数字电路与数字系统实验

实验八

键盘与状态机

计算机科学与技术系

191220008 陈南曈

924690736@qq.com

2020.11.1

一、实验目的

有限状态机 FSM（Finite State Machine）简称状态机，是一个在有限个状态间进行转换和动作的计算模型。有限状态机含有一个起始状态、一个输入列表（列表中包含了所有可能的输入信号序列）、一个状态转移函数和一个输出端，状态机在工作时由状态转移函数根据当前状态和输入信号确定下一个状态和输出。状态机一般都从起始状态开始，根据输入信号由状态转移函数决定状态机的下一个状态。

有限状态机是数字电路系统中十分重要的电路模块，是一种输出取决于过去输入和当前输入的时序逻辑电路，它是组合逻辑电路和时序逻辑电路的组合。其中组合逻辑分为两个部分，一个是用于产生有限状态机下一个状态的次态逻辑，另一个是用于产生输出信号的输出逻辑，次态逻辑的功能是确定有限状态机的下一个状态；输出逻辑的功能是确定有限状态机的输出。除了输入和输出外，状态机还有一组具有“记忆”功能的寄存器，这些寄存器的功能是记忆有限状态机的内部状态，常被称作状态寄存器。

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。

实验内容：

自行设计状态机，实现单个按键的 ASCII 码显示。

1、基本要求：

①七段数码管低两位显示当前按键的键码，中间两位显示对应的 ASCII 码（转换可以考虑自行设计一个 ROM 并初始化）。只需完成字符和数字键的输入，不需要实现组合键和小键盘。
②当按键松开时，七段数码管的低四位全灭。
③七段数码管的高两位显示按键的总次数。按住不放只算一次按键。只考虑顺序按下和放开的情况，不考虑同时按多个键的情况。

2、高级要求（选做）：

①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。
②支持 Shift 键与字母/数字键同时按下，相互不冲突。
③支持输入大写字符，显示对应的 ASCII 码。

二、实验原理（知识背景）

（一）状态机

1、有限状态机

在实际应用中，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。即 Moore 型有限状态机的输出信号 2 是直接由状态寄存器译码得到。Moore 型有限状态机在时钟 CLK 信号有效后经过一段时间的延迟，输出达到稳定值。即使在这个时钟周期内输入信号发生变化，输出也会在这个完整的时钟周期内保持稳定值而不变。输入对输出的影响要到下一个时钟周期才能反映出来。Moore 有限状态机最重要的特点就是将输入与输出信号隔离开来。Mealy 状态机与 Moore 有限状态机不同，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。

2、状态机的编码方式

上一节例子中的状态机的状态寄存器采用顺序二进制编码 binary 方式，即将状态机的状态依次编码为顺序的二进制数，用顺序二进制数编码可使状态向 量的位数最少。如本例中只需要 4 位二进制数来编码。节省了保存状态向量的逻辑资源。但是在输出时要对状态向量进行解码以产生输出（某个状态有特定的输出，因此输出时要对此状态进行解码，满足状态编号时才可输出特定值），这个解码过程往往需要许多组合逻辑。

另外，当芯片受到辐射或者其他干扰时，可能会造成状态机跳转失常，甚至跳转到无效的编码状态而出现死机。如：状态机因异常跳转到某状态，而此状态需要等待输入，并作出应答，此时因为状态运转不正常，不会出现输入，状态机就会进入死等状态。

One-hot 编码也是状态机设计中常用的编码，在 one-hot 编码中，对于任何给定的状态，其状态向量中只有 1 位是“1”，其他所有位的状态都为“0”，n 个状态就需要 n 位的状态向量，所以 one-hot 编码最长。one-hot 编码对于状态的判断非常方便，如果某位为“1”就是某状态，“0”则不是此状态。因此判断状态输出时非常简单，只要一、两个简单的“与门”或者“或门”即可。

One-hot 编码的状态机从一个状态到另一个状态的状态跳转速度非常快， 而顺序二进制编码从一个状态跳转到另外一个状态需要较多次跳转，并且随着 状态的增加，速度急剧下降。在芯片受到干扰时，one-hot 编码一般只能跳转到无效状态（如果跳到另一有效状态必须是当前为“1”的为变为“0”，同时另外一位变成由“0”变为“1”，这种可能性很小），因此 one-hot 编码的状态机稳定 性高。

