一、实验内容

1. 项目内容概括：利用 PS/2 键盘、音响和 Nexys4 开发板完成一架电子琴的设计。

2. 键位设置：采用 3 组全音阶，即加入半音，模拟电子琴的黑白键。每组全音阶 12 个音，共计排布 36 个键位。

3. 实现功能：按下键盘上不同的键位后，通过连接到开发板的单声道音频输出口的音响可以发出对应音调的优美乐声，并且将所按下的键的扫描码在开发板的数码管上显示，同时伴随着 LED 灯的动态效果

4. 工作原理：通过开发板上的 USB 接口和键盘建立主机-从机关系，辅助功能微控制器将 USB HID 协议隐藏在 FPGA 上，并仿效老式 PS/ 2 总线。微控制器的行为就像 PS/2 键盘或鼠标一样。每次按下一个键盘，从机（键盘）将相应的扫描码发送给主机（开发板），主机在接收到特定的扫描码后，根据十二平均律得到的音调-频率对照表可以计算出对应的音调所需要的分频比，将 Nexys4 开发板的时钟晶振进行分频，将分频之后的时钟信号接到 Nexys4 的单声道输出接口上，产生特定频率的方波，将方波连到耳机或者音响的 3.5mm 插头上即可发出相应的音调的声音。

5. 器件介绍

①Nexys 4 DDR Artix-7——由 Xilinx 公司开发出的一款现场可编程门阵列（FPGA）

②Lenovo KBBH21 键盘——使用 PS/2 协议的一款键盘，额定工作电压 5V，额定工作电流 100mA

③索爱音响——用于输出并放大音频的设备

键位	音调	键位	音调	键位	音调
1	低音1	Q	中音1	A	高音1
2	低音#1	W	中音#1	S	高音#1
3	低音2	E	中音2	D	高音2
4	低音#2	R	中音#2	F	高音#2
5	低音3	T	中音3	G	高音3
6	低音4	Y	中音4	H	高音4
7	低音#4	U	中音#4	J	高音#4
8	低音5	I	中音5	K	高音5
9	低音#5	O	中音#5	L	高音#5
0	低音6	P	中音6	;	高音6
-	低音#6	[	中音#6	,	高音#6
+	低音7	]	中音7	ENTER	高音7

图 1.1 键位-音调对照表

音阶		Octave0	Octave1	Octave2	Octave3
Do	C	262	523	1047	2093
	Db	277	554	1109	2217
Re	D	294	587	1175	2349
	Eb	311	622	1245	2489
Mi	E	330	659	1329	2637
Fa	F	349	698	1397	2794
	Gb	370	740	1480	2960
Sol	G	392	784	1568	3136
	Ab	415	831	1661	3322
La	A	440	880	1760	3520
	Bb	466	923	1865	3729
Si	B	494	988	1976	3951

