中国科学技术大学计算机学院

《数字电路实验报告》

实验题目：信号处理及有限状态机

学生姓名：林宸昊

学生学号：PB20000034

完成日期：2021.12.9

计算机实验教学中心制

2020年09月

【实验题目】信号处理及有限状态机

【实验目的】

- 进一步熟悉FPGA开发整体流程
- 掌握几种常见信号处理技巧
- 掌握有限状态机的设计方法
- 能够使用有限状态机设计功能电路

【实验环境】

- vlab.ustc.edu.cn
- fpgaol.ustc.edu.cn
- Logisim
- Vivado

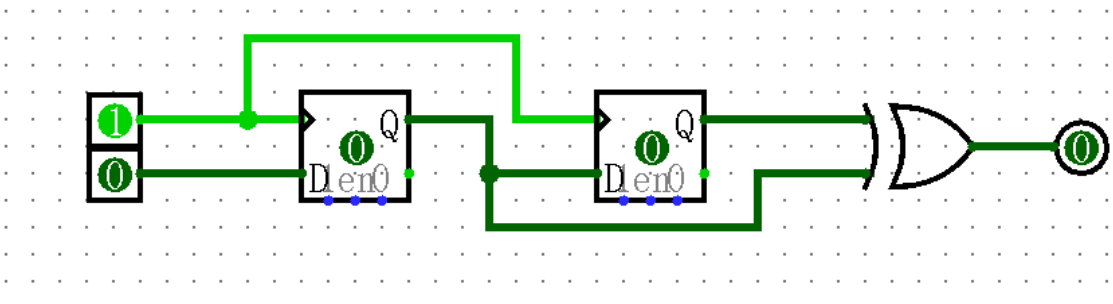
【实验练习】

题目1

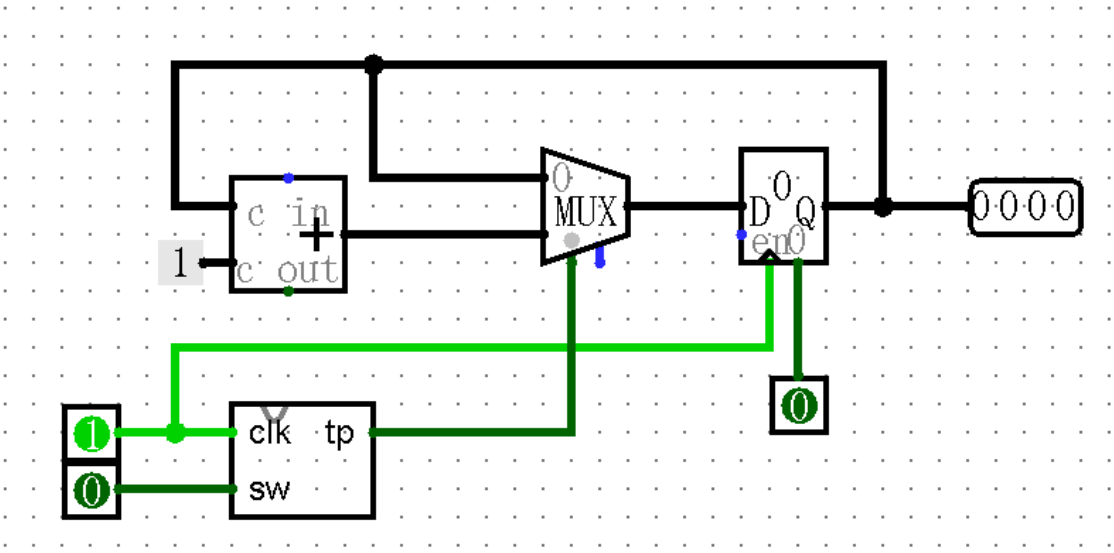
```
module test(
    input clk,rst,
    output led);
    reg [1:0] curr_state;
    reg [1:0] next_state;
    //first part of infinite machine
    always @ (*)
        begin
            case(curr_state)
                2'b00: next_state = 2'b01;
                2'b01: next_state = 2'b10;
                2'b10: next_state = 2'b11;
                default: next_state = 2'b00;
            endcase
        end
    //second part
    always @ (posdege clk or posedge rst_n)
        begin
            if(rst_n)
                cnt <= 2'b0;
            else
                curr_state <= next_state;
        end
    //third part
    assign led = (cnt == 2'b11) ? 1'b1 : 1'b0;
endmodule
```

