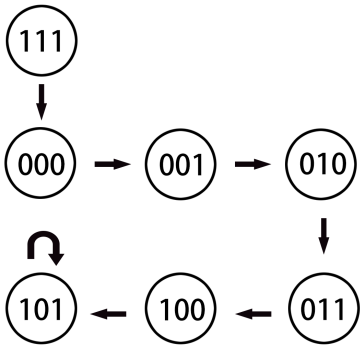
实验二 数据通路与状态机

PB17111623 范睿

逻辑设计

1. 排序

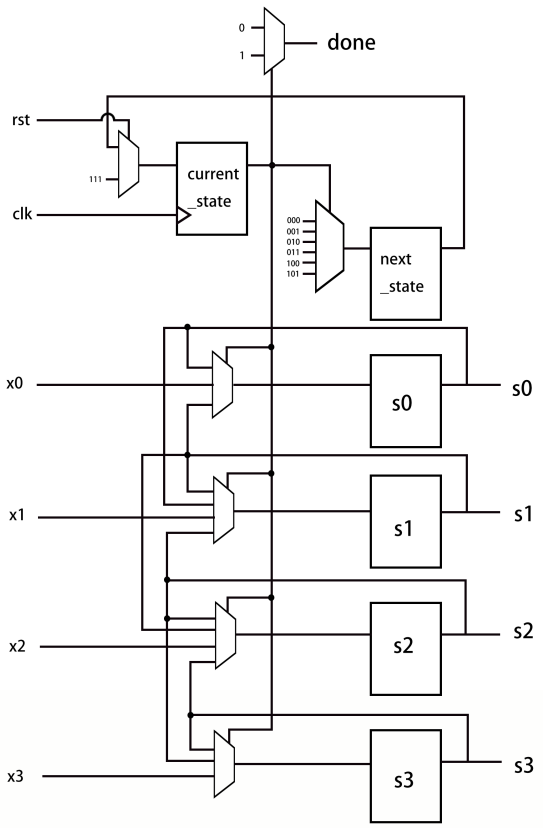
状态图:



状态解释:

共有七中状态（如图）。其中111为初始状态，每当rst为1时，current_state被置为111。当current_state为000,001,010,011,100此5种状态时，对s0~s3进行数据更新。当current_state为101时，有可能正在进行最后一次数据更新，或者进入结束阶段。

数据通路:

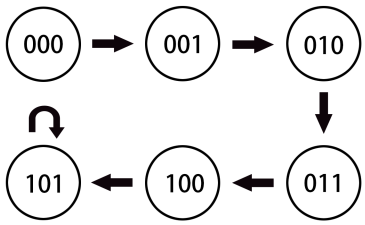


数据通路解释：

1. current_state为一个上升沿触发的寄存器。它基于next_state和rst进行状态更新
2. next_state仅与current_state有关
3. s0根据current_state，与x0，s0，s1有关
4. s1根据current_state，与x1，s0，s1，s2有关
5. s2根据current_state，与x2，s1，s2，s3有关
6. s3根据current_state，与x3，s2，s3有关
7. done根据current_state，被置为0或1

