计算机组成原理 实验报告

姓名：杨佳熹　学号：PB17000050　实验日期：2019-3-21

一、实验题目：

Lab01 运算器

二、实验目的：

设计一算术逻辑运算单元（ALU），实现加减或与异或非功能；利用前述的ALU 模块与适当的硬件电路，完成输出斐波那契数列。

三、实验平台：

Vivado

四、实验过程：

此实验分为两部分 ALU 运算器和 ALU 的简单应用。

实验一代码截图：

利用 always 语句实现基本运算；

```
module ALU(
input [2:0] s,
output reg [5:0]y,
input [5:0] a,b,
output reg [2:0]f
    );
    reg [6:0]t;
    reg [6:0] a1;
    reg [6:0]b1;

always @(*)
begin
    if(s==0)
        begin
            a1=a;
            b1=b;
            t=a1+b1;
            y=a+b;
            f=t[6];
        end
    else if(s==1)
        begin
            y=a-b;
            if(a<b)
            f=1;
            else
            f=0;
        end
    else if(s==2)
        begin
            y[0]=a[0]&&b[0];
            y[2]=a[2]&&b[2];
            y[3]=a[3]&&b[3];
            y[4]=a[4]&&b[4];
            y[5]=a[5]&&b[5];
        end
    else if(s==3)
        begin
            y[0]=a[0]||b[0];
            y[1]=a[1]||b[1];
            y[2]=a[2]||b[2];
            y[3]=a[3]||b[3];
            y[4]=a[4]||b[4];
            y[5]=a[5]||b[5];
        end
    else if(s==4)
        begin
            y[0]=~a[0];
            y[1]=~a[1];
            y[2]=~a[2];
            y[3]=~a[3];
            y[4]=~a[4];
            y[5]=~a[5];
        end
    else if(s==5)
        begin
            y[0]=a[0]^b[0];
            y[1]=a[1]^b[1];
            y[2]=a[2]^b[2];
            y[3]=a[3]^b[3];
            y[4]=a[4]^b[4];
            y[5]=a[5]^b[5];
        end
```

```
module ALU_tb(
    );
    reg [2:0] s;
    wire [5:0]y;
    reg [5:0] a,b;
    wire [2:0]f;
    ALU DUT(.s(s),.y(y),.a(a),.b(b),.f(f));
    initial
        begin
            s=0;a=0;b=1;
            #10 a=1;b=5;
            #10 a=42;b=42;
            #10 s=1;a=15; b=14;
            #10 a=2; b=3;
            #10 s=2;b=3;a=3;
            #10 a=2;b=15;
            #10 s=3;a=14;b=15;
            #10 a=2;b=16;
            #10 s=4;a=34;b=15;
            #10 a=27;b=15;
            #10 s=5;a=25;b=15;
            #10 a=2;b=35;
        end
endmodule
```

实验二 代码截图：

```
module fibs(a,b,y,clk,rst);
input [2:0]a;
input [2:0]b;
output reg [7:0] y;
input clk;
reg [7:0]t;
reg [7:0]a1;
reg [7:0]b1;
input rst;
always @(posedge clk)
begin
if(rst==1)
    begin
        a1=a;
        b1=b;
        y=a1+b1;
    end
    begin
        t=a1+b1;
        a1=b1;
        b1=t;
        y=t;
    end
end

endmodule
```

```
module fibs_tb( );
reg [2:0]a;
reg [2:0]b;
wire [7:0]y;
reg clk;
reg rst;
fibs DUT(.a(a),.b(b),.y(y),.clk(clk),.rst(rst));
initial
    begin
        a=1;b=1;clk=0;rst=0;
        forever
            begin
                #10 clk=1;
                #10 clk=0;
            end
    end
initial
    begin
        #5 rst=1;
        #10 rst=0;
    end

endmodule
```

利用时序逻辑电路把上一个时钟的值进行运算；

累加器代码截图：

```
module acm(a,clk,rst,s);
input [5:0]a;
output reg [5:0]s;
input rst;
input clk;
reg [5:0]t;
always@(posedge clk)
    begin
        if(rst)
            s=0;
        else
        begin
            t=s;
            s=t+a;
        end
    end
endmodule
```

