Fall 2023 CS207 Project

2024/1/2

小组成员分工及贡献：

陈长信 50%（12210731）：

学习机主模块设计，参与蜂鸣器模块设计，歌曲储存模块设计，自由模式模块设计，自动演奏模块设计，学习模式模块设计，分频模块设计，后期代码调试，项目文档补充。

张健豪 50%（12212602）：

参与蜂鸣器模块设计，LED 模块设计，七段数码管模块设计及应用，消抖模块设计，学习机结构化整合以及绑定按键上板测试，后期代码调试，项目文档编写，切换逻辑实现。

Lab 时间：

周三 5-6 节

一、开发计划日程安排和实施情况

1、开发日程

项目初期：小组讨论及分工

项目中期：小组成员根据分工完成各自工作，预计耗时 2 周

项目后期：整合小组成员工作内容，及后续代码调试工作，预计耗时 1 周。

2、实施情况

项目初期：小组讨论项目大体架构，规划需要完成的任务以及具体分工。

项目中期：两人共同编写蜂鸣器模块。一人完成所有三个模式模块，储存模块的编写，另一人完成模块整合，消抖模块，LED 模块，七段数码管模块及其应用，切换功能的逻辑实现。

项目后期：整合及代码调试阶段。主要工作为模块连接，上板测试，代码漏洞的测试及修正。

二、系统功能

系统能在三种模式之间顺序切换，以下是三种模式的具体功能

1、自由模式

基于七个开关按键和一个变调按钮，用户能自由弹奏三个八度总共 21 个音符

2、自动播放模式

学习机进入自动演奏模式，自动演奏歌曲，LED 灯光指示用户演奏位置和持续时间，一首歌曲播放完毕之后自动循环当前歌曲。用户可以通过切换歌曲按钮切换学习机播放的歌曲，以及使用暂停开关进行暂停操作。

3、学习模式

学习机在学习模式中，存在三个用户，学习机根据音符顺序和持续时间点亮琴键上方的 led 灯，当某一个用户 turn on 灯下方对应的键，用户 turn on 后对应的 led 灯会熄灭，然后点亮下一个音符对应的另一盏灯。对于这个用户的演奏，学习机在七段数码管上显示用户的实时演奏评分，并且该用户可以通过储存按钮更新自己的评级。