题目2

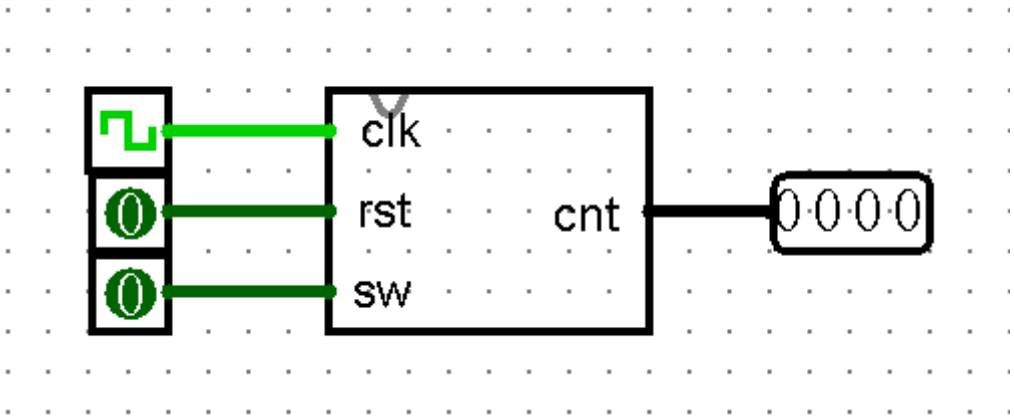
- 取sw信号边沿产生timer_pulse
    - step中所给为取上升沿，那么对于下降沿，观察可以发现利用异或门可以得到上升沿或是下降沿。



- 利用timer_pulse实现有限状态机



- 最后封装



题目3

- 设置去毛刺模块

```
module cleaned_button(
    input clk,
    input button,
    output c_button);
    reg [10:0] clean;
    always @ (posedge clk)
        begin
            if(button == 0)
                clean <= 0;
            else if(clean < 10'h3FF)
    clean <= clean + 1;
    end
    assign button_clean = clean[10];
endmodule
```

- 设置取边沿模块

```
module signal_edge(
    input clk,
    input button,
    output e_button);
    reg button1,button2;
    always @ (posedge clk)
        button1 <= button;
    always @ (posedge clk)
        button2 <= button1;
    assign e_button = button1 & (~button2);
endmodule
```

- 整体代码（不再包括上述模块）

```
module EXP3(
    input clk, rst, sw, button,
    output reg [2:0] ans,
    output reg [3:0] d);
    reg [9:0] count; //作为降频用的计数器
    reg [7:0] cnt;   //用于实际计数
    wire e_button;
    wire c_button;
    cleaned_button clean(clk, button, c_button);
    signal_edge signal(clk, c_button, e_button);
    always @ (posedge clk or posedge rst)
        begin
            if(rst)
                begin
                    cnt[7:4] <= 4'b0001;
                    cnt[3:0] <= 4'b1111;
                end
            else begin
                if(c_button)
                    begin
                        if(sw)
                            cnt <= cnt + 1;
                        else
                            cnt <= cnt - 1;
                        end
                    end
        end
    wire slowed_pulse; //降频后的信号
    always @ (posedge clk)
        count <= count + 1;
    assign slowed_pulse = (count >= 9'h1FF); //设定降频信号
    always @ (posedge clk)
        begin
            if(slowed_pulse)
                begin
                    ans <= 3'b000;
                    d <= cnt[3:0];
                end
            else
                begin
                    ans <= 3'b001;
                    d <= cnt[7:4];
```

```
            end
        end
endmodule
```

