计算机组成原理 实验报告

姓名： 杨佳熹 学号：PB17000050 实验日期：2019-3-28

一、实验题目：

Lab02 数据通路与状态机

二、实验目的：

设计一带有排序功能的时序逻辑电路，实现四个数排序功能；类似设计带有除法功能的电路，通过时钟控制执行。

三、实验平台：

Vivado

四、实验过程：

从 ppt 中三个数排序的算法推广到四个数排序，利用 always 语句实现类似冒泡的排序算法。

排序代码截图：

```
module sort(x0,x1,x2,x3,s0,s1,s2,s3,clk,rst,done);
input [3:0]x0,x1,x2,x3;
output reg [3:0]s0,s1,s2,s3;
input rst,clk;
output reg done;
reg [3:0] flag;
reg [3:0] t0,t1,t2,t3,t;
reg [3:0] state,c_state,n_state;
always @(posedge clk )
begin
    if(rst)
    begin
        s0=x0;
        s1=x1;
        s2=x2;
        s3=x3;
        flag=0;
        done=0;
        t0=x0;
        t1=x1;
        t2=x2;
        t3=x3;
        c_state=0;
    end
    else
        c_state=n_state;
end
always @ (c_state)
begin
    case (c_state)
    0: n_state=c_state+1; //01
    1: n_state=c_state+1; //12
    2: n_state=c_state+1; //23
    3: n_state=c_state+1; //01
    4: n_state=c_state+1; //12
    5: n_state=c_state+1; //01
    endcase
end
always @ (posedge clk)
begin
    case(c_state)
       0: if(t0>t1)  begin t=t0;t0=t1; t1=t; end
       1: if(t1>t2)  begin t=t1;t1=t2; t2=t; end
       2: if(t2>t3)  begin t=t2;t2=t3; t3=t; end
       3: if(t0>t1)  begin t=t0;t0=t1; t1=t; end
       4: if(t1>t2)  begin t=t1;t1=t2; t2=t; end
       5: if(t0>t1)  begin t=t0;t0=t1; t1=t; end
       6: done=1;
    endcase
    s0=t0;
    s1=t1;
    s2=t2;
    s3=t3;
end

endmodule
```

```
module sort_tb( );
reg [3:0]x0,x1,x2,x3;
wire [3:0]s0,s1,s2,s3;
reg rst,clk;
wire done;
sort DUT (x0,x1,x2,x3,s0,s1,s2,s3,clk,rst,done);
initial
    begin
        clk=0;
        x0=4;
        x1=3;
        x2=2;
        x3=1;
        rst=1;
        forever
            #10 clk=~clk;
    end
initial
    begin
    #15
    rst=0;
    end
endmodule
```

设中间变量来进行两个数调换，依次从 0-1 到 2-3 进行比较判断是否交换，与冒泡排序法类似。设计六个状态表示比较进行的程度，最后一个状态为结束状态。

代码截图：

```
module divide(x,y,q,r,clk,rst,done,error);
input [3:0]x,y;
output reg [3:0]q,r;
input clk,rst;
output reg done,error;
reg [3:0]x1,y1,q1,r1,t;
reg [1:0] c_state,n_state;
always @ (posedge clk)
begin
    if(rst)
    begin
        x1=x;
        y1=y;
        done=0;
        q1=0;
        r1=0;
        c_state=0;
        if(y==0)
            error=1;
        else
            error=0;
    end
    else
    begin
        c_state=n_state;
    end
end
always @ (*)
begin
    case(c_state)
    0:begin if(r1<y1) n_state=1; else n_state=0;end
    endcase
end
```

```
    end
end
always @ (*)
begin
    case(c_state)
    0:begin if(r1<y1) n_state=1; else n_state=0;end
    endcase
end
always @ (posedge clk)
begin
    case(c_state)
    0: begin t=q1;q1=t+1; r1=x1-q1*y1;end
    1: done=1;
    endcase
    q=q1;
    r=r1;
end
endmodule
```

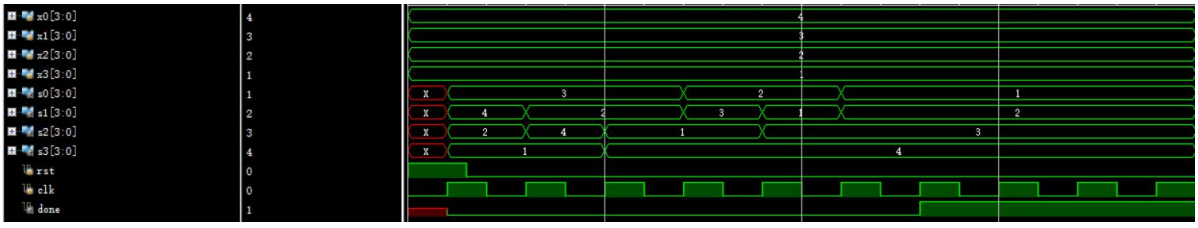
```
module divide_tb();
reg [3:0]x,y;
wire [3:0]q,r;
reg clk,rst;
wire done,error;
divide DUT (x,y,q,r,clk,rst,done,error);
initial
begin
    clk=0;
    x=9;
    y=0;
        forever #10 clk=~clk;
end
initial
begin
    rst=1;
    #15 rst=0;
end
endmodule
```

