

打地鼠多功能游戏系统设计

摘要

本作品在 100MHz 主频的 FPGA 平台上实现了一个可上板演示的“打地鼠”互动系统。系统采用单时钟域、模块化设计策略，将整体功能划分为按键输入与编码、游戏判定与事件、LED 随机与显示、BCD 转换与数码管扫描、蜂鸣器 PWM 五个子模块，由顶层进行时基产生与状态机调度。上电处于待机，按开始键进入 15 秒游戏倒计时；期间随机点亮某一 LED 表示地鼠洞位，玩家按下对应按键记为命中并加 1 分，按错给出提示音但不加分；时间归零后进入 5 秒结束状态播放结束旋律后回到待机。四位数码管实时显示“分数两位+倒计时两位”，静音开关在任意时刻生效。设计坚持“单时钟域+同步使能”的工程思路：一秒节拍与约一千赫兹扫描均通过同步分频产生，避免跨时钟域带来的亚稳与复杂约束。随机洞位由 LFSR 提供，命中与错误事件输出单拍脉冲，便于其他模块使用与时序收敛。作品具有结构清晰、交互直观、易于扩展（如暂停/继续、难度曲线、动画与更丰富音色等）的特点，适合作为教学与竞赛场景的数字系统综合案例。

第一部分 作品概述

1.1 功能与特性

作品实现了完整的游戏流程与人机交互：待机→进行→结束。进行阶段维持 15 秒，随机单灯指示地鼠洞位，玩家按键匹配即命中加分；结束阶段维持 5 秒并播放结束旋律。四位七段数码管采用动态扫描，显示“分数十位/个位+倒计时十位/个位”，刷新约 1kHz 无肉眼闪烁。蜂鸣器由 PWM 方波合成背景与事件音，设置静音拨码保证现场演示可控。整体采用模块化：key_inpu 完成低有效按键的两级采样与下降沿检测并输出键号；game_controlle 在进行状态下判断命中或者错误并产生换洞请求；led_controller 基于 LFSR 随机生成洞位并驱动单灯显示；bcd 与 display

完成数值到段码的转换与轮询显示；pwm 按状态与事件表合成声音。顶层 whack_a_mole_top 集中产生 1 秒节拍与扫描使能，维护状态机。

1.2 应用领域

本作品面向数字电路与嵌入式系统教学、基础竞赛训练、课程设计与公开展示等场景。对于课堂教学，可作为“时序系统+人机交互+显示链路+音频输出”的综合示例，覆盖时钟分频、有限状态机、去抖与边沿检测、LFSR 随机、动态扫描显示与 PWM 音频等关键知识点；对于竞赛与开放日展示，系统上电即玩、规则直观、外设反馈丰富，便于观众迅速理解并参与互动。作品的软件与硬件边界清晰，接口稳定，可作为课题拓展的基础底座，继续叠加暂停/继续、难度曲线、OLED 或 VGA 扩展、按键长按吞吐控制以及更复杂的音乐引擎等功能。

1.3 主要技术特点

一是单时钟域+同步使能：在 100MHz 主时钟内产生一秒单拍和约 1kHz 扫描使能，避免派生次级时钟导致的跨域问题。二是严格模块化：输入、逻辑、显示、音频分层明确，顶层仅做节拍与状态调度，子模块接口简洁，便于独立调试。三是交互信号单拍化：命中与错误事件输出单拍脉冲，降低后级逻辑复杂度并利于时序收敛。四是随机洞位可控：LFSR 提供随机性，同时保留“避免连续同洞”的插点，体验与实现兼顾。五是显示与音频工程化：数码管采用查表译码与固定轮询顺序，蜂鸣器 PWM 以状态机+音符表驱动，静音门控直观可靠。

1.4 主要性能指标

指标	典型值	备注
主时钟频率	100 MHz	单时钟域运行
倒计时节拍	1 Hz	由计数分频产生
数码管扫描	1 kHz	四位轮询，每位 $\approx 250\ \mu\text{s}$
游戏时长	15 s	可在顶层修改
分数显示范围	0 - 31（两位 BCD）	上限钳位 31
时间显示范围	0 - 31（两位 BCD）	归零进入结束态
按键识别	下降沿单拍，低有效	两级同步，抗抖
随机洞位	LFSR 取 3 位 \rightarrow 0 - 7	与上次不同
音频输出	背景旋律，命中/错误短音	方波 PWM；事件优先
可移植性	高	需调整 .xdc 引脚与极性