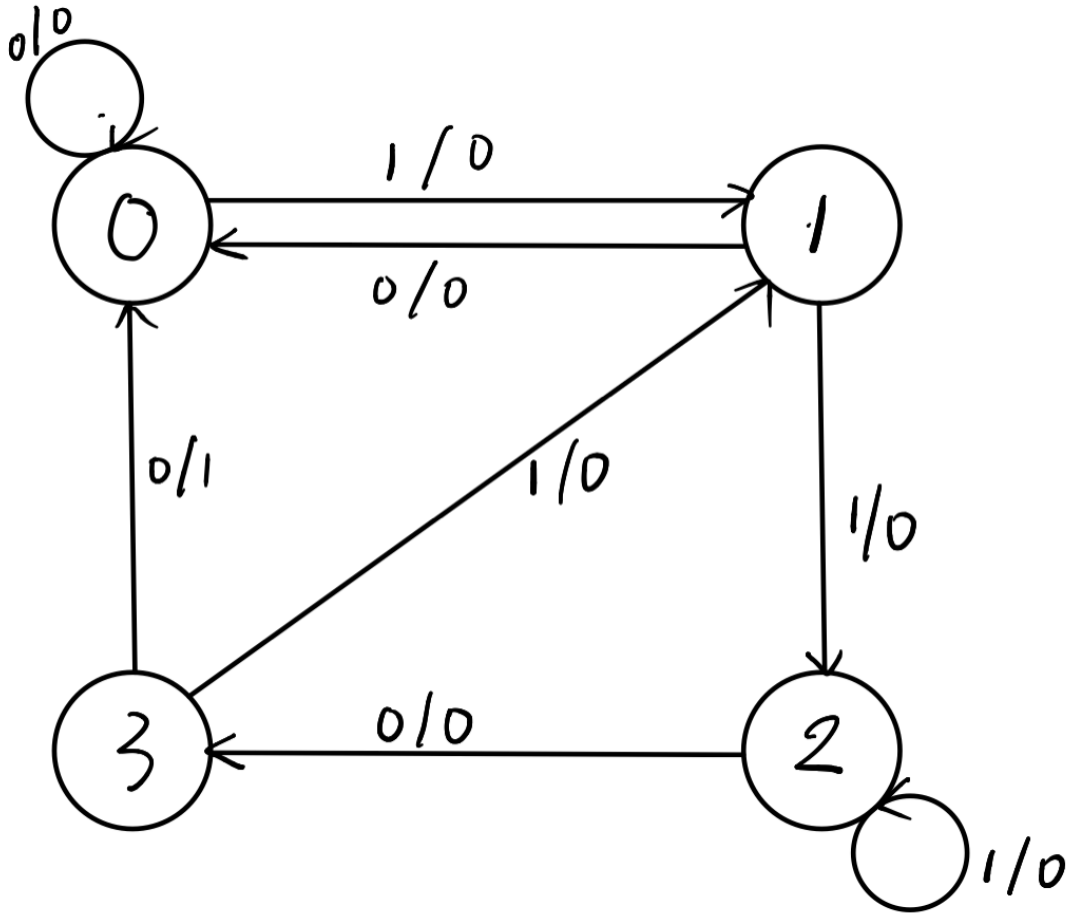
- 约束文件

```
## Clock signal

set_property -dict { PACKAGE_PIN E3    IOSTANDARD LVCMOS33 } [get_ports {
clk }];
set_property -dict { PACKAGE_PIN B18    IOSTANDARD LVCMOS33 } [get_ports {
button }];

## FPGAOL SWITCH

set_property -dict { PACKAGE_PIN D14   IOSTANDARD LVCMOS33 } [get_ports { sw
}];
set_property -dict { PACKAGE_PIN F16   IOSTANDARD LVCMOS33 } [get_ports {
rst }];

## FPGAOL HEXPLAY

set_property -dict { PACKAGE_PIN A14   IOSTANDARD LVCMOS33 } [get_ports {
d[0] }];
set_property -dict { PACKAGE_PIN A13   IOSTANDARD LVCMOS33 } [get_ports {
d[1] }];
set_property -dict { PACKAGE_PIN A16   IOSTANDARD LVCMOS33 } [get_ports {
d[2] }];
set_property -dict { PACKAGE_PIN A15   IOSTANDARD LVCMOS33 } [get_ports {
d[3] }];
set_property -dict { PACKAGE_PIN B17   IOSTANDARD LVCMOS33 } [get_ports {
ans[0] }];
set_property -dict { PACKAGE_PIN B16   IOSTANDARD LVCMOS33 } [get_ports {
ans[1] }];
set_property -dict { PACKAGE_PIN A18   IOSTANDARD LVCMOS33 } [get_ports {
ans[2] }];
```