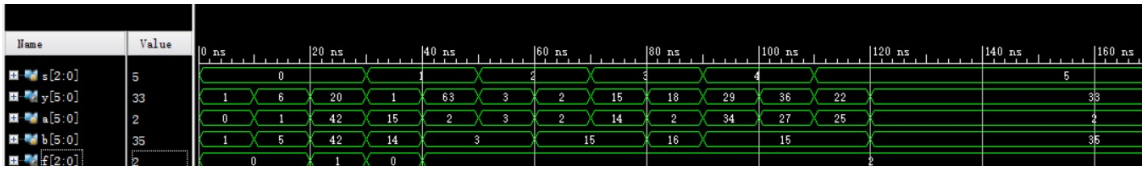
```
module acm_tb( );
reg [5:0]a;
wire [5:0]s;
reg rst;
reg clk;
acm DUT (.a(a),.s(s),.rst(rst),.clk(clk));
initial
    begin
    clk=0;
    forever
        #10 clk=~clk;
    end
initial
    begin
        rst=1;
        a=0;
        #15 a=0;rst=0;
        #10 a=4;
        #10 a=13;
        #10 a=12;
        #10 a=8;
    end
endmodule
```

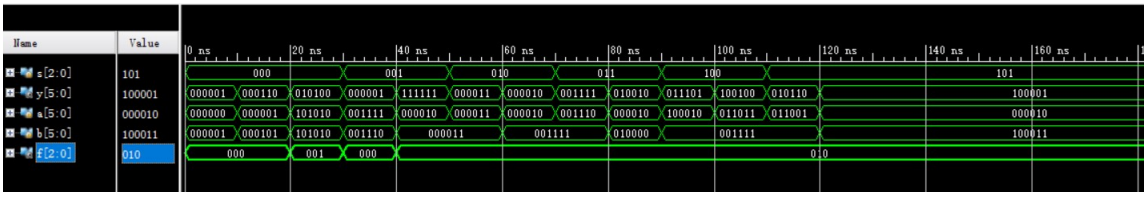
五、实验结果：

此处讲述实验结果，必须附上最后的仿真波形图或是下载到板子上的实拍结果图（视实验要求而定）。

实验一：

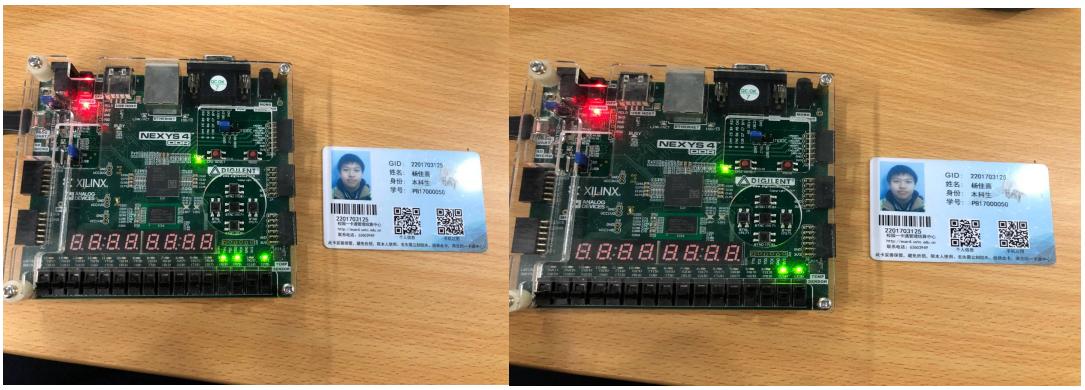
仿真截图：

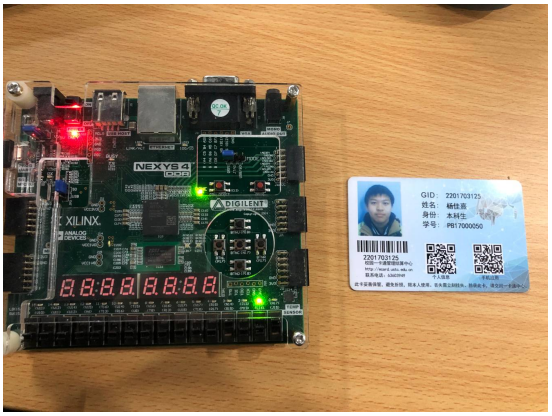