三、系统使用说明（附 I/O 示意图）

1、输入端口

1）复位键（按钮 S6）：

自由模式中，复位键的作用为静音

自动播放模式中，复位键用于从头播放当前歌曲。

学习模式中，复位键能从新开始演奏当前歌曲，并重置实时评分。

2）七个音符拨码开关（开关 SW7-SW1）：

分别为音符 do、re、mi、fa、so、la、xi，turn on 学习机播放对应音符，turn off 停止播放。

注意：同时 turn on 一个以上的音符开关为非法操作，学习机将不会播放对应音符！

3）静音拨码开关（开关 SW0）：

Turn on 学习机静音（仅自动播放模式和学习模式生效），turn off 取消静音。

4）切换模式按钮（按钮 S3）

按下按钮切换到下一个模式，顺序为：自动播放模式->学习模式->自由演奏模式。

5）切换歌曲按钮（按钮 S0）

在自动播放模式和学习模式中，按下按钮切换到下一首歌曲。

6）切换音调按钮（按钮 S4）

在自由模式中，按下按钮调高音调，当音调达到最高时调回最低音调。

7）切换用户按钮（按钮 S1）

在学习模式中，按下按钮切换到下一个用户。

8）存储评级按钮（按钮 S2）

在学习模式中，按下按钮将根据当前评分更新当前用户评级。

2、输出端口

1）音频输出端口（J12）

使用 3.5mm 音频输出设备接入学习机即可获得殿堂级音乐体验。

2）LED 灯（LD2(7-1)）

自由模式中，LED 随用户 turn on 的对应音符开关亮起，用户可以根据灯光位置清楚得知目前发声的音符。

自动播放模式中，LED 会在播放过程中随当前播放音符亮起对应灯光，该音符演奏结束后灯光熄灭，用户可清楚得知当前播放的是哪一个音符。

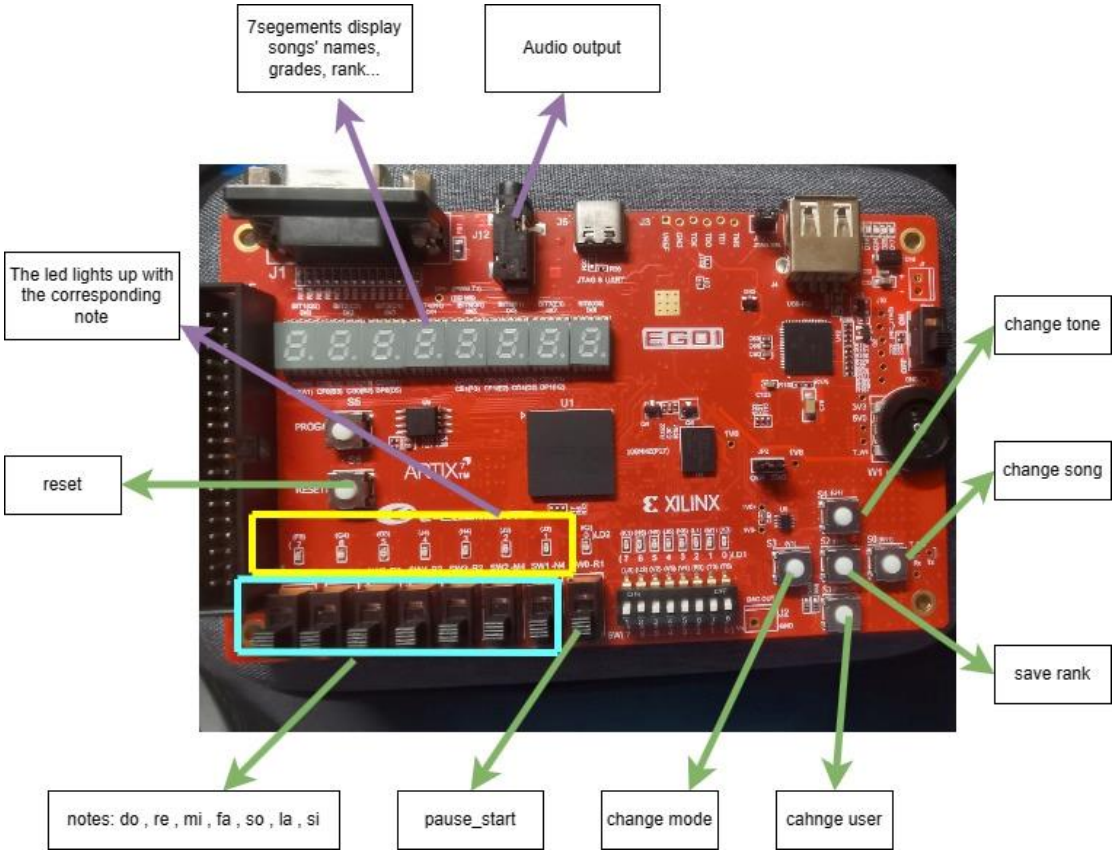
学习模式中，LED 灯引导用户需要弹奏下一个音符，根据音符顺序和持续时琴键上方的 LED 灯亮起，以引导用户正确演奏。用户 turn on 灯下方对应的后对应的 led 灯会熄灭，然下一个音符对应的另一盏 LED 灯将亮起。

3）七段数码管

当学习机处于自由模式时，数码管显示“FrEE”以提示用户当前模式为自由模式。

当学习机处于自动播放模式时，数码管将显示当前播放歌曲的曲名。

当学习机处于学习模式时，数码管第一个位置将显示当前学习的歌曲编号，随后为当前实时演奏评分，第五个位置将显示用户编号，随后为该用户的评级。



四、系统结构说明

1、常量的使用

我们对程序在运行中不会修改的值放入常量文件。这种方法优势如下：

1）可读性和可维护性：通过使用常量，可以用有意义的名称来表示特定的值，使代码更易读，降低出错的可能性，并且如果需要修改该值，只需要修改常量定义处即可，无需在代码的多处进行修改。

2)防止意外修改：常量的值在程序运行期间是不可修改的，这可以防止意外修改而导致程序错误。

3)代码优化：编译器或解释器在处理常量时可以进行一些优化，例如在编译期直接将常量替换为数值，提高程序的执行效率。

2、主模块 Piano_Controller

```
module Piano_Controller(
    input clk,
    input rst_n,
    input switch_tone,
    input [`key_WIDTH - 1:0] key_pitch,
    input changemode,
    input start_pause,
    input changeflag,
    input changeuser,
    input save,
    output beep,
    output [`led_WIDTH  - 1:0]led,
    output sd,
    output  [`segenable_WIDTH - 1:0] tub_sel_out, //Select location
    output  [`seg_WIDTH - 1:0] light_seg, //Select section
    output  [`seg_WIDTH - 1:0] light_seg1
    );
```