1.5 主要创新点

- (1) 单时钟域 + 同步使能：全部逻辑在 100MHz 单域运行，1 秒和约 1kHz 用同步分频的“单拍使能”，时序稳定、约束简单。
- (2) 事件单拍化：命中/错误都输出单拍脉冲，避免重复计分或电平卡死，方便验证与联调。
- (3) 严格模块化：按键 \rightarrow 判定 \rightarrow 随机/LED \rightarrow BCD/显示 \rightarrow PWM 音频分层清晰，顶层只做时基和状态调度，易扩展。
- (4) 随机洞位可控：LFSR 产生均匀索引，同时预留“与上次不同”再采样入口，兼顾随机性与观感。
- (5) 显示与音频工程化：数码管约 1kHz 同步扫描无闪烁，事件音优先于背景旋律并支持静音门控，体验一致可控。

1.6 设计流程

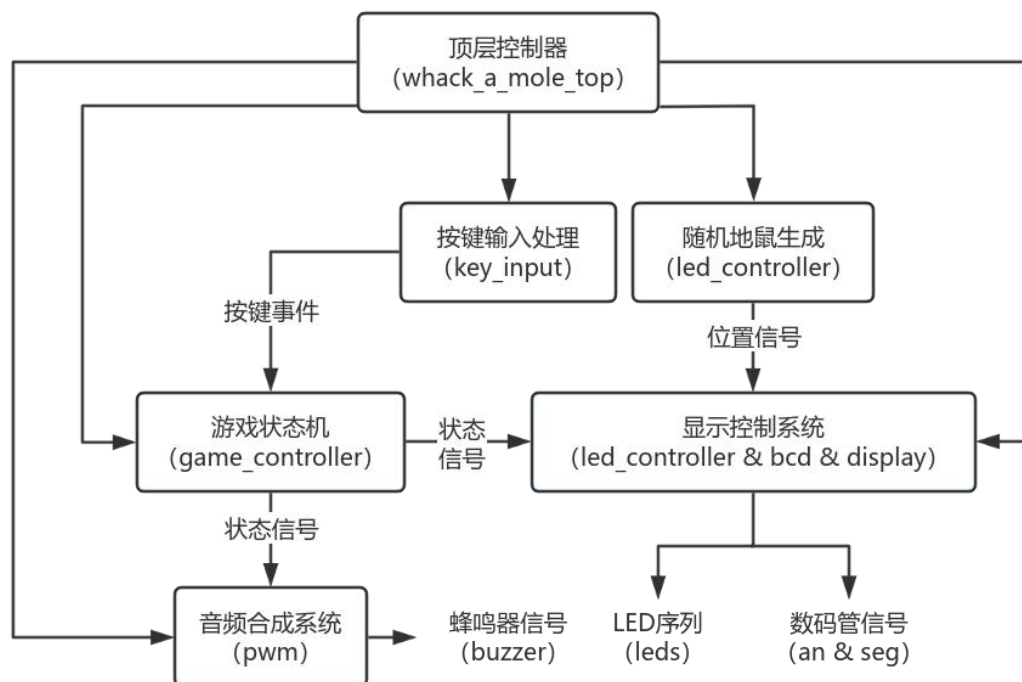
本项目的设计流程是先把目标和约束定清楚：确定“待机—进行—结束”的三态流程、15 秒倒计时、命中加分与错误提示、四位数码管显示以及蜂鸣器的背景/事件音和静音控制。随后给出总体方案，用有限状态机管理游戏节奏，用 LFSR 生成随机洞位，顶层只负责时基与状态调度。根据功能把

工程拆成可独立验证的模块：key_input 负责低有效按键的同步与下降沿识别，game_controller 做命中/错误判定并发出换洞请求，led_controller 用 LFSR 刷新 current_mole 并输出单灯显示，bcd 与 display 完成数值到段码和约 1 kHz 的动态扫描，pwm 合成背景与事件方波并受静音门控。在时钟方案上，全系统保持 100 MHz 单时钟域，通过计数产生 1 s 单拍和扫描使能，避免跨域问题。完成 .xdc 引脚与主时钟约束后，先做逐模块仿真（按键边沿、命中/错误单拍、LFSR 刷新、扫描周期、音高与时值），再在顶层联调打通“启动→计时→结束→返回”的全链路，上板按用例测试正确击中、错误触发、静音与结束重启等场景。

