Lab1_运算器与寄存器 实验报告

姓名：牛岱 学号：PB16051069 实验日期：2019-3-20

实验题目：

运算器与寄存器

实验目的：

用Verilog写一个ALU和register, 再使用它们实现斐波那契额数列。

实验平台：

Vivado

实验过程：

先编写ALU模块：

```
module lab1_1(
    input [2:0] s,
    input [15:0] a,
    input [15:0] b,
    input sign,
    output [15:0] yout,
    output CF,
    output reg V,
    output Z
    );
    wire [16:0] yplus;
    wire [16:0] yminus;
    wire [15:0] yand;
    wire [15:0] yor;
    wire [15:0] ynot;
    wire [15:0] ynotor;
    reg [16:0] y;
    assign yout[0]=y[0];
    assign yout[1]=y[1];
    assign yout[2]=y[2];
    assign yout[3]=y[3];
    assign yout[4]=y[4];
    assign yout[5]=y[5];
    assign yout[6]=y[6];
    assign yout[7]=y[7];
    assign yout[8]=y[8];
    assign yout[9]=y[9];
    assign yout[10]=y[10];
    assign yout[11]=y[11];
    assign yout[12]=y[12];
```

```
    assign yout[13]=y[13];
    assign yout[14]=y[14];
    assign yout[15]=y[15];

    assign yplus=a+b;
    assign yminus=a-b;
    assign yand=a&b;
    assign yor=a|b;
    assign ynot=~a;
    assign ynotor=a^b;
    assign
Z=~y[0]&~y[1]&~y[2]&~y[3]&~y[4]&~y[5]&~y[6]&~y[7]&~y[8]&~y[9]&~y[10]&~y[11]&~y[12]
&~y[13]&~y[14]&~y[15];
    assign CF=~sign&y[16]&~s[2]&~s[1];

    always @ (*)
    begin
        V=0;
        if (s==3'b000)
        begin
            y=yplus;
            if (sign && a[15]==b[15] && a[15]!=yplus[15]&& ~s[2] && ~s[1]) V=1;
        end
        else if (s==3'b001)
        begin
            y=yminus;
            if (sign && a[15]==b[15] && a[15]!=yminus[15] && ~s[2] && ~s[1]) V=1;
        end
        else if (s==3'b010) y=yand;
        else if (s==3'b011) y=yor;
        else if (s==3'b100) y=ynot;
        else if (s==3'b101) y=ynotor;
    end//always
endmodule
```

可以看到, 我们的加法单元是16位的, 然后我们声明输入, a和b为输入的两个数, s为操作选择, 为3位, 000 对应加法, 001 减法, 010 是AND, 011是 OR, 100是 NOT, 101是 XOR, 这些操作都能简单地用一句话解决, 比如XOR, 直接用^位操作符, NOT直接用~操作符。

然后进行 register 模块的编写：

```
module lab1_2(
    input [15:0] in,
    input en,
    input rst,
    input clk,
    output reg [15:0] out
    );
    always @ (posedge clk, posedge rst)
    begin
        if (rst) out=0;
```

```
        else if (en) out=in;
    end
endmodule
```

en为 enable, 只需每个时钟的上涨, 或是reset的上涨, 就触发一次清零, 或者输出之前存储的值。

然后进行Fibonacci数列的编写：

```
module lab1_3_c(
    input clk,
    input [1:0] f0,
    input [1:0] f1,
    input rst,
    output [15:0] fn
    );
    wire CF,V,Z;
    wire [15:0] a;
    wire [15:0] b;
    wire [15:0] sum;
    assign fn = a;

    lab1_2 reg1(.in(rst?f1:a),.en(1),.rst(0),.clk(clk),.out(b));//b<=a
    lab1_2 reg2(.in(rst?f0:sum),.en(1),.rst(0),.clk(clk),.out(a));//a<=sum
    lab1_1 plus(.s(0),.a(a),.b(b),.sign(0),.yout(sum),.CF(CF),.V(V),.Z(Z));

endmodule
```

原理很简单, f0 和 f1 是数列的前两个值, 然后直接让 sum 为 a 和 b的和, 没时钟走过一次, 就让 a = sum b = a 这样每次输出的值, 也就是 fn 都会变成原来的前面两个数列元素的和，原理就是这样。

实验结果：

下载到板子上，运行正常。