

# CS207 Project

2023年秋季学期CS207 Project1 钢琴学习机 项目文档

## 一、基本信息介绍

Member	Student ID	Contribution Rate
刘家铭	12210607	33.33%
温云翔	12211306	33.33%
袁龙	12211308	33.33%

### 团队贡献

刘家铭：顶层模块设计，自由模式的基本功能及标准功能，实现led灯和七段数码显示管

袁龙：自动演奏模式的基本功能及标准功能，实现VGA和录音功能的bonus

温云翔：学习模式的基本功能及标准功能，实现精细化评分的bonus

### 开发计划日程安排

预计于第12周实现项目第一阶段的内容：三种模式的基本功能

预计于第13周实现项目第二阶段的内容：三种模式的标准功能以及led灯、七段数码显示管

预计于第14、15周实现项目第三阶段的内容：bonus部分、项目文档

### 实施情况

11月24日进行第一次项目讨论，确定了团队成员分工及项目实施计划与日程安排；  
12月4日初步实现自动演奏模式；  
12月5日实现自由模式；  
12月6日添加了led灯，修改了自动演奏模式；  
12月7日初步实现学习模式；  
12月10日基本完成各自第一阶段任务，并进行了第二次项目讨论，总结了第一阶段中发现的问题，进一步规划了第二阶段的任务；  
12月16日完成led灯和七段数码显示管，实现了自由模式下的高低音；  
12月17日修改了自动演奏模式的演奏逻辑，合并了Buzzer模块的高低音输入；  
12月20日整合了三种模式，建立的顶层模块；  
12月22日解决了自动演奏模式下bug，扩充了曲库；  
12月23日实现了VGA显示；  
12月24日实现学习模式；  
12月26日修复了数码管显示问题，增添了录音模式，修复了项目存在的一些问题；  
12月27日增加了按键调整模式；  
12月28日对代码进行规范性修改，丰富了VGA显示，解决了录音模式下的bug；  
12月29日进行答辩

## 二．系统功能列表

简述所实现的功能的内容

**自由模式：**蜂鸣器根据选择的音符发声、高低音调整

**自动演奏模式：**蜂鸣器根据曲目发声、曲目之间可以进行切换

**学习模式：**根据曲目中音符的顺序点亮led灯引导用户学习、显示用户的评级、显示用户的账户

**录音模式：**能够记录下用户在学习机中所演奏的内容，在调整到播放模式下会自动演奏

**播放录音模式：**播放所储存的录音

**LED：**能根据音符显示对应的led灯，也能根据模式显示各自的led灯

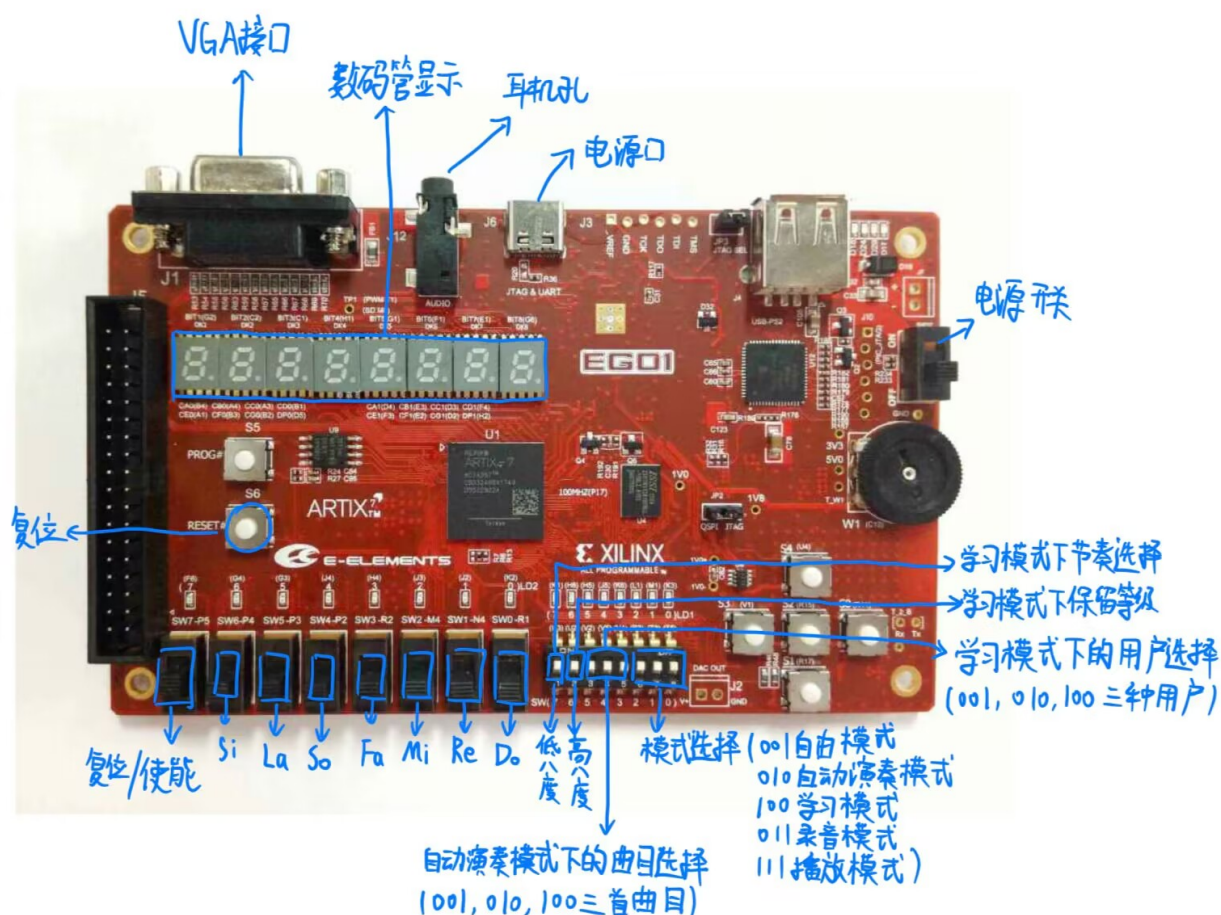
**七段数码显示屏：**能根据学习机当前状态实时更新显示内容，包括模式、当前发声音符（在自由模式、自动演奏模式下）、评级（学习模式下）、评分（学习模式下）、用户历史评级（学习模式下）