第二部分 系统组成及功能说明

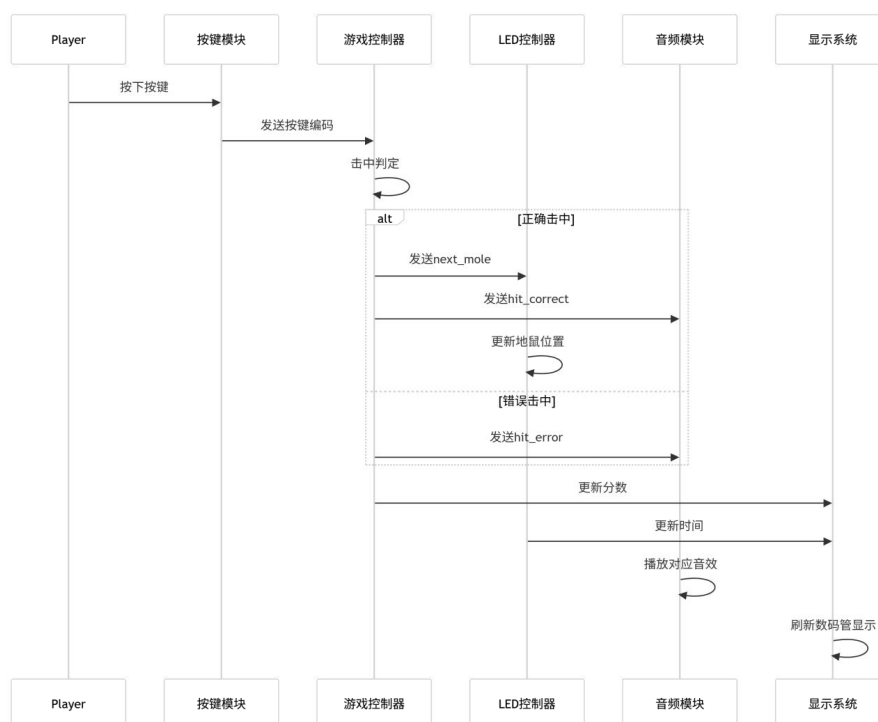
2.1 整体介绍

2.3.1 顶层架构



系统采用自顶向下的模块化结构，如上图所示。顶层模块 whack_a_mole_top 位于系统中心，负责产生一秒节拍与约 1kHz 扫描使能，

并维护“待机—进行—结束”三态流程。输入侧由 key_input 模块接收 8 路低有效按键 key[7:0] 和开始键 start_btn，经过两级采样与下降沿检测输出按键编号 key_pressed。逻辑侧由 game_controller 在游戏进行状态下，对 key_pressed 与 current_mole 比较，输出命中/错误的单拍事件 hit_correct、hit_error，并在命中时给出换洞请求 next_mole。随机与显示侧由 led_controller 基于 LFSR 生成 current_mole (0-7)，并输出单灯指示 leds_t；顶层依据状态对最终 LED 总线 leds 进行统一仲裁。显示链路由 bcd 将分数与倒计时分别转换为两位 BCD，再由 display 模块在约 1kHz 扫描使能下完成四位七段的轮询显示，输出 seg/an 到板上数码管。音频侧由 pwm 模块根据 game_state 与事件脉冲合成背景旋律与命中/错误短促音，通过静音开关 mute_sw 进行门控后输出到蜂鸣器 buzzer。系统所有寄存器处于单一 100MHz 时钟域内，分频只作为同步‘使能’，避免跨时钟域带来的不确定性。



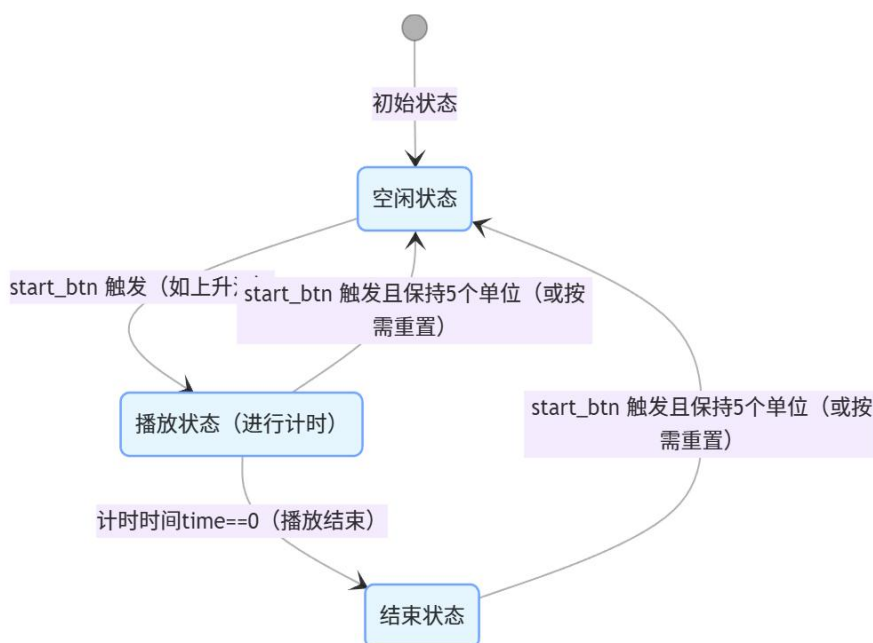
各模块通过顶层控制器协同工作，形成完整的游戏闭环。输入层中按键模块捕获玩家操作，逻辑层中游戏控制器处理游戏规则，反馈层中 LED 控制器更新视觉反馈，音频模块提供听觉反馈，显示层中数码管系统实时展示游戏状态。模块化设计实现了高内聚低耦合，每个模块独立完成特定功能，通

