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：张郑飞扬

学生学号：PB21071416

完成日期：2022.12.1

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： vlab.ustc.eud.cn

FPGAOL:fpgaol.ustc.edu.cn

Logisim

Vivado

【实验过程】

Step1.信号整形及去毛刺

阅读实验手册，了解了信号整形的基本原理是通过一个计数器对高电平持续时间进行计时，当按键输入信号为 0 时，计数器清零，当输入信号为高电平时，计数器进行累加计数，计数达到阈值后则停止计数。通过调整阈值可以改变电路精度。

Step2.取信号边沿技巧

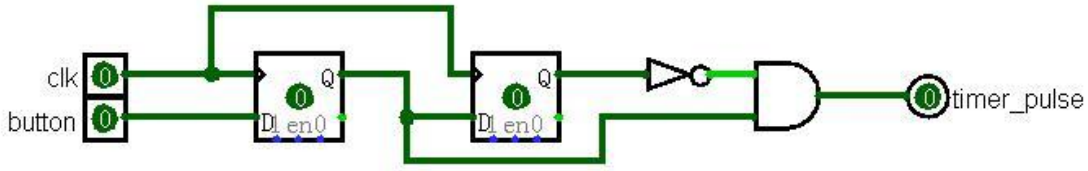
我们知道，除了时钟和异步复位信号外，其它信号都不应放在边沿敏感列表内。那么怎么用其他信号边沿来触发状态跳转呢？实验手册给出了下面的方法：

```
always@(posedge clk)
  if(控制信号==1)
  //状态跳转
  else
  //状态保持
```

上述代码能够实现电路状态转换的功能，但是要求该信号为持续时间为一个时钟周期的脉冲信号， 否则电路状态会一直跳转。而人的反应时间相对于频率快的时钟来说是很慢的，因此要求该信号为持续时间为一个时钟周期的脉冲信号并不现实。

下面学习通过该按键信号生成一个时钟周期宽度的脉冲信号：

在 Logism 中画出电路图如下



其 Verilog 代码如下：

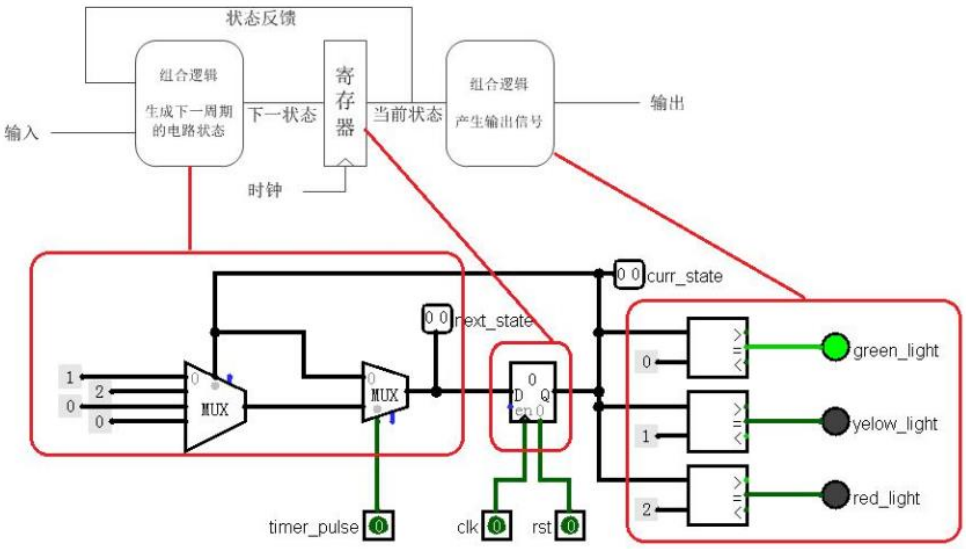
```
module signal_edge(
input clk,
input button,
output button_edge);
reg button_r1,button_r2;
always@(posedge clk)button_r1 <= button;
always@(posedge clk)
button_r2 <= button_r1;
assign button_edge = button_r1 & (~button_r2);
endmodule
```

该信号在 button 信号的上升沿附近为高电平，其余时间均为低电平。解决了上面提出的问题。

Step3、4.有限状态机介绍及实现

对 n 位位宽的寄存器，状态不会超过 2^n 个，即其状态数量是有限的，因此这种电路结构称为有限状态机。有限状态机分为摩尔型和米利型。摩尔型的输出信号仅与当前状态有关，而米利型不仅与当前状态有关，还与输出信号有关。