**VGA显示：**根据模式显示相应模式名称，自动演奏模式下能显示歌曲名称，任何模式下都会在屏幕下方显示

cs207project 如有雷同纯属巧合

## 三．系统使用说明

### 1.1 系统输入、输出端口说明



```

module Home(
    input clk,rst_n,rst,
    input [6:0] note,
    input [2:0] mode,//F 001,A 010,S 100,Ad 101,record 011,read 111
    input [1:0] low_note_control,//10low,01high
    input [2:0] select,
    output [2:0] led_select,
    output [2:0] led_mode,
    output [6:0] led_note,
    output [7:0] seg_left,
    output [7:0] seg_right,
    output [7:0] choose,
    output led_low_note_control,
    output led_high_note_control,
    output speaker,
    output hsync,
    output vsync,
    output [11:0] vga_rgb,
    output vol
);

```

`clk`:系统时钟，绑定P17端口

`rst_n`:复位/使能信号，绑定P5端口

`rst`:录音播放的复位信号，绑定P15端口。在学习模式中，作为暂停信号，使得用户可以在学习状态下查看全部用户的最新成绩

`note`:七个输入音符，do、re、mi、fa、so、la、si依次绑定R1、N4、M4、R2、P2、P3、P4端口

`mode`:输入模式，共3bit，mode[0]、mode[1]、mode[2]依次绑定T5、T3、R3端口

`low_note_control`:高低音控制，low\_note\_control[0]控制高八度，绑定U2端口，low\_note\_control[1]控制低八度，绑定U3端口。在学习模式中，low\_note\_control[0]为储存成绩的按键，low\_note\_control[1]为学习节奏的调整按键

`select`:曲目/用户选择，select[0]、select[1]、select[2]依次绑定V4、V5、V2端口

`led_select`:曲目/用户选择的led灯输出，led\_select[0]、led\_select[1]、led\_select[2]依次绑定K6、J5、H5端口

`led_mode`:模式选择的led灯输出，led\_mode[0]、led\_mode[1]、led\_mode[2]依次绑定K3、M1、L1端口

`led_note`:输入音符的led输出，do、re、mi、fa、so、la、si的led输出依次绑定K2、J2、J3、H4、J4、G3、G4

`seg_left`:单个左侧数码管输出，端口依次绑定为D5、B2、B3、A1、B1、A3、A4、B4

`seg_right`:单个右侧数码管输出，端口依次绑定为H2、D2、E2、F3、F4、D3、E3、D4

`choose`:当前所有数码管输出，从右到左八组数码显示管端口依次绑定为G6、E1、F1、G1、H1、C1、C2、G2

`led_low_note_control`:低音音符led输出，端口绑定为K1

`led_high_note_control`:高音音符led输出，端口绑定为H6

`speaker`:蜂鸣器输出，端口绑定为H17

`hsync`:行同步信号

`vsync`:场同步信号

`vga_rgb`:输出的rgb信息

`vol`:音量输出，端口绑定为M6

## 1.2 详细功能说明

**自由模式：**用户调整到相应模式下，用户可通过打开不同的拨码开关使蜂鸣器发出对应音符的声音，七段数码显示管显示当前模式和发声音符，该音符上方led灯亮起。用户可通过调整拨码开关切换高低音模式，使得蜂鸣器发出更高八度或更低八度的声音。

**自动演奏模式：**用户调整到相应模式下，可通过打开不同的拨码开关使学习机播放不同曲目的音乐，七段数码显示管显示当前模式、曲目和发声音符，发声音符上分的led灯亮起，其中曲库实现了三个八度音的储存，并且对音符持续时间的储存预留了足够的弹性空间（即能实现除滑音以及三连音的所有音符形式）。

