

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	2
2.实验原理	2
2.1 状态机.....	2
2.2 ps2 键盘	4
2.3 扫描码.....	4
3.实验环境	5
4.实验步骤和结果	6
4.1 keyboard 模块.....	6
4.1.1 代码.....	6
4.1.2 接口	8
4.1.3 仿真.....	8
4.2 扫描码转 ascii 码	8
4.3 其余模块	9
5. 实验中遇到的问题及解决办法	9
6.实验启示和建议	10

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

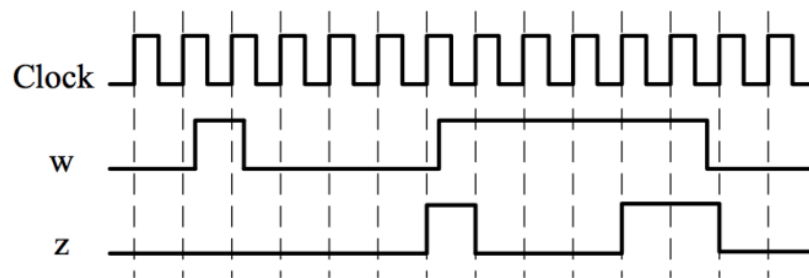
利用 PS/2 键盘输入实现简单状态机的设计

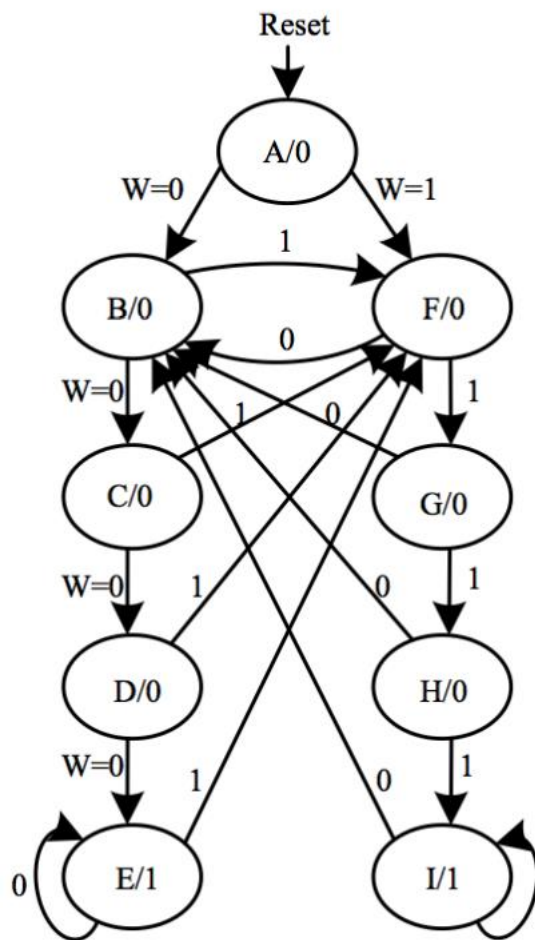
2.实验原理

2.1 状态机

一个简单的状态机：

输入 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。





2.2 ps2 键盘

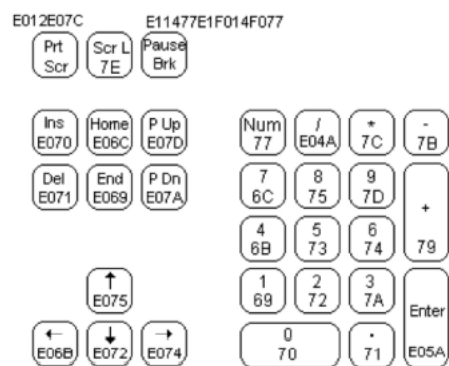
当用户按键或松开时，键盘以每帧 11 位的格式串行传送数据给主机，同时在 PS2_CLK 时钟信号上传输对应的时钟（一般为 10.0–16.7kHz）。第一位是开始位（逻辑 0），后面跟 8 位数据位（低位在前），一个奇偶校验位（奇校验）和一位停止位（逻辑 1）。每位都在时钟的下降沿有效，图 7-4 显示了键盘传送一字节数据的时序。在下降沿有效的主要原因是下降沿正好在数据位的中间，因此可以让数据位从开始变化到接收采样时能有一段信号建立时间。

