关于两份基于 FPGA 的国际象棋课程项目在设计层面相似性的情况说明

- 说明材料提供人：王殊凡（浙江大学，学号：3240105701）

- 涉及课程：
  - 浙江大学《数字逻辑设计》（2024-25 春夏学期）
  - 中国科学技术大学《数字电路教程》（2025-26 秋冬学期）

一、材料目的与说明范围

1. 材料目的

本说明材料旨在**客观呈现**两份基于 FPGA 的国际象棋课程项目在设计与实现层面存在的相似性情况，供相关任课教师在充分了解事实背景的基础上，结合课程要求与学术规范进行独立判断。

本材料的提交目的在于**反映情况、提供信息**，而非对任何个人或项目作出行为性质上的定性评价。材料中所涉及的比较与分析，均基于公开或可核查的信息来源，未对相关人员的主观动机作任何推测。

2. 说明范围

本材料的说明范围主要聚焦于项目在以下层面的设计与实现情况：

- 系统整体架构与模块划分方式；
- 游戏核心逻辑与状态组织思路；
- 图像渲染与显示策略等实现层面的设计选择。

二、项目基本情况与时间线说明

1. FPGA-clock-attached-chess（我方课程项目）

- 所属学校：浙江大学
- 课程名称：《数字逻辑设计》
- 项目内容：基于 FPGA 的附棋钟国际象棋
- 公开代码仓库：https://github.com/VictorWang712/FPGA-clock-attached-chess
- 演示视频：https://www.bilibili.com/video/BV14Q7QznE1D

根据 GitHub 平台公开信息显示，项目 FPGA-clock-attached-chess 的代码仓库于 2025 年 5 月 31 日创建，并一直保持公开，最后一次 commit 时间为 2025 年 6 月 3 日。

此外，本项目的课程演示视频已上传至 B 站，平台显示的视频发布时间为 2025 年 6 月 3 日，该视频内容对项目的核心功能与整体设计进行了展示。

2. 2025-USTC-Digital-Lab-VGA（相关课程项目）

- 所属学校：中国科学技术大学
- 课程名称：《数字电路教程》
- 项目内容：FPGA VGA 实验与国际象棋游戏
- 公开代码仓库：https://github.com/wingtings/2025-USTC-Digital-Lab-VGA
- 演示视频：https://www.bilibili.com/video/BV1pPq5BqE3Q

根据 GitHub 平台公开信息显示，项目 2025-USTC-Digital-Lab-VGA 的代码仓库于 2025 年 12 月 10 日创建，并一直保持公开，最后一次 commit 时间为 2025 年 12 月 17 日。
此外，该项目的课程演示视频已上传至 B 站，平台显示的视频发布时间为 2025 年 12 月 17 日，该视频内容对项目的核心功能与整体设计进行了展示。

3. 时间顺序与说明

结合上述公开时间信息，可以观察到：

我方课程项目 FPGA-clock-attached-chess 的 GitHub 仓库公开时间与课程演示视频发布时间，均早于课程项目 2025-USTC-Digital-Lab-VGA 对应的公开时间；

上述时间信息均来源于公开平台显示的发布时间记录，可由任意第三方进行核查。

需要说明的是，公开发布时间并不必然等同于项目的实际最早完成时间。本材料中所引用的时间信息，仅用于说明两份项目在公开层面上的时间先后关系，不对项目实际开发起始时间作进一步推断。

三、课程背景与合理相似性边界说明

1. 课程背景

两门课程均为数字逻辑电路及相关实验课程，课程目标通常包括学习 FPGA 开发流程、掌握逻辑电路设计与验证方法、以及应用所学知识实现小型项目。课程大作业旨在锻炼学生独立分析问题、设计方案、实现功能和调试优化的能力。

