数字电路与数字系统实验

实验八 键盘

姓名：

学号：

班级：

邮箱：

实验时间：

一、 实验目的

学习状态机的原理，通过 verilog 语言设计、实现一个状态机，实现单个按键的 ascii 码显示。

二、 实验原理

有限状态机被分为两种： Moore（摩尔）型有限状态机和 Mealy（米里）型有限状态机。 Moore 型有限状态机的输出信号只与有限状态机的当前状态有关，与输入信号的当前值无关，输入信号的当前值只会影响到状态机的次态，不会影响状态机当前的输出。Mealy 有限状态机的输出不仅仅与状态机的当前状态有关，而且与输入信号的当前值也有关。 Mealy 有限状态机的输出直接受输入信号的当前值影响，而输入信号可能在一个时钟周期内任意时刻变化，这使得 Mealy 有限状态机对输入的响应发生在当前时钟周期，比 Moore 有限状态机对输入信号的响应要早一个周期。因此，输入信号的噪声可能影响到输出的信号。

三、 实验环境

Quartus 18.1、FPGA 开发板

四、 实验过程

设计思路：

主要分为四个模块，第一个是实验手册上提供 ps2_keyboard，用来接收键盘的输入；第二个模块是 ram，通过 ps2_keyboard 里接收的键盘扫描码来获取对应的 ascii 码；第三个模块是 show，通过标识键盘是否按下的 flag 来决定是否显示输出；第四个模块是顶层模块 keyboard，调用上述三个模块和判断键盘输入的情况。当 ps2_keyboard 输出 ready 时，若 data 为 8’fh0 时，表示松开按键，将所有 flag 置为 0；若 flag2 为 0，表示断码已经过去，此时可以接受新的键盘扫描码，所以将 flag2 置为 1；若 data 不为 8’hf0 且 flag2 为 1，表示已经按下一个键，将 flag1 置为 1，显示新按键的扫描码和 ascii 码。

设计代码：

keyboard. v

```
module keyboard(clk,clrn,ps2_clk,ps2_data,hex0,hex1,hex2,hex3,hex4,hex5);
input clk,clrn,ps2_clk,ps2_data;

output [6:0] hex0,hex1,hex2,hex3,hex4,hex5;

wire ready,overflow;
wire [7:0] data, asc;

reg nextdata_n;
reg flag1 = 1;
reg flag2 = 1;
reg flag3 = 0;
reg always_show = 1;
reg [7:0] count=0;
reg [7:0] outdata=0;

reg [6:0] count_clk = 0;
reg clk2 = 0;
always @(posedge clk)
begin
   if(count_clk == 100)
   begin
      count_clk <= 0;
      clk2 <= ~clk2;
   end
   else
      count_clk <= count_clk + 1;
end

ps2_keyboard P(.clk(clk), .clrn(clrn), .ps2_clk(ps2_clk), .ps2_data(ps2_data),
```

```
ps2_keyboard P(.clk(clk), .clrn(clrn), .ps2_clk(ps2_clk), .ps2_data(ps2_data),
.data(data), .ready(ready), .nextdata_n(nextdata_n), .overflow(overflow));

ram R(.data(outdata), .data_asc(asc), .flag(flag3));

show S0(.flag(flag1), .data(outdata[3:0]), .hex(hex0));
show S1(.flag(flag1), .data(outdata[7:4]), .hex(hex1));
show S2(.flag(flag1), .data(asc[3:0]), .hex(hex2));
show S3(.flag(flag1), .data(asc[7:4]), .hex(hex3));
show S4(.flag(always_show), .data(count[3:0]), .hex(hex4));
show S5(.flag(always_show), .data(count[7:4]), .hex(hex5));

always @ (posedge clk2)
begin
   if (ready)
   begin
      if (data == 8'hf0)
      begin
         outdata <= data;
         count <= count + 1;
         flag2 <= 0;
         flag1 <= 0;
      end
      else if(!flag2)
      begin
         outdata <= data;
         flag2 <= 1;
         flag1 <= 0;
      end
      else if(data != 8'hf0 && flag2)
      begin
```

```
      else if(data != 8'hf0 && flag2)
      begin
         outdata <= data;
         flag1 <= 1;
      end

      if(data == 8'h12 && outdata != 8'hf0) flag3 <= 1;
      else if(data == 8'h12 && outdata == 8'hf0) flag3 <= 0;
      nextdata_n <= 0;
   end
   else nextdata_n <= 1;
end

endmodule
```

