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：李远航

学生学号：PB20000137

完成日期：2021.12.10

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】

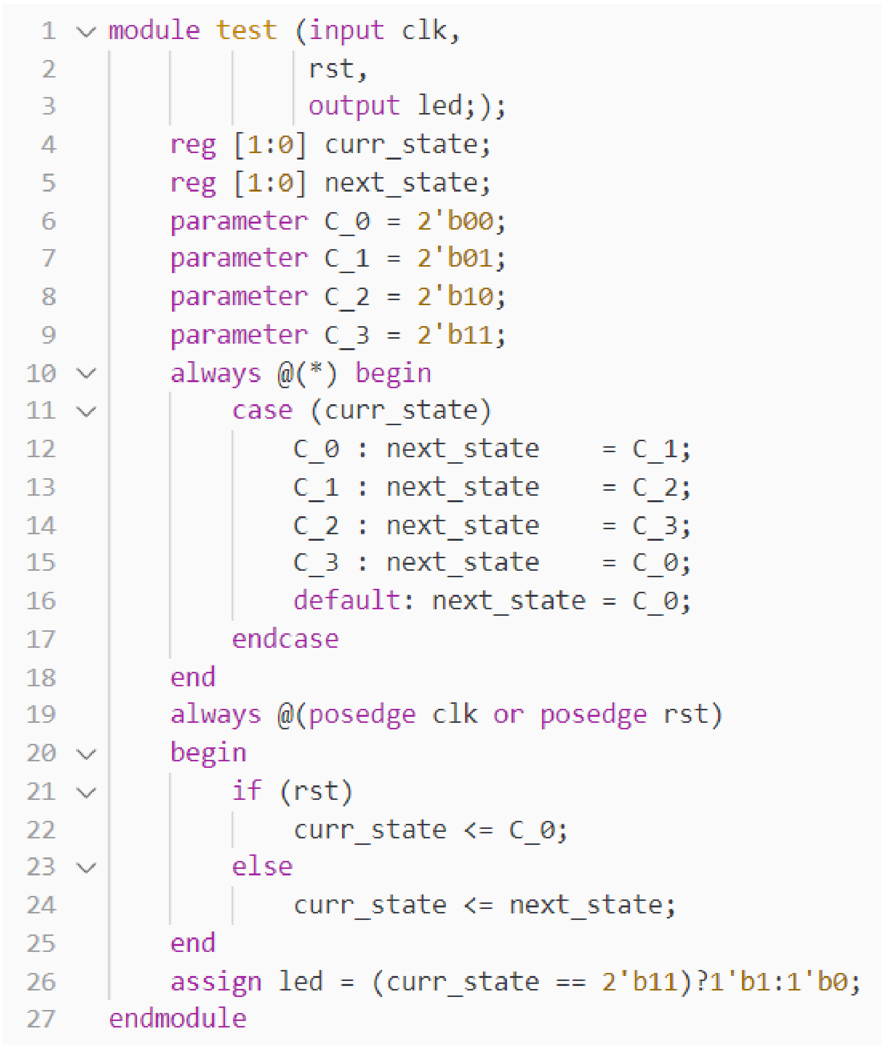
- 进一步熟悉 FPGA 开发的整体流程
- 掌握几种常见的信号处理技巧
- 掌握有限状态机的设计方法
- 能够使用有限状态机设计功能电路
- 学习时分复用的技巧，更加熟练使用 verilog 语言