- 运行截图

题目4

- 绘制状态图



- 设计文件（引用模块同上）

```
module exp4(
    input clk, sw, button,
    output reg [3:0] d,
    output reg [2:0] ans);
    reg [1:0] curr_state;
    reg [1:0] next_state;
    reg [23:0] data; //用于存放输出全部需要输出的数据
    reg [3:0] curr_input;
    reg [3:0] count;
    reg [10:0] times;
```

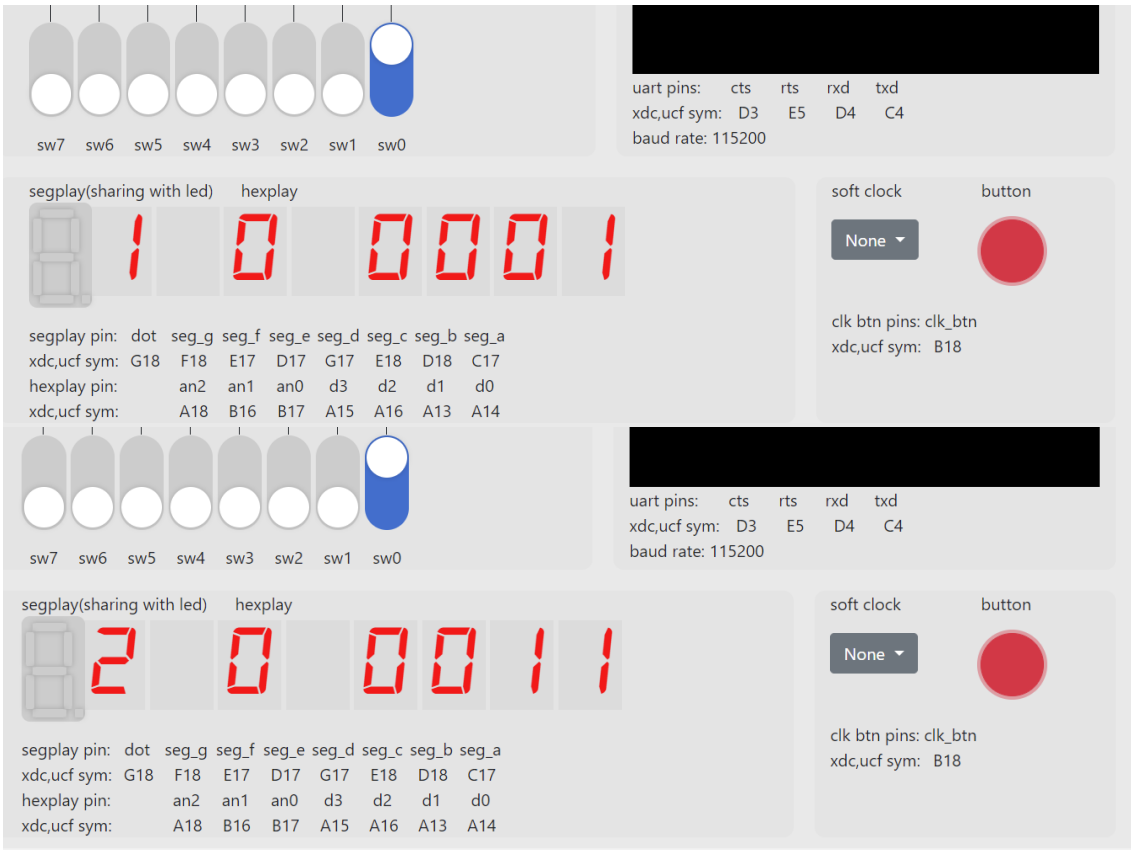
```
    wire e_button;
    wire c_button;
    cleaned_button clean(clk, button, c_button);
    signal_edge signal(clk, c_button, e_button);

    //first part
    always@(*)
        begin
            if(e_button)
                begin
                    case(curr_state)
                        2'b00:
                            begin
                                if(sw)
                                    next_state = s1;
                                else
                                    next_state = s0;
                            end
                        2'b01:
                            begin
                                if(sw)
                                    next_state = s2;
                                else
                                    next_state = s0;
                            end
                        2'b10:
                            begin
                                if(sw)
                                    next_state = s2;
                                else
                                    next_state = s3;
                            end
                        default:
                            begin
                                if(sw)
                                    next_state = s1;
                                else
                                    next_state = s0;
                            end
                    endcase
                end
            else
                next_state = curr_state;
        end

    //second part


    //part 2
    always@(posedge clk)
        begin
            if(e_button)
                begin
                    if(curr_state == 2'b11 & sw == 0)
                        count <= count + 1;
                    curr_state <= next_state;
                end
        end
```

