实验报告三
PB14000556 陈晓彤

实验题目：ISE RAM IP 核学习 及 REG RAM ALU 协调工作实验
实验要求：

从 ram 中 0 地址和 1 地址读取两个数， 分别赋给 reg0 和 reg1
利用第二次实验的结果(ALU+Regfile)进行斐波拉契运算，运算结果保存在对应的寄存器
运算结果同时保存在对应的 ram 地址中，
即 ram[0]<->reg0, ram[1]<->reg1,ram[2]<->reg2,……
结果在仿真中显示

设计思路：

1. 在 ISE 中建立一个 IP 核，选择双端口 RAM，a 为写入 RAM 端口，b 为读出 RAM 端口，
   根据实验要求，创建一个 coe 文件对 RAM 进行初始化，
   格式为 MEMORY_INITIALIZATION_RADIX=10;
   MEMORY_INITIALIZATION_VECTOR=
   内容为 1，1，0；
2. 建立控制模块，实现两阶段工作：
   a) 从 RAM[0],RAM[1]取数至 REG[0],REG[1],完成准备工作
   b) REG 与 ALU 交互工作，完成斐波那契数列计算，同时将结果存入 RAM
   实现方法：
   建立一个小计数器，完成前两个数值的传输；
   建立 3 分频时钟信号，第一周期 ALU<-REG，第二周期 REG<-ALU，RAM<-ALU，第三周期 REG，RAM 地址增一；

源代码：

```
module ctrl(
    input clk,
    input rst_n,
    output reg [31:0] alu_in1,
    output reg [31:0] alu_in2,
    output    [31:0] alu_out
    );
    reg [4:0] reg_aout1;
    reg [4:0] reg_aout2;
    reg [4:0] reg_ain;
    wire [31:0] reg_dout1;
    wire [31:0] reg_dout2;
    reg [31:0] reg_din;
    reg ram_wea,ram_ena,ram_enb;
    reg [5:0] ram_ain;
    reg [5:0] ram_aout;
    reg [31:0] ram_din;
    wire [31:0] ram_dout;
```

```
    parameter ram_rst_n=1'b0;
    alu alu1(
    .alu_a      (alu_in1),
   .alu_b(alu_in2),
   .alu_op     (5'b1),
   .alu_out    (alu_out)
    );
    regfile reg1(
    .clk  (clk),
    .rst_n      (rst_n),
    .r1_addr  (reg_aout1),
    .r2_addr  (reg_aout2),
    .r3_addr  (reg_ain),
    .r3_din     (reg_din),
    .r3_wr      (1'b1),
    .r1_dout  (reg_dout1),
    .r2_dout  (reg_dout2)
    );
    ram ram1(
    .clka       (clk),
    .ena        (ram_ena),
    .wea        (ram_wea),
    .addra      (ram_ain),
    .dina               (ram_din),
    .clkb       (clk),
    .rstb       (ram_rst_n),
    .enb        (ram_enb),
    .addrb      (ram_aout),
    .doutb      (ram_dout)
    );
    reg [31:0] count;
    reg [1:0] clk_ch;
    parameter count1=4,count2=8;
always@(posedge clk or negedge rst_n)
    begin
        if(~rst_n)
            clk_ch<=2'b0;
        else if(clk_ch==2'b10)
            clk_ch<=2'b0;
        else
            clk_ch<=clk_ch+2'b1;
    end

always@(posedge clk or negedge rst_n)
```

```
    begin
        if(~rst_n)
            count<=0;
        else if(count>count2)
            count<=count;
        else
            count<=count+6'b1;
    end

always@(posedge clk or negedge rst_n)
    begin
        if(count<count1)
            begin
                ram_ena<=0;ram_wea<=0;ram_enb<=1;
                ram_aout<=6'b0;

                reg_ain<=5'b0;reg_din<=ram_dout;
            end
        else if(count<count2)
            begin
                ram_ena<=0;ram_wea<=0;ram_enb<=1;
                ram_aout<=6'b1;

                reg_ain<=5'b1;reg_din<=ram_dout;
            end
        else if(count==count2)
            begin
                ram_ena<=1;ram_wea<=1;ram_enb<=0;
                ram_ain<=6'b10;

                reg_ain<=5'b10;reg_aout1<=5'b0;reg_aout2<=5'b1;
            end
        else
            begin
                case(clk_ch)
                2'b00:
                    begin
                        alu_in1<=reg_dout1;alu_in2<=reg_dout2;
                    end
                2'b01:
                    begin
                        reg_din<=alu_out;ram_din<=alu_out;
                    end
                default:
```

```
                        begin

        reg_ain<=reg_ain+5'b1;reg_aout1<=reg_aout1+5'b1;reg_aout2<=reg_aout2+5'b1;
                            ram_ain<=ram_ain+6'b1;
                        end
                    endcase
                end
        end

endmodule

module regfile(
        input       clk,
        input       rst_n,
        input       [4:0]r1_addr,
        input       [4:0]r2_addr,
        input       [4:0]r3_addr,
        input       [31:0] r3_din,
        input       r3_wr,
        output reg [31:0]   r1_dout,
        output reg [31:0]   r2_dout
        );
        reg [31:0] regfile [31:0];
        integer i;
always@(posedge clk or negedge rst_n)
        begin
            if(~rst_n)
            begin
                for(i=0;i<32;i=i+1)
                    regfile[i]<=32'b0;
            end
            else
                if(r3_wr)
                    regfile[r3_addr]<=r3_din;
        end

always@(negedge clk)
        begin
            r1_dout<=regfile[r1_addr];
            r2_dout<=regfile[r2_addr];
        end
endmodule

module alu(
```

```
    input   signed   [31:0] alu_a,
    input   signed   [31:0] alu_b,
    input            [4:0]  alu_op,
    output   reg     [31:0] alu_out
);
        parameter    A_NOP = 5'h00; //空运算
        parameter    A_ADD  = 5'h01; //符号加
        parameter    A_SUB  = 5'h02; //符号减
        parameter    A_AND = 5'h03;  //与
        parameter    A_OR   = 5'h04;  //或
        parameter    A_XOR = 5'h05;  //异或
        parameter    A_NOR = 5'h06;  //或非
        reg sign;
        always@(*)
        begin
            case(alu_op)
                A_ADD:{sign,alu_out}<=alu_a+alu_b;
                A_SUB:{sign,alu_out}<=alu_a-alu_b;
                A_AND:alu_out<=alu_a&alu_b;
                A_OR:alu_out<=alu_a|alu_b;
                A_XOR:alu_out<=alu_a^alu_b;
                A_NOR:alu_out<=~(alu_a|alu_b);
                A_NOP:alu_out<=alu_out;
                default:alu_out<=alu_out;
            endcase
        end
endmodule
```

仿真图：