除法是通过每一个时钟沿减去被除数实现，记录商逐渐加一，判断余数是否小于除数来终止运算。设计两个状态，一种是余数大于除数还可以继续运算，另一个状态是余数小于除数，不能再进行下去，结束运算。

五、实验结果：

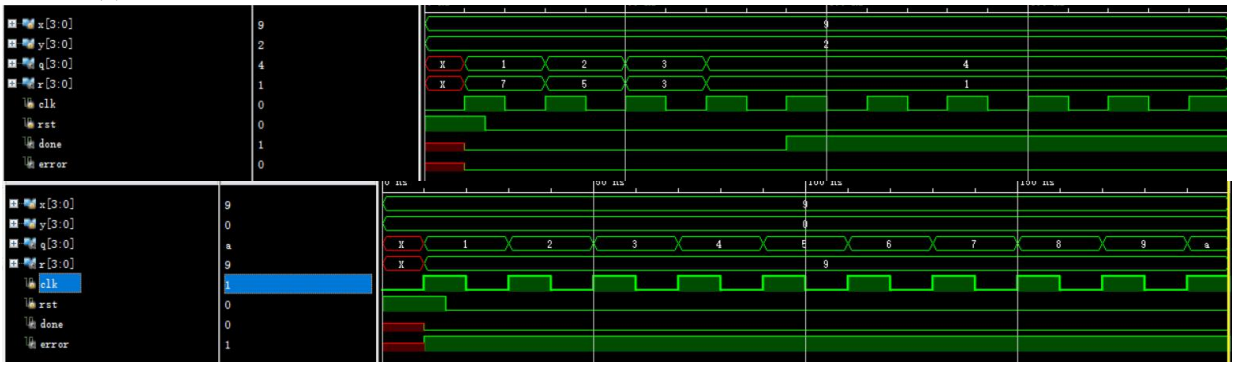
仿真截图：

排序：



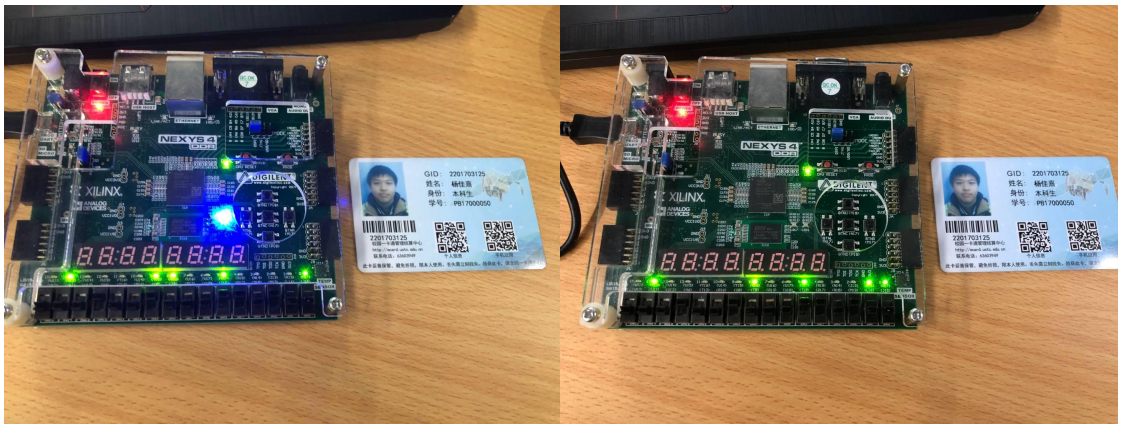
仿真结果可以看出经过三个时钟上升沿，011223 依次进行排序并且排序完成。一共六个时钟周期，所有排序结束。

除法：



仿真结果看出，y=0 时 error 等于一，否则每次时钟沿商加一，三个时钟沿后 done=1，结束运算。

排序下载结果：



蓝灯代表结束排序。

除法下载结果：

六、心得体会：

本次实验让我们复习了有限状态机的写法，并且能够利用三段式实现基本的数据通路。主要遇到的问题有函数的调用不能在 always 里进行，如果想用 always 功能的话只能在函数内通过调用时钟并在函数内写 always；其次有限状态机最好利用三段式，结构清晰并且不需要担心 mutidriven。

改进：除法实验可以利用移位来实现，也借助有限状态机的模型，只不过这次因为图省事用了减法实现。