格雷码 gray-code 也是状态机设计中常用一种编码方式，它的优点是 gray-code 状态机在发生状态跳转时，状态向量只有 1 位发生变化。

一般而言，顺序二进制编码和 gray-code 的状态机使用了最少的触发器，较多的组合

逻辑，适用于提供更多的组合逻辑的 CPLD 芯片。对于具有更多触发器资源的 FPGA，用 one-hot 编码实现状态机则更加有效。所以 CPLD 多使用 gray-code，而 FPGA 多使用 one-hot 编码。对于触发器资源非常丰富的 FPGA 器件，使用 one-hot 是常用的。

（二）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。

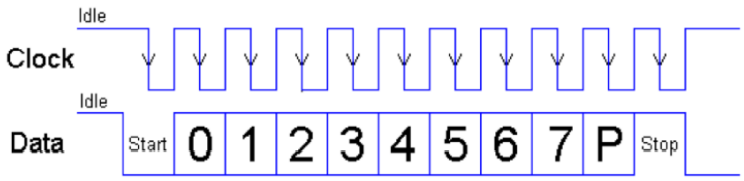


图 8-4: 键盘输出数据时序图

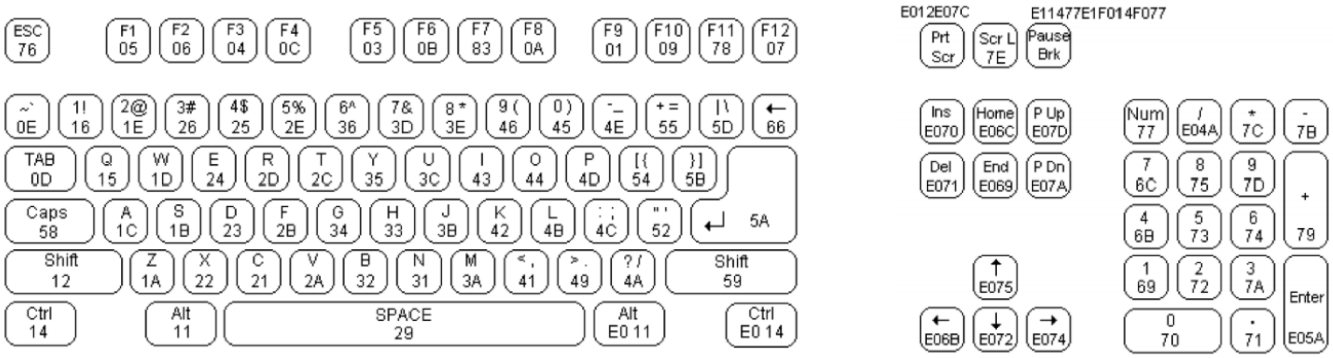
1、PS/2 接口的工作时序

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，每位都在时钟的下降沿有效。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）”。多个键被同时按下时，将逐个输出扫描码。

2、键盘扫描码

每个键都有唯一的通码和断码。键盘所有键的扫描码组成的集合称为扫描 码集。共有三套标准的扫描码集，所有现代的键盘默认使用第二套扫描码。



三、实验环境/器材等

1) 软件环境:

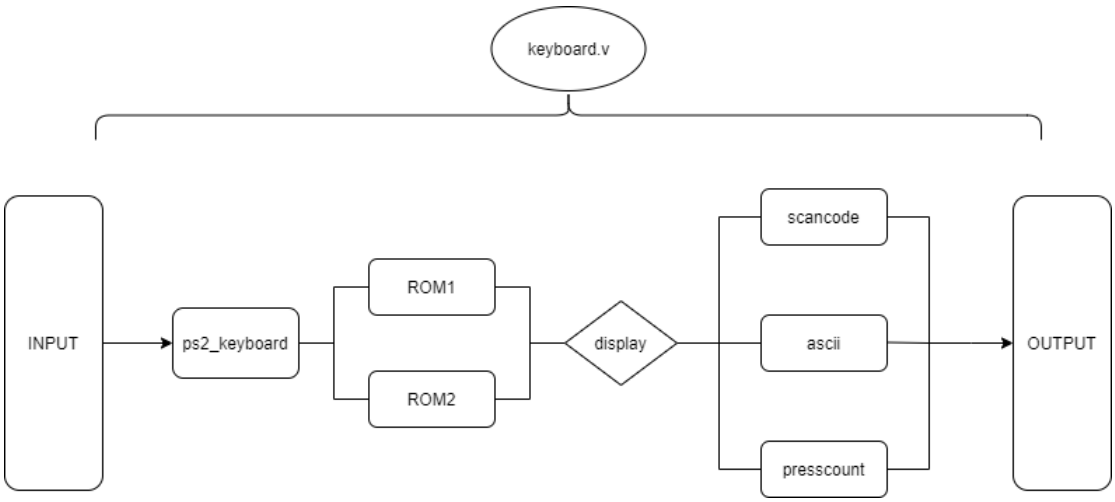
Quartus (Quartus Prime 17.1) Lite Edition