**演奏逻辑：**乐曲对持续时间大于等于四分音符的音符中间有0.1s的空拍，而对其余持续时间小于四分音符的进行其  $\text{持续时间} * 0.025\text{s}$ （以四分音符J持续时间为4个单位作基准）。





















**学习模式：**首先使用了四个状态机来实现用户的账户选择、钢琴的学习过程、用户等级的更新、缓冲状态。账户选择的时候，会在右侧的数码管上面显示当前账户的等级（假如该账户创建后尚未拥有记录，默认记录的等级为F）等级的排序为  $c > d > E > F$ 。钢琴的学习过程实现了根据特定的曲谱，利用led灯指导用户进行学习，当用户输入正确的音符后，led灯熄灭，下一个led灯点亮，达到学习的目的。在最后的信息储存状态，如果用户对于本次等级比较满意，可以选择更新等级，如果选择不更新，在等待8秒后，系统会自动返回到账户选择的状态。在初级节奏中（节奏选择的信号为0），超过4秒未开启正确按键即会判定失效一次，在高级节奏中（节奏选择的信号为1），超过2秒未开启正确按键即会判定失效一次。

**录音模式：**用户调整到相应模式下，将开关打开，此刻处于待定状态，当用户在学习机中开始演奏（使用音符的拨码开关）即录音开始，在用户将模式更改离当前录音模式时，录音即刻结束。（录音的上限约64个音符，单个音符时长应不超过10s）

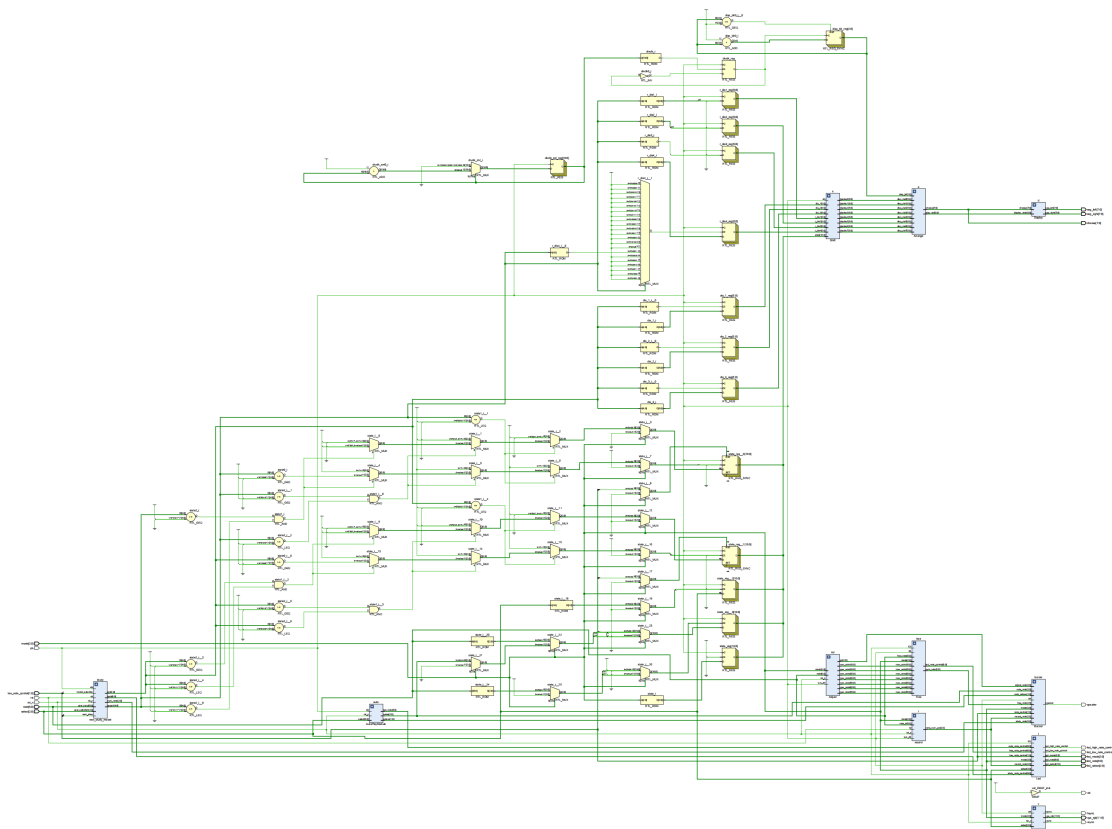
**播放录音模式：**用户调整到相应模式下，自动演奏起所储存的用户演奏内容，当按下相应按键时，会从头演奏。

## 四．系统结构说明

项目采用结构化设计，在主模块Home下实例化其他子模块，整体项目结构图如下：

- ▼  Design Sources (2)
  - ▼  Verilog (1)
    -  constant.v
  - ▼  **Home** (Home.v) (11)
    -  free : Free (Free.v)
    -  auto : AutoPlayModule (AutoPlayModule.v)
    - ▼  study : new\_study\_model (new\_study\_model.v) (2)
      -  counter : counter (counter.v)
      -  counter\_fast : counter\_fast (counter\_fast.v)
    - ▼  r : record (record.v) (1)
      -  c : counter\_100HZ (counter\_100HZ.v)
      -  Ad : Adjust (Adjust.v)
      -  buzzer : Buzzer (Buzzer.v)
      -  s : Shift (Shift.v)
      -  a : Arrange (Arrange.v)
      -  d : Display (display.v)
      -  l : Led (Led.v)
    - ▼  v : VGA (VGA.v) (2)
      -  vga\_c : vga\_ctrl (vga\_ctrl.v)
      -  vga\_p : vga\_pic (vga\_pic.v)

schematic的图示:



## 五. 子模块端口及功能说明功能说明

### 1. Free.v

`clk` :系统时钟信号

`EN` : 拨码开关使能信号

`mode` : 当前模式

`free_note` : 当前拨码开关输入

`new_note1` : 调整模式下新的do音符键位, 初始值默认为原键位(按键调整模式最终并未实现, 故此处及以下部

分的设计均仅表示默认位置)

`new_note2` : 调整模式下新的re音符键位, 初始值默认为原键位  
`new_note3` : 调整模式下新的mi音符键位, 初始值默认为原键位  
`new_note4` : 调整模式下新的fa音符键位, 初始值默认为原键位  
`new_note5` : 调整模式下新的so音符键位, 初始值默认为原键位  
`new_note6` : 调整模式下新的la音符键位, 初始值默认为原键位  
`new_note7` : 调整模式下新的xi音符键位, 初始值默认为原键位  
`num_note` : 输出的演奏音符  
`led_note_control` : led灯输出

功能说明: 用于实现自由模式相关的功能, 可以将用户输入的音符转化为对应的蜂鸣器输出和led灯输出, 同时可以通过使能信号控制输入的开关状态

## 2. AutoPlayModule.v

`clk` : 系统时钟信号  
`rst_n` : 复位信号 (开关)  
`select` : 歌曲选择  
`octave` : 控制音符高低音  
`notes` : 输出的演奏音符  
`led_note` : led灯输出

功能说明: 自动演奏的整个功能实现

## 3. new\_study\_model.v

`clk` : 系统时钟信号  
`rst_n` : 复位信号 (active low)  
`rst` : 暂停信号 (active low) 助于用户在不影响学习的前提下查看成绩  
`user_input` : 用户输入的按键信息  
`user_selection` : 进行不同账户的选择  
`user_store` : 成绩的储存信号  
`model_selection` : 不同学习节奏的选择  
`leds` : led灯的输出  
`num_note` : 输出的演奏音符 (和独热码之间的转化)  
`cnt` : 在学习状态下, 用户被判断失效的次数  
`record` : 用户成绩的输出信号

功能说明: 学习模式的整个功能实现

## 4. Display.v

`display_data` : 当前数码管要显示的数据信息  
`choose` : 当前用于显示的数码管  
`seg_left` : 左侧数码管输出  
`seg_right` : 右侧数码管输出

功能说明: 控制当前数码管的输出显示

## 5. Shift.v

`state` : 数码管状态  
`r_dis4` : 学习模式下显示记录时数码管4的数据信息  
`r_dis3` : 学习模式下显示记录时数码管3的数据信息  
`r_dis2` : 学习模式下显示记录时数码管2的数据信息  
`r_dis1` : 学习模式下显示记录时数码管1的数据信息

`dis_3` : 学习模式下显示分数时数码管3的数据信息  
`dis_2` : 学习模式下显示分数时数码管2的数据信息  
`dis_1` : 学习模式下显示分数时数码管1的数据信息  
`clk` : 系统时钟信号  
`display0` : 数码管0的数据信息  
`display1` : 数码管1的数据信息  
`display2` : 数码管2的数据信息  
`display3` : 数码管3的数据信息  
`display4` : 数码管4的数据信息  
`display5` : 数码管5的数据信息  
`display6` : 数码管6的数据信息  
`display7` : 数码管7的数据信息

功能说明: 用于实现数码管显示的信息的状态转移

## 6. Arrange.v

`disp_bit` : 当前激活的数码管的编号  
`disp_dat0` : 数码管0的数据信息  
`disp_dat1` : 数码管1的数据信息  
`disp_dat2` : 数码管2的数据信息  
`disp_dat3` : 数码管3的数据信息  
`disp_dat4` : 数码管4的数据信息  
`disp_dat5` : 数码管5的数据信息  
`disp_dat6` : 数码管6的数据信息  
`disp_dat7` : 数码管7的数据信息  
`disp_dat` : 当前激活的数码管的数据信息  
`choose` : 当前激活的数码管的独热编号

功能说明: 分配当前需要显示的数码管的显示信息

## 7. Led.v

`free_note_control` : 自由模式下的发声音符  
`auto_note_control` : 自动演奏模式下的发声音符  
`study_note_control` : 学习模式下的待演奏音符  
`record_note` : 记录模式下的发声音符  
`mode` : 模式选择  
`select` : 曲目选择  
`EN` : 使能信号  
`low_note_control` : 高低音信号  
`led_low_note_control` : 低音led灯输出  
`led_high_note_control` : 高音led灯输出  
`led_select` : 曲目选择led灯输出  
`led_note` : 发声音符led灯输出  
`led_mode` : 模式选择led灯输出

功能说明: 统一实现各个模式下led灯相关的输出

## 8. counter.v

使用三个计数器:

`counter_fast` : 生成2HZ的时钟信号  
`counter` : 生成1HZ的时钟信号  
`counter_100HZ` : 生成100HZ的时钟信号



功能说明：系统的时钟信号太快，所以通过减慢来满足设计的需求

## 9. record.v

`sys_clk`：系统时钟信号

`rst_n`：录音功能的总复位信号（控制存档的删除）

`rst`：播放录音功能下的复位信号（用来重播录音）

`mode`：用户所选择的模式（引入mode的原因是为了在同一个模块下区分录制和播放两种状态）

`note_in`：用户在录音模式下输入的音符

`note_num_out`：用户在录音模式下输出的音符

功能说明：实现录音以及录音播放功能的子模块

## 10. VGA.v

`hsync`：行同步信号

`vsync`：场同步信号

`pix_x`：屏幕有效区域内x轴的位置

`pix_y`：屏幕有效区域内y轴的位置

`pix_data`：所储存的显示信息（即rgb 颜色）

`rgb`：输出的rgb信息

功能说明：VGA功能的顶层模块

## 11. vga\_pic

`clk`：系统时钟信号

`rst_n`：复位信号

`mode`：模式选择

`select`：歌曲选择

`pix_x`：屏幕有效区域内x轴的位置

`pix_y`：屏幕有效区域内y轴的位置

`pix_data`：所储存的显示信息（即rgb 颜色）

功能说明：VGA功能中用于储存图像信息的子模块，并根据模式和歌曲选择判断输出相应数据pix\_data

## 12. vga\_ctrl

`clk`：系统时钟信号

`rst_n`：复位信号

`pix_data`：所储存的显示信息（即rgb 颜色）

`pix_x`：屏幕有效区域内x轴的位置

`pix_y`：屏幕有效区域内y轴的位置

`hsync`：行同步信号

`vsync`：场同步信号

`rgb`：输出的rgb信息

功能说明：VGA功能中用于生成扫描位置坐标以及行、场同步信号的子模块

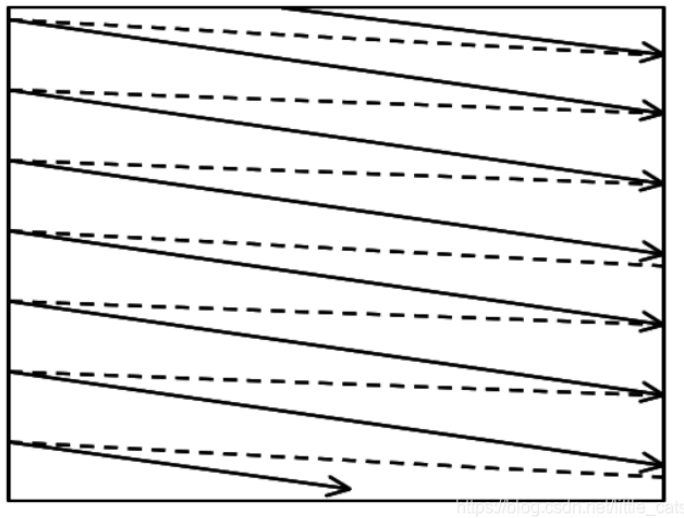
# 六. Bonus的实现说明

## 1. VGA显示

本项目选择用VGA的640\*480分辨率显示各个模式的中文名称以及自动演奏模式下的歌曲名称，其中在该分辨率下的时钟频率应为25.125MHz，而在该项目中选择生成25MHz的时钟信号进行近似替代。

## 1.1 原理介绍

VGA 显示器是依靠阴极射线工作的，也就是高速电子流，当高速电子流扫过显示器表面时，会点亮对应的像素点。通常而电子束的轨迹类似这样。为了点亮整个屏幕，电子束需要在水平偏转系统和垂直偏转系统的共同协调下在极短的时间一行一行扫过整个屏幕。

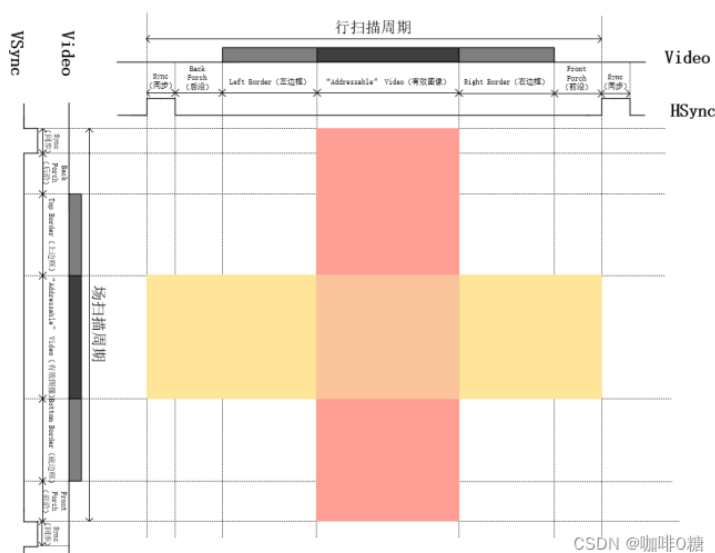


一个完整的行扫描周期，包含 6 部分：Sync（同步）、Back Porch（后沿）、Left Border（左边框）、“Addressable” Video（有效图像）、Right Border（右边框）、Front Porch（前沿），这 6 部分的基本单位是 pixel（像素），即一个像素时钟周期。