2. 合理相似性边界

在学术环境中，基于相同课程要求和技术框架出现一定程度的相似性是合理的。例如：

- 使用 FPGA 开发工具和标准逻辑模块（如寄存器、计数器、状态机）
- 基础模块设计方法，如有限状态机控制棋子移动或棋盘显示
- 对于国际象棋游戏，实现棋盘网格显示、棋子编码及走法规则属于课程常规要求

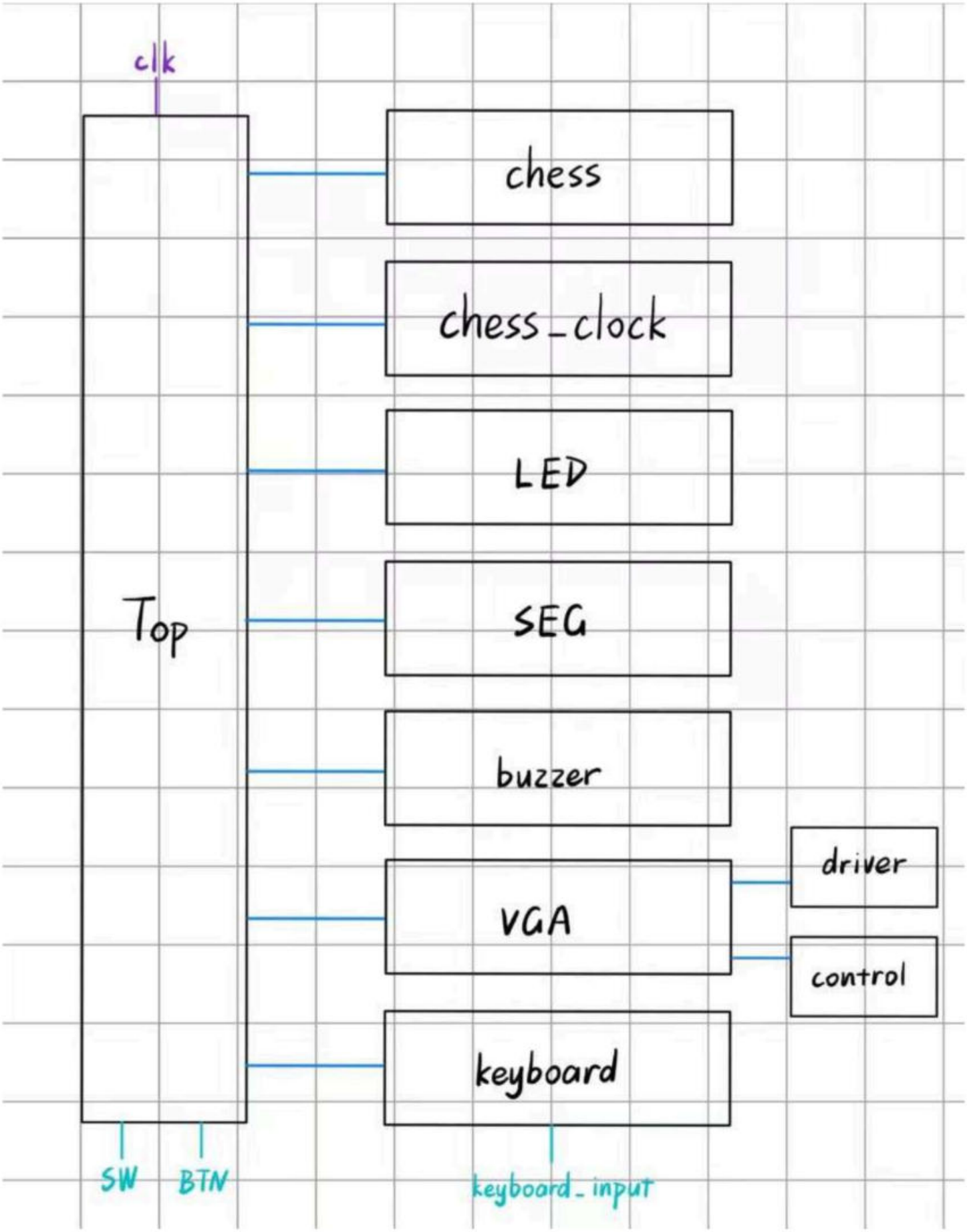
但是，合理相似性应当限于设计方法、模块类型、接口规范等一般性内容，而不应包括核心逻辑实现、具体架构设计、代码实现细节或独特的优化技巧。如果出现了对核心逻辑和具体实现的高度一致，则超出了课程内容指导下的合理相似性边界。