2. 除法

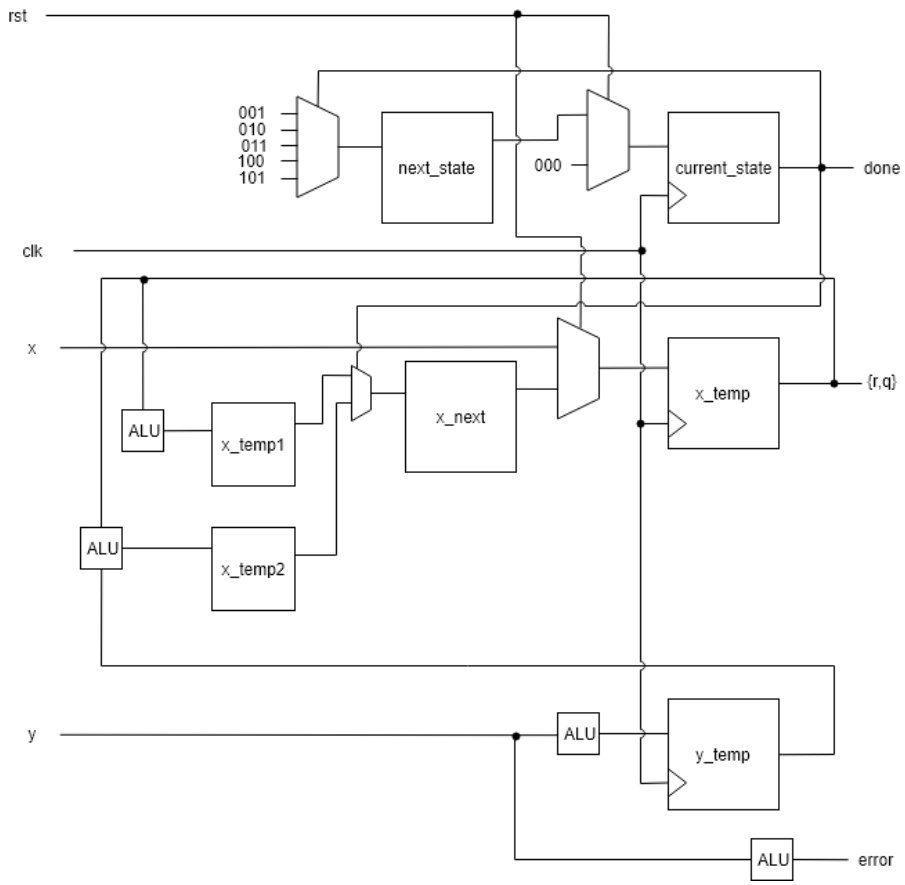
状态图:



状态图解释：

图中共有六种状态。000为初始状态，每当rst有效时将current_state置为000；当current_state为001,010,011,100,101时，对x_next和y_next进行更新。每当clk上升沿到来时，对current_state和x_temp和y_temp进行更新。

数据通路:



数据通路解释：

1. current_state是一个时钟上升沿触发的寄存器，每当时钟上升沿到来时根据next_state进行数据更新。
2. x_temp是一个时钟上升沿触发的寄存器，每当时钟上升沿到来时根据x_next进行数据更新。
3. y_temp是一个时钟上升沿触发的寄存器，每当时钟上升沿到来时根据y_next进行数据更新。
4. next_state根据current_state每时每刻更新
5. x_next是一个寄存器，它根据current_state和x_temp1和x_temp2的大小关系进行更新
6. x_temp1为x_temp左一一位得到的结果
7. x_temp2为x_temp左一一位再减y_temp+1后得到的结果
8. error当y等于0时被置为1
9. done当current_state等于101时被置为1