2) 硬件环境:

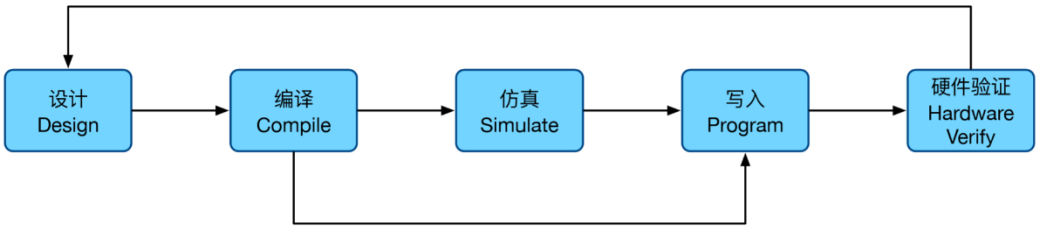
DE10-Standard 开发板
FPGA 部分:
Intel Cyclone V SE 5CSXFC6D6 F31C6N
- 110K 逻辑单元
- 5,761Kbit RAM

HPS 部分:
Dual-core ARM Cortex A9
- 925MHz
- 1GB DDR

四、程序代码或流程图



五、实验步骤/过程



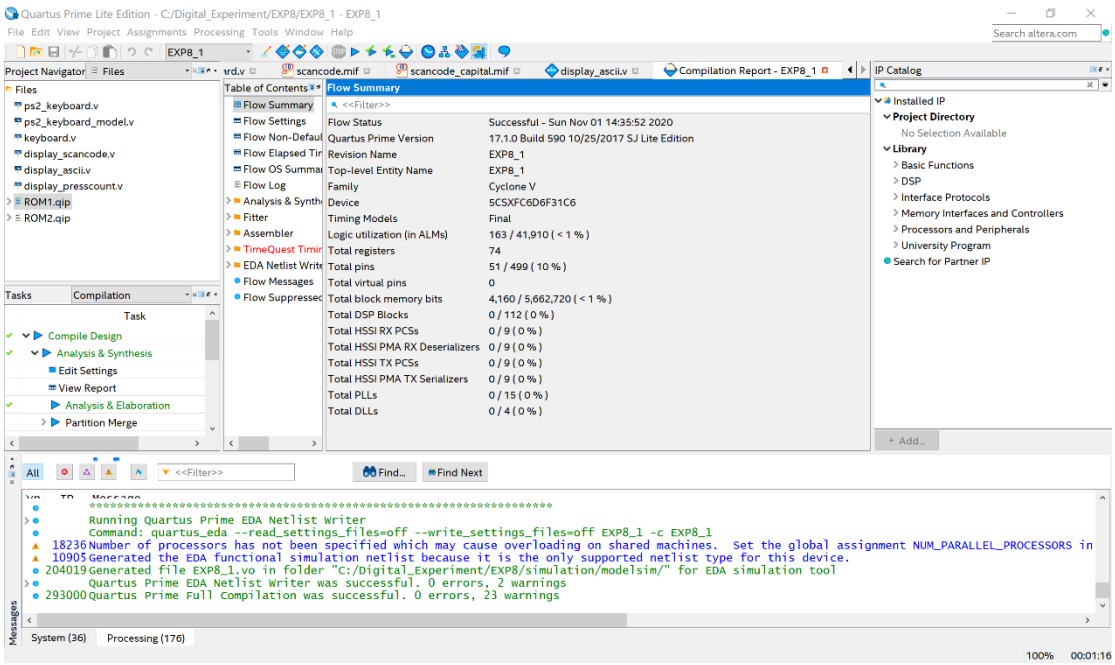
设计：

```
module EXP8_1(
clk, clrn, ps2_clk, ps2_data, ready, overflow, capital, shift, ctrl,
HEX0, HEX1, HEX2, HEX3, HEX4, HEX5,
);
    input clk;
    input clrn;
    input ps2_clk;
    input ps2_data;

    output ready;
    output overflow;
    output [6:0]HEX0;
    output [6:0]HEX1;
    output [6:0]HEX2;
    output [6:0]HEX3;
    output [6:0]HEX4;
    output [6:0]HEX5;

    output reg capital;
    output reg shift;
    output reg ctrl;

    reg capslock;
    reg capslock_flag;

    reg predata; //是否持续按键
    reg nextdata_n;
    wire [7:0] data; //接收从ps2_keyboard中得到的8位键码，作为实例化的wire输出
    reg [7:0] my_data; //接收8位键码
    reg [6:0] press_count; //存储按键次数
    wire [7:0] ascii1; //存储小写ascii码
    wire [7:0] ascii2; //存储大写ascii码

    reg clk_large; //扩大了250倍的时钟
    reg [8:0]count_clk; //用作分频器
```

