计算机组成原理实验报告
LAB03

题目：存储器 RAM

姓名：林祥

学号：PB16020923

实验目的

1、学习如何使用 ISE 的 IP 核

2、学习使用 Xilinx FPGA 内的 RAM 资源

- 例化一个简单双端口的 RAM（32bitx64）
- 使用 coe 文件对 RAM 进行初始化

3、综合利用三次实验的结果，完成以下功能：

- 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,……

实验平台

ISE 14.7

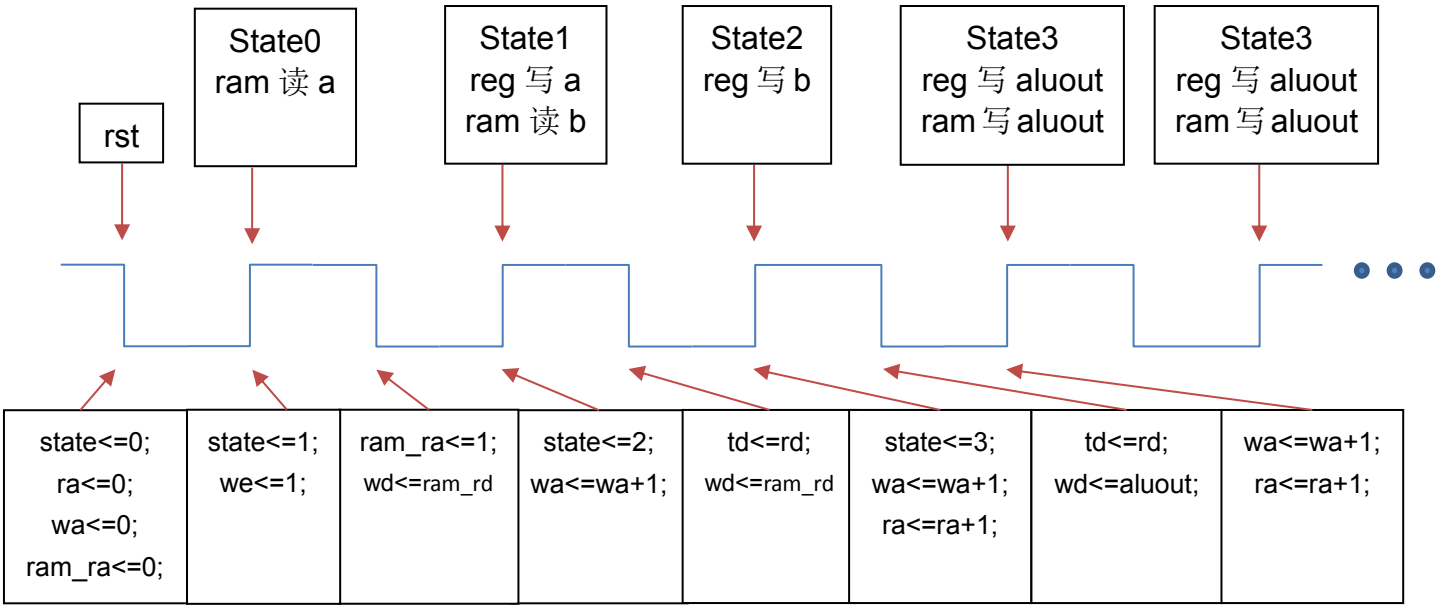
实验过程（分析）

1、模块化设计，一个 alu 模块，一个 regfile 模块，一个 IP 核生成的 ram 模块，一个 control 模块，控制 reg、ram 和 alu，顶层一个 top 模块实例化前几个模块，计算斐波那契数列，初始两个数 a,b 从 ram[0]和 ram[1]读入。

2、alu 模块使用 case 语句判断 7 种操作类型。

3、regfile 模块用组合逻辑读，时序逻辑写。

4、control 模块思路



其中 ra 为 reg 读地址，wa 为 reg/ram 写地址，rd 为 reg 读数据，wd 为 reg/ram 写数据，we 为 reg 写使能，ram_ra 为 ram 读地址，ram_rd 为 ram 读数据，ram_we 为 ram 写使能，td 为临时寄存上一周期的 rd。

我这个想法比较复杂，但还是较清晰的。

实验结果

仿真结果（设 a，b 为斐波那契数列初始值，下面测试两组）

（1）IP 核 ram 设置界面中 Load In 设置 ram 初始化 coe 文件的路径，其中文件内容为

```
MEMORY_INITIALIZATION_RADIX=10;
MEMORY_INITIALIZATION_VECTOR=
2,2,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
```

仿真得 ram 和 reg 数据为

	0	1	2	3
0x0	2	2	4	6
0x4	10	16	26	42
0x8	68	110	178	288
0xC	466	754	1220	1974
0x10	3194	5168	8362	13530
0x14	21892	35422	57314	92736
0x18	150050	242786	392836	635622
0x1C	1028458	1664080	2692538	4356618
0x20	7049156	11405774	18454930	29860704
0x24	48315634	78176338	126491972	204668310
0x28	331160282	535828592	866988874	1402817466
0x2C	2269806340	3672623806	1647462850	1025119360

计算得到的斐波那契数列无误

（2）IP 核 ram 设置界面中 Load In 设置 ram 初始化 coe 文件的路径，其中文件内容为

```
MEMORY_INITIALIZATION_RADIX=10;
MEMORY_INITIALIZATION_VECTOR=
4,7,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
```