图 1.2 音调-频率对照表

二、电子琴数字系统总框图

通过分析我们所需要实现的功能，即：

①通过敲击 PS/2 键盘，使 Nexys4 开发板接收到键盘扫描码

②将键盘扫描码实时记录显示在 Nexys4 开发板的数码管上

③通过获得的键盘扫描码获取相应的分频比

④根据特定的分频比将 100MHz 时钟晶振分频为我们所需要的声音频率

⑤根据获得的键盘码控制 LED 灯的亮灭

可以得到如下的系统总框图：

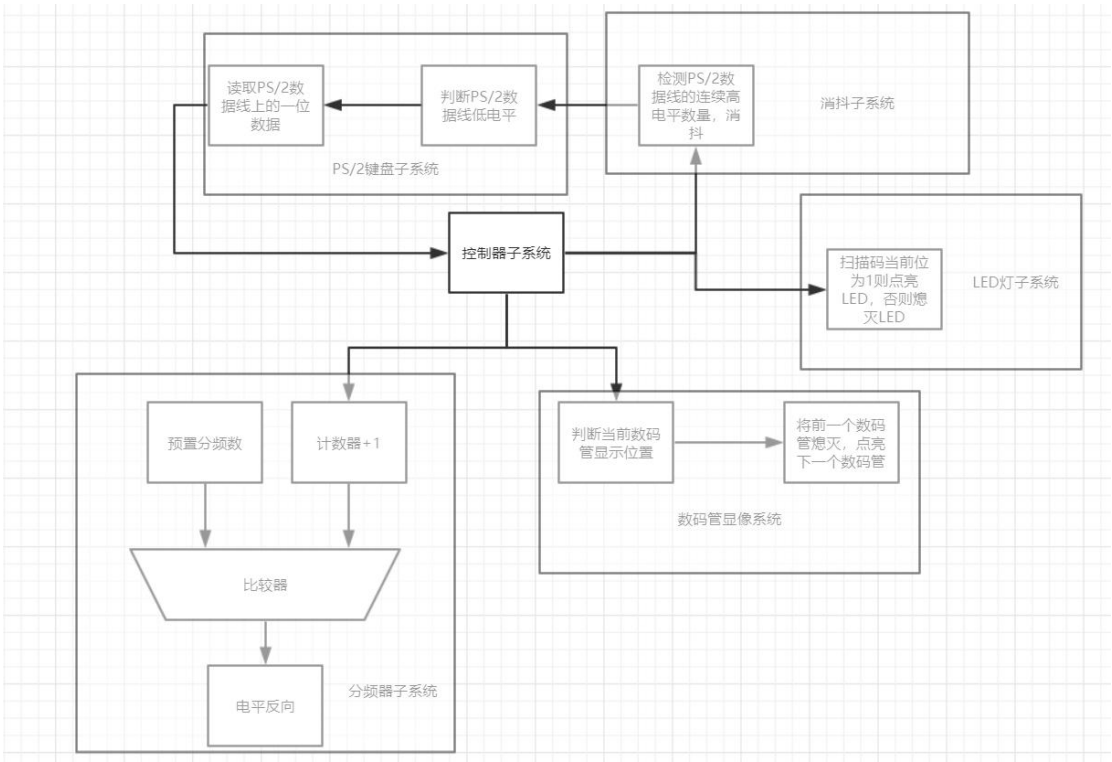


图 2.1 电子琴系统功能总框图

大致可以说明系统的工作工作原理。

RTL 图片如下图所示：

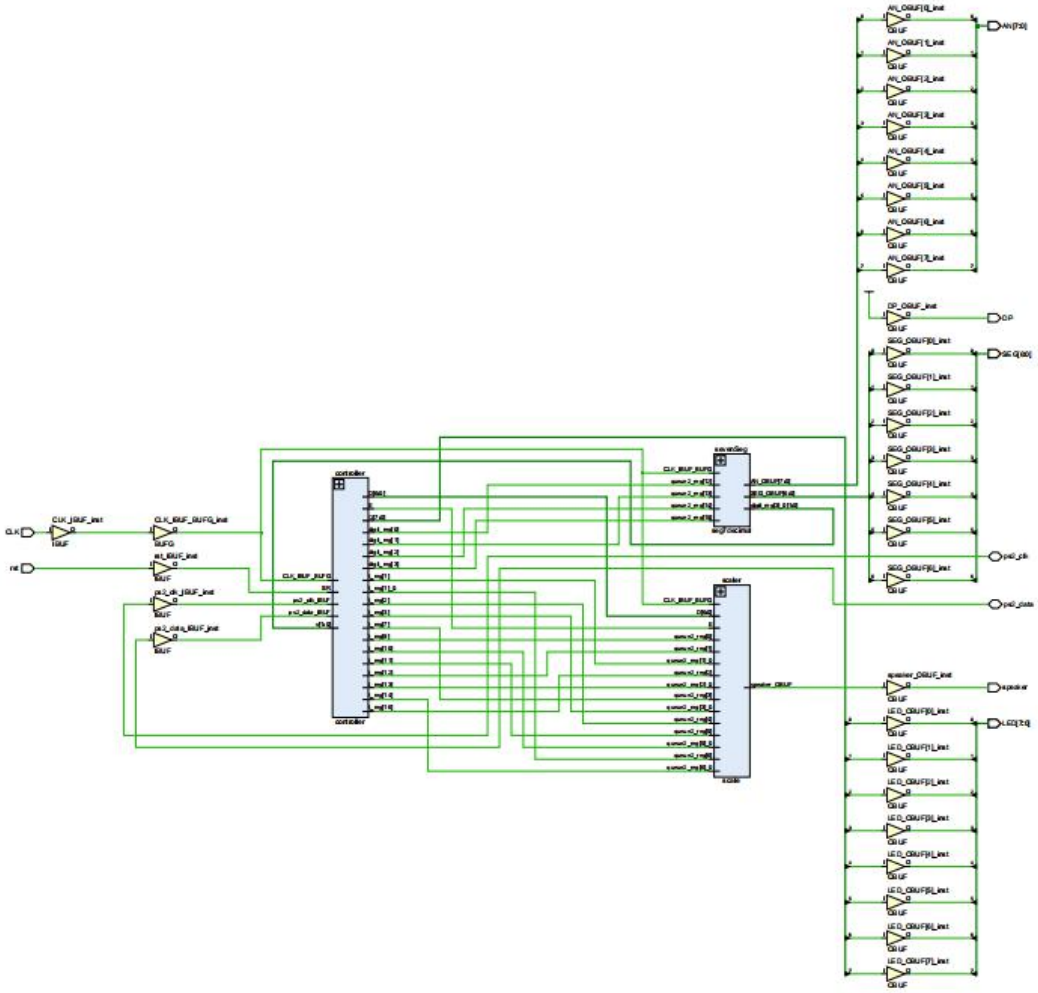


图 2.2 电子琴系统功能总框图

详细模块参考后面截图。

三、系统控制器设计

1.ASM 流程图

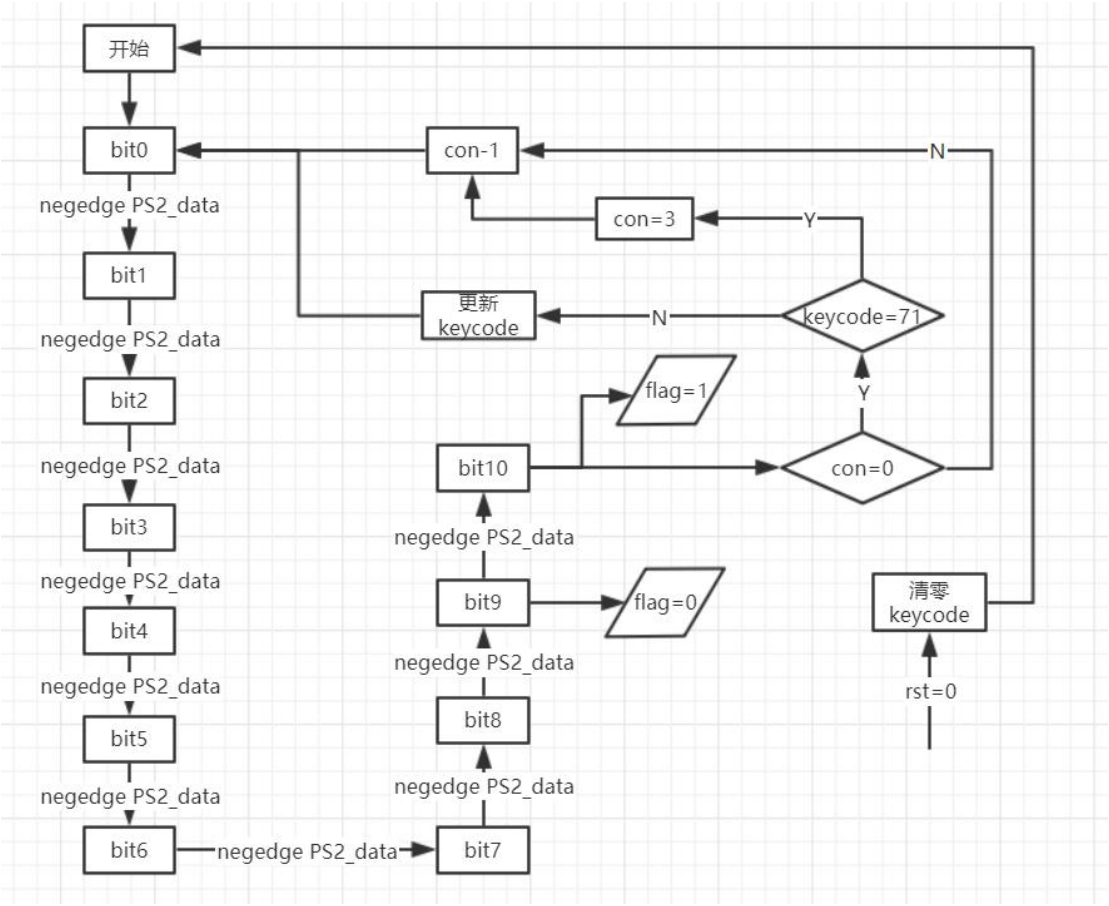


图 3.1 电子琴系统控制器 ASM 流程图

四、子系统模块建模

1. top 顶层模块

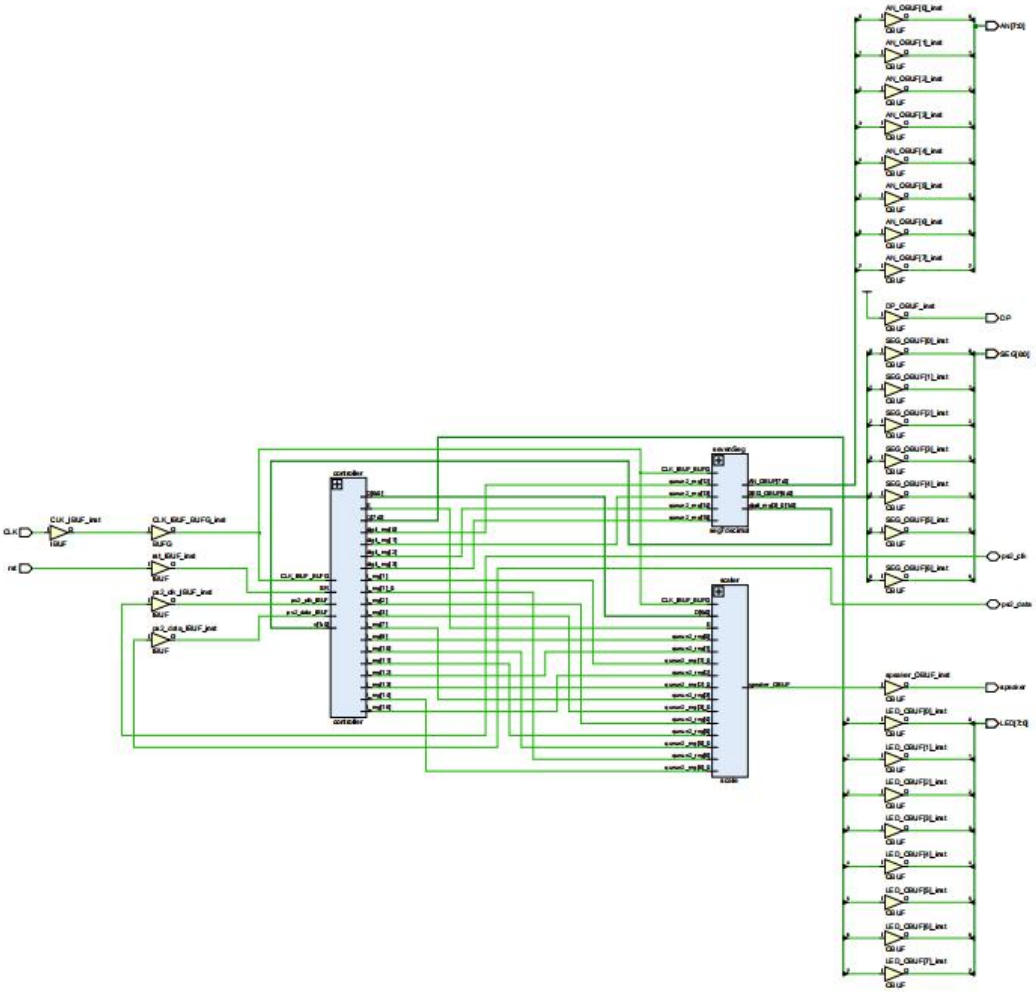
顶层模块，连接各个子模块，列出了所有的与外部交流的数据。

```
module top(CLK,ps2_clk,ps2_data,rst,SEG,AN,DP,LED,speaker);
input CLK;
inout ps2_clk;//PS/2 协议的时钟信号
inout ps2_data;//PS/2 协议的数据信号
input rst;//清零信号
output [6:0] SEG;//单个数码管显像信号
output [7:0] AN;//数码管编号信号
output DP;
output [7:0] LED;//LED 亮灭信号
```

```
output speaker;//音频输出信号
wire [31:0] keycode;//键盘码
wire [31:0] queue;
wire [31:0] con;
integer i;
wire f;

assign read=1;
assign piano=1;
assign display=1;

controller controller(//连接控制器和 PS/2 通信
.read(read),
.clk(CLK),
.kclk(ps2_clk),
.kdata(ps2_data),
.rst(rst),
.keycodeout(keycode[31:0]),
.led(LED),
.queue(queue[15:0]),
.f(f)
);

scale scaler(//连接分频器和输出音频
.piano(piano),
.keycode(keycode),
.clk(CLK),
.key(LED),
.speaker(speaker)
);
```

```
seg7decimal sevenSeg(//连接数码管显像
.display(display),
.x(keycode[31:0]),
.clk(CLK),
.seg(SEG[6:0]),
.an(AN[7:0]),
.dp(DP)
);

Endmodule
```