关于 FSM 的知识已经在数字电路课程上学习过，此处不多作赘述。实验手册上给出了一个控制交通信号灯的有限状态机。给出了 Verilog 代码，并在 Logism 中对其进行了实现。如下图：



FSM 的 Verilog 代码有三种写法：一段式、两段式、三段式。三段式的可读性最好，实验手册也推荐使用三段式写法。

Step5. 有限状态机的另一形式

实验手册在该部分给出了一个电路，它在前面实验就出现过，虽然不是按三段式来写的，但确实是一个有限状态机。有组合部分也有时序部分。

【实验练习】

题目 1.

Verilog 代码如下：

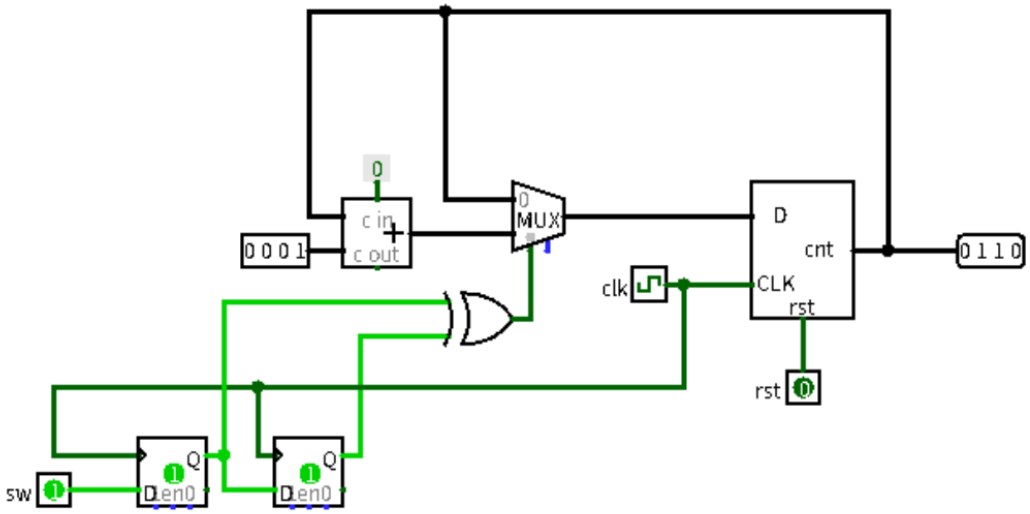
```
module test(
input clk,rst,
output led);
reg [1:0] current_state
reg [1:0] next_state
//第一部分
always@(*)
begin
  next_state = current_state + 2'b01;
end
//第二部分
always@(posedge clk or posedge rst)
begin
  if(rst)
    current_state <= 2'b00;
  else
    current_state <= next_state;
end
//第三部分
assign led = (current_state == 2'b11)? 1'b1 : 1'b0;
endmodule
```

题目 2.

本题思路为，使用 Step2 中的取边沿技巧，在 sw 的上升沿和下降沿附近都输出一个一时钟周期的高电平信号。

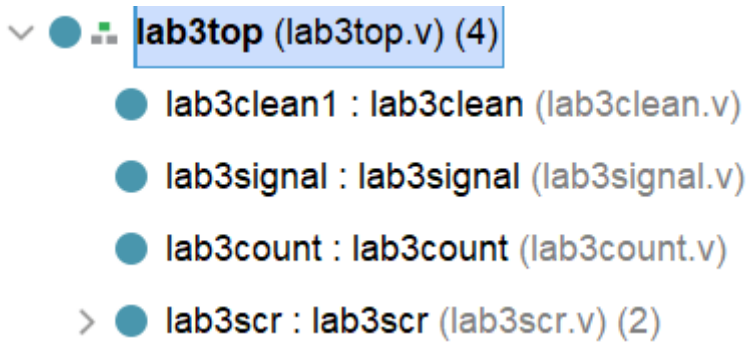
为此，我们需要对 step2 中的电路进行一些改造，即使用一个异或门代替 step2 电路中的与门，从而实现在下降沿附近也能输出高电平信号的功能。

具体电路图如下图所示：



题目 3.