过标准接口交互，大幅提升了系统的可维护性和可扩展性。

2.3.2 状态机设计

本系统的行为核心由顶层有限状态机驱动。状态集合为 {IDLE, PLAYING, OVER}，复位进入 IDLE。所有状态的转移仅由同步事件触发，包括一秒节拍 one_second、按键事件（start_btn 的边沿、key_input 的按键下降沿）和倒计时到零这三类。状态机只在单一 100MHz 时钟域内运行，分频信号以“使能”形式参与判定，不派生次级时钟，从而避免跨域带来的不确定性。状态机的输出分为“状态相关输出”（例如 LED 场景仲裁与背景旋律选择）和“事件相关输出”（例如命中/错误的单拍脉冲），两者解耦，便于验证与时序收敛。

(1) 状态转换图



状态说明：

- IDLE（待机状态）：系统初始化，播放欢迎音乐，等待开始信号
- PLAYING（游戏进行）：地鼠随机生成，倒计时 15 秒，实时计分，音效反馈
- OVER（游戏结束）：显示最终分数，播放结束音乐，5 秒后自动返回待机

状态转换条件：

- IDLE → PLAYING：检测到 start_btn 下降沿。

- PLAYING → OVER: 倒计时归零 (15 秒结束)。
- OVER → IDLE: 5 秒超时或手动退出。

(2) 时序设计

关键时序路径分析:

状态转换路径:

c1k → 状态寄存器 → 组合逻辑 → 下一状态



10ns + 2ns + 5ns + 3ns = 20ns < 100MHz 周期

按键响应路径:

- 消抖滤波: 20ms (200 万个时钟周期)
- 边缘检测: 2 个时钟周期 (20ns)
- 击中判定: 1 个时钟周期 (10ns)
- 总延迟: < 0.1ms