【实验环境】

- PC 一台
- Windows 10 操作系统
- Vivado 2020.02
- fpgaol.ustc.edu.cn

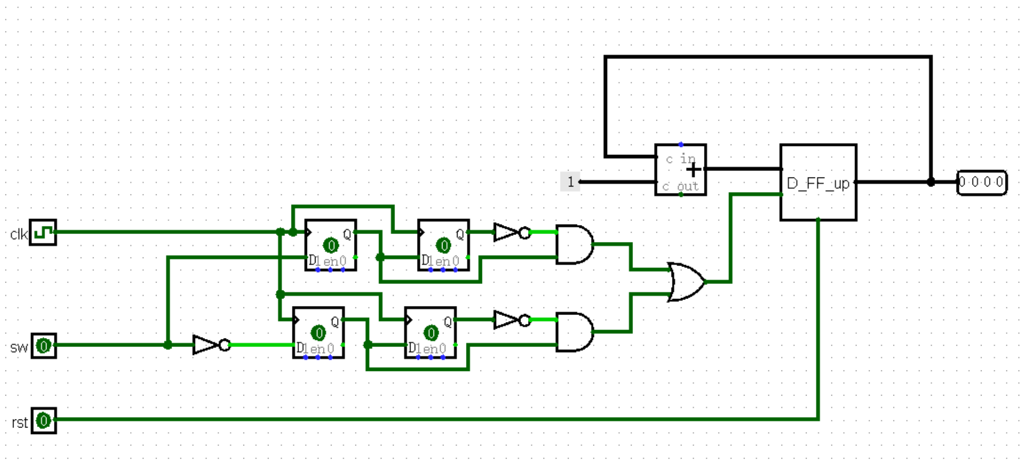
【实验练习】

1. 修改成三段式的 verilog 代码如下所示



```verilog
module test (input clk,
                   rst,
                   output led;);
        reg [1:0] curr_state;
        reg [1:0] next_state;
        parameter C_0 = 2'b00;
        parameter C_1 = 2'b01;
        parameter C_2 = 2'b10;
        parameter C_3 = 2'b11;
        always @(*) begin
            case (curr_state)
                C_0 : next_state     = C_1;
                C_1 : next_state     = C_2;
                C_2 : next_state     = C_3;
                C_3 : next_state     = C_0;
                default: next_state = C_0;
            endcase
        end
        always @(posedge clk or posedge rst)
        begin
            if (rst)
                curr_state <= C_0;
            else
                curr_state <= next_state;
        end
        assign led = (curr_state == 2'b11)?1'b1:1'b0;
endmodule
```

2. 利用实验文档中的介绍，生成一个时钟周期的脉冲，同时需要检测正负，则可以再加上一个非门，电路如下所示：



3. 使用一个计数器来实现时分复用，同时额外定义 reg 变量来对按键信号上升沿进行捕捉，同时根据 xuanze 信号来进行加减计数，代码，运行截图及管口约束文件如下

```verilog
module fun(input clk,
           input xuanze,
           input rst,
           input count,
           output reg [3:0] out,
           output reg [2:0] select);
    reg [7:0] ans;
    reg [4:0] cnt;
    reg [7:0] next;
    reg flag;
    reg fuwei;
    initial next = 8'h1f;
    initial ans  = 8'h1f;
    initial cnt  = 4'd0;
    always@(posedge clk)
    begin
        if (xuanze)
        begin
            if (ans == 8'hff)
                next <= 8'h1f;
            else
                next <= ans+8'h1;
        end
        else
        begin
            if (ans == 8'h00)
                next <= 8'h1f;
            else
                next <= ans-8'h1;
        end
    end
    always@(posedge clk)
    begin
        flag <= count;
    end
    always@(posedge rst or posedge flag)
    begin
        if (rst)
            ans <= 8'h1f;
        else
            ans <= next;
    end
    always @(posedge clk)
    begin
        if (cnt == 4'd9)
            cnt <= 4'd0;
        else
            cnt <= cnt+4'd1;
    end
    always@(posedge clk)
        if (cnt>4'd5)
        begin
            select <= 3'b001;
            out    <= ans[7:4];
        end
        else
        begin
            select <= 3'b000;
            out    <= ans[3:0];
        end
endmodule
```