在一个完整的行扫描周期中,Video 图像信息在 HSync 行同步信号的同步下完成一行图像的扫描显示，Video 图像信息只有在“Addressable” Video（有效图像）阶段，图像信息有效，其他阶段图像信息无效。

HSync 行同步信号在 Sync(同步)阶段，维持高电平，其他阶段均保持低电平，在下一个行扫描周期的 Sync(同步)阶段,HSync 行扫描信号会再次拉高，其他阶段拉低，周而复始。

场同步与上述行同步类似。



图中的红色区域表示在一个完整的行扫描周期中,Video 图像信息只在此区域有效，黄色区域表示在一个完整的场扫描周期中,Video 图像信息只在此区域有效，两者相交的橙色区域，就是 VGA 图像的最终显示区域。

## 1.2 VGA汉字显示

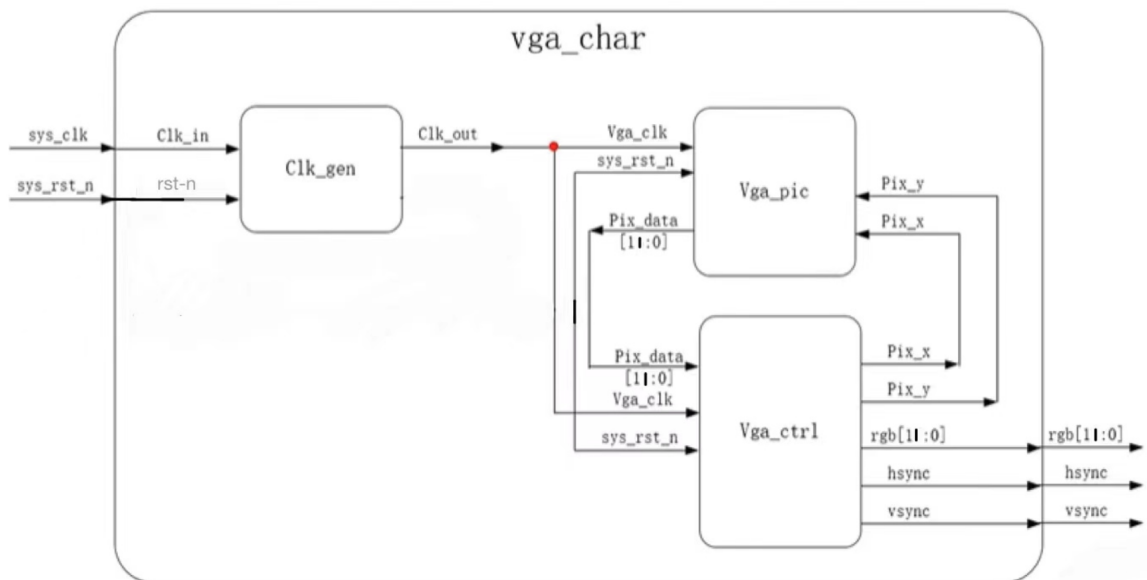
在清楚VGA的基本工作原理之后，本项目选择用VGA的显示汉字来代表各个模式以及自动演奏模式下的歌曲名称。

我们使用 `PCtoLCD2002完美版` 该工具进行取字模操作，形成字高64、字宽64的若干汉字（屏幕下方小字为32字高、32字宽）。

在 `vga_pic` 模块中分别用 `[255:0] char [63:0]`、`[511:0] cchar [63:0]`、`[511:0] a_char [31:0]` 进行储存。

### 1.3 VGA功能实现

VGA项目结构图如下



- `hsync`：行同步信号
- `vsync`：场同步信号
- `pix_x`：屏幕有效区域内x轴的位置
- `pix_y`：屏幕有效区域内y轴的位置
- `pix_data`：所储存的显示信息（即rgb 颜色）
- `rgb`：输出的rgb信息

其中在 `vga_ctrl` 里引入 `cnt_h` 和 `cnt_v` 分别作为行与场的位置的计数，当 `cnt_h` 和 `cnt_v` 到达有效区域时，用组合逻辑给 `pix_x` 和 `pix_y` 赋值。

在 `vga_pic` 里进行模式选择，对不同模式下的所储存的显示数据进行调用，值得注意的是这里调用采用的是时序逻辑，这是因为时序逻辑能用于确保像素数据的传输与显示器的时钟同步，以便于确保数据的稳定性和正确性（这在VGA显示中十分重要）。

此外在 `vga_pic` 中

```
if(((pix_x >= `CHAR_B_H - 1) && (pix_x < `CHAR_B_H + `CHAR_W - 1))
    && ((pix_y >= `CHAR_B_V) && (pix_y < `CHAR_B_V + `CHAR_H))
    && (char[char_y][255-char_x] == 1'b1))//Need to reverse
    pix_data <= `WHITE;
```