1）主模块涵盖了学习机所有的输入输出端口。主模块内容包含以下三个部分：

【1】变量声明：声明主模块需要使用到的变量。

【2】切换模式：切换模式的逻辑

【3】结构化建模：对不同子模块给出其所需的输入，并获取其输出完成学习机主模块的功能实现，

a. 消抖模块给未消抖信号以获得经过消抖的稳定信号，

```
wire s_t;//Debounced switch_tone
Debounce switch_tone0(clk,switch_tone,s_t);
wire cm;//Debounced changemode
Debounce changemode0(clk,changemode,cm);
wire cf;//Debounced changeflag
Debounce changeflag0(clk,changeflag,cf);
wire cu;//Debounced changeuser
Debounce changeuser0(clk,changeuser,cu);
wire save_user;//Debounced save
Debounce save_userd(clk,save,save_user);
```

b. 三种模式子模块以实现学习机功能，

```
//free-play
Free_play freeplay(clk,rst_n,s_t,key_pitch,notes_free);
//auto_play
wire [`seginput_WIDTH - 1:0]song_name;
Auto_play autoplay(clk,rst_n,cf,start_pause,notes_auto, song_name);
//study mode
wire [`seginput_WIDTH - 1:0]grades_rank;
Studymode study(clk,rst_n,start_pause,key_pitch, cf, cu, save_user, notes_learn, grades_rank);
```

- 自由模式：给定 切换音调、音符键盘 输入，获得 自由模式音符输出。
- 自动播放模式：给定 切换歌曲、静音 输入，获得 自动播放模式音符、歌曲名（用于数码管） 输出。
- 学习模式：给定 静音、音符键盘、切换歌曲、切换用户、储存评级 输入，获得 学习模式音符、分数及评级（用于数码管） 输出。

c. 三个主要输出模块：LED、蜂鸣器、七段数码管，

```
Buzzer buzzer(clk,notes,beep);
Led led_light(notes,led);
Light_seg autolight0(clk,rst_n,seg_in[39:20],light_seg,tub_sel_out[7:4]);
Light_seg autolight1(clk,rst_n,seg_in[19:0],light_seg1,tub_sel_out[3:0]);
```

- 蜂鸣器模块：给定 音符 输入，获得 声音信号 输出。
- LED 模块：给定 音符 输入，获得 LED 输出。
- 七段数码管模块：给定 显示内容 输入，获得 段选、位选 输出。

d. 自动播放与学习模式中使用 library 存储的歌曲以及使用分频器

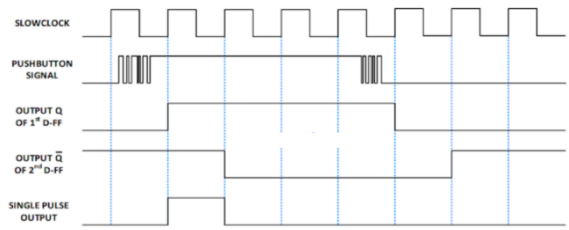
```
Library lib(flag,cnt,pause_start,next_note);

//then after we get cnt and flag and the time we
Library lib(flag,cnt,start_pause,notes);

wire clk_out;
Fre_div frediv(clk,clk_out);

Fre_div #(50000000) fre2(clk,clk_out);
```

- 使用 Library 模块输入对应的 flag 对应歌曲，然后输入 cnt 表示音符编号，最后可以输出供蜂鸣器输出的方波信号。
- 使用分频器模块输入 100hz 的时钟信号，输出 0.5s 一个周期的信号，为时序逻辑做分频。

五、子模块功能说明

1、Debounce（消抖模块）

```
module Debounce (
  input wire clk,
  input wire button,
  output wire debounced
);
```



用上图所示原理，$debounced = q1 \,\&\, (\sim q2)$获得稳定的消抖后信号。

2、Buzzer（蜂鸣器模块）

```
module Buzzer(
  input clk,
  input [`notes_WIDTH - 1:0] note,//001 do 010 re 011 mi 100 fa 101 so 110 la 111 xi
  output wire speaker
  );
```

根据不同音符频率不同，使用计数器即可得到不同的音频信号。

3、Led（LED 模块）

```
module Led(
  input [4:0] notes,
  output reg [6:0] led
  );
```

根据给定音符的输入确定 led 的输出，三种音调总共 21 个音符每个 LED 灯对应三个不同音调的同一音符。

4、Light_seg（七段数码管模块）

```
module Light_seg(
input clk,
input rst_n,
input [`seginput_WIDTH/2 - 1:0] x,
output reg [`seg_WIDTH - 1:0] seg_out,
output reg [`segenable_WIDTH/2 - 1:0] seg_en
);
```

