数字电路与数字系统实验

实验十一 字符输入界面

姓名：

学号：

班级：

邮箱：

实验时间：2020.12.6

一、 实验目的

本实验将利用前面实现过的键盘和显示器功能来搭建一个简单的字符输入界面，通过该系统的实现深入理解多个模块之间的交互和接口的设计。

二、 实验原理

字符显示：
每个字符高为 16 个点，宽为 9 个点。因此单个字符可以用 6 个 9bit 数来表示，每个 9bit 数代表字符的一行，因此，我们只需要 256×16×9≈37kbit 的空间即可存储整个点阵。

显存读写：
有了字符点阵后，系统就不再需要记录屏幕上每个点的颜色信息了，只需要记录屏幕上显示的 ASCII 字符即可。在显示时，根据当前屏幕位置，确定应该显示那个字符，再查找对应的字符点阵即可完成显示。对于 640×480 的屏幕，可以显示 30 行（30×16=480）， 70 列（70 × 9=630）的 ASCII 字符。系统的显存只需要 30 × 70 大小，每单元存储 8bit 的 ASCII 字符即可。这样，我们的字符显存只需要 2.1kByte。

扫描显示：
1. 根据当前扫描位置，获取对应的字符的 x,y 坐标，以及扫描到单个字符点阵内的行列信息
2. 根据字符的 x,y 坐标，查询字符显存，获取对应 ASCII 编码
3. 根据 ASCII 编码和字符内的行信息，查询点阵 ROM，获取对应行的 9bit 数据
4. 根据字符内的列信息，取出对应的 bit，并根据该 bit 设置颜色。此处可以显示黑底白字或其他彩色字符，只需要按自己的需求分别设置背景颜色和字符颜色即可。

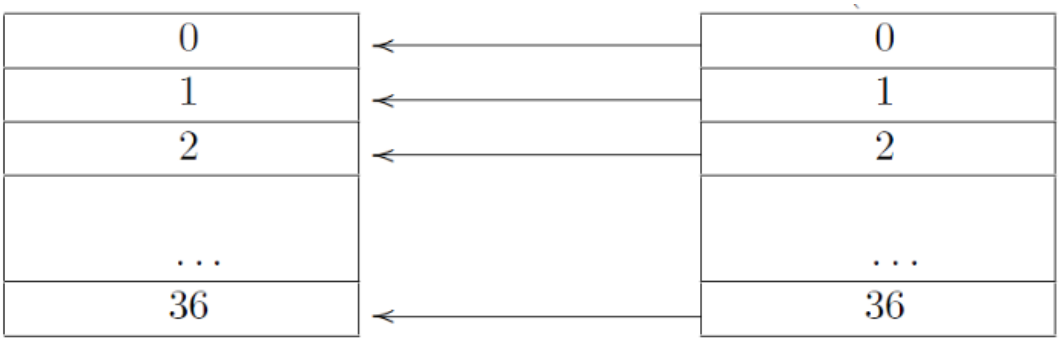
三、 实验环境

Quartus 18.1、FPGA 开发板

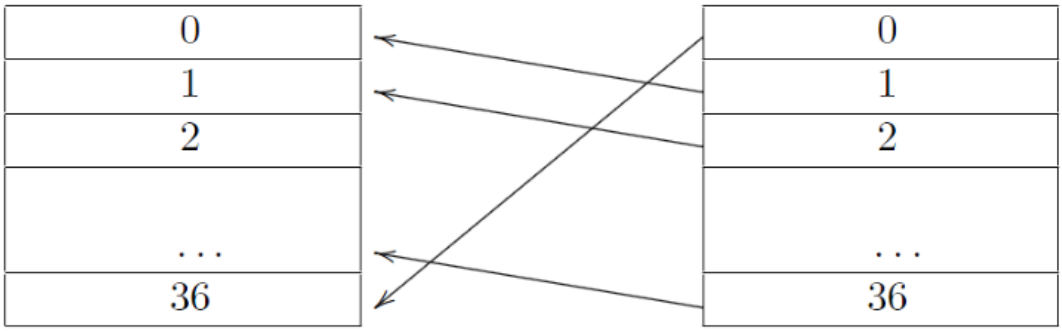
四、 实验过程

设计思路：
显存读写：设置 x，y 来存储当前的字符的坐标， x=h_addr/9，y=v_addr>>4，Ram 地址为 x+(y<<6)+(y<<2)+(y<<1)。x，y 偏移量 offset_x 为 h_addr-(x<<3)-x，offset_y 为 v_addr-(y<<4)。字符点阵 rom_font 读地址 romfont_addr 为{ascii,4'b0000}+offset_y，font_data 为 rom_font[romfont_addr]。通过 font_data[offset_x]从一行字符点阵中读取每一个点对应的 bit。
大写：设置一个 cap 标志位，每按下一次大写键 cap 就取反，cap=1 时即为大