功能框图：



2.PS/2 通信模块

从 PS/2 设备向主机发送一个字节可按照下面的步骤进行：

(1)检测时钟线电平，如果时钟线为低，则延时 5 0μs；

(2)检测判断时钟信号是否为高，为高，则向下执行，为低，则转到(1)；

(3)检测数据线是否为高，如果为高则继续执行，如果为低，则放弃发送（此时ＰＣ机在向 ＰＳ／２设备发送数据，所以ＰＳ／２设备要转移到接收程序处接收数据）；

(4)延时２０μs（如果此时正在发送起始位，则应延时４０μs）；

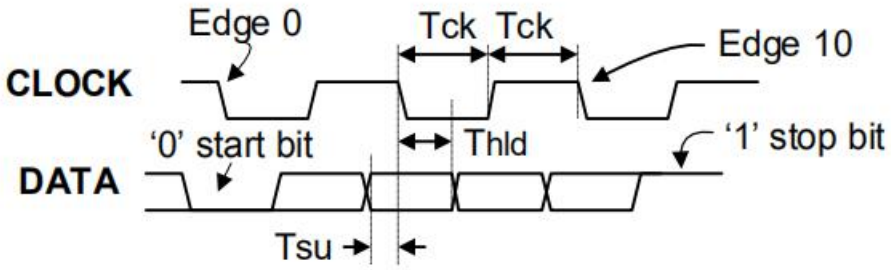
（５）输出起始位（０）到数据线上。这里要注意的是：在送出每一位后都要检测时钟线，以确保ＰＣ机没有抑制ＰＳ／２设备，如果有则中止发送；

（６）输出８个数据位到数据线上；

（７）输出校验位；

（８）输出停止位（１）；

（９）延时３０μｓ（如果在发送停止位时释放时钟信号则应延时５０μｓ）；



Symbol	Parameter	Min	Max
T_{CK}	Clock time	30us	50us
T_{SU}	Data-to-clock setup time	5us	25us
T_{HLD}	Clock-to-data hold time	5us	25us

通过以下步骤可发送单个位：

（１）准备数据位（将需要发送的数据位放到数据线上）；

（２）延时２０μｓ；

（３）把时钟线拉低；

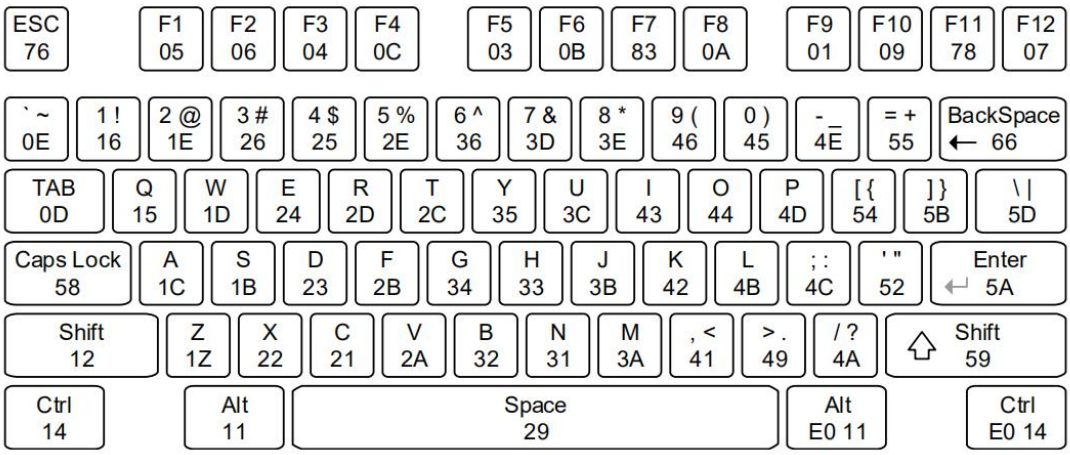
（４）延时４０μｓ；

（５）释放时钟线；

（６）延时２０μｓ。

PS/2 型键盘使用扫描码传送按键数据。为每个按键分配一个发送的代码每当按下键。如果按住键，扫描代码将重复发送，大约每一次 100 毫秒。当释放钥匙时，发送一个 F0 向上键代码，然后是释放钥匙的扫描代码。如果一

