实验报告lab7

姓名 林凡琪

学号 211240042

班级 匡亚明学院

邮箱 211240042@smail.nju.edu.cn

实验目的

本实验的目的是学习状态机的工作原理，了解状态机的编码方式，并利用PS/2键盘输入实现简单状态机的设计。

实验原理（知识背景）

- 状态机
  - 有限状态机
  - 简单状态机
  - 状态机的编码方式
- PS/2接口控制器及键盘输入

PS/2是个人计算机串行I/O 接口的一种标准，因其首次在IBM PS/2（Personal System/2）机器上使用而得名，PS/2 接口可以连接PS/2键盘和PS/2鼠标。所谓串行接口是指信息是在单根信号线上按序一位一位发送的。

PS/2 接口使用两根信号线，一根信号线传输时钟 PS2_CLK，另一根传输数据 PS2_DAT。时钟信号主要用于指示数据线上的比特位在什么时候是有效的。键盘和主机间可以进行数据双向传送，这里只讨论键盘向主机传送数据的情况。当 PS2_DAT 和 PS2_CLK 信号线都为高电平（空闲）时，键盘才可以给主机发送信号。如果主机将 PS2_CLK 信号置低，键盘将准备接受主机发来的命令。在我们的实验中，主机不需要发命令，只需将这两根信号线做为输入即可。
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␣屡 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）”。

详情可见课程网站给出的实验指导.

键盘输出数据的时序图如下，实验讲义中已经给出键盘控制器模块ps2_keyboard 的完整实现代码

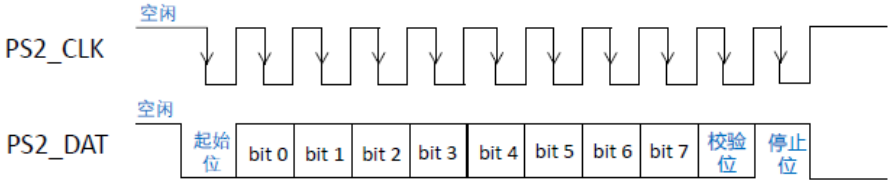


图 7–4: 键盘输出数据时序图

实验环境/器材等

Verilog 2022.1

Windows 10

Nexys开发板

程序代码或流程图

```
29  // add your definitions here
30  reg  state;
31  wire [7:0] keydata;
32  wire ready;
33  reg nextdata_n;
34  wire overflow;
35  reg jump;
36  reg [7:0] key_count;
37  reg [7:0] cur_key;
38  wire [7:0] ascii_key;
39
40  //----DO NOT CHANGE BEGIN----
41  //scancode to ascii conversion, will be initialized by the testbench
42  scancode_ram myram(clk, cur_key, ascii_key);
43  //PS2 interface, you may need to specify the inputs and outputs
44  ps2_keyboard mykey(clk, clrn, ps2_clk, ps2_data, keydata, ready, nextdata_n,
    overflow);
45  //---DO NOT CHANGE END-----
46
47  //segment
48  sevenseg my_7seg(clk_10k, 8'b00111111, {8'b0, key_count, ascii_key, cur_key},
    AN, HEX);
49  // add you code here
50  initial
51  begin
52      key_count = 8'b0;
53      cur_key = 8'b0;
54      //ascii_key = 8'b0;
55      state = 0;
56      nextdata_n = 1'b0;
57      //keydata = 8'b0;
58      jump = 1'b0;
59  end
60
61      always @(posedge clk) begin
62          if (!clrn) begin
63              key_count <= 8'b0;
64              cur_key <= 8'b0;
65              nextdata_n <= 1'b1;
66              state <= 0;
67              jump <= 1'b0;
68          end
69          else begin
70             // nextdata_n <= 1'b1;
71              if(ready)
72              begin
```

```
69          else begin
70             // nextdata_n <= 1'b1;
71              if(ready)
72              begin
73
74                  if (keydata == 8'hf0)
75                  begin
76                      state <= 1'b0;
77                      //cur_key <= 8'b0;
78                      jump <= 1'b1;
79                  end
80                  else
81                  begin
82                      if (state == 1'b0)//
83                      begin
84                          if (jump == 1'b1 && keydata == cur_key)//
85                          begin
86                              cur_key <= 8'b0;
87                              state <= 1'b0;
88                              jump <= 1'b0;//
89                              //key_count <= key_count + 10;
90                          end
91                          else begin//
92                              key_count <= key_count + 1;
93                              cur_key <= keydata;
94                              state <= 1'b1;
95                          end
96                      end
97                      else begin//
98                          cur_key <= keydata;
99                          //key_count <= key_count + 10;
100                         state <= 1'b1;
101                     end
102                 end
103                  //nextdata_n <= 1'b0;
104             end
105         end
106     end
107 endmodule
```