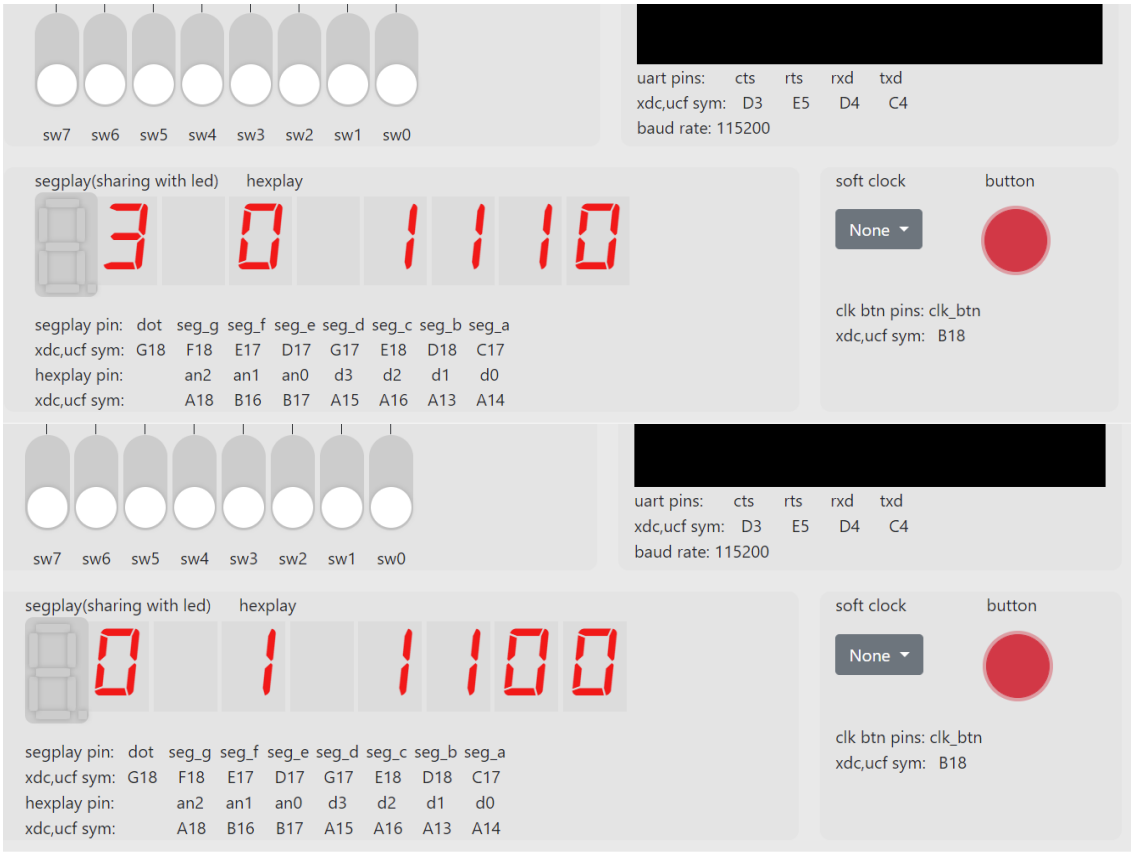
```
    //part 3
    always@(posedge clk)
        begin
            if(e_button == 1)
                begin
                    curr_input <= {curr_input[2:0],sw};
                end
            data[3:0] <= {3'b000,curr_input[0]};
            data[7:4] <= {3'b000,curr_input[1]};
            data[11:8] <= {3'b000,curr_input[2]};
            data[15:12] <= {3'b000,curr_input[3]};
            data[19:16] <= count;
            case (curr_state)
                2'b00: data[23:20] <= 2'b00;
                2'b01: data[23:20] <= 2'b01;
                2'b10: data[23:20] <= 2'b10;
                default: data[23:20] <= 2'b11;
            endcase
        end
    always@(posedge clk)
        begin
            times <= times +1;
            if(times == 11'd341)
                begin
                    d <= data[3:0];
                    ans <= 0;
                end
            else if(times == 11'd682)
                begin
                    d <= data[7:4];
                    ans <= 1;
                end
            else if(times == 11'd1023)
                begin
                    d <= data[11:8];
                    ans <= 2;
                end
            else if(times == 11'd1364)
                begin
                    d <= data[15:12];
                    ans <= 3;
                end
            else if(times == 11'd1705)
                begin
                    d <= data[19:16];
                    ans <= 5;
                end
            else if(times == 11'd2046)
                begin
                    d <= data[23:20];
                    ans <= 7;
                end
        end
endmodule
```