四、证据对比与分析

本节将具体陈述两项目的相似与一致之处。以下分析仅针对设计选择本身进行技术层面的对照描述，不对相关实现形成原因作出推断。

1. 项目代码架构的设计思路

FPGA-clock-attached-chess



这一架构图在 B 站演示视频 (https://www.bilibili.com/video/BV14Q7QznE1D) 中就出现过。

3. Part 1: VGA 基础实验

本部分主要用于学习 VGA 时序控制和图像显示原理。

3.1 核心模块

*   **`DST.v` (Display Sync Timing)**: 显示时序发生器。负责产生 VGA 标准的时序信号（`hs`, `vs`）以
*   **`DDP.v` (Display Data Processor)**: 显示数据处理器。根据当前的扫描坐标和输入数据，生成最终的

3.2 顶层模块

*   **`RGB.v`**: 基础测试模块。演示如何驱动 VGA 接口显示颜色。
*   **`VGA.v`**: 静态图片显示模块。实例化 Block RAM (`VRAM`) 存储图片数据，并通过 `DDP` 模块在屏幕
*   **`Vedio.v`**: 视频播放模块。实例化多个存储器 IP 核，通过定时器 (`timer_cnt`) 循环切换显示的图

---

4. Part 2: 国际象棋 (Chess) 游戏设计

本部分实现了一个功能完整的双人对战国际象棋游戏。

4.1 系统架构

系统采用模块化设计，顶层模块 `Chess.v` 协调输入、逻辑处理、图形渲染和音频输出。

*   **`Chess.v` (Top Module)**: 顶层模块，负责时钟分频、模块实例化及信号互联。

4.2 模块详细说明

4.2.1 顶层控制 (`Chess.v`)
*   **功能**: 系统的总入口。

4.2.2 输入系统 (`Keyboard.v`)
*   **功能**: 处理 PS/2 协议，解码扫描码并生成按键事件。

4.2.3 核心逻辑 (`Play.v`)
*   **功能**: 维护游戏状态、棋盘数据和规则判断。

4.2.4 图形渲染 (`DDP.v` & `DST.v`)
*   **`DST.v`**: 产生 640x480 @ 60Hz 的 VGA 时序。
*   **`DDP.v`**: 根据 `Play.v` 提供的 `board_data`、光标位置 `cursor_x/y` 和游戏状态 `state` 进行

4.2.5 音效系统 (`Sound.v`)
*   **功能**: 接收 `sound_code`，通过 PWM 驱动蜂鸣器播放对应的音效（如移动声、吃子声、胜利声）。

虽然很多代码文件命名不同，但核心的游戏逻辑以及总体架构都有相似性，其中**大部分**代码文件更是可以一一对应。两个项目都采用几乎相同的模块划分：

1. 顶层模块：协调所有子模块
2. 游戏逻辑模块：处理规则和状态
3. 键盘输入模块：PS/2 解码
4. 显示渲染模块：都分为时序控制和数据处理两个独立代码文件

这种特定的模块划分方式反映了高度相似的设计思维过程，其是否属于正常的独立设计结果，需结合课程教学内容与项目要求，由任课教师进一步判断。

2. 棋子图像处理方法

FPGA-clock-attached-chess

在我方项目中，这是一个核心的创新点。由于一般的 `VGA` 接口代码只默认支持 RGB 三色显示，而没有预留额外位实现透明效果，因此我方采用了：**将透明像素点全部预留为正红色（255, 0, 0），并在渲染工程中额外添加逻辑判断。这一处理并非常规处理方法，属于独特的优化技巧。