该处代码中对所储存的数组索引进行了反转，这样的索引反转是为了匹配实际硬件而设计的。在图形处理中，有时需要对图像数据进行转换，以适应特定显示设备的扫描顺序或其他特殊的要求。

### 1.4 期待实现部分

直接采用了512bit和256bit位宽的储存器数组来记录图象信息，可能会造成资源的浪费，同时使得FPGA芯片的储存成本变高，灵活性降低，十分不利于后续开发与应用。

该项目迫于时间压力没有对其采取优化，后续可对该部分信息储存进行分块处理。

## 2. 录音功能

本功能未采用vivado中的ip核

### 2.1 设计思路

受启发于lab作业3的p10，我们选择使用一个临时的储存器数组 `reg [2:0] note_data_in [127:0]` 来记录用户的输入。

不同于作业中的自动递增逻辑，在实际的录音中需要记录演奏音符的时长，因此引出了

`reg [9:0] cnt_data_in [127:0]` 给储存器数组来记录时长。

这之后受困于录制模式中用户输入音符时长的不确定性，我们无法直接自然的将用户演奏时所改变的音符信息储存到数组的下一个索引位置，因此我们采取了一个敏感于时钟下降沿的时序逻辑语块，使用状态机的方式来进行按位延迟操作，通过比较前后音符的不同来实现索引的递进。

```
reg [6:0] now_note;
reg [6:0] next_note;
always@(posedge clk , negedge rst_n) begin
//Determine state transitions, i.e. note input transitions

if(!rst_n) begin
now_note <= 0;
next_note <= 0;
end
else if (mode == `mode_record) begin
now_note <= next_note;
next_note <= note_in;
end
end

always@(negedge clk, negedge rst_n) begin
//Index incrementally by note transformation

if(!rst_n)begin
index <= 0;
end else if (mode == `mode_record) begin
if(now_note != next_note) begin
index <= index + 1;
end
end
end
end
```

幸运的是在我们的设计中0作为合法的空拍是能被记录进去的，这样也恰好可以记录用户演奏各个音符之间时间间隔，进而达到完美的还原用户所录制的内容。

### 2.2 功能实现

```

) module record(
    input sys_clk,
    input rst_n, //deletion record
    input rst, //replay
    input [2:0] mode,
    input [6:0] note_in, //hot Note input
    output reg [2:0] note_num_out //Note output
);

```

`sys_clk`：系统时钟信号

`rst_n`：录音功能的总复位信号（控制存档的删除）

`rst`：播放录音功能下的复位信号（用来重播录音）

`mode`：用户所选择的模式（引入mode的原因是为了在同一个模块下区分录制和播放两种状态）

`note_in`：用户在录音模式下输入的音符

`note_num_out`：用户在录音模式下输出的音符

该功能采用了一个新的100Hz的时钟用于其中的所有时序逻辑部分，其原因是为了增大时钟上升沿与下降沿之间的时间间隔，避免因间隔时间过短而发生逻辑混乱（模块中部分时序逻辑敏感于上升沿，部分敏感于下降沿）。

### 2.3 期待实现的部分

该部分未实现多段录音储存和删除功能，同时也只对用户演奏的一个八度音进行储存。储存和播放的逻辑与自动演奏模式类似，其中的困难主要集中于索引的选择与递进，实现起来的难度并不大。

迫于时间压力未能实现，期待后续的补充。

## 3. 学习模式下的节奏调节

### 3.1 设计思路

通过改变时钟信号的频率，来实现判定系统反应时间的改变，进而改变整个学习状态的节奏，当用户输入正确的时候，灯会十分迅速地改变，而且对于判定失败的标准有所提升，刺激用户加快输入的节奏，以此来实现节奏的调整。

### 3.2 功能实现

```

module counter(
    input clk,
    input rst_n,
    output clk_bps
);
    reg[14:0] cnt_first, cnt_second;
    always@(posedge clk, negedge rst_n)
        if (~rst_n)
            cnt_first <= 15'd0;
        else if (cnt_first==15'd10000)
            cnt_first <= 15'd0;
        else
            cnt_first <= cnt_first + 1'b1;
    always@(posedge clk, negedge rst_n)
        if (~rst_n)
            cnt_second <= 15'd0;
        else if (cnt_second==15'd10000)

```

```

        cnt_second <= 15'd0;
    else if (cnt_first == 15'd10000)
        cnt_second <= cnt_second + 1'b1;
    else
        cnt_second <= cnt_second;
    assign clk_bps = cnt_second == 15'd10000;
endmodule

```

这段代码为其中一个计数器，将100MHZ的时钟信号转变成了1HZ的信号，以此来满足系统判定的时间需求；稍快的节奏则采取了2HZ的信号，加快了整个学习过程的进行。

## 4. 学习模式下的分数详细显示

### 4.1 设计思路

在学习状态下，通过对用户判定失误次数的统计，来将子模块的cnt信号传输到顶层模块。在顶层模板中，将cnt的值传递给数码管，根据不同的case来使state获得不同的值，以此来实现用户成绩的实时变化。