个按键可以通过按住 Shift 键以产生新字符（如大写字母），那么除了扫描之外还发送移位字符。本实验所采用的键盘的扫描码集合如下图所示：

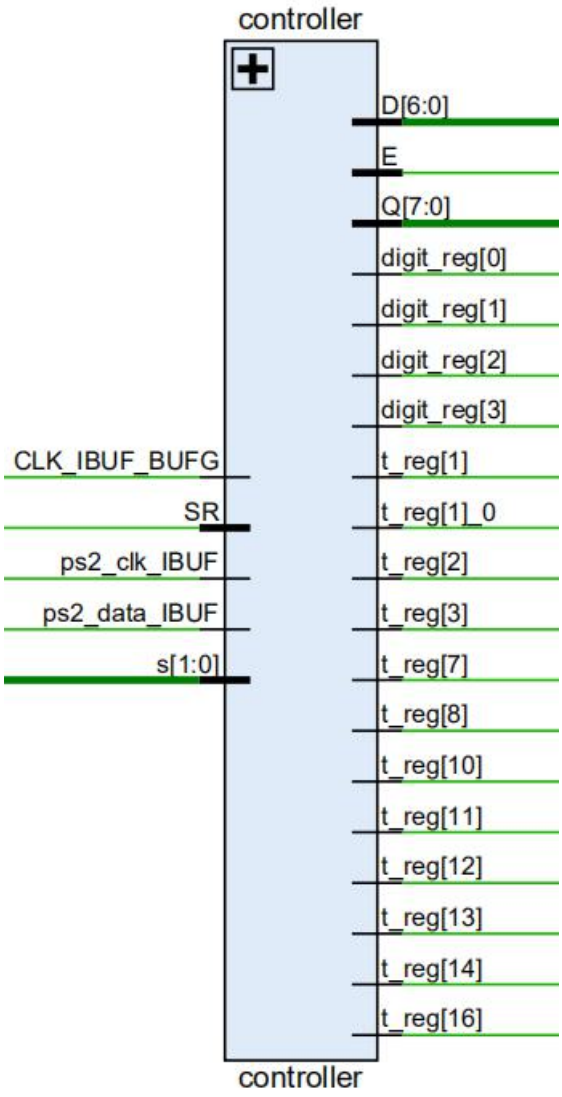



```
module controller(read,clk,kclk,kdata,rst,keycodeout,led,queue,f);
input read;//允许键盘读入信号
input clk;//时钟信号
input kclk;//PS/2 时钟信号
input kdata;//PS/2 数据信号
input rst;//清零信号
output [31:0] keycodeout;//键盘码
output [7:0] led;//les 信号
output [31:0] queue;
output f;
reg [31:0] c;
wire kclkf, kdataf;
reg [7:0] datacur;
reg [7:0] dataprev;
reg [3:0] cnt=0;
reg [31:0] keycode=0;
reg flag=0,flag_old=0,fl;//记录是否读完一次数据包的标志信号
reg [31:0] queue2=0;
integer con;
```

```
debouncer debounce(.clk(clk),.I0(kclk),.I1(kdata),.O0(kclkf),.O1(kdataf));

initial begin
   fl=f;
   con=0;
end

always@(negedge kclkf) begin
   case(cnt)
      0:;//获得第 1 位信号
      1:datacur[0]<=kdataf;//获得第 2 位信号
      2:datacur[1]<=kdataf;//获得第 3 位信号
      3:datacur[2]<=kdataf;//获得第 4 位信号
      4:datacur[3]<=kdataf;//获得第 5 位信号
      5:datacur[4]<=kdataf;//获得第 6 位信号
      6:datacur[5]<=kdataf;//获得第 7 位信号
      7:datacur[6]<=kdataf;//获得第 8 位信号
      8:datacur[7]<=kdataf;//获得第 9 位信号
      9:flag<=1'b1;//获得第 10 位信号
      10:flag<=1'b0;//获得第 11 位信号
   endcase
   if(cnt<=9) cnt<=cnt+1; else cnt<=0;
end

always @(posedge clk ) begin
   flag_old<=flag;
   if (rst==1) begin keycode<=0;con<=0;end
      else begin
         if ((flag_old==0)&&(flag==1)&&(read==1)) begin
```

```
            if ((con==0)&&(datacur==8'h71)) con=3;//过滤掉“.”键位，保持通信顺畅
            if (con==0) begin
               keycode[31:24]<=keycode[23:16];//数码管显像区第 1 个键盘码
               keycode[23:16]<=keycode[15:8];//数码管显像区第 2 个键盘码
               keycode[15:8]<=dataprev;//数码管显像区第 3 个键盘码
               keycode[7:0]<=datacur;//数码管显像区第 4 个键盘码
               dataprev<=datacur;//记录上一个键盘码
            end
            else con=con-1;//“.”键按下释放会产生 3 个键盘码，一个个过滤
         end
      end
end

assign keycodeout=keycode;
assign led=keycode[7:0];
assign queue=queue2;
assign f=fl;

Endmodule
```

功能框图：



3.音调产生模块

```
module scale(piano,keycode,clk,key,speaker);
input piano;
input [31:0] keycode;//键盘码
input clk;//时钟信号
input [7:0] key;//最后一个键盘码
output speaker;//音频信号
reg sound,fl;
integer count,t,p,f;
```

```
initial f=1;

always @(posedge clk)
        if (keycode[7:0]==8'hf0) f<=0;
          else begin
              case (keycode[7:0]) //根据最后一个键盘码，获取相应的分频数
                  8'h16:t<=190839;
                  8'h1e:t<=180505;
                  8'h26:t<=170068;
                  8'h25:t<=160771;
                  8'h2e:t<=151515;
                  8'h36:t<=143266;
                  8'h3d:t<=135135;
                  8'h3e:t<=127551;
                  8'h46:t<=120481;
                  8'h45:t<=113636;
                  8'h4e:t<=107296;
                  8'h55:t<=101214;
                  8'h15:t<=95602;
                  8'h1d:t<=90252;
                  8'h24:t<=85178;
                  8'h2d:t<=80385;
                  8'h2c:t<=75872;
                  8'h35:t<=71633;
                  8'h3c:t<=67567;
                  8'h43:t<=63775;
                  8'h44:t<=60168;
                  8'h4d:t<=56818;
                  8'h54:t<=54171;
                  8'h5b:t<=50607;
```

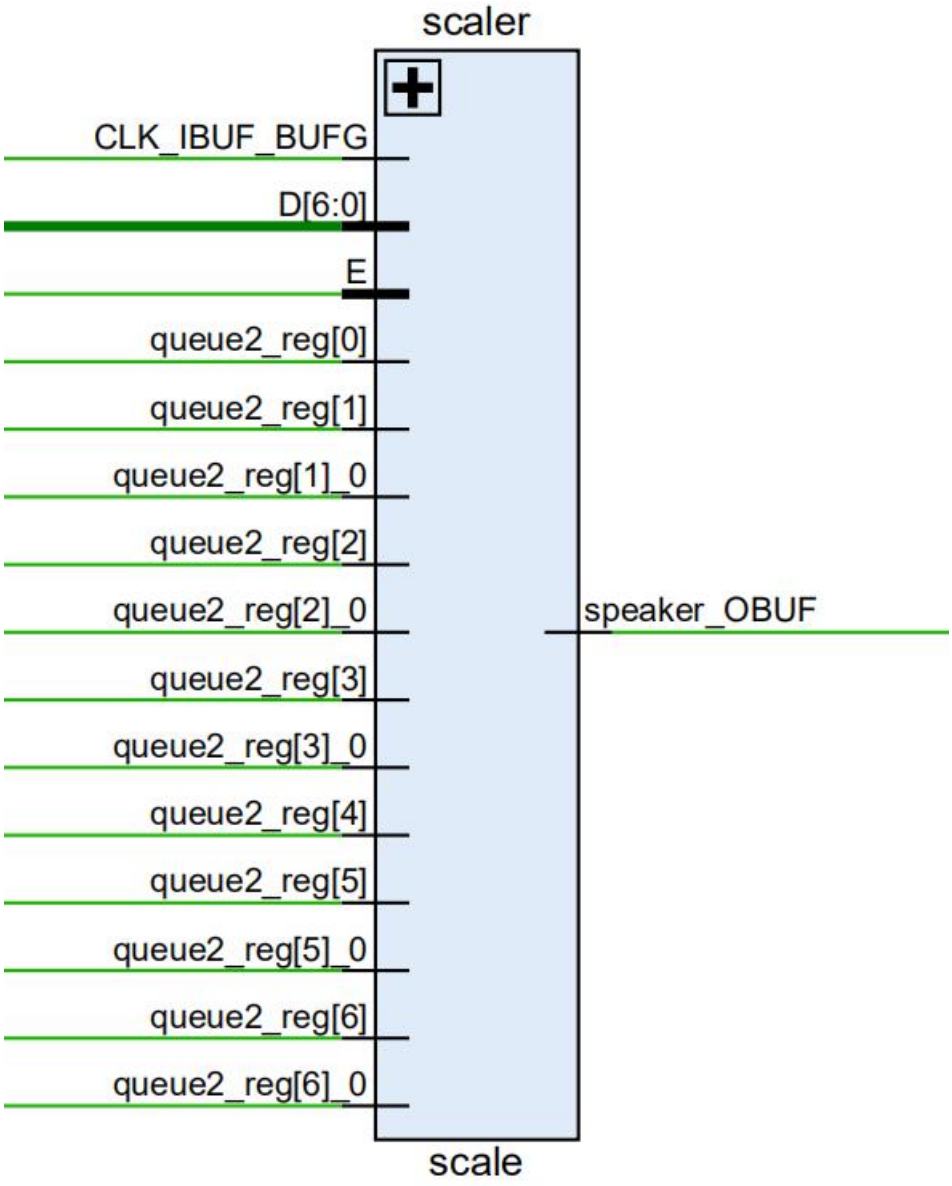
```
                8'h1c:t<=47755;
                8'h1b:t<=45085;
                8'h23:t<=42553;
                8'h2b:t<=40160;
                8'h34:t<=37622;
                8'h33:t<=35790;
                8'h3b:t<=33783;
                8'h42:t<=31887;
                8'h4b:t<=30102;
                8'h4c:t<=28409;
                8'h52:t<=26809;
                8'h5a:t<=25303;
                8'h29:t<=0;
                8'h00:t<=0;
                default:;
            endcase
            if (f==0) begin t<=0;f<=1;end
        end
```

always @(posedge clk) 根据获取得到的分频数对时钟信号进行分频，得到对应音调的频率的方波信号，作为音频输出

```
    if (t>1) begin
        if (count>t) begin
            sound<=~sound;
            count<=0;
        end
        else count<=count+1;
    end

assign speaker=sound&piano;
```

