

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： 黄瑞轩

学生学号： PB20111686

完成日期： 2021.12.08

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

## 实验环境

- VLAB 平台: vlab.ustc.edu.cn
- FPGAOL 实验平台: fpgaol.ustc.edu.cn
- Logisim
- Vivado 工具

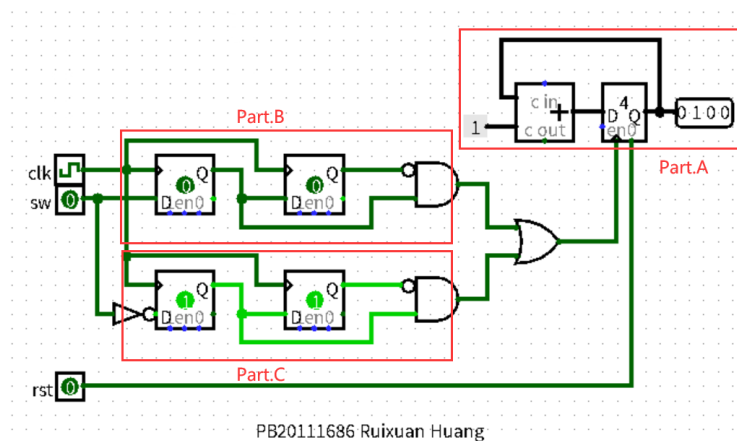
## 实验练习

【题目 1】改成三段式后的代码如下。

```
module test(input clk, rst, output led);
    parameter ZE = 2'b00;
    parameter UN = 2'b01;
    parameter DE = 2'b10;
    parameter TR = 2'b11;
    reg [1:0] CS = 2'b00; //实际电路中不需要，但是为了仿真需要给 CS 赋初值
    reg [1:0] NS;
    always@(*) begin
        case(CS)
            ZE: NS = UN;
            UN: NS = DE;
            DE: NS = TR;
            TR: NS = ZE;
        endcase
    end
    always@(posedge clk or posedge rst) begin
        if(rst)
            CS <= ZE;
        else
            CS <= NS;
        end
    assign led = (CS == TR) ? 1'b1 : 1'b0;
endmodule
```

## 【题目 2】

- 设计的电路图如下。



- 下面解释电路设计的思路：Part.A 部分是一个寄存器，Part.B 部分是捕获一个 sw 信号的上升沿，Part.C 部分是捕获一个 sw 信号的下降沿，或门来获得 sw 信号电平的变动（p 变 n+n 变 p），并以一个时钟周期上升沿的形式输出给寄存器。

### 【题目 3】

- 十六进制计数器在前面的实验做过类似的，这里不再详细解释十六进制计数器的详细设计，只着重谈题目要求的部分是如何实现的。下面是设计文件，解释部分以注释的形式给出。

```
module test(
    input CLK100MHZ,
    input dir,
    input rst,
    input BTN,
    output reg [2:0] hexplay_an,
    output reg [3:0] hexplay_data
);
reg [7:0] data;
reg [32:0] hexplay_cnt;
always@(posedge CLK100MHZ) begin
    if (hexplay_cnt >= (2000000 / 8)) hexplay_cnt <= 0;
    else hexplay_cnt <= hexplay_cnt + 1;
end
always@(posedge CLK100MHZ) begin
    if (hexplay_cnt == 0) begin
        if (hexplay_an == 1) hexplay_an <= 0;
        else hexplay_an <= hexplay_an + 1;
    end
end
always@(*) begin
    case(hexplay_an)
        0: hexplay_data = data[3:0];
        1: hexplay_data = data[7:4];
    endcase
end
reg [26:0] timer_cnt;
always@(posedge CLK100MHZ) begin
    if (timer_cnt >= 100000000) timer_cnt <= 0;
    else timer_cnt <= timer_cnt + 1;
end
//给 BTN 信号去毛刺
reg [3:0] cnt;
always@(posedge CLK100MHZ) begin
    if (BTN == 1'b0) cnt <= 4'h0;
    else if (cnt < 4'h8) cnt <= cnt + 1;
end
wire BTN_c;
assign BTN_c = cnt[3];
```

```

//获得 BTN_c 信号的边沿
reg BTN_R1, BTR_R2;
wire BTN_EDGE;
always @(posedge CLK100MHZ) BTN_R1 <= BTN_c;
always @(posedge CLK100MHZ) BTR_R2 <= BTN_R1;
assign BTN_EDGE = BTN_R1 & (~BTR_R2);
//状态转移
always@(posedge CLK100MHZ) begin
    if(rst) data <= 8'h1F;
    else if (BTN_EDGE) begin
        if (dir) begin
            data <= data - 1;
        end
        else begin
            data <= data + 1;
        end
    end
end
endmodule

