Lab1_运算器与寄存器实验报告

PB17111623 范睿

一、 逻辑设计与核心代码

1. 运算实现

a) 加法：verilog 加

b) 减法：减数求反加一后利用 verilog 加

c) 与：verilog 与

d) 或：verilog 或

e) 非：verilog 非

f) 异或：verilog 异或

2. 进位/借位判断：

a) 进位：

```
y1 = a + b;
//判断进位
   {f[2], y} = y1;
```

y1 的位数比 a 与 b 多 1，若 y1 最高位为 1，说明产生进位，将 f[2]赋值为 1；若 y1 最高位为 0，说明不产生进位，则将 f[2]赋值为 0。

b) 借位：

```
b1 = ~b + 1;//b1=-b
y1 = a + ~b + 6'b000001;
{f[2], y} = y1;//判断借位
```

将减数求反。

y1 的位数比 a 与 b 多 1。将 y1 赋值为 a 与（-b）的和。

若 y1 最高位为 1，说明产生借位，将 f[2]赋值为 1；若 y1 最高位为 0，说明产生借位，将 f[2]赋值为 0。

3. 溢出判断

a) 加法判断溢出：

```
//判断溢出
   if(a[5]==0 && b[5]==0)begin //ab均正
        if(y[5]==1) f[1] = 1;//溢出
        else f[1] = 0;
   end
   else if(a[5]==1 && b[5]==1) //ab均负
        if(y[5]==0) f[1] = 1;
        else f[1]=0;
 end
```

加法时，只有 a 与 b 均为正或 a 与 b 均为负时才有机会产生溢出。

若 ab 均正，则当结果 y 最高位为 1 时，即 y 为负时产生溢出。

若 ab 均负，则当结果 y 最高位为 0 时，即 y 为正时产生溢出。

b) 减法判断溢出：

```
//判断溢出
if(a[5]==0 && b1[5]==0)begin //a,-b均正
    if(y[5]==1) f[1] = 1;//溢出
    else f[1] = 0;
end
else if(a[5]==1 && b1[5]==1) //a,-b均负
    if(y[5]==0) f[1] = 1;
    else f[1]=0;
//由于100000求反+1后仍为100000，因此单独考虑
if(b==6'b100000)begin
    if(a[5]==0) f[1]=1;
    else f[1]=0;
end
```

减法时，只有在 a 与-b 均为正数或负数时才有机会产生溢出。

若 a，-b 均正，则当结果 y 最高位为 1 时，即 y 为负时产生溢出。

若 a，-b 均负，则当结果 y 最高位为 0 时，即 y 为正时产生溢出。

另外需要将单独讨论减数为 100000，即 b=-32 时的情况。因为将 100000 求反加一后仍为 100000，达不到取反的效果。所以若减数为 100000，当被减数 a 为正数时一定会溢出，f[1]置为 1。

4. 零标志判断

```
f[0] = (&(~y));//zero flag
```

若 y 为 000000，零标志为 1。

5. 斐波那契数列实现

a) 初始化

```
initial begin
    f1_old = 0;
    f2_old = 0;
    flag = 0;
    s = 3'b001;
end
```

将 f1_old，f2_old, flag 均初始化为零。

f1_old,f2_old 为斐波那契数列中任意连续三个值的前两个值。

flag 用来判断当前需要将 f1_old 和 f2_old 循环更新还是更新为输入值。

s 为输入给 ALU 的参数，赋值为 1 表示让 ALU 做加法操作。

b) 计算操作数

```
always@(*)begin
    if(rst == 1) flag = 0;
    else if(flag == 0)begin
        f1_old = f1;
        f2_old = f2;
        flag = 1;
    end
    else if(flag == 1)begin
        f1_old = f2_old;
        f2_old = fn;
    end
end
```

若 rst 有效（为 0），将 flag 置为 0，这时两操作数将被置为输入值，flag 置为 1。

若 rst 无效（为 1），将 f1_old 置为 f2_old, f2_old 置为 fn。（fn 为上一次的计算结果）

c) 计算结果

```
always@(*)begin
    if(rst==1)fn = 0;
    else fn=fn_;
end
ALU L(.a(f1_old),.b(f2_old),.s(s),.y(fn_),.f(f));
```