写，将字母的 ascii 码减去 0x20
彩色显示：改变 vga_data 的值即可
自动滚屏：我设想的是写满一屏幕之后，改变读地址和写地址的映射关系，将所有行都能往上移一行，如下图所示。但是由于本人能力有限，没有实现这个功能…



写满之后改为：



设计代码：
ps2_keyboard.v

```
    reg [7:0] rom_ascii[255:0];
    initial begin
        $readmemh("C:/intelFPGA_lite/exp/exp11/rom_ascii.txt",rom_ascii,0,255);
    end

    show S0(1'b1,waddr[3:0],hex0);
    show S1(1'b1,waddr[7:4],hex1);
    show S2(1'b1,out_ascii[3:0],hex2);
    show S3(1'b1,out_ascii[7:4],hex3);

    always @(posedge clk) begin
        ps2_clk_sync <= {ps2_clk_sync[1:0],ps2_clk};
    end

    wire sampling = ps2_clk_sync[2] & ~ps2_clk_sync[1];

    always @(posedge clk)
    begin
        if (sampling) begin
            if (count == 4'd10) begin
                if ((buffer[0] == 0) && // start bit
                    (ps2_data) && // stop bit
                    (^buffer[9:1])) begin // odd parity
                    data <= buffer[8:1]; // kbd scan code
                    temp <= data;
                    if(cap) ascii<=rom_ascii[buffer[8:1]]-8'h20;
                    else ascii<=rom_ascii[buffer[8:1]];
                    if(buffer[8:1]==8'h58)begin
```

```
                temp <= data;
                if(cap) ascii<=rom_ascii[buffer[8:1]]-8'h20;
                else ascii<=rom_ascii[buffer[8:1]];
                if(buffer[8:1]==8'h58)begin
                   cap<=~cap;
                end
                else if(buffer[8:1]==8'h5a)begin //enter
                   row=(row==29?0:row+1);
                   col<=0;
                   waddr<=row*70+col;
                end
                else
                begin
                   waddr<=row*70+col;
                   if(col==69)
                   begin
                      col<=0;
                      row<=(row==29?0:row+1);
                   end
                   else col<=col+1;
                end
             end
             else data<=buffer[8:1];
             count <= 0; // for next
          end
          else begin
             buffer[count] <= ps2_data; // store ps2_data
             count <= count + 3'b1;
          end
       end
    end
```

vga.v

```
 9   reg [7:0] ram_vga[2099:0];
10   reg [11:0] rom_font[4095:0];
11   initial begin
12      $readmemh("C:/intelFPGA_lite/exp/exp11/vga_font.txt",rom_font,0,4095);
13   end
14
15   wire [9:0] x,y;
16   wire [11:0] font_data;
17   wire [11:0] ramaddr;
18   wire [7:0] ascii;
19   wire [11:0] romfont_addr,offset_x,offset_y;
20   assign x=h_addr/9;
21   assign y=v_addr>>4;
22   assign ramaddr=x+(y<<6)+(y<<2)+(y<<1);
23   assign ascii=ram_vga[ramaddr];
24   assign offset_x=h_addr-(x<<3)-x;
25   assign offset_y=v_addr-(y<<4);
26   assign romfont_addr={ascii,4'b0000}+offset_y;
27   assign font_data=rom_font[romfont_addr];
28
29   always@(posedge clk)begin
30      if(asciires!=8'h0) ram_vga[waddr]<=asciires;
31
32      if(h_addr >= 630) vga_data = 24'h0;
33      else if(font_data[offset_x]==1'b1) vga_data = 24'hffffff;
34      else vga_data = 24'h0;
35   end
36
37   endmodule
38
```

