数字电路与数字系统

实验报告

实　　验: 状态机及键盘输入

姓　名: 周心同

学　　号: 201220069

目录

1.实验目的

学习状态机的工作原理

了解状态机的编码方式

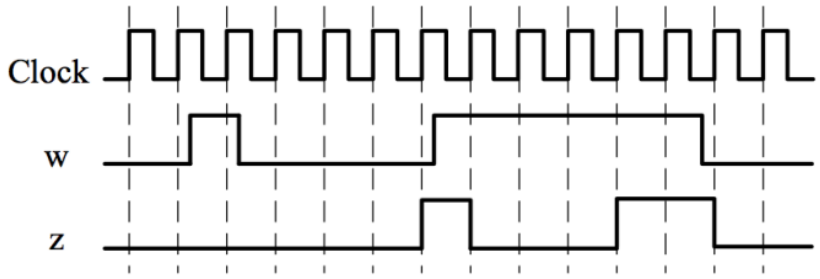
利用 PS/2 键盘输入实现简单状态机的设计

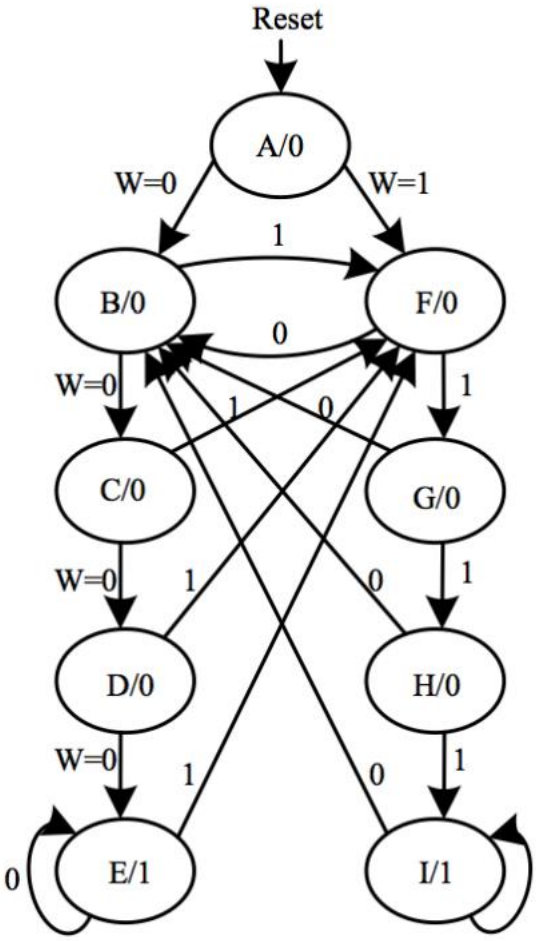
2.实验原理

2.1 状态机

一个简单的状态机:

入 w 和一个输出 z。当 w 是 4 个连续的 0 或 4 个连续的 1 时，输出 z=1，否则 z=0，时序允许重叠。即：若 w 是连续的 5 个 1 时，则在第 4 个和第 5 个时钟之后，z 均为 1。图 7-1是这个有限状态机的时序图。




Reset
A/0
W=0
W=1
1
B/0
F/0
0
W=0
1
0
1
C/0
G/0
W=0
1
0
1
D/0
H/0
W=0
1
0
1
0
E/1
I/1