核心代码

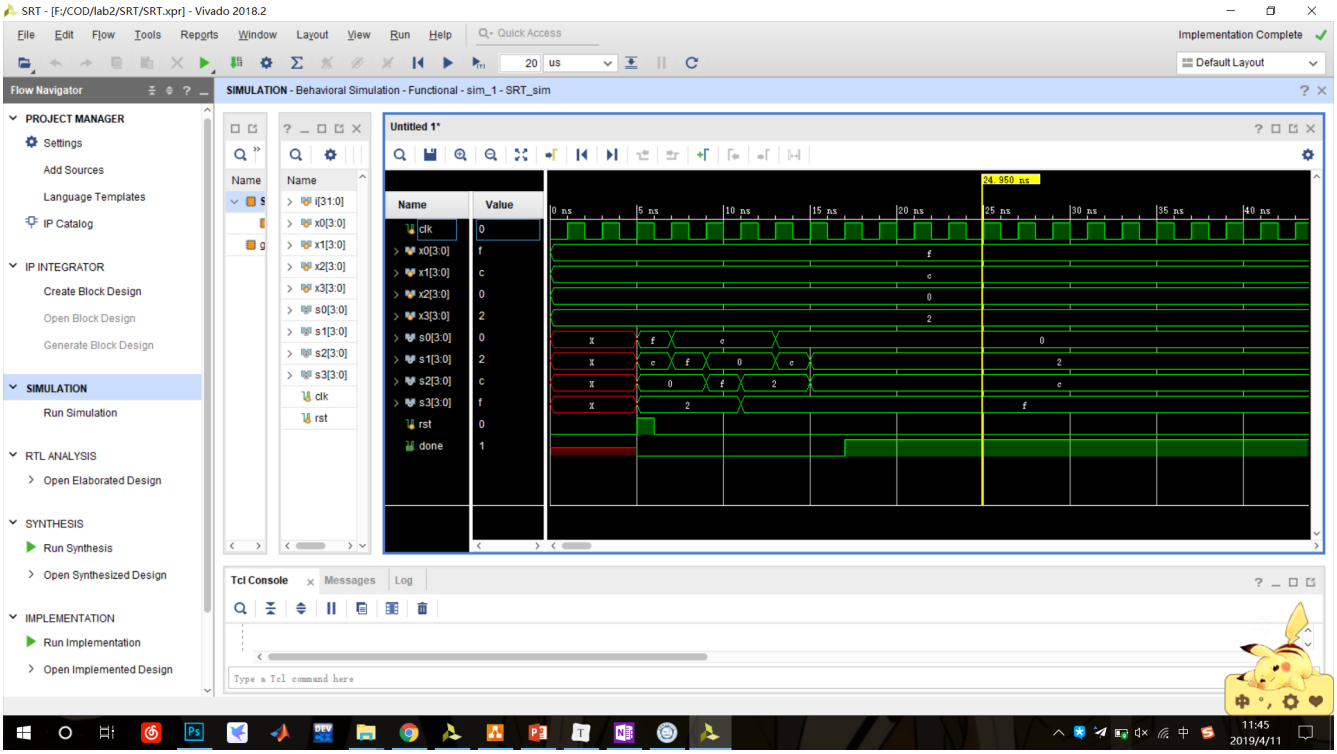
```
always @(*) begin
    case (current_state)
        //若current_state为000，表示初始状态，将x_next，y_next置为初始值
        3'b000:begin
            x_next = {4'b0000,x};
            y_next = {y,4'b0000};
        end
        //若当前状态非初始状态，根据x_temp1的高四位和y的大小关系将x_next置为相应值
        3'b001,3'b010,3'b011,3'b100:begin
            if (x_temp1[7:4] >= y)
                x_next = x_temp2;
            else
                x_next = x_temp1;
        end
    endcase
end

always@(*)begin
    //x_temp1为x_temp左移一位的结果
    x_temp1 = x_temp << 1;
    //x_temp2位x_temp左移一位再减去y_temp+1后的结果
    x_temp2 = (x_temp << 1) - y_temp + 8'b00000001;
end
```