加法进位和减法借位均满足结果。

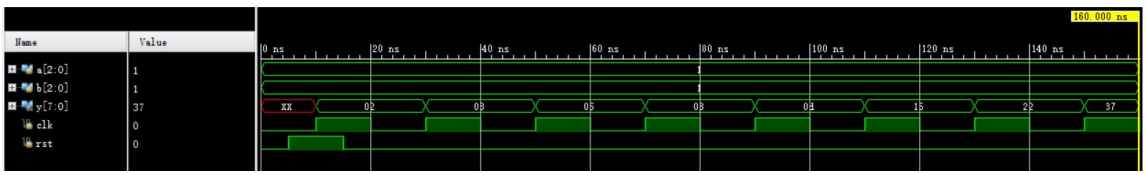
下载结果：





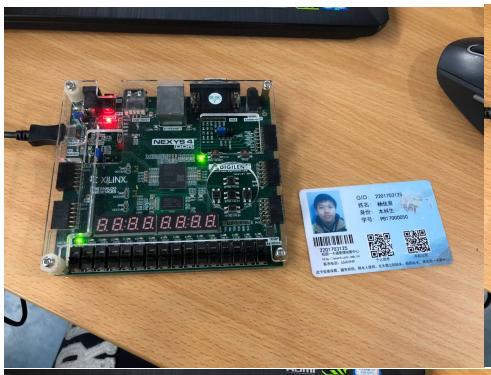
实验二：

仿真截图：



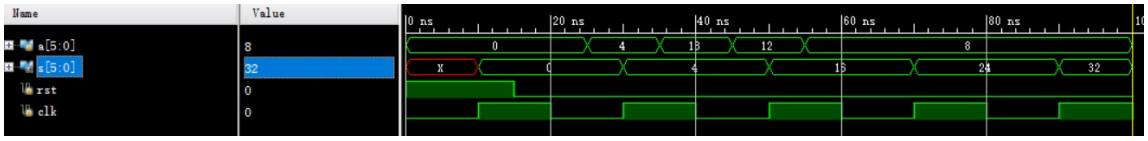
在 rst=1 时时间上升沿；赋初值 a+b；之后每经历上升沿生成下一项。

下载结果：

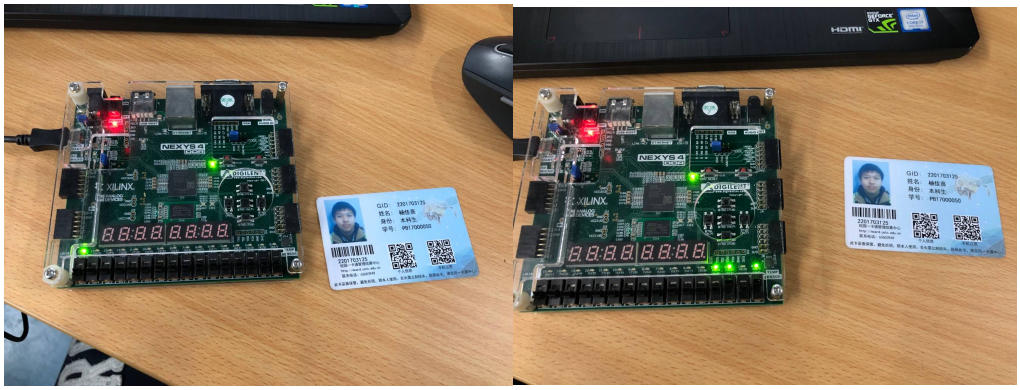


累加器：

仿真截图：



下载结果：



在 rst 经历时间上升沿是清零，之后没经历一个上升沿会加 a；

六、心得体会：

此次实验为第一次实验，难度不大，主要让我们复习了 verilog 语句的基本语法。在实验中遇到了 mult-driven 的问题（一个变量在不同的 always 里赋值），只需把两个 always 合在一起即可通过。另外，debug 的时候可以打开提示中所给路径的文件来查看问题所在，遗憾的是，上学期的数电实验没有发现这个

技巧从而浪费了很多时间。在赋值硬件引脚时，简单回顾了手动控制 clk 的技巧，在 xdc 文件内加上一句代码即可实现。