```

- 约束文件 test.xdc 如下:

```

set_property -dict { PACKAGE_PIN E3      IOSTANDARD LVCMOS33 } [get_ports { CLK100MHZ }];
create_clock -add -name sys_clk_pin -period 10.00 -waveform {0 5} [get_ports { CLK100MHZ }];

set_property -dict { PACKAGE_PIN D14     IOSTANDARD LVCMOS33 } [get_ports { dir }];
set_property -dict { PACKAGE_PIN F16     IOSTANDARD LVCMOS33 } [get_ports { rst }];
set_property -dict { PACKAGE_PIN B18     IOSTANDARD LVCMOS33 } [get_ports { BTN }];

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

```

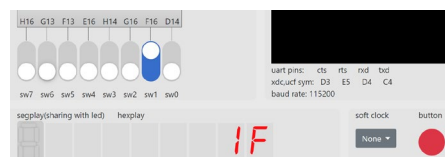
- 按实验指导手册上的实验过程生成.bit 文件，上传到 fpgaol 平台烧写，结果如下所示。



累加模式点击按钮 0x25 下



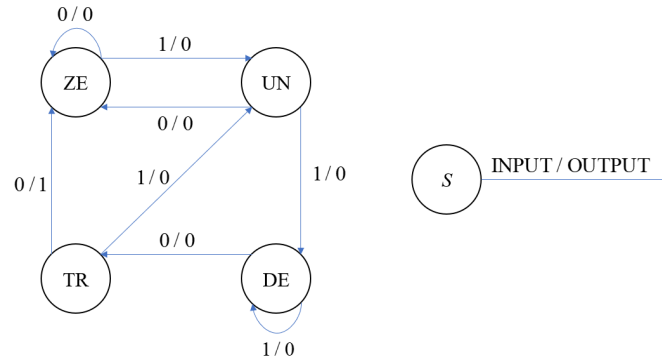
换到递减模式再点击按钮 0x8 下



复位

#### 【题目 4】

- 先画出状态图，如下图所示。



- 根据状态图设计电路，设计文件如下，思路解释部分以注释的形式给出。

```

module test(
    input CLK100MHZ,
    input dir,
    input BTN,
    output reg [2:0] hexplay_an,
    output reg [3:0] hexplay_data
);

    parameter ZE = 4'h0;
    parameter UN = 4'h1;
    parameter DE = 4'h2;
    parameter TR = 4'h3;
    reg [4:0] CS = 4'h0;
    reg [4:0] NS;
    reg in = 0;
    reg [3:0] NUM = 4'h0;
    reg [15:0] HISTORY = 16'h0000;
    //下面是给 BTN 信号去毛刺
    reg [3:0] cnt;
    always@(posedge CLK100MHZ) begin
        if(BTN == 1'b0) cnt <= 4'h0;
        else if(cnt < 4'h8) cnt <= cnt+1;
    end
    wire BTN_c;
    assign BTN_c = cnt[3];
    //下面是获得 BTN 按下的边沿
    reg BTN_R1,BTN_R2;
    wire BTN_EDGE;
    always@(posedge CLK100MHZ) begin
        BTN_R1 <= BTN_c;
        BTN_R2 <= BTN_R1;
    end
    assign BTN_EDGE = BTN_R1 & (~BTN_R2);
    //下面是状态转移部分
    always@(posedge CLK100MHZ) begin
        if(BTN_EDGE) begin
            HISTORY <= {HISTORY[11:0], 3'b0, dir};
            if(dir == 0) begin
                case(CS)
                    ZE: NS <= ZE;
                    UN: NS <= ZE;
                    DE: NS <= TR;
                end
            end
        end
    end