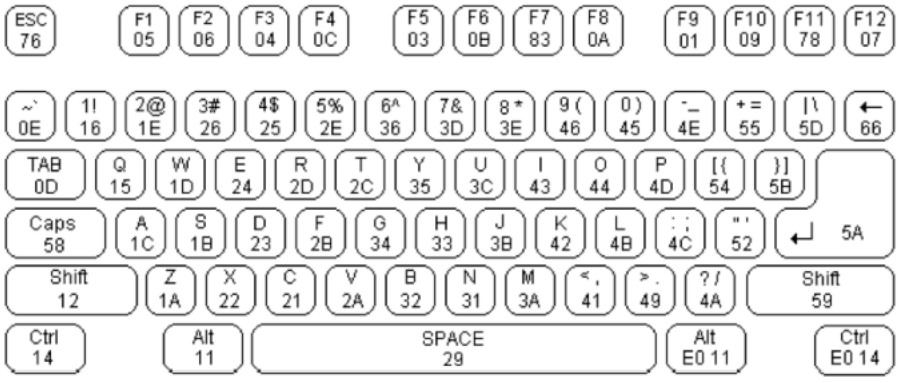
2.2 ps2 键盘

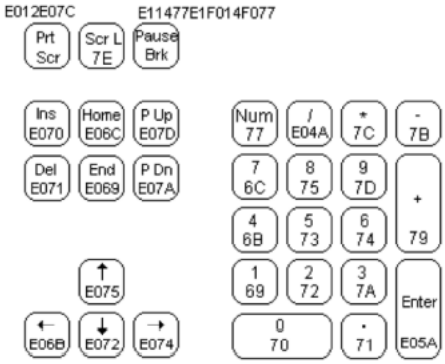
当用户按键或松开时，键盘以每帧 11 位的格式串行传送数据给主机，同时在 PS2_CLK 时钟信号上传输对应的时钟（一般为 10.0–16.7kHz）。第一位是开始位（逻辑 0），后面跟 8 位数据位（低位在前），一个奇偶校验位（奇校验）和一位停止位（逻辑 1）。每位都在时钟的**下降沿**有效，图 7-4显示了键盘传送一字节数据的时序。在下降沿有效的主要原因是下降沿正好在数据位的中间，因此可以让数据位从开始变化到接收采样时能有一段信号建立时间。

键盘通过 PS2_DAT 引脚发送的信息称为扫描码，每个扫描码可以由单个数据帧或连续多个数据帧构成。当按键被按下时送出的扫描码被称为“通码（Make Code）”，当按键被释放时送出的扫描码称为“断码（Break Code）”。以“W”键为例，“W”键的通码是 1Dh，如果“W”键被按下，则 PS2_DAT 引脚将输出一帧数据，其中的 8 位数据位为 1Dh，如果“W”键一直没有释放，则不断输出扫描码 1Dh 1Dh . . . 1Dh，直到有其他键按下或者“W”键被放开。某按键的断码是 F0h 加此按键的通码，如释放“W”键时输出的断码为 F0h 1Dh，分两帧传输。

多个键被同时按下时，将逐个输出扫描码，如：先按左“Shift”键（扫描码为 12h）、再按“W”键、放开“W”键、再放开左“Shift”键，则此过程送出的全部扫描码为：12h 1Dh F0h 1Dh F0h 12h。

2.3 扫描码





3.实验环境

- 软件环境

Quartus 17.1 Lite

- 硬件环境

开发板：DE10 Standard

FPGA：Intel Cyclone V SE 5CSXFC6D6F31C6N

4.实验步骤和结果

4.1 keyboard 模块

4.1.1 代码

```
module keyboard(input clk,
    input clrn,
    input ps2_clk,
    input ps2_data,
    output reg [7:0] key_count,
    output reg [7:0] cur_key,
    output [7:0] ascii_key,
    output reg flag_ctrl,
    output reg flag_shift,
    output reg flag_cur,
    output reg flag_caps
);

// add your definitions here
reg [7:0] lastdata;
wire [7:0] keydata;
wire ready;
reg nextdata_n;
wire overflow;



//----DO NOT CHANGE BEGIN----
//scancode to ascii conversion, will be initialized by the testbench
scancode_ram myram(clk, cur_key, flag_shift,flag_caps,ascii_key);
//PS2 interface, you may need to specify the inputs and outputs
ps2_keyboard mykey(clk, clrn, ps2_clk, ps2_data, keydata, ready, nextdata_n, overflow);
//---DO NOT CHANGE END-----

// add you code here
reg flag=0;
```

```
always @(posedge ps2_clk)
begin
if(flag==0)//初始化
   begin
   cur_key=0;
   lastdata=0;
   flag=1;
   flag_cur=0;
   flag_caps=0;
   flag_ctrl=0;
   flag_shift=0;
   end
if(ready&&nextdata_n)//&& nextdata_n
   begin
   lastdata=cur_key;
   cur_key=keydata;
   nextdata_n=0;


     if(lastdata!=cur_key && lastdata!=8'hf0 && cur_key!=8'hf0)
        begin
        key_count=key_count+1;
        if(cur_key==8'h12)
        flag_shift=1;
        if(cur_key==8'h14)
        flag_ctrl=1;
        if(cur_key==8'h58)
        flag_caps=1-flag_caps;
        if(cur_key!=8'h12 && cur_key!=8'h14 && cur_key!=8'h58)
        flag_cur=1;
        end

    else if(lastdata==8'hf0 && cur_key!=0 ) //
        begin
        if(cur_key==8'h12)flag_shift=0;
        if(cur_key==8'h14)flag_ctrl=0;
        if(cur_key!=8'h12 && cur_key!=8'h14)
        flag_cur=0;
        lastdata=cur_key;
        cur_key=0;
        end



   end
else
   begin
   nextdata_n=1;
   end
if(clrn==0)
   key_count=0;
end

endmodule
```

4.1.2 接口

```
keyboard k(CLOCK_50,
    SW[0],//清零
    PS2_CLK,
    PS2_DAT,
    a,
    c,
    b,
    LEDR[0],//ctrl
    LEDR[1],//shift
    flag,
    LEDR[2]//caps
);
```