- 约束文件

```
## Clock signal
set_property -dict { PACKAGE_PIN E3    IOSTANDARD LVCMOS33 } [get_ports {clk}];
set_property -dict { PACKAGE_PIN B18    IOSTANDARD LVCMOS33 } [get_ports
{button}];

## FPGAOL SWITCH

set_property -dict { PACKAGE_PIN D14   IOSTANDARD LVCMOS33 } [get_ports { sw }];

## FPGAOL HEXPLAY

set_property -dict { PACKAGE_PIN A14   IOSTANDARD LVCMOS33 } [get_ports { d[0]
}];
set_property -dict { PACKAGE_PIN A13   IOSTANDARD LVCMOS33 } [get_ports { d[1]
}];
set_property -dict { PACKAGE_PIN A16   IOSTANDARD LVCMOS33 } [get_ports { d[2]
}];
set_property -dict { PACKAGE_PIN A15   IOSTANDARD LVCMOS33 } [get_ports { d[3]
}];
set_property -dict { PACKAGE_PIN B17   IOSTANDARD LVCMOS33 } [get_ports { ans[0]
}];
set_property -dict { PACKAGE_PIN B16   IOSTANDARD LVCMOS33 } [get_ports { ans[1]
}];
set_property -dict { PACKAGE_PIN A18   IOSTANDARD LVCMOS33 } [get_ports { ans[2]
}];
```

- 运行截图



- 

-

【总结与思考】

- 实验收获
  - 掌握如何去除任意时钟周期内的毛刺；
  - 掌握如何取不在边沿敏感列表中的信号的边沿，包括上升沿，下降沿，或者二者皆有。
  - 掌握有限状态机的实现，包括逻辑图与代码。
- 实验难易及任务量
  - 难度中等。只要详细阅读实验步骤，并且在上一次实验能真正掌握，基本不会感到困难，主要新增知识点还是有限状态机的实现。
  - 总体来说任务量不重，只要准确把握题目意思即可。
- 实验建议
  - 也许是毛刺现象不好体现，那么可以提出一些具体要求以考察毛刺的去除（比如多少时钟周期内的）。
- 实验吐槽

  两次实验隔太长了……第8次实验大量用到第七次实验的内容，我又隔了两个星期做第8次，结果直接基本忘光光，还研究了好久代码……还是一周一次比较好，大概。