

Lab1-ALU 与寄存器的实现与应用实验报告

PB17010420 庞继泽

(1) 逻辑设计

首先是 ALU 的设计方面，常规运算可以直接使用 Verilog 定义的运算实现，但应注意 ALU 的输出要求包含一个标志信号，在 lab1 实验要求中，这个信号应该能够标志加法有无进位，减法有无借位，有符号数加减法有无溢出。考虑到该标志信号设置为三位，确定 000 为 0 标志，001 为进位标志，010 为借位标志，100 为溢出标志。

进位和借位较易判断，有符号数溢出的判断原则是两个有符号数相加，最高位进位与次高位进位取异或结果为 1，则为溢出。

寄存器设计较为简单，按下不表。

下面说说 ALU 和寄存器的应用。

首先是借助 ALU 实现比较，首先将两数相减直接可以判断是否相等，否则将两数相减，由结果是正是反得到无符号数大小关系，然后比较两数最高位，若不同则直接得到有符号数大小关系，若相同则直接进行无符号数减法得到这两个有符号数的大小关系。

然后是求多个数累加和，连接 ALU 和一个寄存器，ALU 做加法的结果存入寄存器，另一个家数从输入读取，不需要设为有限状态机。

最后是斐波那契数列计算，使用两个寄存器加一个 ALU，采用并行和串行结合方式，ALU 从两个寄存器并行读数做运算，结果存入第一个寄存器，第一个寄存器将上一个数串行输向第二个寄存器即可。

(2) 核心代码

```
always@(*)
begin
    case(s)
        A_ADDU:
        begin
            {f,y} = {3'b0,a} + {3'b0,b}; //如果有进位则f置为1
        end
        A_SUBU:
        begin
            {temp,y} = {1'b1,a}-{1'b0,b};
            if(temp == 0)
            begin
                f = 3'b010;
            end
            else
            begin
                f = 3'b000;
            end
        end
    end
end
```

```

A_ADD:
begin
    if(a[5] == 1)begin
        tempnum[4:0] = a[4:0];
        tempa[4:0] = tempnum[4:0]^5'b11111;
        tempnum[4:0] = tempa[4:0];
        tempa[4:0] = tempnum[4:0] + 5'b00001;
        tempa[5] = a[5];

    end
    else begin
        tempa = a;
    end
    if(b[5] == 1)begin
        tempnum[4:0] = b[4:0];
        tempb[4:0] = tempnum[4:0]^5'b11111;
        tempnum[4:0] = tempb[4:0];
        tempb[4:0] = tempnum[4:0] + 5'b00001;
        tempb[5] = b[5];

    end
    else begin
        tempb = b;
    end
    y = tempa + tempb;
    if((a[5] == b[5]) && (y[5] != a[5])) f = 3'b100;
    else f = 3'b000;
    if(y[5] == 1)begin
        tempnum[4:0] = y[4:0];
        y[4:0] = tempnum[4:0]^5'b11111;
        tempnum[4:0] = y[4:0];
        y[4:0] = tempnum[4:0] + 5'b00001;

    end
    else begin
        tempa = y;
    end
end

A_AND:
begin
    y = a&b;
    f = 3'b000;
end
A_OR:
begin
    y = a|b;
    f = 3'b000;
end
A_NOT:
begin
    y = ~a;
    f = 3'b000;
end
A_XOR:
begin
    y = a^b;
    f = 3'b000;
end
endcase