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

ESC 76	F05	F06	F04	F4 0C	F03	F0B	F83	F8 0A	F9 01	F10 09	F11 78	F12 07		
~ 0E	1! 16	2@ 1E	3# 26	4\$ 25	5% 2E	6^ 36	7& 3D	8* 3E	9(46	0) 45	-_ 4E	+= 55	[\ 5D	← 66
TAB 0D	Q 15	W 1D	E 24	R 2D	T 2C	Y 35	U 3C	I 43	O 44	P 4D	[{ 54]} 5B		
Caps 58	A 1C	S 1B	D 23	F 2B	G 34	H 33	J 3B	K 42	L 4B	:: 4C	"" 52	↵ 5A		
Shift 12	Z 1A	X 22	C 21	V 2A	B 32	N 31	M 3A	<, 41	>. 49	?/ 4A	Shift 59			
Ctrl 14	Alt 11	SPACE 29								Alt E0 11	Ctrl E0 14			



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 cs1ab 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;
```

选择Microsoft Visual Studio 调试控制台

```
8'h6D: outdata <= 8'h00;
8'h6E: outdata <= 8'h00;
8'h6F: outdata <= 8'h00;
8'h70: outdata <= 8'h30;
8'h71: outdata <= 8'h2E;
8'h72: outdata <= 8'h32;
8'h73: outdata <= 8'h35;
8'h74: outdata <= 8'h36;
8'h75: outdata <= 8'h38;
8'h76: outdata <= 8'h00;
8'h77: outdata <= 8'h00;
8'h78: outdata <= 8'h00;
8'h79: outdata <= 8'h2B;
8'h7A: outdata <= 8'h33;
8'h7B: outdata <= 8'h2C;
8'h7C: outdata <= 8'h2A;
8'h7D: outdata <= 8'h39;
8'h7E: outdata <= 8'h00;
8'h7F: outdata <= 8'h00;
8'h80: outdata <= 8'h00;
8'h81: outdata <= 8'h00;
8'h82: outdata <= 8'h00;
8'h83: outdata <= 8'h00;
8'h84: outdata <= 8'h00;
8'h85: outdata <= 8'h00;
8'h86: outdata <= 8'h00;
8'h87: outdata <= 8'h00;
8'h88: outdata <= 8'h00;
8'h89: outdata <= 8'h00;
8'h8A: outdata <= 8'h00;
8'h8B: outdata <= 8'h00;
8'h8C: outdata <= 8'h00;
8'h8D: outdata <= 8'h00;
8'h8E: outdata <= 8'h00;
8'h8F: outdata <= 8'h00;
8'h90: outdata <= 8'h00;
8'h91: outdata <= 8'h00;
8'h92: outdata <= 8'h00;
8'h93: outdata <= 8'h00;
8'h94: outdata <= 8'h00;
8'h95: outdata <= 8'h00;
8'h96: outdata <= 8'h00;
8'h97: outdata <= 8'h00;
8'h98: outdata <= 8'h00;
8'h99: outdata <= 8'h00;
8'h9A: outdata <= 8'h00;
8'h9B: outdata <= 8'h00;
8'h9C: outdata <= 8'h00;
8'h9D: outdata <= 8'h00;
8'h9E: outdata <= 8'h00;
8'h9F: outdata <= 8'h00;
8'ha0: outdata <= 8'h00;
8'ha1: outdata <= 8'h00;
8'ha2: outdata <= 8'h00;
8'ha3: outdata <= 8'h00;
8'ha4: outdata <= 8'h00;
8'ha5: outdata <= 8'h00;
8'ha6: outdata <= 8'h00;
8'ha7: outdata <= 8'h00;
8'ha8: outdata <= 8'h00;
8'ha9: outdata <= 8'h00;
8'haA: outdata <= 8'h00;
8'haB: outdata <= 8'h00;