lab07.v文件

```
23  module keyboard_sim;
24
25    parameter [31:0] clock_period = 10;
26    reg clk, clrn;
27    wire [7:0] data;
28    wire ready, overflow;
29    wire kbd_clk, kbd_data;
30    reg nextdata_n;
31    reg [7:0] AN;
32    reg [7:0] hex;
33    wire [7:0] key_count;
34
35    ps2_keyboard_model model(
36      .ps2_clk(kbd_clk),
37      .ps2_data(kbd_data)
38    );
39
40    ps2_keyboard inst(
41      .clk(clk),
42      .clrn(clrn),
43      .ps2_clk(kbd_clk),
44      .ps2_data(kbd_data),
45      .data(data),
46      .ready(ready),
47      .nextdata_n(nextdata_n),
48      .overflow(overflow)
49    );
50
51    lab07 i1(
52    .clk(clk),
53    .clrn(clrn),
54    .ps2_clk(ps2_clk),
55    .ps2_data(ps2_data),
56    .key_count(key_count));
57
58    initial begin
59      clk = 0;
60      forever
61        #(clock_period/2) clk = ~clk;
62    end
63
64    initial begin
65      clrn = 1'b0;  #20;
66      clrn = 1'b1;  #20;
67      model.kbd_sendcode(8'h1C);
68      #20 nextdata_n = 1'b0; #20  nextdata_n = 1'b1;
69      model.kbd_sendcode(8'hF0);
```

```
64    initial begin
65      clrn = 1'b0;  #20;
66      clrn = 1'b1;  #20;
67      model.kbd_sendcode(8'h1C);
68      #20 nextdata_n = 1'b0; #20  nextdata_n = 1'b1;
69      model.kbd_sendcode(8'hF0);
70      #20 nextdata_n = 1'b0; #20  nextdata_n = 1'b1;
71      model.kbd_sendcode(8'h1C);
72      #20 nextdata_n = 1'b0; #20  nextdata_n = 1'b1;
73      model.kbd_sendcode(8'h1B);
74      #20 model.kbd_sendcode(8'h1B);
75      #20 model.kbd_sendcode(8'h1B);
76      model.kbd_sendcode(8'hF0);
77      model.kbd_sendcode(8'h1B);
78      #20;
79      $stop;
80    end
81
82  endmodule
83
```