### 4.2 功能实现

```

        if (user_input[key[current_note] - 1]) begin
            leds[key[current_note] - 1] <= 1'b0;
            duration_counter <= 4'd0;
            if (current_note == `study_model_last_note)
                state <= `store_state;
            else
                current_note <= current_note + 1;
        end
    else begin
        if (duration_counter == 4'd3) begin // automatically switch after 4 s
            if (current_note == `study_model_last_note)
                state <= `store_state;
            else
                current_note <= current_note + 1;
            if (cnt == `out_cnt)
                cnt <= `out_cnt;
            else
                cnt <= cnt + 1'b1;
            leds[key[current_note] - 1] <= 1'b0;
            duration_counter <= 4'd0;
        end
        else
            duration_counter <= duration_counter + 1;
    end
end

```

这段代码为子模块中对cnt赋值的逻辑语句，if语句满足的条件是用户在学习时输入了正确的音符信号，else语句满足的条件是用户在给定的反应时间内仍没有输入正确的值，那么cnt就会加一，也就是判定失败加一次。

```

always@(posedge clk) begin
    //display for points in study mode
    if(mode==`n_fa)begin
        if(cnt<=`cnt5) state[10:8]<=`n_do;
        else if(cnt>=`cnt6&&cnt<=`cnt10) state[10:8]<=`n_re;
        else if(cnt>=`cnt11&&cnt<=`cnt15) state[10:8]<=`n_mi;
        else if(cnt>=`cnt16) state[10:8]<=`n_fa;
    end
    else state[10:8]<=3'b000;
    if(mode==`n_fa)begin

```

```
        if(low_note_control[0])begin
            state[10:8]<= `n_space;
        end
    end
end
end
```

这段在顶层模块的代码，则通过将不同cnt的值转变为不同的state值，然后通过一系列的case语句，实现不同数字在数码管中的显示。

## 七. 项目总结

刘家铭：在项目开始前先确定主模块的输入输出以及与各个子模块之间的接线关系，可以使项目总体架构清晰，也有助于后续项目的推进。第一次尝试使用verilog语言完成一个较大的项目工程使我加深了对HDL以及数字逻辑这门课中许多知识点的理解，例如利用人眼的视觉残留效应实现多个数码管的同时显示、FSM在时序逻辑电路中的重要作用等。同时，在小组合作完成项目工程的过程中，我也充分体会到了团队合作的力量和重要性，看似庞大又复杂的问题在组内成员的分工合作和沟通交流下迎刃而解。

袁龙：在整个项目之中，我体会到了verilog作为一种硬件语言与java等高级编程语言的不同之处。verilog的测试床debug往往需要通过值的设定来进行试错，并且对包括中间临时信号中的每个信号的检查都是至关重要的，当确保仿真有输出且难以看出问题的时候，上板操作出现的情况更能从实际角度反应出代码问题。

此外，在这个项目中我对时序逻辑部分的认识变得更加深刻了，在设计中体会到了敏感链表的重要所在，通过上升沿和下降沿的时序差异来避免逻辑关系的冲突，同时适当延长时钟信号能避免上板时出现的逻辑混乱问题（我在debug录音部分时，开始直接使用的系统时钟信号，在代码没有任何问题的情况下，我上板测试了许久，其中播放录音的逻辑一直十分混乱，持续时间和演奏的音符都与所期待的有很大差异，当我采用100Hz的时钟之后，这个问题就迎刃而解了）。

温云翔：通过本次digital logic的project，对于团队的合作有了更多的感悟，每个人在负责各自任务的同时，还要花费精力去考虑同伴的任务进度，尽量保证团队里面的每个人一起前进，不能有人止步不前，影响整个团队的任务进度。同时，经常的沟通可以有效避免很多的错误发生，例如：xx按键是我要用来实现xx功能的，你记得给我留出来啊。这可以提高整个团队的工作效率。在完成project之后，和团队成员的同学情谊也得到了加深，对于个人的人生发展也有着很大的意义。

## 八. 想法和建议

袁龙：用FPGA做一个贪吃蛇小游戏，数码管上显示蛇头的坐标、蛇身的长度以及豆子所在的坐标。个人感觉VGA的学习并不复杂，在传统project中的重复性工作较高（卷这个bonus感觉意义不是很大，个人觉得有这个功能就足够了）、代码过于冗长，因此可以作为新项目的必要部分。或者将整个项目分为初级版和进阶版，初级版不考虑蛇碰到蛇身子，而进阶版会有碰到蛇身子和边界就死掉的情况。这样的项目感觉对时序理解的考察更加突出，同时VGA部分不至于为拿高分而做过多相同的事情。（自己想的）

温云翔：可以将project拆解成为两个小的project，这样能分担一些期末考试的压力，可以让任务量不全部堆积在后半学期，将压力均匀分开。具体实现可参考CS307数据库原理的两次project。

刘家铭：用FPGA实现俄罗斯方块小游戏，类似本学期的另一个project原神厨房，使用按键控制下落方块的移动，同时可以要求实现用户功能和存档功能，保存当下的游戏情况并可以被读档，蜂鸣器类似本次project的自动演奏功能，可以用于实现不同游戏模式下不同的bgm，数码管显示游戏相关的信息，例如当前模式、分数等。有助于帮助学生HDL产生更加深刻且全面的理解。