```

```

        TR: begin
            NS <= ZE;
            NUM = NUM + 1;
        end
    endcase
end
else begin
    case(CS)
        ZE: NS <= UN;
        UN: NS <= DE;
        DE: NS <= DE;
        TR: NS <= UN;
    endcase
end
end
else NS <= NS;
end
//下面是状态赋值部分
always@(posedge CLK100MHZ) begin
    CS <= NS;
end

reg [4:0] hexplay_cnt;

always@(posedge CLK100MHZ) begin
    if (hexplay_cnt == 5'b10111)
        hexplay_cnt <= 0;
    else
        hexplay_cnt <= hexplay_cnt + 1;
end

always@(posedge CLK100MHZ) begin
    hexplay_an = hexplay_cnt[4:2];
    case(hexplay_an)
        3'b000: hexplay_data = CS;
        3'b001: hexplay_data = NUM;
        3'b010: hexplay_data = HISTORY[3:0];
        3'b011: hexplay_data = HISTORY[7:4];
        3'b100: hexplay_data = HISTORY[11:8];
        3'b101: hexplay_data = HISTORY[15:12];
        default: hexplay_data = ZE;
    endcase
end
endmodule

```

● 约束文件 test.xdc 如下:

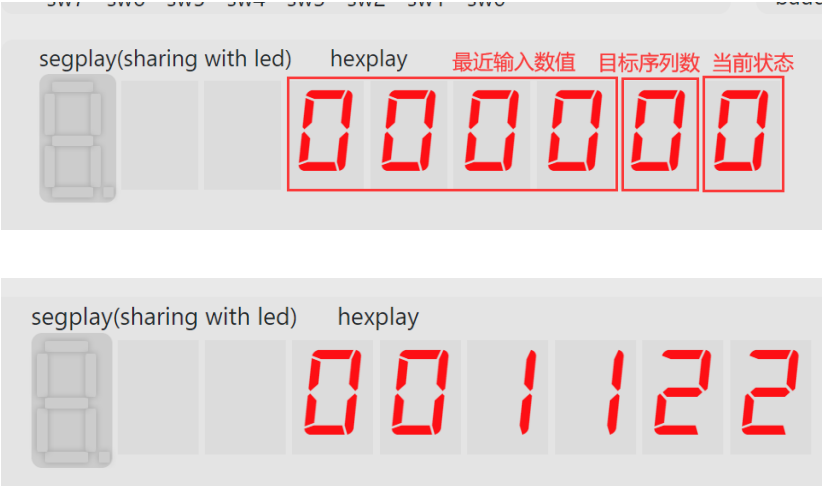
```

set_property -dict { PACKAGE_PIN E3      IOSTANDARD LVCMOS33 } [get_ports { CLK100MHZ }];
create_clock -add -name sys_clk_pin -period 10.00 -waveform {0 5} [get_ports { CLK100MHZ }];
set_property -dict { PACKAGE_PIN D14     IOSTANDARD LVCMOS33 } [get_ports { dir }];
set_property -dict { PACKAGE_PIN B18     IOSTANDARD LVCMOS33   } [get_ports { BTN }];

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

```

- 按实验指导手册上的实验过程生成.bit 文件，上传到 fpgaol 平台烧写，结果如下所示。



输入 “0011001110011” 后



## 总结与思考

- 本次实验中我学会了 `button` 的使用和相关的处理方法，除了 `sw[7:0]` 之外又学会了一种操控 FPGA 的方法，收获很大。
- 本次实验是在前面的实验基础上的综合练习，总体来说难度中等。
- 本次实验有 4 个小题，但是前两道小题比较简单，因此总体来说任务量不大；Vivado 的 Generate Bitstream 每次需要消耗的时间较长，会导致多出很多调试时间。
- 改进建议：  
给 fpgaol 平台的开关抖动情况创建一个直接可见波形的窗口，方便这门课程的同学进行直观理解和调试。