Endmodule

功能框图：



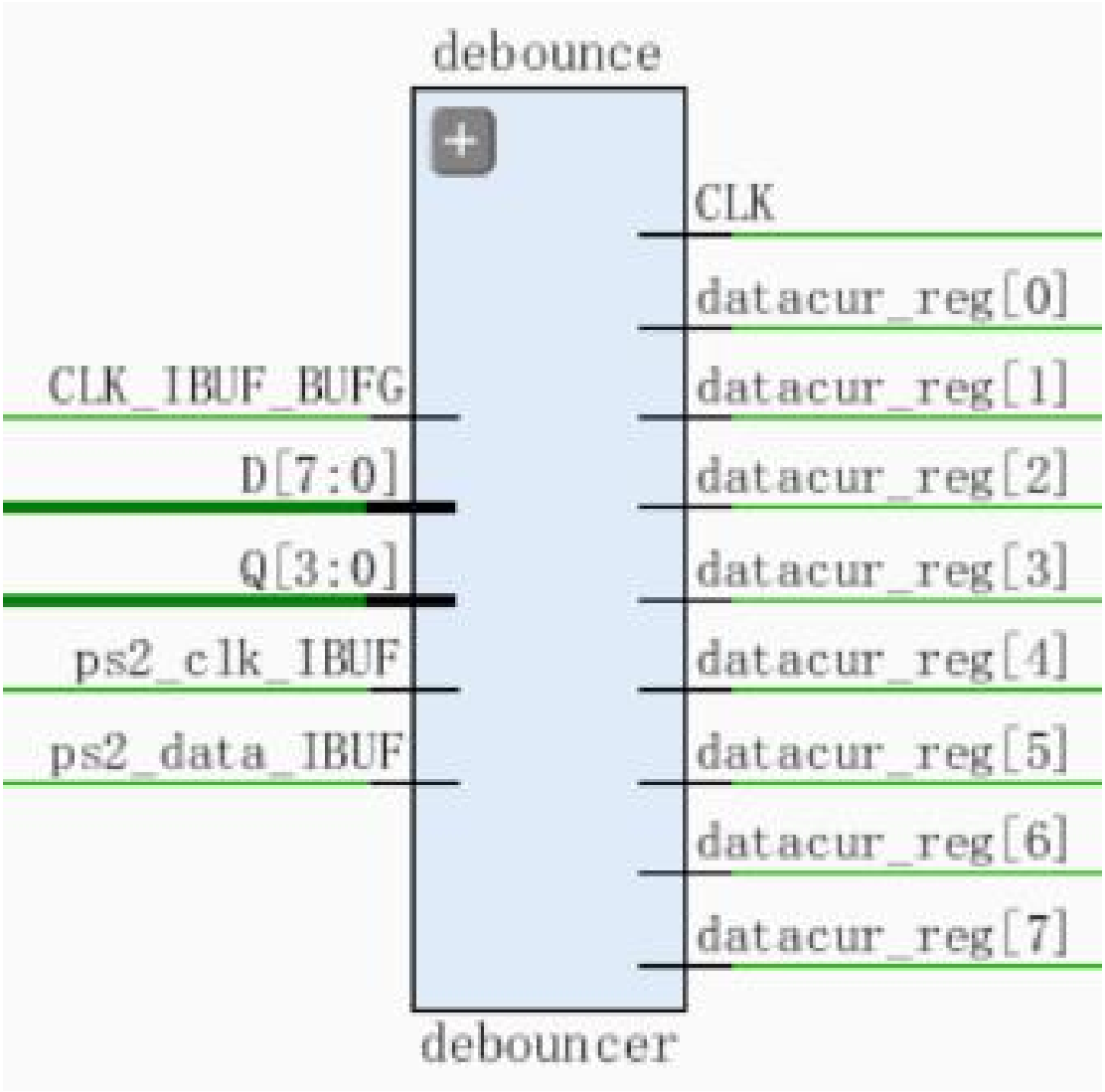
4.消抖模块

module debouncer(clk,I0,I1,O0,O1);

input clk;

```
input I0;//消抖前的低电平信号
input I1;//消抖前的高电平信号
output reg O0;//消抖后的低电平信号
output reg O1;//消抖后的高电平信号

integer cnt0,cnt1;//连续的低电平/高电平的数目
reg Iv0=0,Iv1=0;
reg out0, out1;

always@(posedge clk) begin
   if (I0==Iv0)begin
      if (cnt0==19) O0<=I0; else cnt0<=cnt0+1;
   end
   else begin cnt0<=0;Iv0<=I0;end
   if (I1==Iv1) begin
      if (cnt1==19) O1<=I1; else cnt1<=cnt1+1;
   end
   else begin cnt1<=0;Iv1<=I1;end
End

Endmodule
```

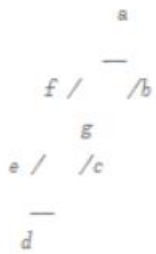
功能框图：



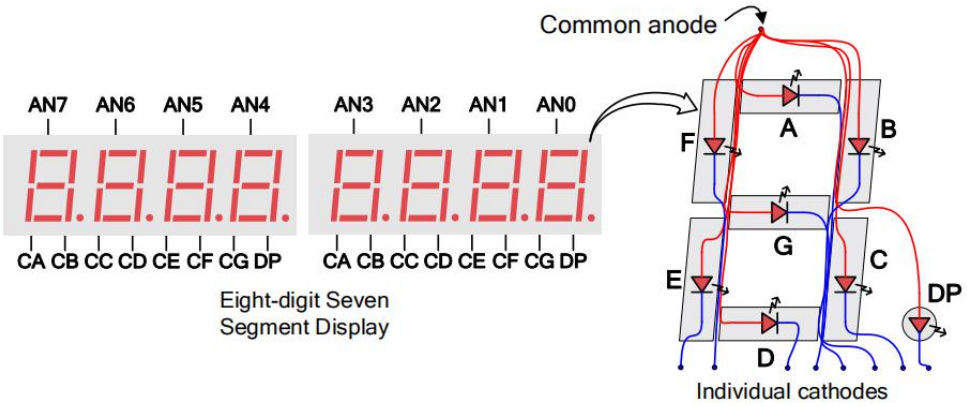
5.数码管显像模块：

要想让八位数码管分别显示不同的数字，需要利用人眼的视觉残留效应。利用高频率的时钟信号，每一个上升沿时在不同位的数码管上显示数字，看上去就像是八位数码管在同时显示一样。

Nexys4 开发板上的八位七段数码管是共阳极的，只需要控制每个上升沿时刻给不同的位的阳极给低电平，就可以确保在下一个时钟周期内只有这个位置的数码管能够发光。确定了发光的数码管的位之后，再将数码管按照下图方式进行编号：