此代码块实现x_next和y_next的更新。

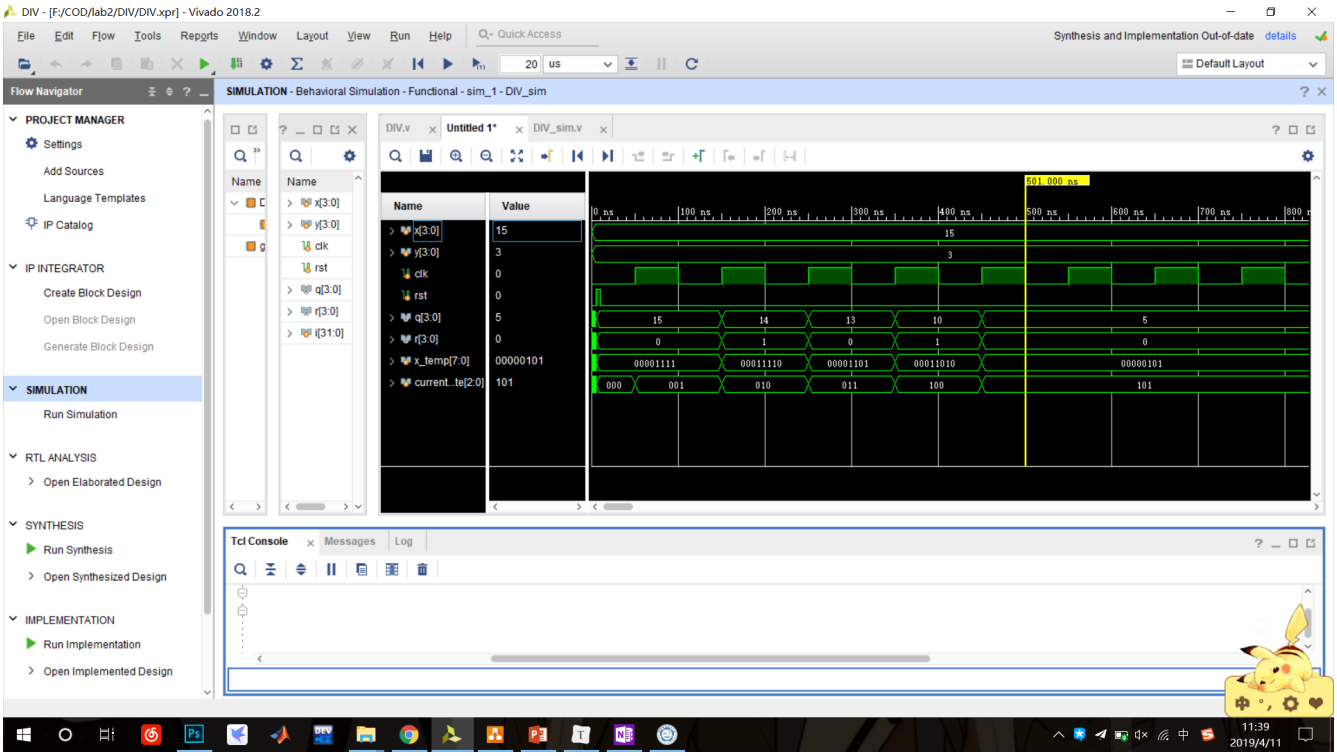
仿真及下载结果

排序仿真：



排序下载：

除法仿真：



除法下载：

结果分析

根据仿真及结果来看，排序和除法的结果均正确。

实验总结

在本次试验中，我复习了状态机的实现和学习了除法的实现。状态机的实现的关键在于确定状态的个数以及状态更新的时刻及顺序。一般来说状态机在时钟上升沿更新，其他寄存器均是状态的函数。但在本次除法的实现中，我犯了一个错误。我本来没有这事x_next及y_next，而直接将上一次的x_temp赋给下一次的x_temp。这样做的结果是Behavior Simulation没有问题，但是Post-Implementation Functional Simulation和下载中，x_temp跳跃的很厉害，结果也产生（疯狂）跳跃。经过询问同学，我发现了问题，于是将x_temp和y_temp也设置成上升沿触发赋值的寄存器，新增x_next和y_next进行数据更新，解决了问题。

设计及仿真代码

排序设计代码：

```
module RST(
    input [3:0]x0,
    input [3:0]x1,
    input [3:0]x2,
    input [3:0]x3,
    input rst,
    input clk,
    output reg [3:0]s0,
    output reg [3:0]s1,
    output reg [3:0]s2,
    output reg [3:0]s3,
    output reg done
    );
    reg [2:0]current_state;
    reg [2:0]next_state;
    reg [3:0]tmp;
    reg process_flag;
    reg [3:0]done_process;

    always@(current_state)begin
        case(current_state)
            3'b111:next_state=3'b000;
            3'b000:next_state=3'b001;
            3'b001:next_state=3'b010;
            3'b010:next_state=3'b011;
            3'b011:next_state=3'b100;
            3'b100:next_state=3'b101;
            3'b101:next_state=3'b101;
        endcase
    end

    always@(posedge clk or posedge rst)begin
        if(rst)begin
            current_state=3'b111;
        end
        else current_state=next_state;
    end

    always@(posedge clk or posedge rst)begin
        if(rst)begin
            s0 = x0;
            s1 = x1;
            s2 = x2;
            s3 = x3;
```

```
        end
        else begin
            case(current_state)
                3'b000:begin if(s0>s1) begin tmp = s0; s0 = s1; s1 = tmp; end end
                3'b001:begin if(s1>s2) begin tmp = s1; s1 = s2; s2 = tmp; end end
                3'b010:begin if(s2>s3) begin tmp = s2; s2 = s3; s3 = tmp; end end
                3'b011:begin if(s0>s1) begin tmp = s0; s0 = s1; s1 = tmp; end end
                3'b100:begin if(s1>s2) begin tmp = s1; s1 = s2; s2 = tmp; end end
                3'b101:begin if(s0>s1) begin tmp = s0; s0 = s1; s1 = tmp; end end
            endcase
        end
    end
    always@(current_state)begin
        if(current_state==3'b101) done=1;
        else done=0;
    end
endmodule
```