4.1.3 仿真

由于是七段数码管显示数字，见于上板实验。

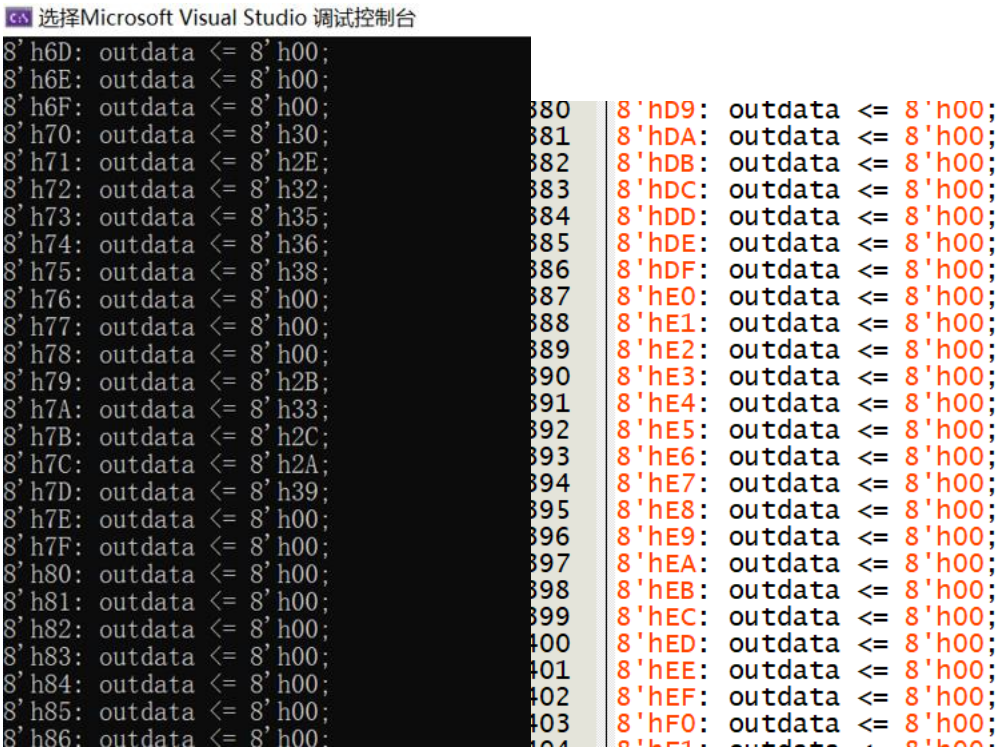
4.2 扫描码转 ascii 码

由于我在看到 cslab 的 mif 之前，就已经自己写好了个 ram，但同时又缺了一些数据，避免一个个比对，就自己编了一个 c 程序将原先 mif 文件中的扫描码和 ascii 码对应传进来，然后转换成 verilog 格式的代码。

```
string s;
vector<string> vec;
for (int i = 0; i < 0xff;i++)//3 4 8 9
{
    getline(cin, s);
    //8'h15: outdata <= 8'h71;
    string temp = "8'h";
        temp+= s[3];
        temp+= s[4];
        temp += ": outdata <= 8'h";
        temp += s[8];
        temp += s[9];
        temp+=';';
    cout << temp<<endl;
    vec.push_back(temp);

}

for (int i = 0; i < vec.size(); i++)
    cout << vec[i] << endl;
```



4.3 其余模块

ps2 按照头歌所给框架所写，七段数码管按照前面实验所述。

这里要特别注意 ps2 的时钟周期的规律，要与 keyboard 模块相对应。

5. 实验中遇到的问题及解决办法

问题：在按下 shift+W 再松开 W 此时后四位全灰，再松开 shift 此时后四位反而会亮。

解决办法：开始的时候， 把按键判断标准改为 lastdata!=cur_key && lastdata!=8'hf0 && cur_key!=8'hf0，修复了 shift 的 bug，原因未知。后来根据实验证实，先 shift+某按键然而松开该按键再松开 shift 后，得到的码是 0f 12 12 而不是 0f 12（而只按 shift 则不会）， 多一个 12，原因未知。另外 ctrl 和 shift 一样实现，上板却不太一样，里面可能也有类似问题。再后来经过调试，发现是判断错误，if(ready)判断的话可能时钟周期内会多读一次之前的数据，正确的判断方法是 if(ready&&nextdata_n)。

6.实验启示和建议

这个实验的键盘组合按键如果有一些奇怪的 bug（和 pdf 所述不符），很可能是对实验给定的 ps2 模块理解不够，这里的时钟沿确实比较复杂。