将该 FSM 分为如下几个模块：



其中，clean 模块是去毛刺模块，原理同 step1，代码如下：

```
23 module lab3clean(
24 input clk,
25 input button,
26 output button_clean
27     );
28 reg [19:0] cnt;
29 always@(posedge clk)
30 begin
31    if(button == 1'b0)
32       cnt <= 20'b0;
33    else if(cnt < 20'h80000)
34       cnt <= cnt+1;
35 end
36 assign button_clean = cnt[19];
37 endmodule
```

signal 是产生边缘信号模块，原理同 step2，代码如下：

```
23  module lab3signal(
24  input clk,
25  input button,
26  output button_edge
27      );
28  reg r1,r2;
29  always@(posedge clk)
30     r1 <= button;
31  always@(posedge clk)
32     r2 <= r1;
33  assign button_edge = r1 & (~r2);
34  endmodule
```

count 模块是计数器模块，其实现比较简单，代码如下：

```
23  module lab3count(
24  input clk,
25  input signal,
26  input model,
27  input rst,
28  output reg [7:0]out);
29  always@(posedge clk)
30  begin
31     if(rst)
32        out <= 8'h1f;
33     else if(signal)
34     begin
35           if(model)
36              out <= out + 1;
37           else
38              out <= out - 1;
39     end
40  end
41  endmodule
```

此外，为了实现在数码管上的呈现，需要和实验七中一样采用时分复用的方法，其中分频等操作的原理都和实验七中的原理一样。不再赘述。另外，信号的产生也需要使用上面 signal 模块里的取信号边沿功能。综上，我们可以设计出 scr 模块，其代码如下：

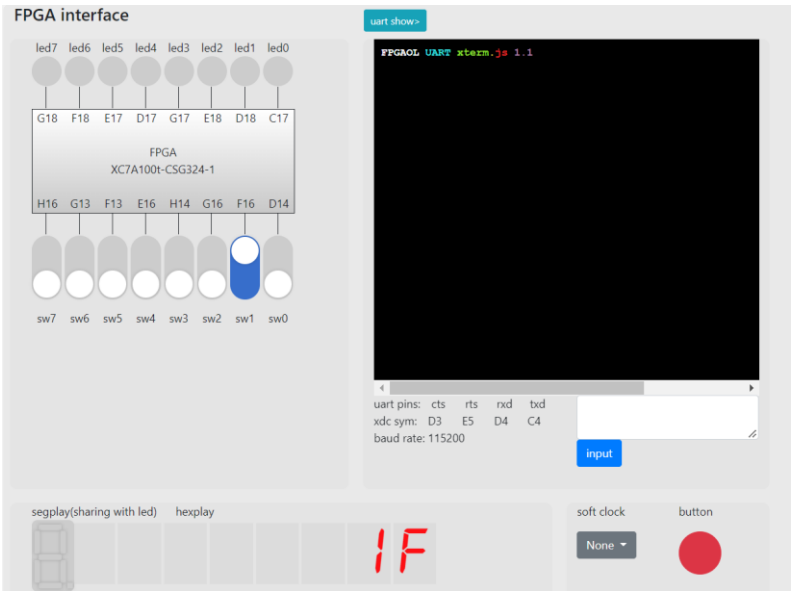
```
module lab3scr(
input clk,
input [7:0]in,
output reg [2:0]sel,
output reg [3:0]data
    );
wire [18:0]k;
wire clkd;
wire signal2;
assign k = 19'b111_1010_0001_0010_0000;//5*10^5 clkd周期为5ms
fenpin fenpin(//产生片选信号
.clk(clk),
.mag(k),
.clkout(clkd));
lab3signal lab3signal2(//产生信号
.clk(clk),
.button(clkd),
.button_edge(signal2));
reg choose;
initial choose = 1'b0;
always@(posedge clk)
begin
   if(signal2)
     case(choose)
     1'b0:choose <= 1'b1;
     1'b1:choose <= 1'b0;
     default:choose <= 1'b0;
     endcase
end
```

```
always@(*)
begin
   case(choose)
   1'b0:
     begin
       sel = 3'b000;
       data = in[3:0];
     end
   1'b1:
     begin
       sel = 3'b001;
       data = in[7:4];
     end
  default:
     begin
       sel = 3'b010;
       data = 4'b0000;
     end
   endcase
 end
endmodule
```