排序仿真代码：

```
module SRT_sim(

    );
    integer i;
    reg [3:0]x0;
    reg [3:0]x1;
    reg [3:0]x2;
    reg [3:0]x3;
    wire [3:0]s0;
    wire [3:0]s1;
    wire [3:0]s2;
    wire [3:0]s3;
    reg clk;
    reg rst;

 RST
R(.x0(x0),.x1(x1),.x2(x2),.x3(x3),.rst(rst),.clk(clk),.s0(s0),.s1(s1),.s2(s2),.s3(s3));

    initial begin
        clk = 0;
        forever #1 clk = ~clk;
    end
    initial begin
        x0 = 4'b1111;//5
        x1 = 4'b1100;//3
        x2 = 4'b0000;//c
        x3 = 4'b0010;//a
    end
    initial begin
        rst = 0;
        #5 rst = 1;
        #1 rst = 0;
        #100000 rst = 1;
        #1 rst = 0;
    end
endmodule
```

除法设计代码：

```
module DIV(
    input [3:0]x,
```

```
    input [3:0]y,
    input clk,
    input rst,
    output [3:0]q,
    output [3:0]r,
    output error,
    output done
    );
    reg [7:0] x_next, y_next;
    reg [7:0]x_temp;
    reg [7:0]y_temp;
    reg [7:0]x_temp1;
    reg [7:0]x_temp2;
    reg [2:0]current_state;
    reg [2:0]next_state;
    always@(current_state)begin
        case(current_state)
            3'b000:next_state=3'b001;
            3'b001:next_state=3'b010;
            3'b010:next_state=3'b011;
            3'b011:next_state=3'b100;
            3'b100:next_state=3'b101;
            3'b101:next_state=3'b101;
            default:next_state=3'b101;
        endcase
    end

    assign {r, q} = x_temp;

    always@(posedge clk or posedge rst)begin
        if (rst) begin
            current_state <= 3'b000;
            x_temp <= {4'b0000, x};
            y_temp <= {y, 4'b0000};
        end else begin
            current_state <= next_state;
            x_temp <= x_next;
            y_temp <= y_next;
        end
    end
    assign error = (y == 0);
    assign done = (current_state == 3'b101);

    always @(*) begin
        case (current_state)
            3'b000:begin
                x_next = {4'b0000,x};
                y_next = {y,4'b0000};
            end
            3'b001,3'b010,3'b011,3'b100:begin
                if (x_temp1[7:4] >= y)
                    x_next = x_temp2;
                else
                    x_next = x_temp1;
            end
        endcase
    end

    always@(*)begin
        x_temp1 = x_temp << 1;
```

```
        x_temp2 = (x_temp << 1) - y_temp + 8'b00000001;
    end
endmodule
```

除法仿真代码：

```
module DIV_sim(

    );
    reg [3:0]x;
    reg [3:0]y;
    reg clk;
    reg rst;
    wire [3:0]q;
    wire [3:0]r;
    integer i;
    DIV L(.x(x),.y(y),.clk(clk),.rst(rst),.q(q),.r(r));
    initial begin
        clk = 0;
        forever #50 clk = ~clk;
    end
    initial begin
        x = 4'b1111;
        y = 4'b0011;
    end
    initial begin
        rst = 0;
        #5 rst = 1;
        #5 rst = 0;
    end
endmodule
```