具体来说，我方项目中的 `FPGA-clock-attached-chess\project\images\mem` 文件内，可见文件开头（图片左上角）都是如下数据：

```
00F
00F
00F
...
```

即对应正红色。同时，在我方项目中的 `FPGA-clock-attached-chess\project\project.srcs\sources_1\new\vga_ctrl.v` 中，有这样的控制逻辑（568-600 行）：

```
if (in_wp0 && wp_rom_data[wp0_rlv_addr] != 12'h00F) pixel_data <= wp_rom_data[wp0_rlv_addr];
    else if (in_wp1 && wp_rom_data[wp1_rlv_addr] != 12'h00F) pixel_data <= wp_rom_data[wp1_rlv
    else if (in_wp2 && wp_rom_data[wp2_rlv_addr] != 12'h00F) pixel_data <= wp_rom_data[wp2_rlv
    else if (in_wp3 && wp_rom_data[wp3_rlv_addr] != 12'h00F) pixel_data <= wp_rom_data[wp3_rlv
    else if (in_wp4 && wp_rom_data[wp4_rlv_addr] != 12'h00F) pixel_data <= wp_rom_data[wp4_rlv
    else if (in_wp5 && wp_rom_data[wp5_rlv_addr] != 12'h00F) pixel_data <= wp_rom_data[wp5_rlv
    ...
```

即对正红色（ `00F` ）做了特殊的逻辑判断以实现透明效果。

2025-USTC-Digital-Lab-VGA

其棋子图像文件存放在 `2025-USTC-Digital-Lab-VGA\img` 中，列举几图如下：

可见其将图片应该透明显示的区域填充以绿色。同时，在该项目中的 `2025-USTC-Digital-Lab-VGA\Part2\DDP.v` 中，有这样的控制逻辑（325-336、349-360、373-384、403-414 行）：

```
case (type)
    4'b0001: rgb = (douta[0] == 12'h0F2) ? (((m-60)/60 + (n-60)/60) % 2 == 1 ? 12'h555 : 12'hC
    4'b0010: rgb = (douta[1] == 12'h0F2) ? (((m-60)/60 + (n-60)/60) % 2 == 1 ? 12'h555 : 12'hC
    4'b0011: rgb = (douta[2] == 12'h0F2) ? (((m-60)/60 + (n-60)/60) % 2 == 1 ? 12'h555 : 12'hC
    4'b0100: rgb = (douta[3] == 12'h0F2) ? (((m-60)/60 + (n-60)/60) % 2 == 1 ? 12'h555 : 12'hC
    4'b0101: rgb = (douta[4] == 12'h0F2) ? (((m-60)/60 + (n-60)/60) % 2 == 1 ? 12'h555 : 12'hC
    ...
```

即对绿色（ `0F2` ）做了特殊的逻辑判断以实现透明效果。

虽然颜色不同，但是其处理方法几乎一致。在传统 RGB 显示中，实现透明效果的方法有很多。而对图像透明部分填充特定色彩是一种不常规的方法，这一处理方式**并非 FPGA 教学中常见或标准的实现路径**，而是在工程实现中具有一定**个体选择性**的方案。

3. 光标图像的设计

FPGA-clock-attached-chess

在我方项目中，为了表示棋子被选中，采用了独立的光标图像 `FPGA-clock-attached-chess\project\images\raw\slc.png` ，如下图：

2025-USTC-Digital-Lab-VGA

在该项目中，为了表示棋子被选中，采用了独立的光标图像
`2025-USTC-Digital-Lab-VGA/refs/heads/main/img/r_box.png` ，如下图：

UI 设计本应是**方案最自由、选择最丰富**的部分。就算是现有的主流网络国际象棋平台（如 chess.com、Lichess）等，也都没有采用这种四角光标图像。