有了以上这些模块，我们就可以实现该计数器了，其顶层代码如下：

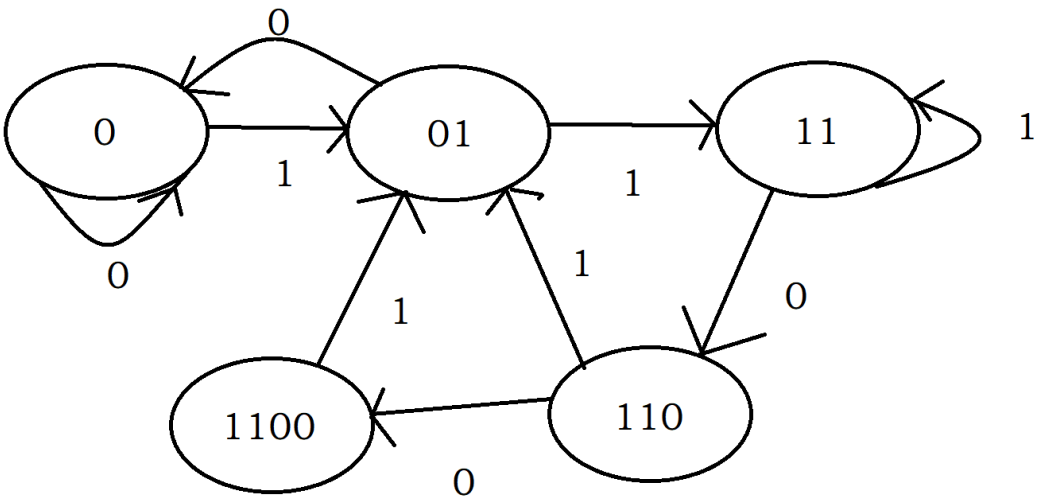
```
module lab3top(
input clk,
input model,
input rst,
input button,
output [2:0] sel,
output [3:0] data
    );
wire button_clean;
wire signal;
wire [7:0] temp;
lab3clean lab3clean1(//毛刺
.clk(clk),
.button(button),
.button_clean(button_clean));
lab3signal lab3signal(//产生信号
.clk(clk),
.button(button_clean),
.button_edge(signal));

lab3count lab3count(
.clk(clk),
.signal(signal),
.model(model),
.rst(rst),
.out(temp));

lab3scr lab3scr(
.in(temp),
.clk(clk),
.sel(sel),
.data(data));


endmodule
```

烧写到 FPGA 上，成功实现了所需功能：

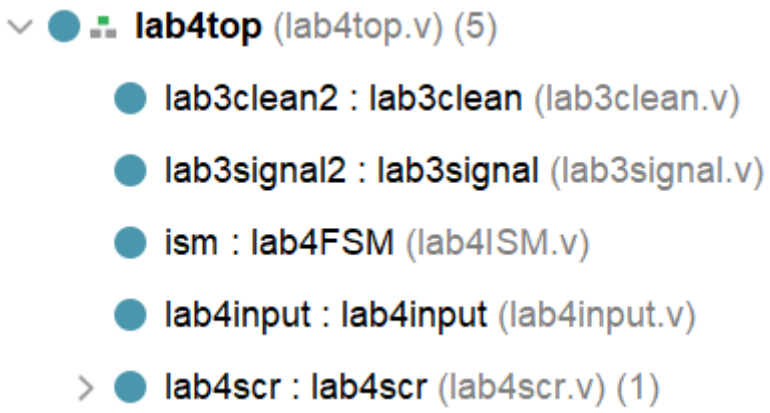


题目 4.

对于该 FSM，其状态图如下：



接下来，分模块设计该 FSM，分析得知，类似题目 3，我们需要去毛刺模块、取信号边沿模块、数码管的时分复用模块，此外，我们还需要设计一个输入模块，在该模块中，对数据进行移位处理，根据 button 是否被按下来决定数据是否移位，并产生新的 4 位数据。于是，模块的分配如下图所示：



对去毛刺模块（clean）和取信号边沿（signal）模块不再赘述，和题目 3 大体相同。对数码管的时分复用模块（scr），我们规定低 4

位为计数器，中 16 位为前四次 sw，高 4 位为当前状态，用六个数码管显示全部所需数据。以上三模块的代码都和题目三差不多，不再展示。接下来展示 input 模块，该模块也比较简单：

```
module lab4input(
input clk,
input sw,
input button,
output reg [15:0] out
    );
always@(posedge clk)
begin
   if(button)
   begin
      out[15:4] <= out[11:0];
      out[3:0] <= sw;
   end
   else out[15:0] <= out[15:0];
end
endmodule
```