若 rst 有效（为 0），将结果 fn 置为 0。

若 rst 无效（为 1），将结果 fn 置为 ALU 的计算结果 fn_。

对 ALU 的模块调用为：将 f1_old 和 f2_old 作为两操作数的输入，s 为控制输入，fn_为结果输出，f 为进位溢出和零标志判断。

6. 数码管对斐波那契数列的实现

a) 斐波那契数列实现

同前

b) 时钟分频

```
clk_wiz_0 clk1(clk_5M,rst,locked,clk_100M);

always@(posedge clk_5M)begin
    if(count>=21'd9999)begin
        count<=0;
    end
    else begin
        count<=count+1;
    end
end
always@(posedge clk_5M)begin
    clk_500<=(count>=21'd4999)?1:0;
end
```

调用时钟模块产生 5MHz 的时钟，再利用分频模块产生 500Hz 的时钟进行扫描。

c) 更新位选信号 an

```
always@(posedge clk_500)begin
    case(an)
        8'b11111110:an<=8'b11111101;
        8'b11111101:an<=8'b11111011;
        8'b11111011:an<=8'b11110111;
        8'b11110111:an<=8'b11101111;
        8'b11101111:an<=8'b11011111;
        8'b11011111:an<=8'b10111111;
        8'b10111111:an<=8'b01111111;
        8'b01111111:an<=8'b11111110;
    endcase
end
```

每当 500Hz 时钟上升沿到达时，跟新 an 的值。为 0 的位表示当前扫描到第几个位置。（an 赋初值为 8'b1111110）

d) 更新段选信号 seg

```
integer num;
always@(*)begin
    num = 1*fn[0] + 2*fn[1] + 4*fn[2] + 8*fn[3] + 16*fn[4] + 32*fn[5];
    case(an)
        8'b11111110:num = (num/10000000)%10;
        8'b11111101:num = (num/1000000)%10;
        8'b11111011:num = (num/100000)%10;
        8'b11110111:num = (num/10000)%10;
        8'b11101111:num = (num/1000)%10;
        8'b11011111:num = (num/100)%10;
        8'b10111111:num = (num/10)%10;
        8'b01111111:num = num%10;
    endcase
end
```

fn 为计算好的结果的输入，num 为 fn 转为十进制的值，为整型。

根据 an 位选的不同，选择将 num 的那个位置上的值提取出来。

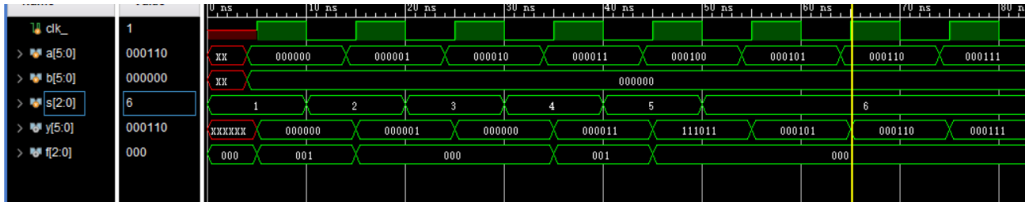
```
always@(num)begin
    case(num)
        0:seg = 7'b1000000;
        1:seg = 7'b1111001;
        2:seg = 7'b0100100;
        3:seg = 7'b0110000;
        4:seg = 7'b0011001;
        5:seg = 7'b0010010;
        6:seg = 7'b0000010;
        7:seg = 7'b1111000;
        8:seg = 7'b0000000;
        9:seg = 7'b0010000;
    endcase
      seg = ~seg;
end
```