通过给 abcdefg 不同的电平就可以控制七段数码管产生不同的明灭状态。从而实现了 4 个键盘码在八位数码管上分别显示的问题。

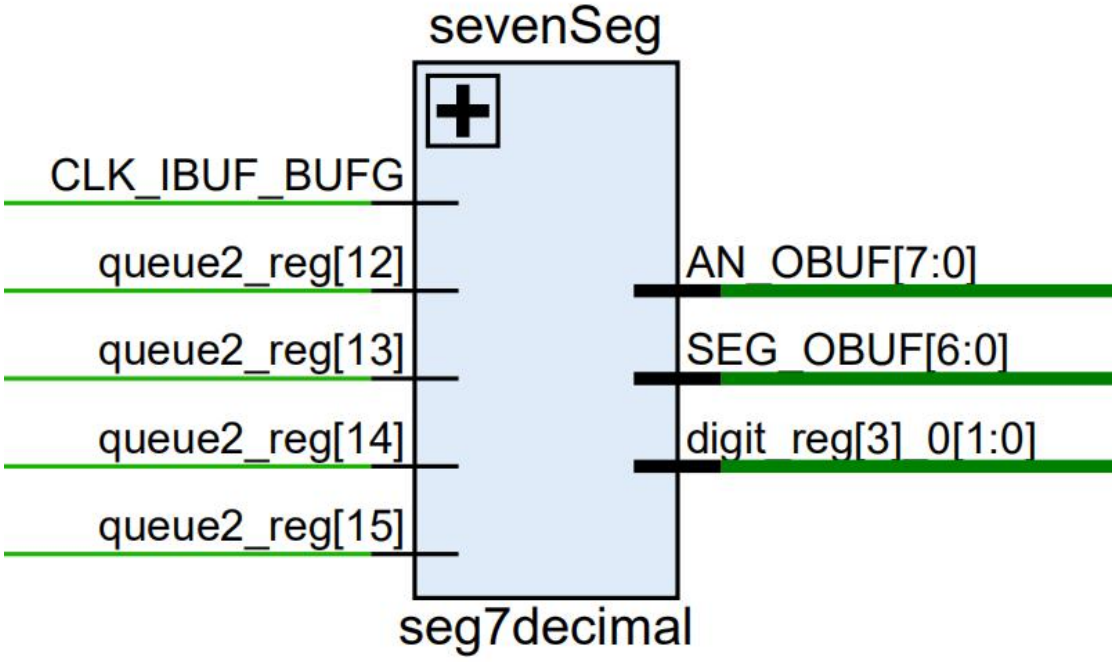



```
`timescale 1ns / 1ps
module seg7decimal(display,x,clk,seg,an,dp);
input display;
input [31:0] x;
input clk;
output reg [6:0] seg;
output reg [7:0] an;
output wire dp;
wire [2:0] s;
reg [3:0] digit;
wire [7:0] aen;
reg [19:0] clkdiv;
```

```
assign dp = 1;
assign s = clkdiv[19:17];
assign aen = 8'b11111111;

always @(posedge clk)
      if (display)
    case(s)//截取 4 个键盘码共 8 个字符的信息，分个显示
        0:digit = x[3:0];
        1:digit = x[7:4];
        2:digit = x[11:8];
        3:digit = x[15:12];
        4:digit = x[19:16];
            5:digit = x[23:20];
            6:digit = x[27:24];
            7:digit = x[31:28];
        default:digit = x[3:0];
    endcase
always @(*)
      if (display)
         case(digit)//按照 gfedcba  的顺序表明各管电平，其中低电平表示该管发光
              0:seg = 7'b1000000;
              1:seg = 7'b1111001;
              2:seg = 7'b0100100;
              3:seg = 7'b0110000;
              4:seg = 7'b0011001;
              5:seg = 7'b0010010;
              6:seg = 7'b0000010;
              7:seg = 7'b1111000;
              8:seg = 7'b0000000;
```

```
            9:seg = 7'b0010000;
            'hA:seg = 7'b0001000;
            'hB:seg = 7'b0000011;
            'hC:seg = 7'b1000110;
            'hD:seg = 7'b0100001;
            'hE:seg = 7'b0000110;
            'hF:seg = 7'b0001110;
            default: seg = 7'b0000000;
        endcase

always @(*)
    if (display) begin
        an=8'b11111111;
        if (aen[s] == 1) an[s] = 0;//表明每位数码管的阳极的电平，为低电平表示该数码管可以发光
    end

always @(posedge clk) clkdiv <= clkdiv+1;

endmodule
```

功能框图：



五、测试模块建模

本实验仅需要对 PS2 通信模块进行测试即可：

```
module ps2_tb;
//系统输入
reg clk;//系统输入时钟
reg rst_n;//系统复位
reg ps2_clk;//ps2 的时钟
reg ps2_data_in;//ps2 的数据
//系统输出
wire [7:0] ps2_data_out;//按键的通、断码
wire valid;//通、断码有效信号
initial begin
    clk = 1;
    rst_n = 0;
    ps2_clk = 1;
    ps2_data_in = 1;
```

```
#200.1 rst_n = 1;
//数字“1”的通码
ps2_data_in = 0;//起始位“0”
#60 ps2_clk = 0;
#120 ps2_clk = 1;
#60 ps2_data_in = 1;//“1”
#60 ps2_clk = 0;
#120 ps2_clk = 1;
#60 ps2_data_in = 0;//“0”
#60 ps2_clk = 0;
#120 ps2_clk = 1;
#60 ps2_data_in = 0;//“0”
#60 ps2_clk = 0;
#120 ps2_clk = 1;
#60 ps2_data_in = 1;//“1”
#60 ps2_clk = 0;
#120 ps2_clk = 1;
#60 ps2_data_in = 0;//“0”
#60 ps2_clk = 0;
#120 ps2_clk = 1;
#60 ps2_data_in = 1;//“1”
#60 ps2_clk = 0;
#120 ps2_clk = 1;
#60 ps2_data_in = 1;//“1”
#60 ps2_clk = 0;
#120 ps2_clk = 1;
#60 ps2_data_in = 0;//“0”
#60 ps2_clk = 0;
#120 ps2_clk = 1;
#60 ps2_data_in = 1;//奇偶校验位“1”
```

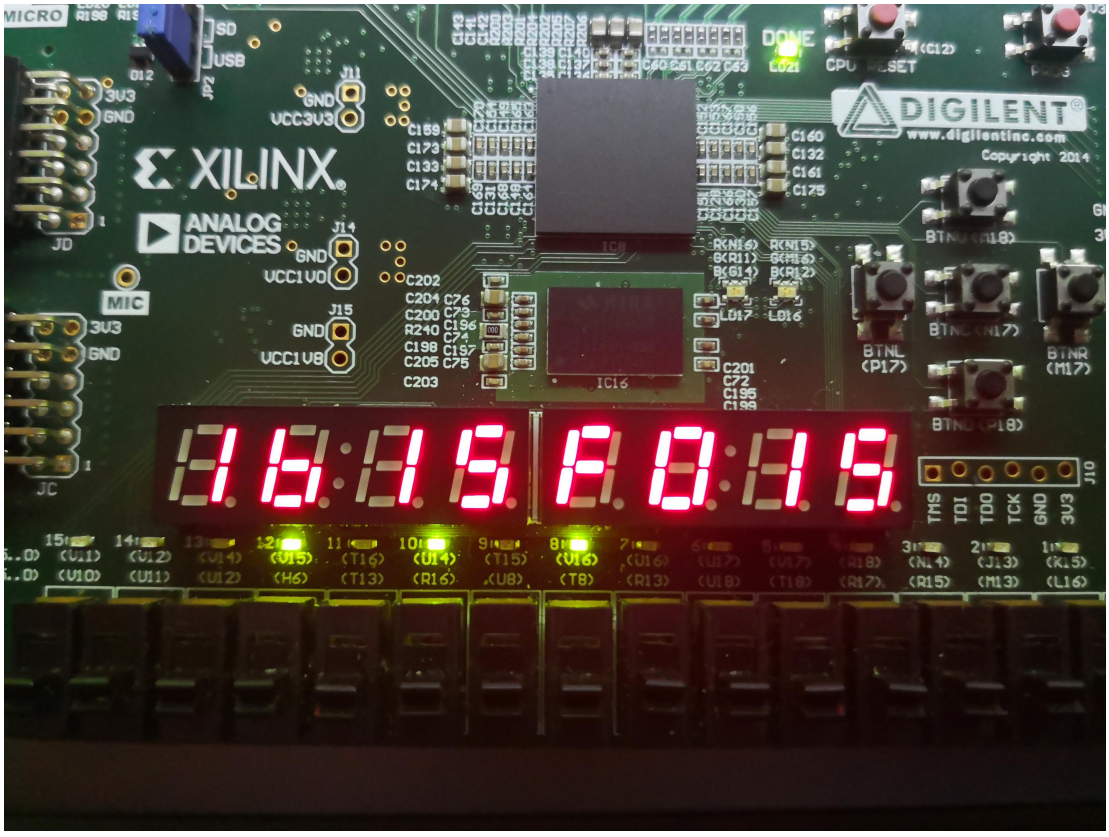
```
#60 ps2_clk = 0;
#120 ps2_clk = 1;
#60 ps2_data_in = 1;//停止位“1”
#60 ps2_clk = 0;
#120 ps2_clk = 1;
#2000
//断码中的“f0”
ps2_data_in = 0;//起始位“0”
#60 ps2_clk = 0;
#120 ps2_clk = 1;
#60 ps2_data_in = 0;//“0”
#60 ps2_clk = 0;
#120 ps2_clk = 1;
#60 ps2_data_in = 0;//“0”
#60 ps2_clk = 0;
#120 ps2_clk = 1;
#60 ps2_data_in = 0;//“0”
#60 ps2_clk = 0;
#120 ps2_clk = 1;
#60 ps2_data_in = 0;//“0”
#60 ps2_clk = 0;
#120 ps2_clk = 1;
#60 ps2_data_in = 1;//“1”
#60 ps2_clk = 0;
#120 ps2_clk = 1;
#60 ps2_data_in = 1;//“1”
#60 ps2_clk = 0;
#120 ps2_clk = 1;
#60 ps2_data_in = 1;//“1”
#60 ps2_clk = 0;
```