```

上图为 ALU 的基本运算实现，注意其中有符号数运算是先换成补码进行运算后再换回成源

码表示的，因而那一部分代码较为繁琐。

```
assign tempsx[4:0] = x[4:0];
assign tempsy[4:0] = y[4:0];
assign tempsx[5] = 1'b0;
assign tempsy[5] = 1'b0;
ALU U1(.s(3'b010),.a(x),.b(y),.y(temp1),.f(rend1));
ALU U2(.s(3'b010),.a(tempsx),.b(tempsy),.y(temp2),.f(rend2));

always@(*)begin
    if(temp1 == 6'b000000)begin
        eq = 1'b1;
        ug = 1'b0;
        ul = 1'b0;
        sg = 1'b0;
        sl = 1'b0;
    end
    else begin
        if(rend1 == 3'b010)begin
            ul = 1'b1;
            ug = 1'b0;
            eq = 1'b0;
        end
        else begin
            ul = 1'b0;
            ug = 1'b1;
            eq = 1'b0;
        end
        if((x[5] < y[5]) || ((x[5] == y[5]) && (rend2 == 3'b000)))begin
            sg = 1'b1;
            sl = 1'b0;
            eq = 1'b0;
        end
        else begin
            sg = 1'b0;
            sl = 1'b1;
            eq = 1'b0;
        end
    end
end
end
```

上图是使用 ALU 判断两个数的代码

```
module ALU_register_add(
    input [5:0] x,
    input rst,
    input clk,
    output [5:0] s
);
    wire [3:0] temp;
    wire [5:0] in;
    ALU U1(.s(3'b001),.a(x),.b(s),.y(in),.f(temp));
    register U2(.in(in),.rst(rst),.clk(clk),.en(1'b1),.out(s));

endmodule
```

上图是使用 ALU 和 register 求累加数和的代码

```
module ALU_register_fn(
    input [5:0] f0,
    input [5:0] f1,
    input rst,
    input clk,
    output [5:0] out0,
    output [5:0] out1
);
    wire [5:0] temp0;
    wire [5:0] temp1;
    wire [5:0] alu_out;
    //wire [5:0] out0;
    //wire [5:0] out1;

    wire [2:0] f;

    assign temp0 = rst?f1:alu_out;
    assign temp1 = rst?f0:out0;
    register U0(.rst(1'b0),.in(temp0),.out(out0),.en(1'b1),.clk(clk));
    register U1(.rst(1'b0),.in(temp1),.out(out1),.en(1'b1),.clk(clk));
    ALU A0(.s(3'b001),.a(out0),.b(out1),.y(alu_out),.f(f));

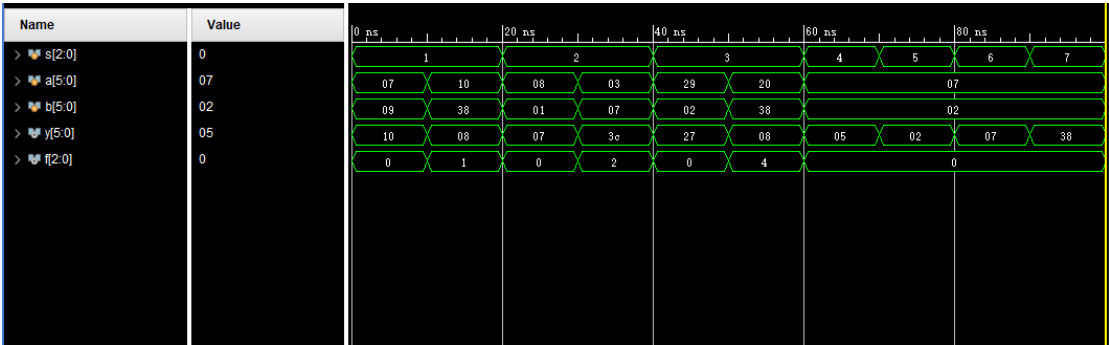
endmodule
```