根据 num 的不同，将段选信号 seg 赋值为不同值以显示正确数字。

二、 仿真与下载结果

1. ALU

仿真：



S 为 1,2,3,4,5,6 分别对应加，减，与，或，非，异或。

下载：

加法：

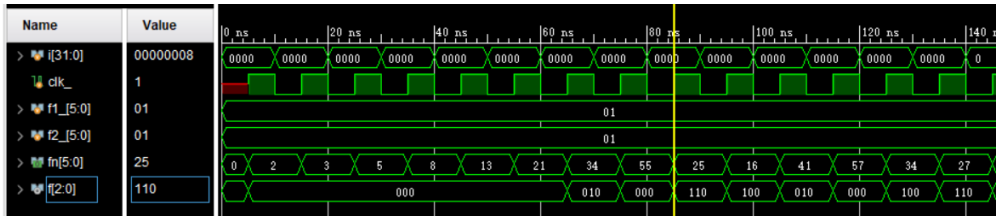
输入为 011111 和 011111。Led 灯左端显示溢出。

减法

输入为 000000 和 000000。Led 灯左端显示借位（左一）和零标志（左三）。

2. 斐波那契

仿真：



黄线之后可以由最后一行的 f 值看出已经溢出。

下载：

输入为 1 和 1

3. 数码管实现斐波那契

输入为 1 和 1

（灭掉的数码管是因为照相机可以分辨出来当前的扫描频率，500Hz，但人眼分辨不出）

三、 结果分析和实验总结

从斐波那契的实验结果和仿真结果来看，ALU 的运算结果没有问题。

但我觉得我 ALU 的实现有点繁琐。一个是溢出判断。不是通过最高位和次高位的进位判断的，而是通过结果和操作数的数值，在后期 cpu 的实现上可能会拖慢速度。另一个是灵活性不太好。我这个 ALU 只能运算六位及以下的操作数，如果要运算六位以上的数需要大改，不好。

四、完整代码

1.ALU

```
module ALU(
    input clk,
    input [5:0] a,
    input [5:0]b,
    input [2:0]s,
    output reg [5:0]y,
    output reg [2:0]f//[进位/借位，溢出，零标志]
    );
    initial begin
        f = 3'b000;
    end
    reg [6:0]y1;
    reg [5:0]_y;
    reg [5:0]b1;
    reg c;
    always@(posedge clk)begin
    f = 3'b000;
        case(s)
            3'b001://add
                begin
                y1 = a + b;
                //判断进位
                    {f[2], y} = y1;
                //判断溢出
                    if(a[5]==0 && b[5]==0)begin //ab 均正
                        if(y[5]==1) f[1] = 1;//溢出
                        else f[1] = 0;
                    end
                    else if(a[5]==1 && b[5]==1) //ab 均负
                        if(y[5]==0) f[1] = 1;
                        else f[1]=0;
                 end
            3'b010://sub
                begin
                    b1 = ~b + 1;//b1=-b
                    y1 = a + ~b + 6'b000001;
                    {f[2], y} = y1;//判断借位
                    //判断溢出
                    if(a[5]==0 && b1[5]==0)begin //a,-b 均正
                        if(y[5]==1) f[1] = 1;//溢出
                        else f[1] = 0;
                    end
                    else if(a[5]==1 && b1[5]==1) //a,-b 均负
                        if(y[5]==0) f[1] = 1;
                        else f[1]=0;
                    //由于 100000 求反+1 后仍为 100000，因此单独考虑
                    if(b==6'b100000)begin
                        if(a[5]==0) f[1]=1;
                        else f[1]=0;
                    end
                end
            3'b011://and
                begin y <= a&b; end
            3'b100://or
                begin y <= a|b; end
            3'b101://not
                begin y <= ~a; end
            3'b110://xor
                begin y <= a^b; end
        endcase
        f[0] = (&(~y));//zero flag
    end
endmodule
```

ALU 仿真

```
module add_sim(

    );
    reg clk_;
    integer i,j;
    reg [5:0]a;
    reg [5:0]b;
    reg [2:0]s;
    wire [5:0]y;
    wire [2:0]f;
    ALU L(.clk(clk_), .a(a), .b(b), .s(s), .y(y), .f(f));
    initial begin
        for(i=0;i<62;i=i+1) begin
            #5 assign clk_=1;
            #5 assign clk_=0;
        end
    end
    initial begin
        #4 a = 6'b000000;
        for(j=0;j<62;j=j+1) begin
            #10 a <= a + 6'b000001;
        end
    end
    initial begin
        #4 b = 6'b000000;
//          #320 b = 6'b111010;
    end
    initial begin
        s = 3'b001;
        #10 s = 3'b010;
        #10 s = 3'b011;
        #10 s = 3'b100;
        #10 s = 3'b101;
        #10 s = 6'b110;
        end
endmodule
```

2. 斐波那契

```
module FIB(
    input [5:0]f1,
    input [5:0]f2,
    input clk,
    input   rst,
    output reg [5:0]fn
    );
    reg [5:0]f1_old;
    reg [5:0]f2_old;
    reg flag;
    reg [2:0]s;
    wire [2:0]f;
    wire [5:0]fn_;
    initial begin
        f1_old = 0;
        f2_old = 0;
        flag = 0;
        s = 3'b001;
    end
    always@(posedge clk or posedge rst)begin
        if(rst == 1) flag = 0;
        else if(flag == 0)begin
            f1_old = f1;
            f2_old = f2;
            flag = 1;
        end
        else if(flag == 1)begin
             f1_old = f2_old;
             f2_old = fn;
        end
    end
    always@(*)begin
        if(rst==1)fn = 0;
        else fn=fn_;
    end
    ALU L(.a(f1_old),.b(f2_old),.s(s),.y(fn_),.f(f));
endmodule
```