8'haC: outdata <= 8'h00;
8'haD: outdata <= 8'h00;
8'haE: outdata <= 8'h00;
8'haF: outdata <= 8'h00;
8'hb0: outdata <= 8'h00;
8'hb1: outdata <= 8'h00;
8'hb2: outdata <= 8'h00;
8'hb3: outdata <= 8'h00;
8'hb4: outdata <= 8'h00;
8'hb5: outdata <= 8'h00;
8'hb6: outdata <= 8'h00;
8'hb7: outdata <= 8'h00;
8'hb8: outdata <= 8'h00;
8'hb9: outdata <= 8'h00;
8'hba: outdata <= 8'h00;
8'hbb: outdata <= 8'h00;
8'hbc: outdata <= 8'h00;
8'hbd: outdata <= 8'h00;
8'hbe: outdata <= 8'h00;
8'hbf: outdata <= 8'h00;
8'hc0: outdata <= 8'h00;
8'hc1: outdata <= 8'h00;
8'hc2: outdata <= 8'h00;
8'hc3: outdata <= 8'h00;
8'hc4: outdata <= 8'h00;
8'hc5: outdata <= 8'h00;
8'hc6: outdata <= 8'h00;
8'hc7: outdata <= 8'h00;
8'hc8: outdata <= 8'h00;
8'hc9: outdata <= 8'h00;
8'hca: outdata <= 8'h00;
8'hcb: outdata <= 8'h00;
8'hcc: outdata <= 8'h00;
8'hcd: outdata <= 8'h00;
8'hce: outdata <= 8'h00;
8'hcf: outdata <= 8'h00;
8'hd0: outdata <= 8'h00;
8'hd1: outdata <= 8'h00;
8'hd2: outdata <= 8'h00;
8'hd3: outdata <= 8'h00;
8'hd4: outdata <= 8'h00;
8'hd5: outdata <= 8'h00;
8'hd6: outdata <= 8'h00;
8'hd7: outdata <= 8'h00;
8'hd8: outdata <= 8'h00;
8'hd9: outdata <= 8'h00;
8'hda: outdata <= 8'h00;
8'hdb: outdata <= 8'h00;
8'hdc: outdata <= 8'h00;
8'hdd: outdata <= 8'h00;
8'hde: outdata <= 8'h00;
8'hdf: outdata <= 8'h00;
8'he0: outdata <= 8'h00;
8'he1: outdata <= 8'h00;
8'he2: outdata <= 8'h00;
8'he3: outdata <= 8'h00;
8'he4: outdata <= 8'h00;
8'he5: outdata <= 8'h00;
8'he6: outdata <= 8'h00;
8'he7: outdata <= 8'h00;
8'he8: outdata <= 8'h00;
8'he9: outdata <= 8'h00;
8'hea: outdata <= 8'h00;
8'heb: outdata <= 8'h00;
8'hec: outdata <= 8'h00;
8'hed: outdata <= 8'h00;
8'hee: outdata <= 8'h00;
8'hef: outdata <= 8'h00;
8'hf0: outdata <= 8'h00;
8'hf1: outdata <= 8'h00;
8'hf2: outdata <= 8'h00;
8'hf3: outdata <= 8'h00;
8'hf4: outdata <= 8'h00;
8'hf5: outdata <= 8'h00;
8'hf6: outdata <= 8'h00;
8'hf7: outdata <= 8'h00;
8'hf8: outdata <= 8'h00;
8'hf9: outdata <= 8'h00;
8'hfa: outdata <= 8'h00;
8'hfb: outdata <= 8'h00;
8'hfc: outdata <= 8'h00;
8'hfd: outdata <= 8'h00;
8'hfe: outdata <= 8'h00;
8'hff: outdata <= 8'h00;
```

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+ 某按键然后松开该按键再松开 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。