上图为求斐波那契额数的代码，为方便检测，这里设置了两个输出，其实就是两个寄存器的值。

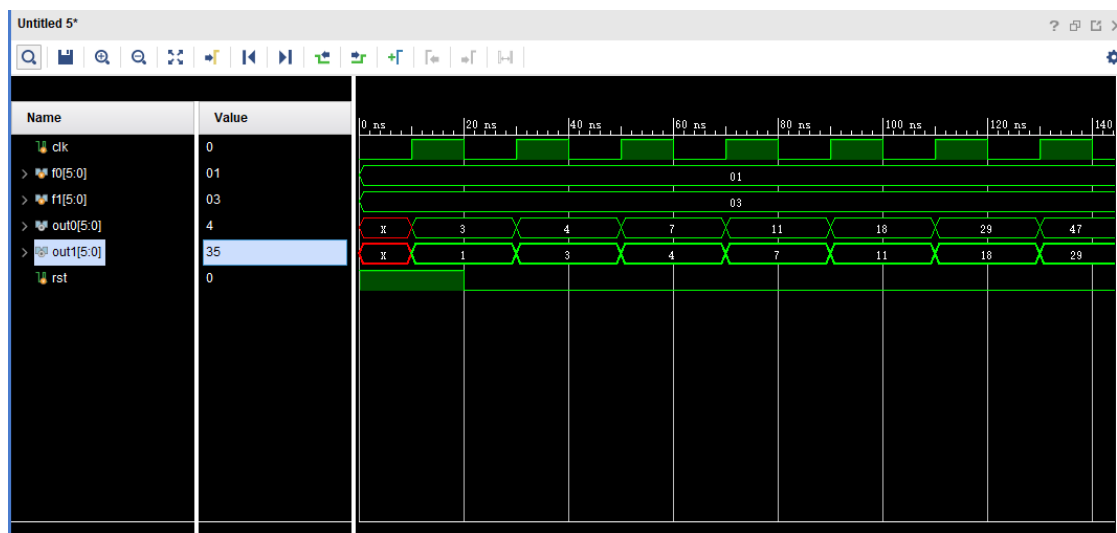
(3) 仿真结果与结果分析

```
parameter A_ADDU = 3'b001; //加无符号数
parameter A_SUBU = 3'b010; //减无符号数
parameter A_ADD = 3'b011; //加有符号数
parameter A_SUB = 3'b100; //减有符号数
parameter A_AND = 3'b101; //与
parameter A_OR = 3'b110; //或
parameter A_NOT = 3'b111; //非
parameter A_XOR = 3'b000; //异或

//对于标志位f, 1进位, 2借位, 4溢出, 0表示0标志
```