show.v

```
module show(flag,data,hex);
input flag;
input [3:0] data;

output reg [6:0] hex;

always @ (*)
   if(flag)
      case(data)
         4'h0: hex = 7'b1000000;
         4'h1: hex = 7'b1111001;
         4'h2: hex = 7'b0100100;
         4'h3: hex = 7'b0110000;
         4'h4: hex = 7'b0011001;
         4'h5: hex = 7'b0010010;
         4'h6: hex = 7'b0000010;
         4'h7: hex = 7'b1111000;
         4'h8: hex = 7'b0000000;
         4'h9: hex = 7'b0010000;
         4'ha: hex = 7'b0001000;
         4'hb: hex = 7'b0000011;
         4'hc: hex = 7'b1000110;
         4'hd: hex = 7'b0100001;
         4'he: hex = 7'b0000110;
         4'hf: hex = 7'b0001110;
         default: hex = 7'b1111111;
      endcase
   else hex = 7'b1111111;

endmodule
```

ram.v

```
module ram(data, data_asc,flag);
   input [7:0] data;
   input flag;
   output reg [7:0] data_asc;
   reg [7:0] ram [255:0];

   initial
   begin
      $readmemh("C:/intelFPGA_lite/exp/exp08/ram.txt", ram, 0, 255);
   end

   always
   begin
      data_asc = ram[data];
      if(flag) data_asc = data_asc - 8'h20;
   end
endmodule
```

ps2_keyboard.v 和实验手册上的一样…

ModelSim 仿真：
没有进行这一步，直接在开发板上进行验证…

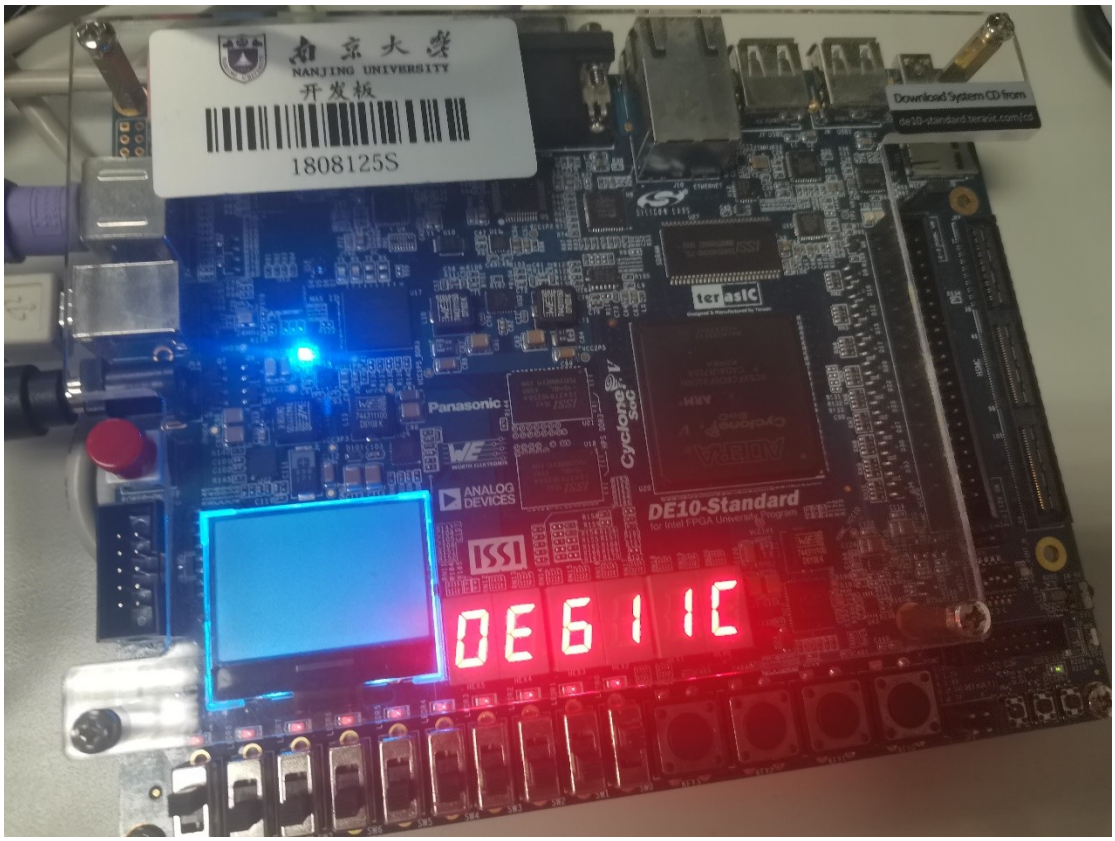
引脚分配：

Node Name	Direction	Location	I/O Bank
clk	Input	PIN_AF14	3B
clrn	Input	PIN_AB30	5B
hex0[6]	Output	PIN_AH18	4A
hex0[5]	Output	PIN_AG18	4A
hex0[4]	Output	PIN_AH17	4A
hex0[3]	Output	PIN_AG16	4A
hex0[2]	Output	PIN_AG17	4A
hex0[1]	Output	PIN_V18	4A
hex0[0]	Output	PIN_W17	4A
hex1[6]	Output	PIN_V17	4A
hex1[5]	Output	PIN_AE17	4A
hex1[4]	Output	PIN_AE18	4A
hex1[3]	Output	PIN_AD17	4A
hex1[2]	Output	PIN_AE16	4A
hex1[1]	Output	PIN_V16	4A
hex1[0]	Output	PIN_AF16	4A
hex2[6]	Output	PIN_W16	4A