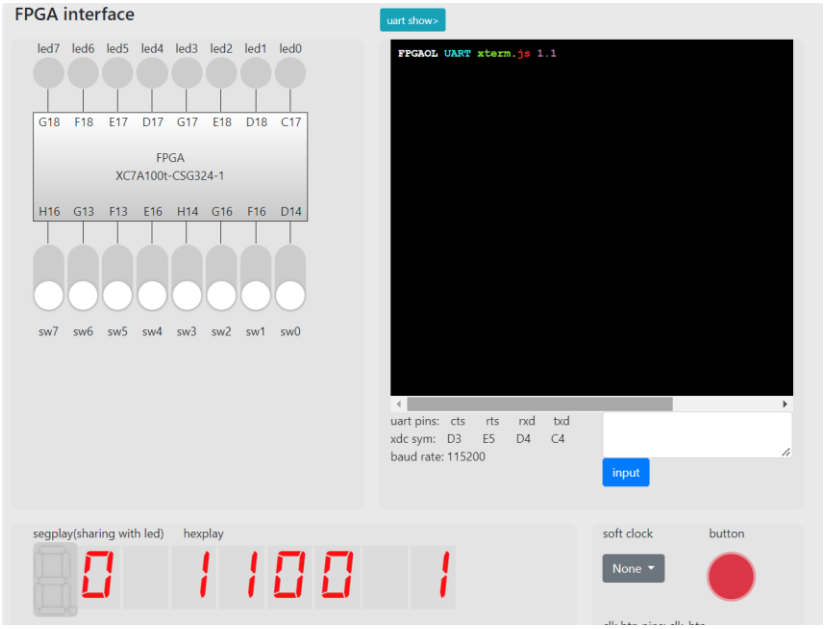
有了这些模块，我们就可以构建状态转换模块（FSM）：

```
module lab4FSM(
input sw,//输入
input button,//按键
input clk,
output reg [3:0] out,//计数器
output reg [3:0] state
    );
reg [2:0]cs;
reg [2:0]ns;//current state & next state
parameter state0=3'b000;
parameter state1=3'b001;
parameter state2=3'b010;
parameter state3=3'b011;
parameter state4=3'b100;
always@(posedge clk)
begin
    if(cs == state4)
    begin
      out <=out+1;
    end
    cs <= ns;
    state <= ns;
end
always@(*)
begin
    case(cs)
    state0:
    begin
      if(button)
         begin
            if(sw) ns = state1;
            else ns = state0;
         end
      else ns = cs;
    end
    state1:
    begin
      if(button)
         begin
            if(sw) ns = state2;
            else ns =state0;
         end
      else ns = cs;
    end
    state2:
    begin
       if(button)
       begin
            if(~sw) ns = state3;
            else ns =state2;
       end
       else ns = cs;
    end
    state3:
    begin
       if(button)
       begin
            if(~sw)//1100
              ns = state4;
            else ns = state1;//1101
       end
       else ns =cs;
    end
    state4:
    begin
       ns = state0;
    end
    default:ns = state0;
  endcase
end
endmodule
```

最终，顶层代码如下图所示：

```
module lab4top(
input sw,
input clk,
input button,
output [2:0] sel,
output [3:0] out);
wire [23:0]temp;
wire button_clean;
wire signal;
lab3clean lab3clean2(
.clk (clk),
.button (button),
.button_clean (button_clean)
    );
lab3signal lab3signal2(
.clk (clk),
.button (button_clean),
.button_edge (signal)
    );
lab4FSM ism(
.sw (sw),
.button (signal),
.clk (clk),
.out (temp[3:0]),//计数部分
.state(temp[23:20])
    );
lab4input lab4input(
.clk (clk),
.sw (sw),
.button (signal),
.out (temp[19:4])
    );
lab4scr lab4scr(
.clk (clk),
.sw (temp),
.sel (sel),
.out (out)
    );
endmodule
```

烧写到 FPGA 上，成功实现了所需功能：



【总结与思考】

本次实验主要系统学习了有限状态机的 Verilog 编码实现，并且学习了一些处理信号的技巧，比如去毛刺和取信号边沿，这些技巧对设计电路起到了很大的帮助。