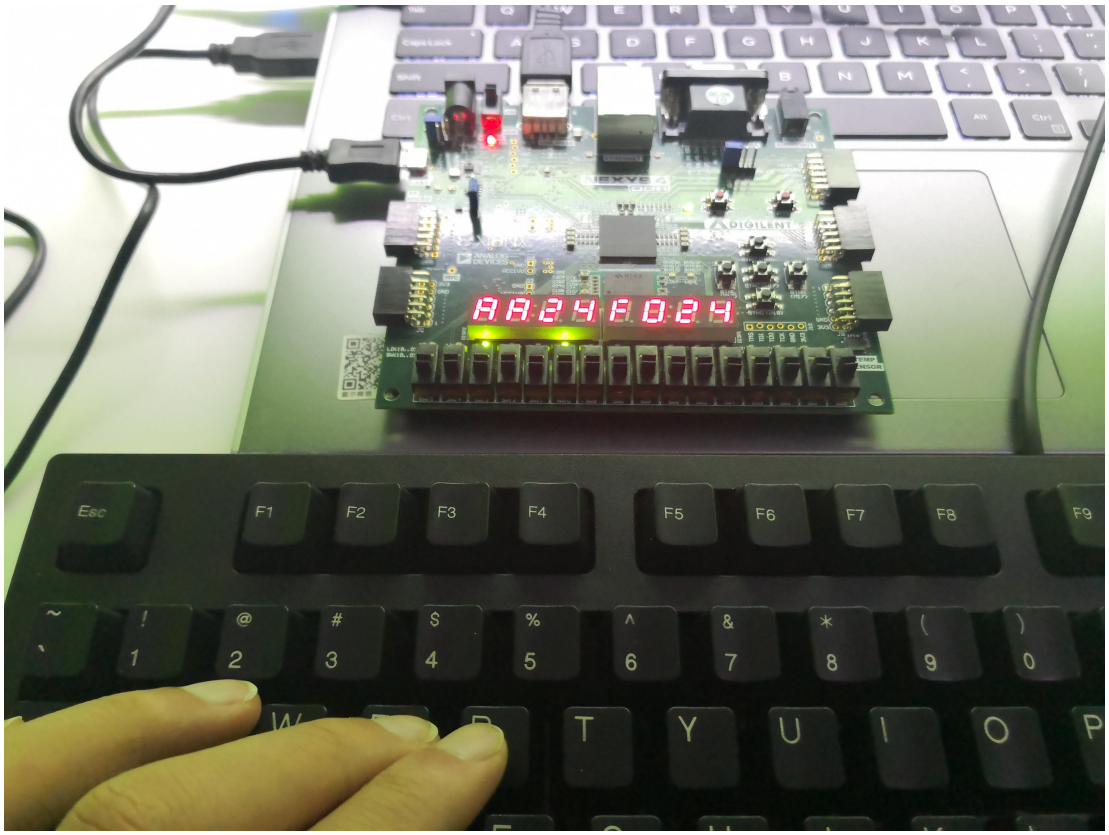
```
#120 ps2_clk = 1;
#60 ps2_data_in = 1;//“1”
#60 ps2_clk = 0;
#120 ps2_clk = 1;
#60 ps2_data_in = 1;//奇偶校验位“1”
#60 ps2_clk = 0;
#120 ps2_clk = 1;
#60 ps2_data_in = 1;//停止位“1”
#60 ps2_clk = 0;
#120 ps2_clk = 1;
#2000
//数字“1”的通码
ps2_data_in = 0;//起始位“0”
#60 ps2_clk = 0;
#120 ps2_clk = 1;
#60 ps2_data_in = 1;//“1”
#60 ps2_clk = 0;
#120 ps2_clk = 1;
#60 ps2_data_in = 0;//“0”
#60 ps2_clk = 0;
#120 ps2_clk = 1;
#60 ps2_data_in = 0;//“0”
#60 ps2_clk = 0;
#120 ps2_clk = 1;
#60 ps2_data_in = 1;//“1”
#60 ps2_clk = 0;
#120 ps2_clk = 1;
#60 ps2_data_in = 0;//“0”
#60 ps2_clk = 0;
#120 ps2_clk = 1;
```

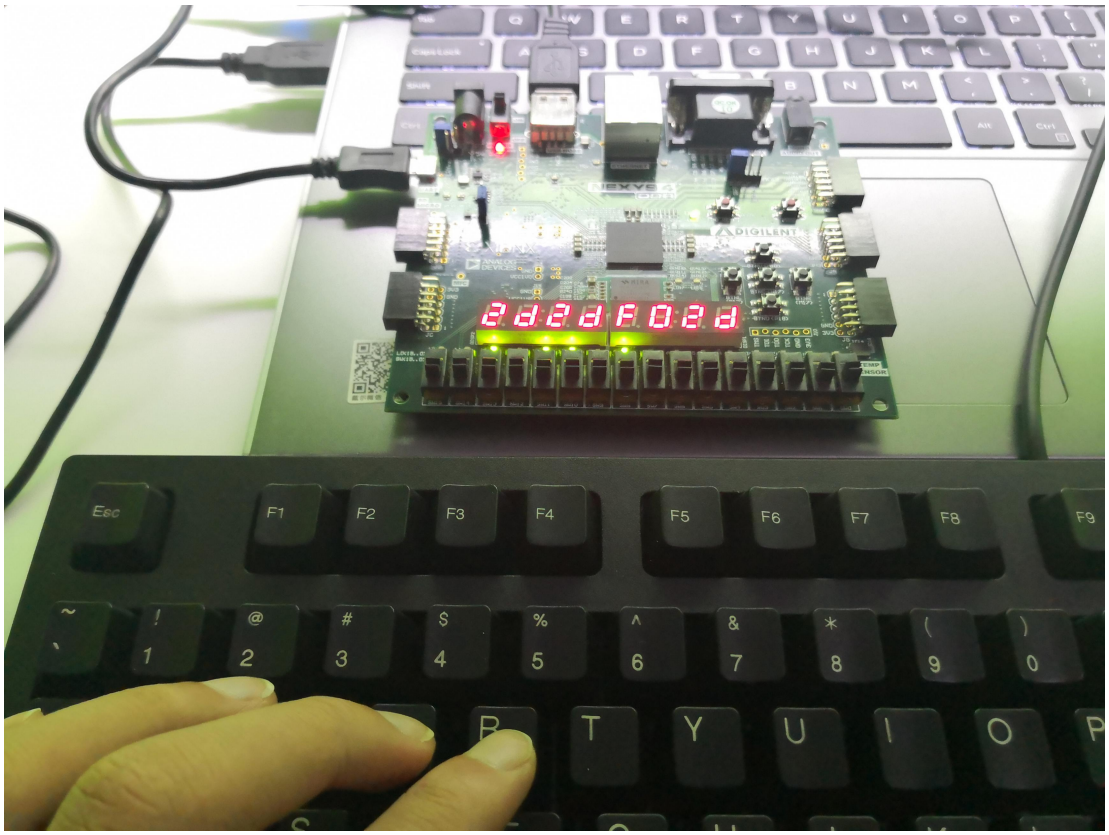
```
    #60 ps2_data_in = 1;//“1”
    #60 ps2_clk = 0;
    #120 ps2_clk = 1;
    #60 ps2_data_in = 1;//“1”
    #60 ps2_clk = 0;
    #120 ps2_clk = 1;
    #60 ps2_data_in = 0;//“0”
    #60 ps2_clk = 0;
    #120 ps2_clk = 1;
    #60 ps2_data_in = 1;//奇偶校验位“1”
    #60 ps2_clk = 0;
    #120 ps2_clk = 1;
    #60 ps2_data_in = 1;//停止位“1”
    #60 ps2_clk = 0;
    #120 ps2_clk = 1;
end

always # 5 clk = ~clk;//100M 的时钟

PS2receiver  controller(
.clk(clk), //系统输入时钟
.rst_n(rst_n),//系统复位
.ps2_data_in(ps2_data_in),//ps2 的数据
.ps2_data_out(ps2_data_out),//按键的通、断码
.ps2_clk(ps2_clk),//ps2 的时钟
);

endmodule
```