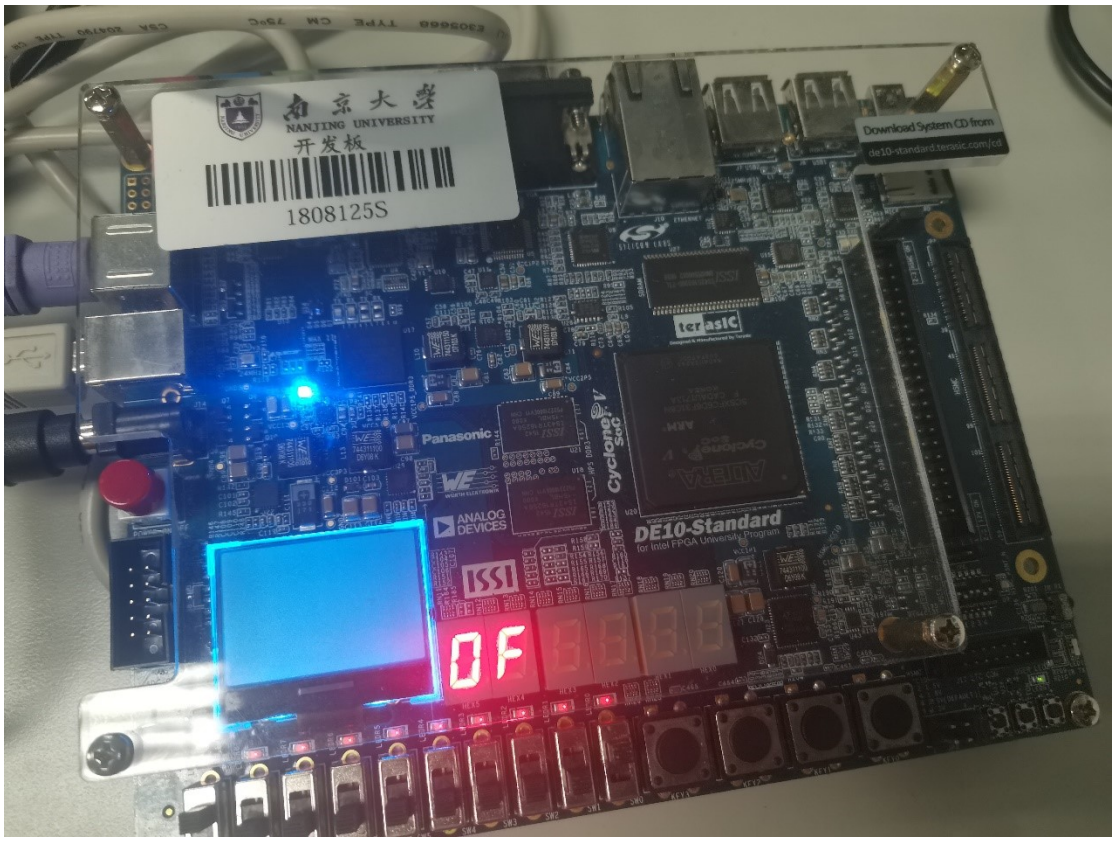
Node Name	Direction	Location	I/O Bank
hex2[5]	Output	PIN_AF18	4A
hex2[4]	Output	PIN_Y18	4A
hex2[3]	Output	PIN_Y17	4A
hex2[2]	Output	PIN_AA18	4A
hex2[1]	Output	PIN_AB17	4A
hex2[0]	Output	PIN_AA21	4A
hex3[6]	Output	PIN_AD20	4A
hex3[5]	Output	PIN_AA19	4A
hex3[4]	Output	PIN_AC20	4A
hex3[3]	Output	PIN_AA20	4A
hex3[2]	Output	PIN_AD19	4A
hex3[1]	Output	PIN_W19	4A
hex3[0]	Output	PIN_Y19	4A
hex4[6]	Output	PIN_AH22	4A
hex4[5]	Output	PIN_AF23	4A
hex4[4]	Output	PIN_AG23	4A
hex4[3]	Output	PIN_AE23	4A

out hex4[2]	Output	PIN_AE22	4A
out hex4[1]	Output	PIN_AG22	4A
out hex4[0]	Output	PIN_AD21	4A
out hex5[6]	Output	PIN_AB21	4A
out hex5[5]	Output	PIN_AF19	4A
out hex5[4]	Output	PIN_AE19	4A
out hex5[3]	Output	PIN_AG20	4A
out hex5[2]	Output	PIN_AF20	4A
out hex5[1]	Output	PIN_AG21	4A
out hex5[0]	Output	PIN_AF21	4A
in ps2_clk	Input	PIN_AB25	5A
in ps2_data	Input	PIN_AA25	5A
<<new node>>			