由于八个七段数码管中，前四个数码管与后四个数码管的段引脚不同，所以此处七段数码管模块仅控制四个数码管，结构化建模时需要据情况实例化两次。

该模块的实现主要基于四个小模块：

- 计数器模块：s 随时钟信号在 00、01、10、11 四个状态循环，配合下面位选模块和段选模块能够做到四个位置的七段数码管循环亮起，此频率使人眼难以辨认，以达到多个七段数码管同时亮起且显示不同内容的效果。

```
wire [`s_WIDTH - 1:0]s;//time of one seg light   `define s_WIDTH        2
reg[`d_WIDTH - 1:0]clkdiv;//`define d_WIDTH        21
//counter decide how long each digital tube will be lighting
always @(posedge clk, negedge rst_n) begin
   if(~rst_n)
      clkdiv<=0;
   else
      clkdiv<=clkdiv+1;
end
assign s = clkdiv[20:19];//2bit represent four positions
```

- 位选模块：

```
//select location
always@*begin
   seg_en = 4'b0000;
   seg_en[s]=1;
end
```

- 段选模块：

```
//select content
always@*begin
   case(s)
      0:in=x[4:0];
      1:in=x[9:5];
      2:in=x[14:10];
      3:in=x[19:15];
      default:;
   endcase
end
```

- 输出内容模块：根据数码管的所在位置，列出所需的显示样式以便调用。

- Library 模块：即存储歌曲，输入对应歌曲编号，以及歌曲的音符编号，这样能够给出对应歌曲的 5 比特音符给主模块，最后利用 buzzer 发声。

```
module Library(
    input [1:0] flag,
    input [7:0] cnt,
    input pause,
    output reg [4:0]notes
    );
```

- 自由模式模块：时钟信号，复位，变换音高以及音符输入，输出对应的 notes，即利用不同音高对应按键输出即可。

```
module Free_play(
    input clk,
    input rst_n,
    input s_t,
    input [6:0] key_pitch,
    output reg [4:0] notes
    );
```

- 自动播放模式模块：利用分频器，将每个音间隔 0.5s，并在音之间添加空白音，这样间隔更明显，最后利用计数器调用 library 中对应歌曲的音符，最后发声。

```
module Auto_play(
    input clk,
    input rst_n,
    input cf,
    input start_pause,
    output [`notes_WIDTH - 1:0] notes,
    output reg [39:0]seg_in//songs name
    );
```

- 学习模式模块：也是利用分频器，然后选择对应歌曲，如果输入按键对应是正确的，那么会跳到下一个音，如果不正确，会被计入错误次数，实时更新用户的分数，并且提供用户分数等级存储功能，每次需要选择用户进行演奏，演奏结束或者复位会自动存入用户演奏记录，并且更新用户评级

```
module Studymode(
    input clk,
    input rst_n,
    input pause_start,
    input [6:0] key_pitch,
    input cf,
    input cu,
    input save,
    output [4:0] next_note,
    output [39:0]seg_in//grades and rank
    );
```

六、项目总结

1、团队合作

在本次项目中，我们采用了分工合作的方式来完成各自的任务。我们相互协作、沟通顺畅，及时解决出现的问题，并且能够在项目进度出现偏差时快速调整。同时，我们也不定期召开会议，讨论项目进展和下一步的计划，保证了项目的顺利进行。

2、开发

在开发阶段，我们首先明确了项目需求，确定了主模块的输入输出接口，通过对项目进行了细致的规划和设计，以及对项目的功能模块和开发流程的确认，我们最终实现了预期的功能。

3、测试

测试环节分为两个环节：第一个环节是由于开发板数量的限制，我们采取两人先后工作的模式，使得小组成员工作时都能够及时上板测试，提高了个人工作部分的测试效率。第二个环节是小组成员各自负责工作完成后的最终代码测试阶段，该阶段下小组成员增加组会次数，及时同步信息，代码调试效率大大提高，最终完成了项目要求。

七、方案建议

1、Project 主题：基于 Vivado 和 Ego1 的密码库设计

2、密码库功能：

1）基本功能：密码库可存储一定数量用户的账号及其对应密码。

对新用户，该用户可通过注册行为为自己设置一个密码库账号和密码。

对已有密码库账号的用户，其可以通过登录行为登录自己的密码库，然后可以写入该用户需要保存的账号和对应密码，读取操作来获取已保存的账号和密码。

2）标准功能

在基础功能的前提下，用户登录后可以通过文件 I/O 来读取外部设备文件中的账户密码信息，导出板内该用户存储的账号密码到外部设备。

3）附加功能

为存入密码库的账号和密码加密以确保数据安全等。