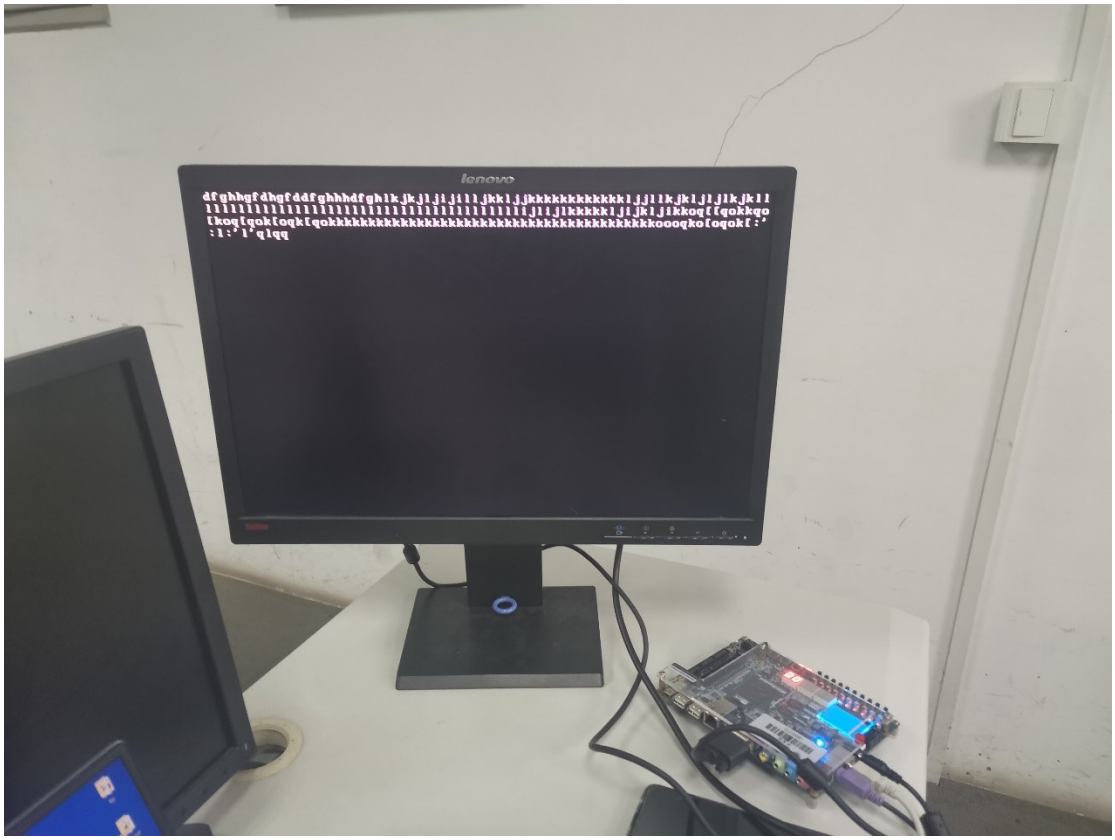
vga_crtl.v 没有什么改变

exp11.v

```
47   //=======================================================
48   //  REG/WIRE declarations
49   //=======================================================
50   assign VGA_SYNC_N=0;
51   assign HEX4=7'b1111111;
52   assign HEX5=7'b1111111;
53   wire [7:0] asciires;
54   wire [11:0] waddr;
55   wire [23:0] vga_data;
56   wire [9:0] h_addr,v_addr;
57
58   //=======================================================
59   //  Structural coding
60   //=======================================================
61   clkgen #(25000000) vclk(CLOCK_50,1'b0,1'b1,VGA_CLK);
62   ps2_keyboard pk(CLOCK_50,PS2_CLK,PS2_DAT,asciires,waddr,HEX0,HEX1,HEX2,HEX3);
63   vga_ctrl vc(VGA_CLK,1'b0,vga_data,h_addr,v_addr,VGA_HS,VGA_VS,VGA_BLANK_N,
64   VGA_R,VGA_G,VGA_B);
65   vga v(VGA_CLK,asciires,waddr,h_addr,v_addr,vga_data);
66
67   endmodule
```

ModelSim 仿真：
没有进行这一步，直接在开发板上进行调试

实验结果：



五、 实验中遇到的问题及解决办法

1、

如果用寄存器来实现所需的存储器而不用片上的 M10K 或 MLAB 有可能会占用大量资源，造成 FPGA 资源紧张，编译时间大大增加。

编译完了才看到这个提示…确实编译了非常久…

2、 一开始写的时候思路不清晰，导致模块很混乱，后来认真读了实验手册理清楚思路之后再写就好很多

3. 大写的实现还有点问题

六、 启示

一定要想好了再写代码，不然容易思路混乱，反复修改，调试要有耐心，尽量每种情况都考虑到…

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。