斐波那契仿真

```
module FIB_sim(

    );
integer i;
reg clk_;
reg [5:0]f1_;
reg [5:0]f2_;
wire [5:0]fn_;
reg rst;
FIB L(.f1(f1_),.f2(f2_),.clk(clk_),.fn(fn_),.rst(rst));
initial begin
        for(i=0;i<62;i=i+1) begin
            #5 assign clk_=1;
            #5 assign clk_=0;
        end
    end
initial begin
    f1_=6'b000001;      #500 f1_=6'b000011;end
    initial begin
    f2_=6'b000001;
    #500 f2_=6'b000011;
    end
initial begin
    rst = 0;
    #500 rst = 1;
    #5 rst = 0;
end
endmodule
```

3.斐波那契数码管

```
module FIB_tube(
    input [5:0]f1,
    input [5:0]f2,
    input clk_100M,
    input clk,
    input   rst,
    output [6:0]seg,
    output reg [7:0]an
    );
    wire clk_5M;
    wire locked;
    reg [5:0]fn;
    reg clk_500;
    reg [21:0]count;

    reg [5:0]f1_old;
    reg [5:0]f2_old;
    reg flag;
    reg [2:0]s;
    wire [2:0]f;
    wire [5:0]fn_;

    clk_wiz_0 clk1(clk_5M,rst,locked,clk_100M);

    always@(posedge clk_5M)begin
        if(count>=21'd9999)begin
            count<=0;
        end
        else begin
            count<=count+1;
        end
    end
    always@(posedge clk_5M)begin
        clk_500<=(count>=21'd4999)?1:0;
    end
```

```
    initial begin
        f1_old = 0;
        f2_old = 0;
        flag = 0;
        s = 3'b001;
        an = 8'b11111110;
        count = 0;
    end
    always@(posedge clk or posedge rst)begin
        if(rst == 1) flag = 0;
        else if(flag == 0)begin
            f1_old = f1;
            f2_old = f2;
            flag = 1;
        end
        else if(flag == 1)begin
            f1_old = f2_old;
            f2_old = fn;
        end
    end
    always@(*)begin
        if(rst==1)fn = 0;
        else if(clk==1) fn = fn_;
    end

    always@(posedge clk_500)begin
        case(an)
            8'b11111110:an<=8'b11111101;
            8'b11111101:an<=8'b11111011;
            8'b11111011:an<=8'b11110111;
            8'b11110111:an<=8'b11101111;
            8'b11101111:an<=8'b11011111;
            8'b11011111:an<=8'b10111111;
            8'b10111111:an<=8'b01111111;
            8'b01111111:an<=8'b11111110;
        endcase
    end

    B2D ff(.fn(fn),.an(an),.seg(seg));

    ALU L(.a(f1_old),.b(f2_old),.s(s),.y(fn_),.f(f));
endmodule
```

```
module B2D(
    input [5:0]fn,
    input [7:0]an,
    output reg [6:0]seg
    );

    integer num;
    always@(*)begin
        num = 1*fn[0] + 2*fn[1] + 4*fn[2] + 8*fn[3] + 16*fn[4] + 32*fn[5];
        case(an)
            8'b11111110:num = (num/10000000)%10;
            8'b11111101:num = (num/1000000)%10;
            8'b11111011:num = (num/100000)%10;
            8'b11110111:num = (num/10000)%10;
            8'b11101111:num = (num/1000)%10;
            8'b11011111:num = (num/100)%10;
            8'b10111111:num = (num/10)%10;
            8'b01111111:num = num%10;
        endcase
    end

    always@(num)begin
        case(num)
            0:seg = 7'b1000000;
            1:seg = 7'b1111001;
            2:seg = 7'b0100100;
            3:seg = 7'b0110000;
            4:seg = 7'b0011001;
            5:seg = 7'b0010010;
            6:seg = 7'b0000010;
            7:seg = 7'b1111000;
            8:seg = 7'b0000000;
            9:seg = 7'b0010000;
        endcase
//          seg = ~seg;
    end

endmodule
```