test文件

```
## Clock signal
set_property -dict { PACKAGE_PIN E3    IOSTANDARD LVCMOS33 } [get_ports { clk }]; #IO_L12P_T1_MRCC_35 Sch=clk100mhz
#create_clock -add -name sys_clk_pin -period 10.00 -waveform {0 5} [get_ports {CLK100MHZ}];


##Switches
set_property -dict { PACKAGE_PIN J15   IOSTANDARD LVCMOS33 } [get_ports { clrn }]; #IO_L24N_T3_RS0_15 Sch=sw[0]
#set_property -dict { PACKAGE_PIN L16   IOSTANDARD LVCMOS33 } [get_ports { SW[1] }]; #IO_L3N_T0_DQS_EMCCLK_14 Sch=sw[1]
#set_property -dict { PACKAGE_PIN M13   IOSTANDARD LVCMOS33 } [get_ports { SW[2] }]; #IO_L6N_T0_D08_VREF_14 Sch=sw[2]
#set_property -dict { PACKAGE_PIN R15   IOSTANDARD LVCMOS33 } [get_ports { SW[3] }]; #IO_L13N_T2_MRCC_14 Sch=sw[3]
#set_property -dict { PACKAGE_PIN R17   IOSTANDARD LVCMOS33 } [get_ports { SW[4] }]; #IO_L12N_T1_MRCC_14 Sch=sw[4]
#set_property -dict { PACKAGE_PIN T18   IOSTANDARD LVCMOS33 } [get_ports { SW[5] }]; #IO_L7N_T1_D10_14 Sch=sw[5]
#set_property -dict { PACKAGE_PIN U18   IOSTANDARD LVCMOS33 } [get_ports { SW[6] }]; #IO_L17N_T2_A13_D29_14 Sch=sw[6]
#set_property -dict { PACKAGE_PIN R13   IOSTANDARD LVCMOS33 } [get_ports { SW[7] }]; #IO_L5N_T0_D07_14 Sch=sw[7]
#set_property -dict { PACKAGE_PIN T8    IOSTANDARD LVCMOS18 } [get_ports { SW[8] }]; #IO_L24N_T3_34 Sch=sw[8]
#set_property -dict { PACKAGE_PIN U8    IOSTANDARD LVCMOS18 } [get_ports { SW[9] }]; #IO_25_34 Sch=sw[9]
#set_property -dict { PACKAGE_PIN R16   IOSTANDARD LVCMOS33 } [get_ports { SW[10] }]; #IO_L15P_T2_DQS_RDWR_B_14 Sch=sw[10]
#set_property -dict { PACKAGE_PIN T13   IOSTANDARD LVCMOS33 } [get_ports { SW[11] }]; #IO_L23P_T3_A03_D19_14 Sch=sw[11]
#set_property -dict { PACKAGE_PIN H6    IOSTANDARD LVCMOS33 } [get_ports { SW[12] }]; #IO_L24P_T3_35 Sch=sw[12]
#set_property -dict { PACKAGE_PIN U12   IOSTANDARD LVCMOS33 } [get_ports { SW[13] }]; #IO_L20P_T3_A08_D24_14 Sch=sw[13]
#set_property -dict { PACKAGE_PIN U11   IOSTANDARD LVCMOS33 } [get_ports { SW[14] }]; #IO_L19N_T3_A09_D25_VREF_14 Sch=sw[14]
#set_property -dict { PACKAGE_PIN V10   IOSTANDARD LVCMOS33 } [get_ports { SW[15] }]; #IO_L21P_T3_DQS_14 Sch=sw[15]

## LEDs
#set_property -dict { PACKAGE_PIN H17   IOSTANDARD LVCMOS33 } [get_ports { ready }]; #IO_L18P_T2_A24_15 Sch=led[0]