在工程实践中，当多个实现方案在细节层面高度一致时，通常需要进一步结合背景与实现路径进行审慎评估。

4. 显示器图像渲染方法

FPGA-clock-attached-chess - `FPGA-clock-attached-chess\project\project.srcs\sources_1\new\vga_ctrl.v`

我方项目中，采用了自顶向下，“棋子/光标 -> 棋盘高亮 -> 棋盘正常 -> 其余区域”的 4 层渲染逻辑。由于代码冗长，可以在 `FPGA-clock-attached-chess\project\project.srcs\sources_1\new\vga_ctrl.v` 中查阅。

2025-USTC-Digital-Lab-VGA - `2025-USTC-Digital-Lab-VGA\Part2\DDP.v`

在该项目的显示逻辑代码中，出现了以下注释：

```
(291 行)//首先看光标
...
(316 行)//然后看棋子
...
(388 行)//最后渲染棋盘
...
(419 行)//渲染侧边
```

无论是注释还是实际的代码逻辑，都可以表明该代码采用了类似的 4 层渲染逻辑。

同理，显示器的渲染方法多种多样，这种用一个完整的 4 级逻辑，自顶向下的方法更是**毫无显著性**，完全可以采用分类渲染、自底向上等其它方法。这种特定的渲染顺序并非课程中唯一或必然的实现选择，而两个项目在该层面的实现方式表现出高度一致性。

5. 棋子数据表示方法

FPGA-clock-attached-chess - `FPGA-clock-attached-chess\project\project.srcs\sources_1\new\chess.v`

```
// 棋子数据结构：8位
// [7]: 存在位 (1: 存在, 0: 被吃)
// [6]: 选中标志 (1: 选中)
// [5:3]: X坐标 (0-7)
// [2:0]: Y坐标 (0-7)
output reg [7:0] wp0, wp1, wp2, wp3, wp4, wp5, wp6, wp7;
output reg [7:0] wr0, wr1, wn0, wn1, wb0, wb1, wq, wk;
```

2025-USTC-Digital-Lab-VGA - `2025-USTC-Digital-Lab-VGA\Part2\Play.v`

```
// 棋盘寄存器堆：8x8，每个寄存器 8 位
// [4]: 有效位 (1: 有棋子, 0: 无棋子)
// [3]: 阵营 (0: 白方, 1: 黑方)
// [2:0]: 棋子类型
reg [7:0] board [7:0][7:0];
```

虽然具体声明不同，但 8 位表示棋子状态的逻辑及对应位的含义相同。这种数据结构的创新思路高度相似，不是常见的标准国际象棋表示方法。而且这些数据信息，无论是分成若干单独的变量存储，还是重新分组以不同的位宽存储，都在技术上是自然且可行的，**不存在 8 位存储存在优越性的情况**。

6. 其它零碎的细节

- 复位逻辑：都使用相似的初始化模式
- 输入处理：都使用边沿检测处理按键
- 图像坐标计算：都使用相似的算术运算

这些内容较为零碎，且分布在各处代码细节处，故归纳在此。

五、总结性说明

综上所述，本说明材料基于公开、可核查的信息，对两份基于 FPGA 的国际象棋课程项目在**项目时间线**及**若干设计与实现层面的相似性情况**进行了客观整理与对比分析。相关分析重点关注了部分非课程教材或课堂示例中必然要求的设计选择，旨在呈现事实背景，而非对具体行为作出价值判断。

需要再次强调的是，本材料仅用于**如实反映情况、提供参考信息**。相关相似性是否属于正常的独立实现，或是否涉及不当借鉴，应结合课程具体要求、教学安排以及学术规范，由任课教师进行综合评估与判断。本材料不对任何个人的主观动机或行为性质作出预设性结论。

如任课教师认为有必要进一步了解相关情况，本人愿意在合适范围内补充说明或提供已有材料以供参考。无论最终判断结果如何，本人均充分尊重并服从任课教师的专业意见。