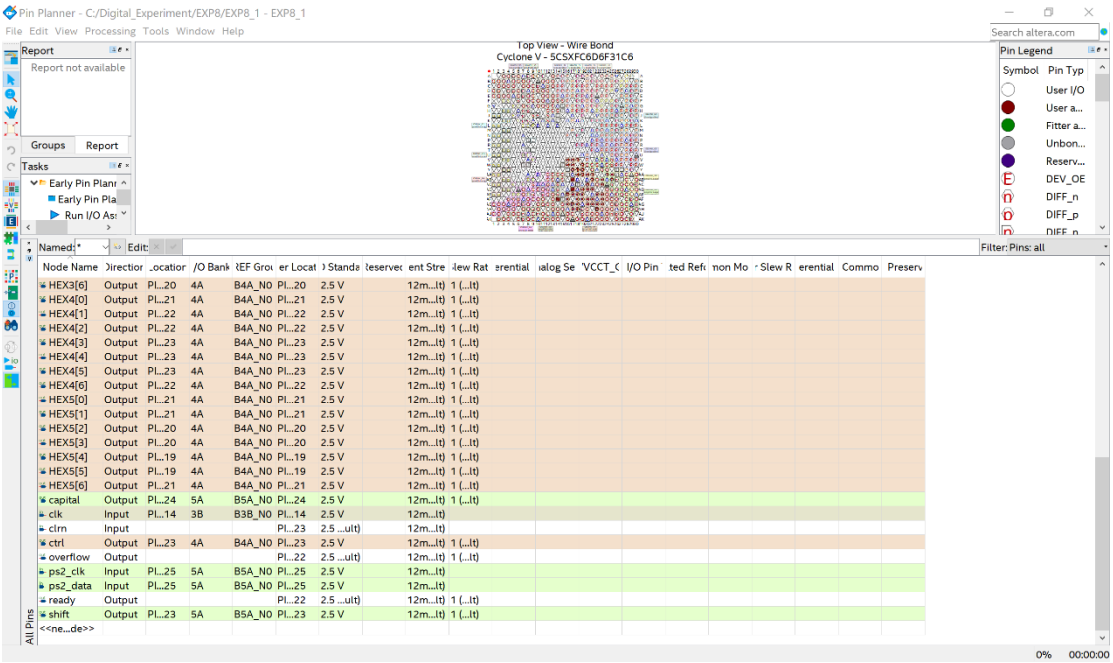
测试：

```
initial begin /* clock driver */
   clk = 0;
   forever
      #(clock_period/2) clk = ~clk;
end

initial begin
   clrn = 1'b0; #20;
   clrn = 1'b1; #20;
   kbd_sendcode(8'h1C); // press 'A'
   #20 nextdata_n =1'b0; #20 nextdata_n =1'b1;//read data
   kbd_sendcode(8'hF0); // break code
   #20 nextdata_n =1'b0; #20 nextdata_n =1'b1; //read data
   kbd_sendcode(8'h1C); // release 'A'
   #20 nextdata_n =1'b0; #20 nextdata_n =1'b1; //read data
   kbd_sendcode(8'h1B); // press 'S'
   #20 kbd_sendcode(8'h1B); // keep pressing 'S'
   #20 kbd_sendcode(8'h1B); // keep pressing 'S'
   kbd_sendcode(8'hF0); // break code
   kbd_sendcode(8'h1B); // release 'S'
   #20;
   $stop;
end

endmodule
```