#set_property -dict { PACKAGE_PIN K15   IOSTANDARD LVCMOS33 } [get_ports {  }]; #IO_L24P_T3_RS1_15 Sch=led[1]
#set_property -dict { PACKAGE_PIN J13   IOSTANDARD LVCMOS33 } [get_ports { key_count[1] }]; #IO_L17N_T2_A25_15 Sch=led[2]
#set_property -dict { PACKAGE_PIN N14   IOSTANDARD LVCMOS33 } [get_ports { key_count[2] }]; #IO_L8P_T1_D11_14 Sch=led[3]
#set_property -dict { PACKAGE_PIN R18   IOSTANDARD LVCMOS33 } [get_ports { key_count[3] }]; #IO_L7P_T1_D09_14 Sch=led[4]
#set_property -dict { PACKAGE_PIN V17   IOSTANDARD LVCMOS33 } [get_ports { key_count[4] }]; #IO_L18N_T2_A11_D27_14 Sch=led[5]
#set_property -dict { PACKAGE_PIN U17   IOSTANDARD LVCMOS33 } [get_ports { key_count[5] }]; #IO_L17P_T2_A14_D30_14 Sch=led[6]
#set_property -dict { PACKAGE_PIN U16   IOSTANDARD LVCMOS33 } [get_ports { key_count[6] }]; #IO_L18P_T2_A12_D28_14 Sch=led[7]
#set_property -dict { PACKAGE_PIN V16   IOSTANDARD LVCMOS33 } [get_ports { key_count[7] }]; #IO_L16N_T2_A15_D31_14 Sch=led[8]
```

```
#set_property -dict { PACKAGE_PIN T15   IOSTANDARD LVCMOS33 } [get_ports { LED[9] }]; #IO_L14N_T2_SRCC_14 Sch=led[9]
#set_property -dict { PACKAGE_PIN U14   IOSTANDARD LVCMOS33 } [get_ports { LED[10] }]; #IO_L22P_T3_A05_D21_14 Sch=led[10]
#set_property -dict { PACKAGE_PIN T16   IOSTANDARD LVCMOS33 } [get_ports { LED[11] }]; #IO_L15N_T2_DQS_DOUT_CSO_B_14 Sch=led[11]
#set_property -dict { PACKAGE_PIN V15   IOSTANDARD LVCMOS33 } [get_ports { LED[12] }]; #IO_L16P_T2_CSI_B_14 Sch=led[12]
#set_property -dict { PACKAGE_PIN V14   IOSTANDARD LVCMOS33 } [get_ports { LED[13] }]; #IO_L22N_T3_A04_D20_14 Sch=led[13]
#set_property -dict { PACKAGE_PIN V12   IOSTANDARD LVCMOS33 } [get_ports { LED[14] }]; #IO_L20N_T3_A07_D23_14 Sch=led[14]
set_property -dict { PACKAGE_PIN V11   IOSTANDARD LVCMOS33 } [get_ports { combine }]; #IO_L21N_T3_DQS_A06_D22_14 Sch=led[15]

## RGB LEDs
#set_property -dict { PACKAGE_PIN R12   IOSTANDARD LVCMOS33 } [get_ports { LED16_B }]; #IO_L5P_T0_D06_14 Sch=led16_b
#set_property -dict { PACKAGE_PIN M16   IOSTANDARD LVCMOS33 } [get_ports { LED16_G }]; #IO_L10P_T1_D14_14 Sch=led16_g
#set_property -dict { PACKAGE_PIN N15   IOSTANDARD LVCMOS33 } [get_ports { LED16_R }]; #IO_L11P_T1_SRCC_14 Sch=led16_r