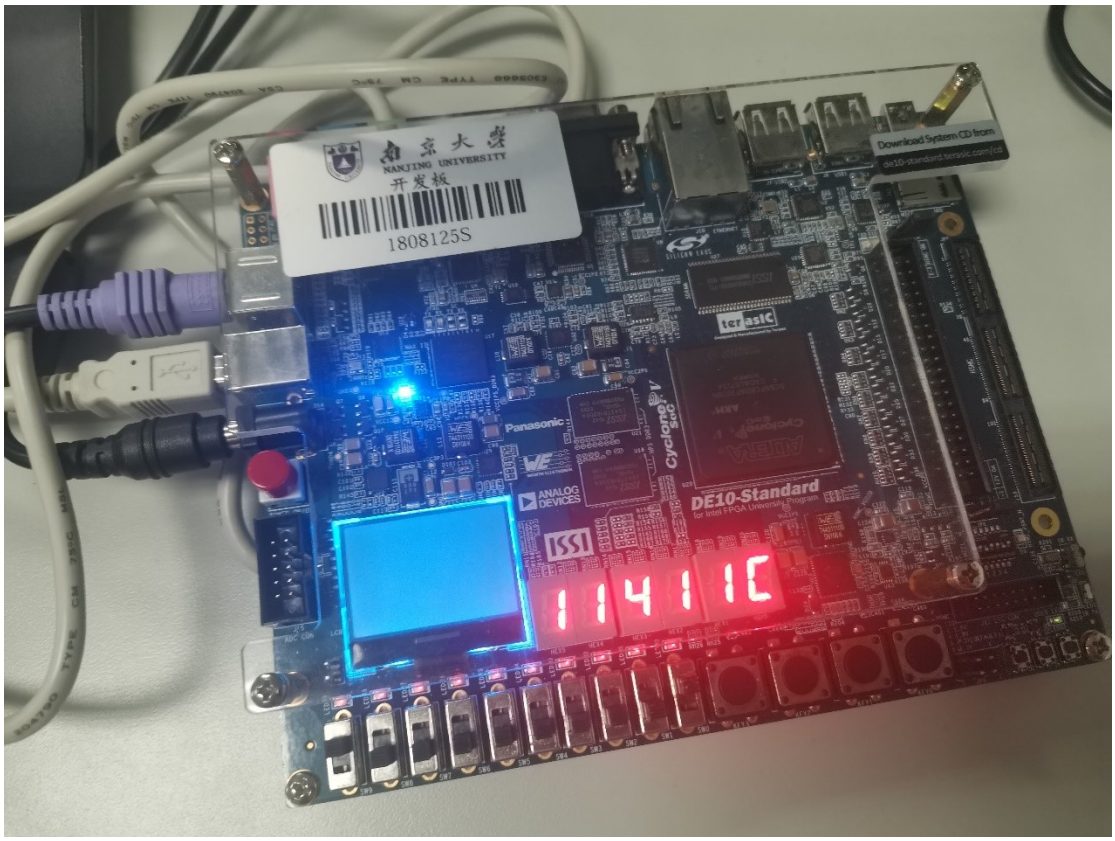
实验结果：
按一个 a：



松开 a：



按 shift+a，显示大写 A：



五、 实验中遇到的问题及解决办法

1. 一开始只设置了一个 flag 来判断是否应该显示和计数，导致扫描码变化且每次计数都不是增加 1，后来经过调试，增加了一个 flag，解决了这个问题。

六、 启示

☞ 高级要求（选做）

- 支持 Shift，CTRL 等组合键，在 LED 上显示组合键是否按下的状态指示
- 支持 Shift 键与字母/数字键同时按下，相互不冲突
- 支持输入大写字符，显示对应的 ASCII 码

通过 flag3 判断是否按下 shift，若按下，则输出大写字母的 ascii 码。

keyboard.v

```
if(data == 8'h12 && outdata != 8'hf0) flag3 <= 1;
else if(data == 8'h12 && outdata == 8'hf0) flag3 <= 0;
```

ram.v

```
always
begin
    data_asc = ram[data];
    if(flag) data_asc = data_asc - 8'h20;
end
endmodule
```

七、 意见与建议

虽然之前我并没有上过数字电路这门课，但是实验手册前面的讲解非常的清楚，由浅入深，帮助我学习和完成了这次实验。