上图为 ALU 的仿真结果，可以看到，计算结果符合实验要求。



上图为斐波那契数列仿真情况，可以看到，实验结果符合要求。

(4) 下载结果与结果分析



http://ecard.ustc.edu.cn
联系电话: 63603949

手机应用



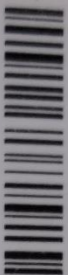
昇成



减法

此卡妥善保管，避免折损，限本人使用。丢失需立刻挂失。拾获此卡，请交回一卡通中心。

2201704008
校园一卡通管理结算中心
<http://ecard.ustc.edu.cn>
联系电话：63603949

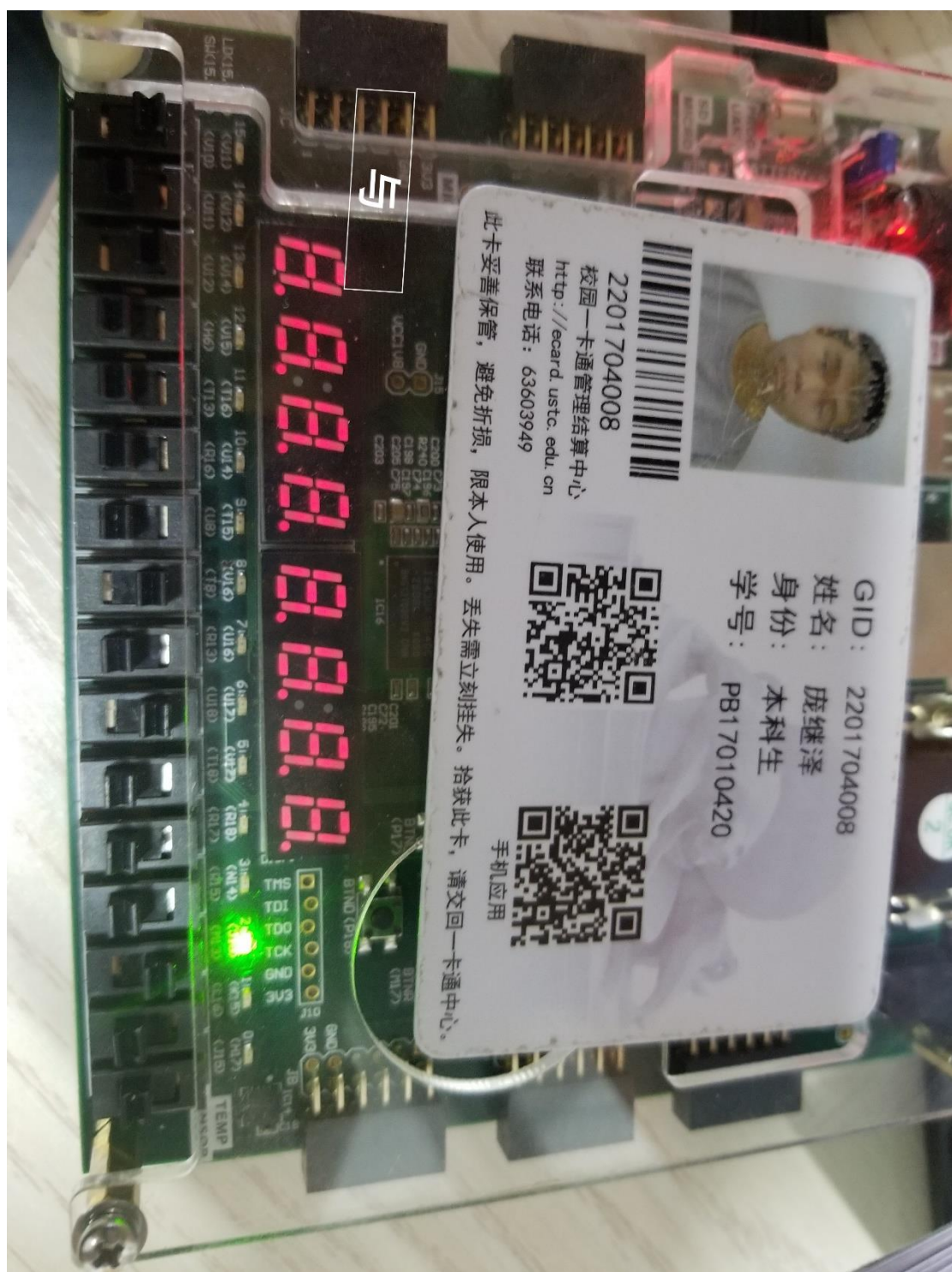


GIID : 2201704008
姓名：庞继泽
身份：本科生
学号：PB17010420

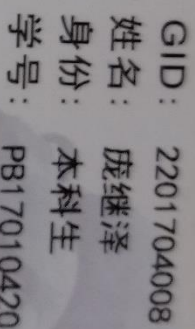


手机应用





上面三张图是 ALU 的下载结果，操作已在图片上说明，下载结果符合实验预期。



校园一卡通管理结算中心
http://ecard.ustc.edu.cn
联系电话: 63603949



手机肉旺

此項保險，雖免折損，限本人使用。若失離位居住者，亦不得以圖一失而兩失。







上面三张图是斐波那契数列下载结果，实验要求符合实验预期。

(5) 实验总结

首先这次实验让我复习了 verilog 的基本写法和仿真文件写法与实验流程，这次实验让我对 ALU 的结构与寄存器的结构有了更深的认识，并且认识到模块组合带来的巨大作用。在实验进行过程中，我复习了进位借位溢出的判断方法，对于 ALU 的实现有了更具体的认识。

(6) 建议

请务必让实验要求更明确。