#set_property -dict { PACKAGE_PIN G14   IOSTANDARD LVCMOS33 } [get_ports { LED17_B }]; #IO_L15N_T2_DQS_ADV_B_15 Sch=led17_b
#set_property -dict { PACKAGE_PIN R11   IOSTANDARD LVCMOS33 } [get_ports { LED17_G }]; #IO_0_14 Sch=led17_g
#set_property -dict { PACKAGE_PIN N16   IOSTANDARD LVCMOS33 } [get_ports { LED17_R }]; #IO_L11N_T1_SRCC_14 Sch=led17_r

##7 segment display
set_property -dict { PACKAGE_PIN T10   IOSTANDARD LVCMOS33 } [get_ports { hex[0] }]; #IO_L24N_T3_A00_D16_14 Sch=ca
set_property -dict { PACKAGE_PIN R10   IOSTANDARD LVCMOS33 } [get_ports { hex[1] }]; #IO_25_14 Sch=cb
set_property -dict { PACKAGE_PIN K16   IOSTANDARD LVCMOS33 } [get_ports { hex[2] }]; #IO_25_15 Sch=cc
set_property -dict { PACKAGE_PIN K13   IOSTANDARD LVCMOS33 } [get_ports { hex[3] }]; #IO_L17P_T2_A26_15 Sch=cd
set_property -dict { PACKAGE_PIN P15   IOSTANDARD LVCMOS33 } [get_ports { hex[4] }]; #IO_L13P_T2_MRCC_14 Sch=ce
set_property -dict { PACKAGE_PIN T11   IOSTANDARD LVCMOS33 } [get_ports { hex[5] }]; #IO_L19P_T3_A10_D26_14 Sch=cf
set_property -dict { PACKAGE_PIN L18   IOSTANDARD LVCMOS33 } [get_ports { hex[6] }]; #IO_L4P_T0_D04_14 Sch=cg
set_property -dict { PACKAGE_PIN H15   IOSTANDARD LVCMOS33 } [get_ports { hex[7] }]; #IO_L19N_T3_A21_VREF_15 Sch=dp
set_property -dict { PACKAGE_PIN J17   IOSTANDARD LVCMOS33 } [get_ports { AN[0] }]; #IO_L23P_T3_FOE_B_15 Sch=an[0]
set_property -dict { PACKAGE_PIN J18   IOSTANDARD LVCMOS33 } [get_ports { AN[1] }]; #IO_L23N_T3_FWE_B_15 Sch=an[1]
set_property -dict { PACKAGE_PIN T9    IOSTANDARD LVCMOS33 } [get_ports { AN[2] }]; #IO_L24P_T3_A01_D17_14 Sch=an[2]
set_property -dict { PACKAGE_PIN J14   IOSTANDARD LVCMOS33 } [get_ports { AN[3] }]; #IO_L19P_T3_A22_15 Sch=an[3]
set_property -dict { PACKAGE_PIN P14   IOSTANDARD LVCMOS33 } [get_ports { AN[4] }]; #IO_L8N_T1_D12_14 Sch=an[4]
set_property -dict { PACKAGE_PIN T14   IOSTANDARD LVCMOS33 } [get_ports { AN[5] }]; #IO_L14P_T2_SRCC_14 Sch=an[5]
set_property -dict { PACKAGE_PIN K2    IOSTANDARD LVCMOS33 } [get_ports { AN[6] }]; #IO_L23P_T3_35 Sch=an[6]
set_property -dict { PACKAGE_PIN U13   IOSTANDARD LVCMOS33 } [get_ports { AN[7] }]; #IO_L23N_T3_A02_D18_14 Sch=an[7]
```