仿真得 ram 和 reg 数据为

	0	1	2	3
0x0	4	7	11	18
0x4	29	47	76	123
0x8	199	322	521	843
0xC	1364	2207	3571	5778
0x10	9349	15127	24476	39603
0x14	64079	103682	167761	271443
0x18	439204	710647	1149851	1860498
0x1C	3010349	4870847	7881196	12752043
0x20	20633239	33385282	54018521	87403803
0x24	141422324	228826127	370248451	599074578

计算得到的斐波那契数列无误

附录:

一、 模块源代码

top.v

```
module top(
    input clk,rst_n,

);

    wire [5:0] ra;
    wire [5:0] wa;

    wire [31:0] aluout;

    reg we=1'b1;
    wire [31:0] rd,
    wire [31:0] wd,
    wire [31:0] td,

    wire clkb;
    wire ram_we;
    wire [5:0] ram_ra;
    wire [31:0] ram_rd;

    alu alu1(td,rd,5'h01,aluout);
    regfile regfile1(clk,rst_n,ra,wa,wd,we,rd);
    control control1(clk,rst_n,aluout,ra,rd,wa,wd,td,ram_we,ram_ra,ram_rd);
    ram ram1(clk,ram_we,wa,wd,clk,ram_ra,ram_rd);

endmodule
```

alu.v

```
parameter A_NOP =5'h00; //nop
parameter A_ADD =5'h01; //sign_add
parameter A_SUB =5'h02; //sign_sub
parameter A_AND =5'h03; //and
parameter A_OR  =5'h04; //or
parameter A_XOR =5'h05; //xor
parameter A_NOR =5'h06; //nor

module alu(
```

```
    input [31:0] alu_a,
    input [31:0] alu_b,
    input [4:0] alu_op,
    output reg [31:0] alu_out
    );
    always@(*)
        case (alu_op)
            A_NOP: alu_out = 0;
            A_ADD: alu_out = alu_a + alu_b;
            A_SUB: 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);
            default: alu_out = 0;
        endcase
endmodule
```

regfile.v

```
module regfile(
    input  clk,
    input rst_n,
    input [5:0] rAddr1,//读地址
    input [5:0] wAddr,//写地址
    input [31:0] wDin,//写数据
    input wEna,//写使能
    output [31:0] rDout1//读数据 1
);
    reg [31:0] data [0:63];
    integer i;
    assign rDout1=data[rAddr1];//读

    always@(posedge clk or rst_n)//写和复位
        if(~rst_n)
        begin
            for(i=0; i<64; i=i+1) data[i]<=0;
        end
        else
        begin
            if(wEna)
                data[wAddr]<=wDin;
        end
endmodule
```

control.v

```
module control(
    input clk,rst_n,
    input [31:0] aluout,
    output reg [5:0] ra=6'd0,//reg read addr
    input [31:0] rd,//reg read data
    output reg [5:0] wa=6'd0,
    output reg [31:0] wd,//reg write data
    output reg [31:0] td,//tmp data
    output reg ram_we,//reg write enable
    output reg [5:0] ram_ra,//reg read addr
    input [31:0] ram_rd//reg read data
    );
    reg [1:0] state=2'b11;//state 变化 3->0->1->2->2->2...
    always@(negedge clk) //下降沿改变数据(reg 的 we 总为 1，ram 的 we 在读 a,b 之后开启)

        if(state==2'b11)
            begin
                ram_ra<=6'b0;//从 ram 读 a(改 ram 读地址为 0，下一个上升沿将读 ram[0]=a)
            end
        else if(state==2'b00)
            begin
                wd<=ram_rd;//向 reg 写 a(改写数据，下一个上升沿将写 reg[0]=a)
                ram_ra<=6'b1;//从 ram 读 b(改 ram 读地址为 1，下一个上升沿将读 ram[1]=b)
            end
        else if(state==2'b01)
            begin
                wd<=ram_rd;//向 reg 写 b(改写数据，下一个上升沿将写 reg[1]=b)
                td<=rd; //寄存前一个 rd
            end
        else if(state==2'b10)
            begin
                wd<=aluout;//向 reg,ram 写 aluout(改写数据为 aluout，下一个上升沿将写 reg,ram)
                td<=rd; //寄存前一个 rd
                ram_we=1'b1;//开启 ram 的 we 从 ram[2]开始写
            end

    always@(posedge clk or negedge rst_n) //上升沿改变地址
    begin
        if(~rst_n)//初始化
            begin
                wa<=6'd0;
                ra<=6'd0;
                state<=2'b11;
            end
```

```
        else
            begin
                if(state==2'b11)//这个状态下只有 ram 操作，reg 不操作
                    begin
                        state<=2'b00;
                    end
                else if(state==2'b00)//写 a 状态，reg 读地址不+1，使 reg 读地址落后 reg 写地址一周期
                    begin
                        state<=2'b01;
                        wa<=wa+6'd1;
                    end
                else if(state==2'b01)//写 b 状态
                    begin
                        ra<=ra+6'd1;
                        wa<=wa+6'd1;
                        state<=2'b10;
                    end
                else if(state==2'b10) //写 aluout 状态
                    begin
                        ra<=ra+6'd1;
                        wa<=wa+6'd1;
                    end

            end
    end

endmodule
```

test.v

```
module test(
    );
    reg clk,rst_n;
    top test(
        .clk(clk),
        .rst_n(rst_n),
    );
    always #10 clk=~clk;
    initial begin
        clk=0;
        rst_n=0;
        #20;
        rst_n=1;
    end
endmodule
```