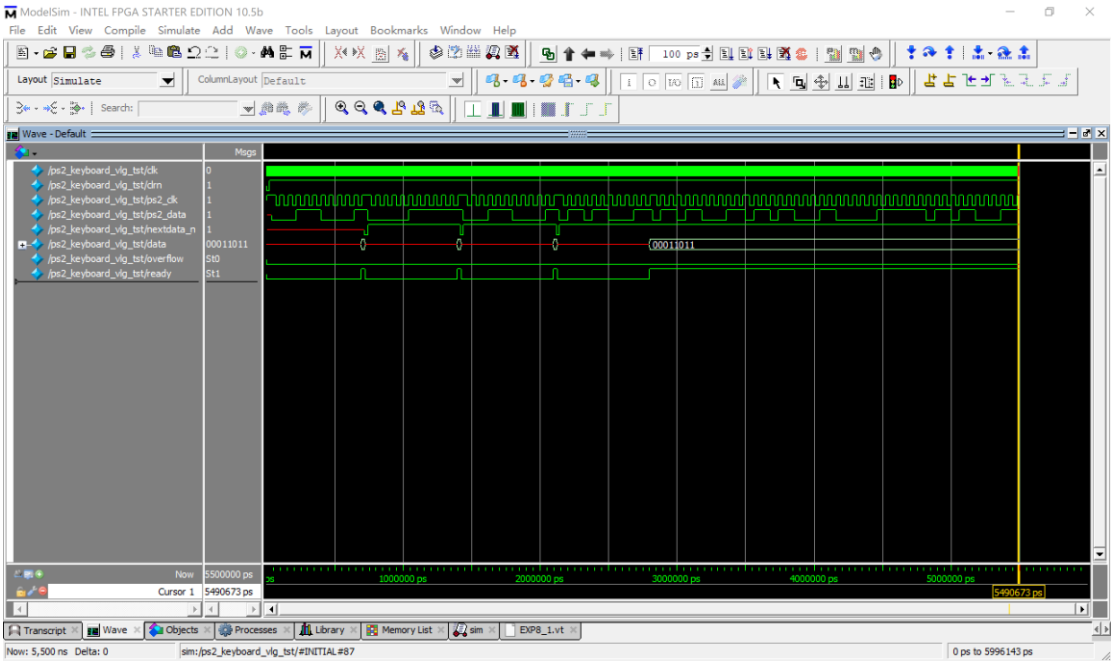
编译：



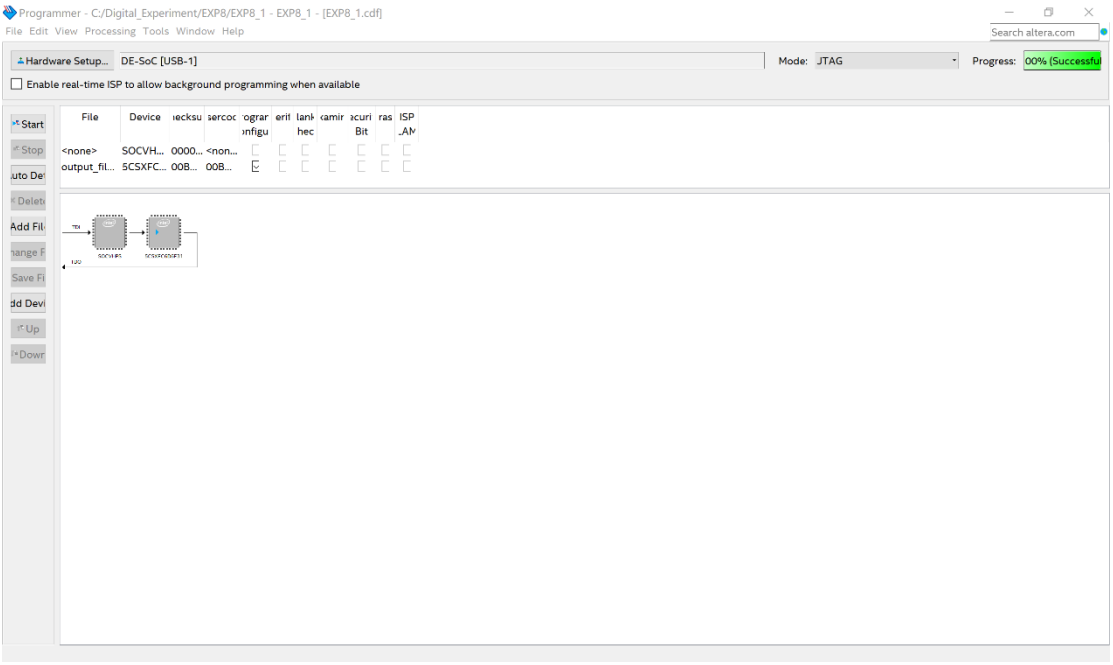
引脚分配:


Pin Planner - C:/Digital_Experiment/EXP8/EXP8_1 - EXP8_1
File Edit View Processing Tools Window Help
Top View - Wire Bond
Cyclone V - 5CSXFC6D6F31C6
Pin Legend
Report not available
Groups Report
Tasks
Node Name
All Pins

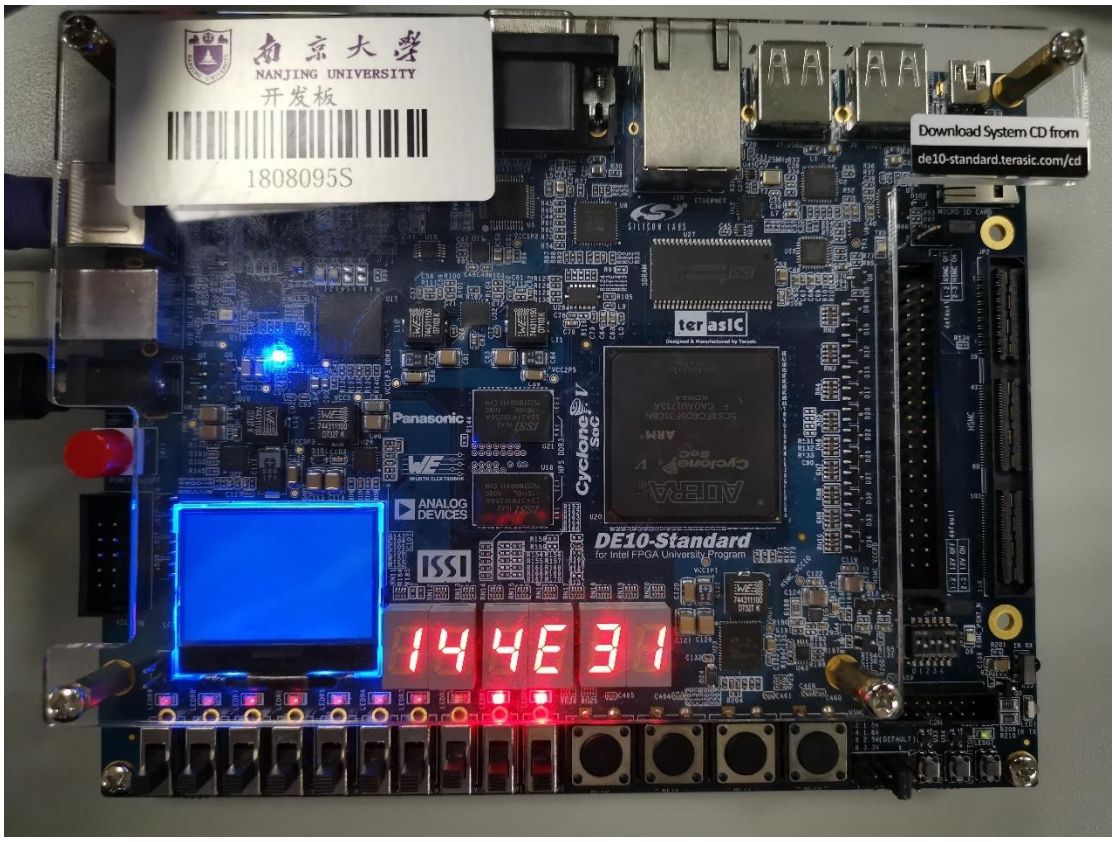
仿真:


ModelSim - INTEL FPGA STARTER EDITION 10.5b
File Edit View Compile Simulate Add Wave Tools Layout Bookmarks Window Help
Wave - Default
/ps2_keyboard_vlg_tst/clk
/ps2_keyboard_vlg_tst/drn
/ps2_keyboard_vlg_tst/ps2_clk
/ps2_keyboard_vlg_tst/ps2_data
/ps2_keyboard_vlg_tst/nextdata_n
/ps2_keyboard_vlg_tst/data
/ps2_keyboard_vlg_tst/overflow
/ps2_keyboard_vlg_tst/ready
00011011
Cursor 1
5490673 ps

写入：


Programmer - C:/Digital_Experiment/EXP8/EXP8_1 - EXP8_1 - [EXP8_1.cdf]
File Edit View Processing Tools Window Help
Search altera.com
Hardware Setup...
DE-SoC [USB-1]
Mode: JTAG
Progress: 00% (Successful
Enable real-time ISP to allow background programming when available
Start
Stop
Add Fil
Save Fi
Up
Down
File
Device
Bit
<none>
SOCVH...
output_fil...
5CSXFC...

硬件验证：


南京大学
NANJING UNIVERSITY
开发板
1808095S
Download System CD from
de10-standard.terasic.com/cd
terasIC
Panasonic
ANALOG DEVICES
ISSI
Cyclone V SoC
DE10-Standard
for Intel FPGA University Program
14 4E 31

六、测试方法

Test Bench

传入'A'和'S'的扫描码，观察 data 变化情况。

```
initial begin /* clock driver */
   clk = 0;
   forever
      #(clock_period/2) clk = ~clk;
end

initial begin
   clrn = 1'b0; #20;
   clrn = 1'b1; #20;
   kbd_sendcode(8'h1C); // press 'A'
   #20 nextdata_n =1'b0; #20 nextdata_n =1'b1;//read data
   kbd_sendcode(8'hF0); // break code
   #20 nextdata_n =1'b0; #20 nextdata_n =1'b1; //read data
   kbd_sendcode(8'h1C); // release 'A'
   #20 nextdata_n =1'b0; #20 nextdata_n =1'b1; //read data
   kbd_sendcode(8'h1B); // press 'S'
   #20 kbd_sendcode(8'h1B); // keep pressing 'S'
   #20 kbd_sendcode(8'h1B); // keep pressing 'S'
   kbd_sendcode(8'hF0); // break code
   kbd_sendcode(8'h1B); // release 'S'
   #20;
   $stop;
end

endmodule
```

七、实验结果

仿真结果、实际结果和预期结果完全一致，能通过键盘输入在数码管上显示对应案件次数、ascii 码和扫描码。同时，按住 shift 或 ctrl 有相应变化，capital 也正常改变。

八、实验中遇到的问题及解决办法

·1、讲义中给出的代码参数很多，难以弄清参数的含义和参数之间的关系。

解决办法：先将讲义中的解释看清楚，然后将过程模块化，列出每个模块需要的输入输出接口，然后和代码中的参数意义对应。

2、不知道讲义中的仿真代码如何写到测试文件中去。

解决办法：在网上搜索了 task 的相关资料，按自己的理解将仿真模型和测试代码写在一起，经过多次的调试后，成功编译。

九、实验得到的启示

1、越是复杂的问题，越是需要进行模块化处理，然后针对每个模块进行逻辑上的梳理，最后通过模块之间的关系，将模块间的接口连接起来，得到最终的结果。

2、对于适用 test bench 的实验，要多利用仿真进行检验，并充分考虑多种情况来编写测试代码，针对实验的需求，有针对性的给出一些输入。

十、意见和建议

1、希望对于测试代码的写法给出一些框架上的提示，不然不太明白该如何组织讲义中给出的测试代码。

2、希望可以给出一些较为复杂的实验的效果展示小视频或图片示例，不然难以判断自己的理解是否有问题，可能会导致做无用功。

3、提供的 mif 文件里部分按键的 ascii 码有问题，希望可以修正。同时，希望可以明确指出按住 shift 会改变 ascii 码的按键。