实验步骤/过程

首先根据要求写出lab07.v文件

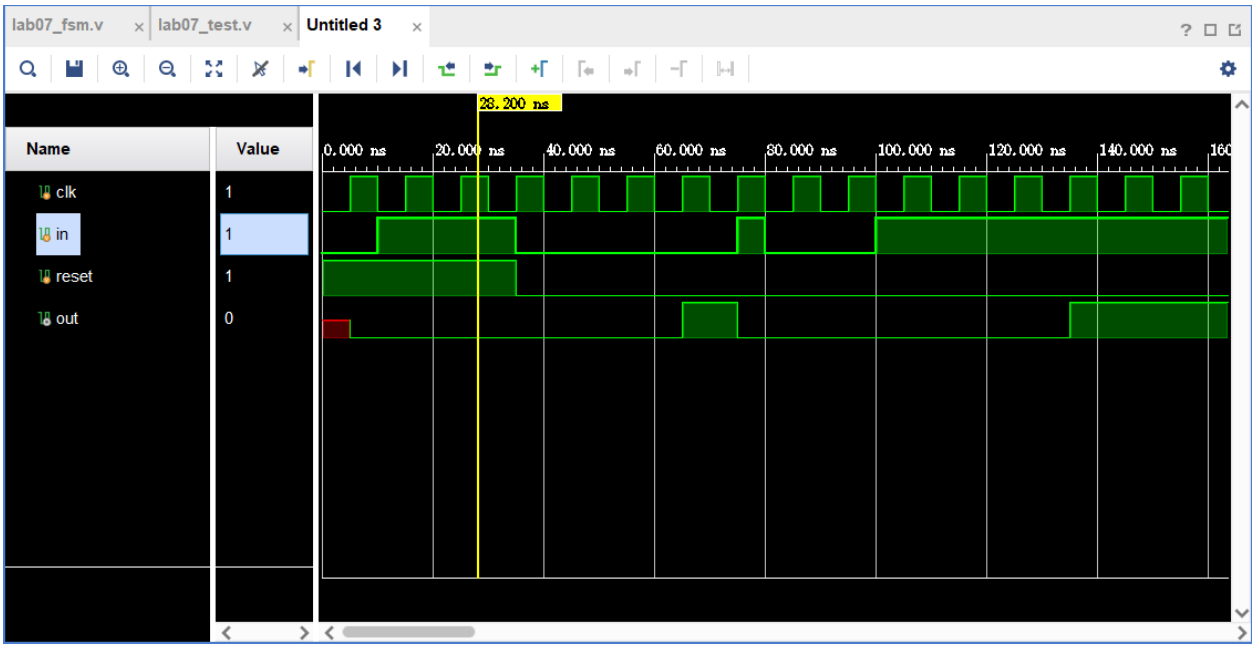
然后写出test.v文件进行仿真模拟

再编写缺省文件连接引脚

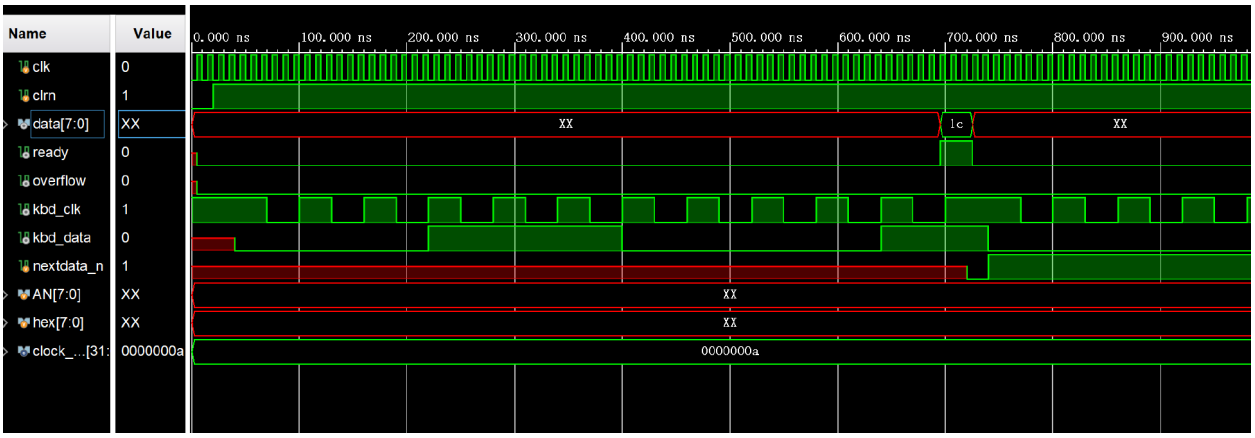
测试方法

利用test bench进行测试，并且在开发板上进行实验。

实验结果



状态机仿真



仿真

实验中遇到的问题及解决办法

各种各样奇怪的报错

根据报错上网搜索/询问同学得到解决方案

实验得到的启示

根据实验指导，我们学习了状态机的工作原理，了解了状态机的编码方式，并利用PS/2键盘输入实现简单状态机的设计。在实验过程中，我们遇到了各种各样的问题，但通过上网搜索和询问同学，我们最终解决了这些问题。通过本次实验，我们深入了解了状态机和PS/2接口控制器及键盘输入的相关知识.

意见和建议