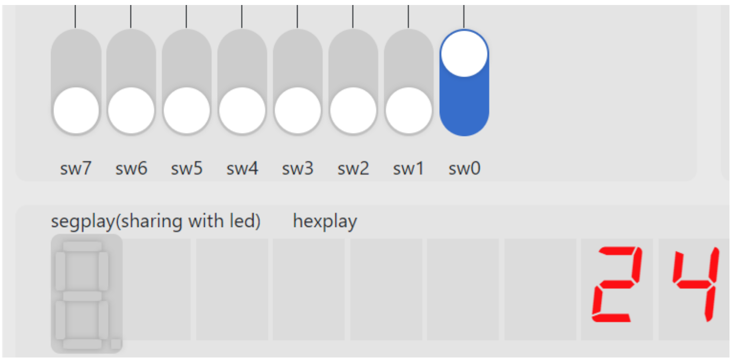


图 3.1

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

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

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

set_property -dict { PACKAGE_PIN B18   IOSTANDARD LVCMOS33 } [get_ports { count }];
```

4. 通过以下状态图编写 verilog 代码，代码管脚约束文件及运行截图如下：

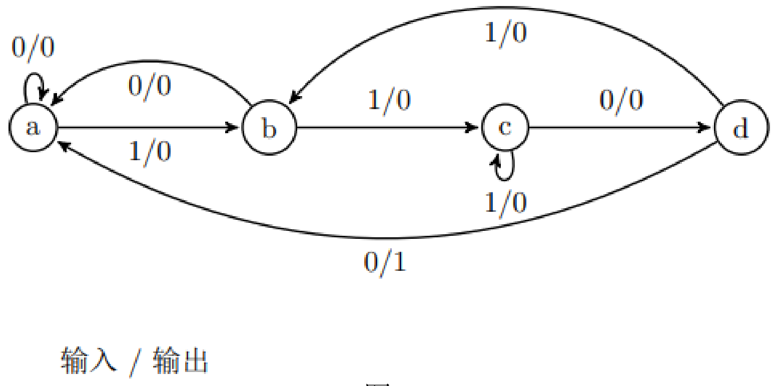


图 4.1

第一个数码管表示 1100 个数
第三个数码管表示状态：0,1,2,3 对应 a,b,c,d
最后四个数码管表示近期输入的四个数字
下图为输入序列：0011001110011 后的情况

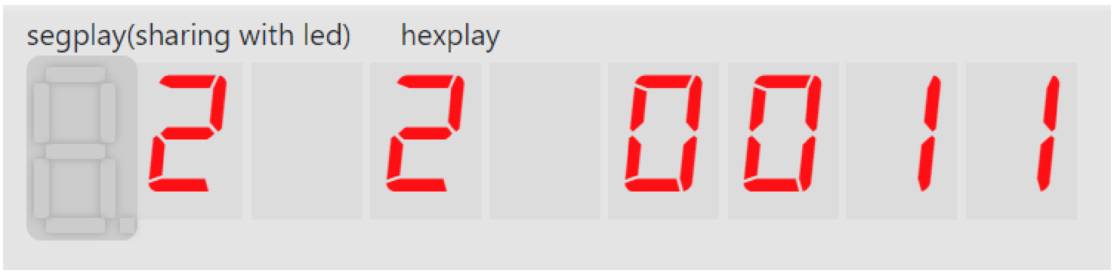


图 4.2

Verilog 代码：

```
module judge(input clk,
          input string,
          input get_num,
          output reg [2:0] sel,
          output reg [3:0] out);
```

```
reg [4:0] cnt;
reg [15:0] last_num;
initial last_num = 16'd0;
initial cnt <= 5'd0;

reg [1:0] curr_state;
reg [1:0] next_state;
parameter a        = 2'b00;
parameter b        = 2'b01;
parameter c        = 2'b10;
parameter d        = 2'b11;
initial curr_state = 2'b00;

always@(*)
begin
    case(curr_state)
        a :
        begin
            if (string == 1)
                next_state = b;
            else
                next_state = a;
        end
        b :
        begin
            if (string == 1)
                next_state = c;
            else
                next_state = a;
        end
        c :
        begin
            if (string == 1)
                next_state = c;
            else
                next_state = d;
        end
        d :
        begin
            if (string == 1)
             next_state = b;
            else
                next_state = a;
        end
    endcase
end
```

```
reg flag;
always@(posedge clk)
    flag <= get_num;

always@(posedge flag)
begin
    last_num[15:12] <= last_num[11:8];
    last_num[11:8]  <= last_num[7:4];
    last_num[7:4]   <= last_num[3:0];
    last_num[3:0]   <= {3'b0,string};
    curr_state      <= next_state;
end

reg [3:0] num_now;
initial num_now = 4'b0;
reg [3:0] state_now;
always@(*)
    state_now = {2'b00,curr_state};

always@(posedge flag)
begin
    if (last_num[11:0] == 12'h110&&string == 0)
    begin
        num_now <= num_now+4'b1;
    end
end
always@(posedge clk)
begin
    if (cnt == 5'd24)
        cnt <= 5'd0;
    else
        cnt <= cnt+5'd1;
end
always@(posedge clk)
begin
    if (cnt<5'd4)
    begin
        sel <= 3'b111;
        out <= num_now;
    end
    else if (cnt<5'd8)
    begin
        sel <= 3'b101;
        out <= state_now;
    end
```

```
        else if (cnt<5'd12)
        begin
            sel <= 3'b011;
            out <= last_num[15:12];
        end
        else if (cnt<5'd16)
        begin
            sel <= 3'b010;
            out <= last_num[11:8];
        end
        else if (cnt<5'd20)
        begin
            sel <= 3'b001;
            out <= last_num[7:4];
        end
        else if (cnt<5'd24)
        begin
            sel <= 3'b000;
            out <= last_num[3:0];
        end
    end
endmodule
```

管脚约束文件:

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

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

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

set_property -dict { PACKAGE_PIN B18   IOSTANDARD LVCMOS33 } [get_ports { get_num }];
```

【总结与思考】

- 本次实验难度较大，任务量较多
- 进一步熟悉了 FPGA 的开发流程
- 学会设计并例化状态机
- 学会处理不同的信号
- 对 Verilog 语言有了更深的认识