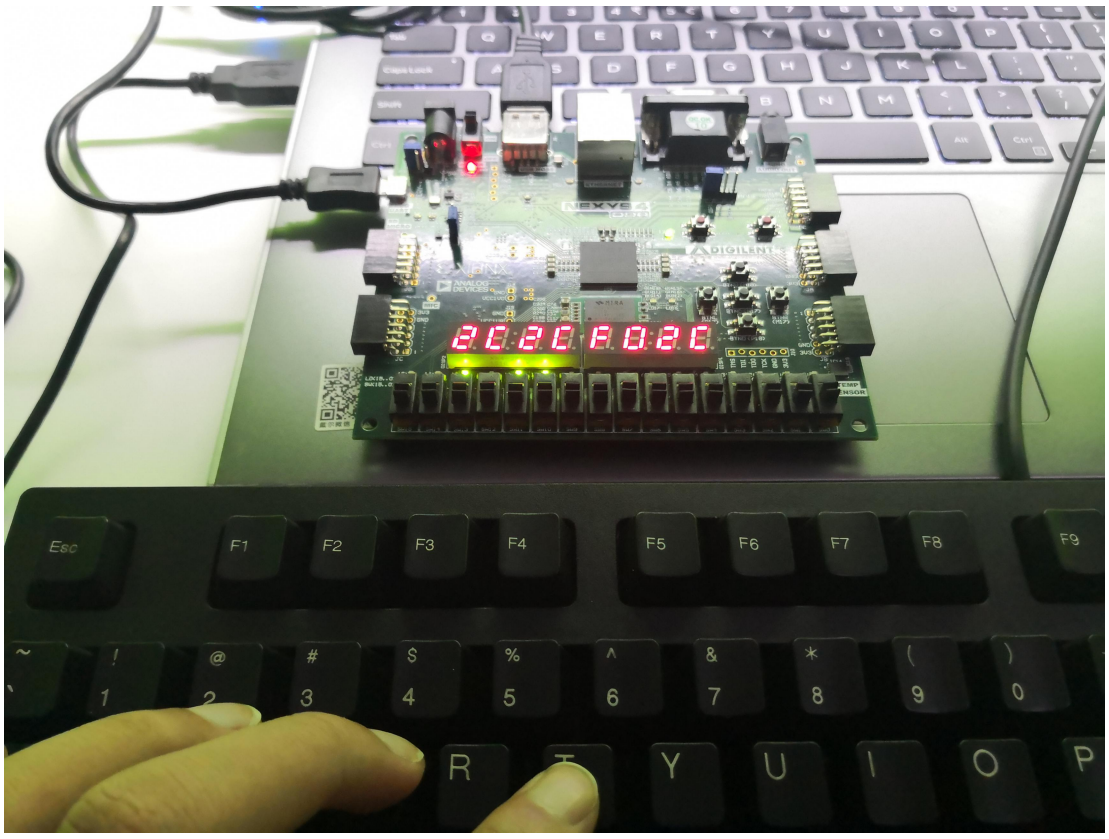
六、实验结果

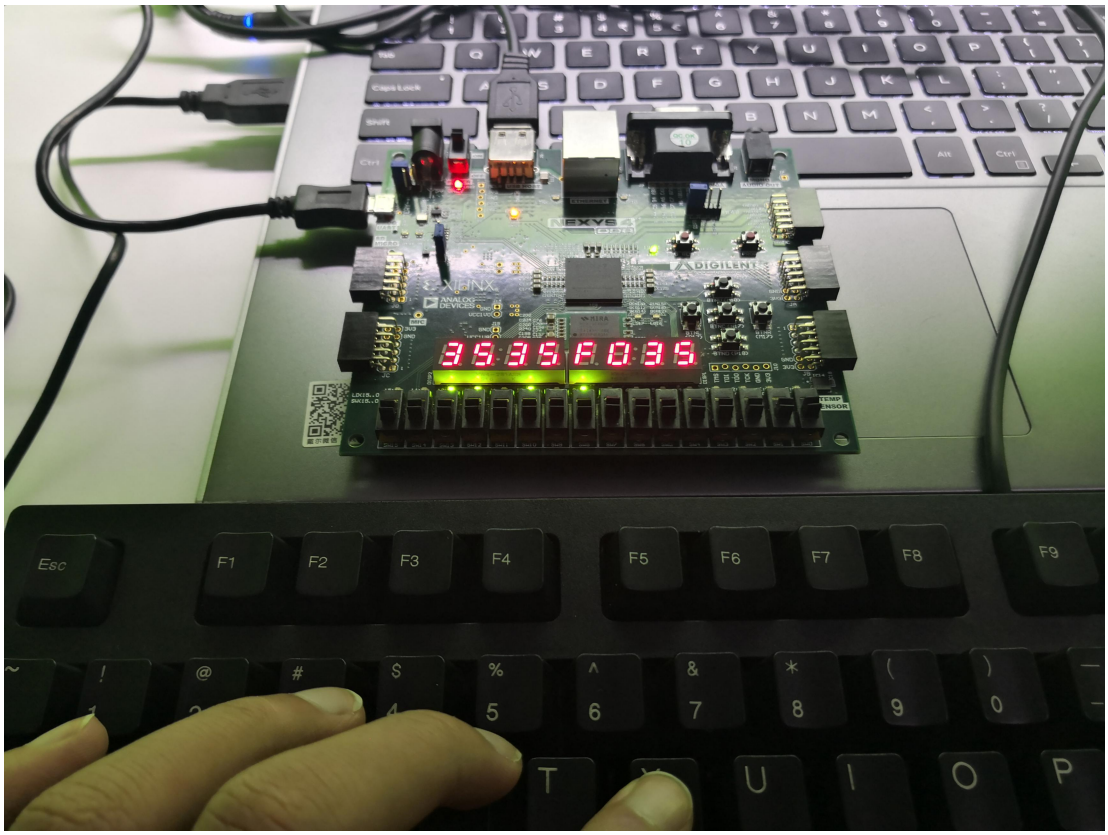
直接下板检验更为明显(音频此处无法展示)：






2d2aF02d


2C2CF02C



七、结论

1.由于 PS/2 键盘的扫描码集并不是只有一套，而是有三套，所以当我按照开发板指导手册上的实例扫描码集合来进行 verilog 程序编写时，使用许多键盘在下板后都没有反应。直到我将读入的扫描码结果显示在数码管上才意识到这个问题。于是开始尝试各种键盘，希望找到和开发板手册上的示例相同的扫描码集合，因为这套集合较为简洁规整，易于编写。

2.不同的键盘额定输出电流和电压各不相同，Nexys4 开发板承载能力有限，当接上 100mA 额定电流的键盘时，若一段时间没有更换按键或者没有按键，则 PS/2 数据线将会被开发板拉低，直到按下新键之后的一段时间才能重新恢复正常的通信状态。这个问题通过人为消除一个键——“.”，在弹奏时不停地按下“.”键即可保持 PS/2 通信持续进行。而被过滤掉的“.”并不会影响弹奏出的音调以及数码管上显示的扫描码。

3.经过测试，电子琴数字系统基本能够实现演奏乐曲的功能，且经过一定程度的练习，可以替代真正的电子琴进行使用。实验取得了成功。