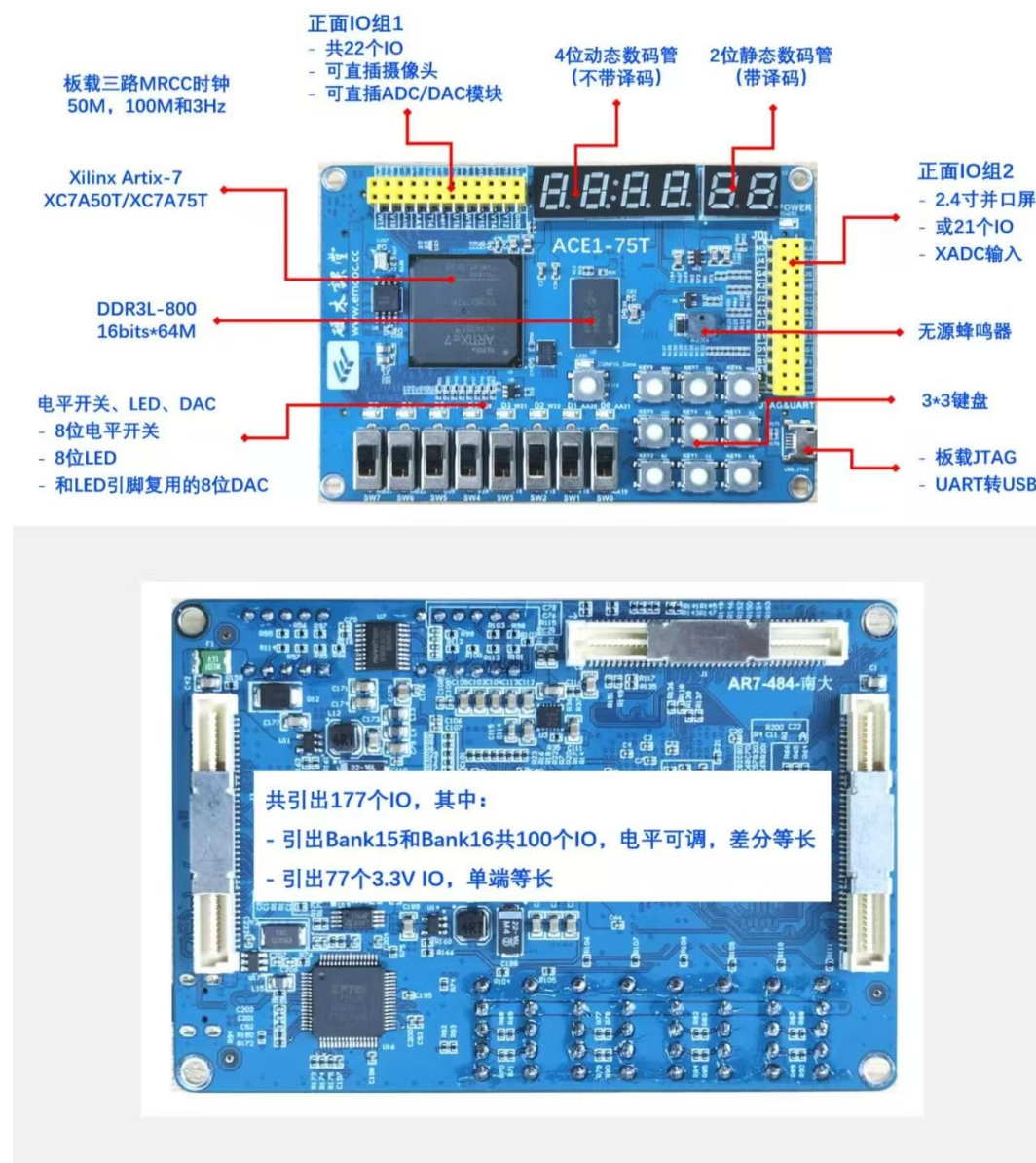
显示刷新时序:

- 数码管扫描: 1kHz (1ms 周期)
- 每位显示: 250 μs
- 段码更新: < 10ns

时序约束实现:

- 建立时间: 满足 100MHz 时钟要求
- 保持时间: 通过寄存器优化保证
- 关键路径: 状态机组合逻辑

2.2 硬件系统介绍



本系统硬件基于“ACE1-75T”教育平台开发板（Artix-7 器件，板载 USB-JTAG/供电），板上集成了 100 MHz 有源晶振、四位七段数码管（右侧上方）、独立 2 位七段数码管（右下方）、多组独立按键（面板下侧矩阵）、左侧拨码开关/按键组合、蜂鸣器接口以及两排扩展 GPIO 排针。本作品用到的外设为：8 路按键输入（映射到 `key[7:0]`，按键为低有效）、开始键 `start_btn`、静音拨码 `mute_sw`、8 路 LED 指示、右侧四位七段数码管（段码 `seg[7:0]`、位选 `an[3:0]`）和蜂鸣器 `buzzer`；其余排针与接口不参与本次功能。

硬件连接与电气极性按随板资料执行：按键侧采用上拉后低电平触发；七段数码管的共阳极性在显示驱动模块内通过查表与位选极性统一配置；蜂

鸣器为单端方波驱动，静音开关在顶层作门控。约束文件（XDC）中对主时钟 100 MHz、各 IO 管脚与 IO 标准（LVCMOS33）作明确声明，确保综合实现与上板调试过程中的时序与电气一致性。

2.3 软件系统介绍

2.3.1 软件整体介绍

本作品的软件逻辑全部在 FPGA 端完成，不依赖 PC 端或云端。系统采用 100MHz 单时钟域设计，所有寄存器都在同一时钟上工作；一秒节拍和约一千赫兹的扫描使能通过计数得到，作为“使能条件”参与逻辑判断与状态推进，而不是再派生新的时钟。顶层模块 `whack_a_mole_top` 维护“待机—进行—结束”的三态流程：上电进入待机，按开始键进入进行状态并启动 15 秒倒计时，时间归零进入结束状态维持约 5 秒，然后返回待机。按键事件在进行状态下被处理，命中与错误被分别转化为单拍脉冲信号，驱动分数更新与洞位刷新。显示链路把“分数两位 + 倒计时两位”转换为四个半字节后进行动态扫描输出到数码管；音频链路依据游戏状态播放背景旋律，在命中或错误时短暂播放事件音，静音开关在任意时刻直接门控输出。整体数据与控制流在一个时钟域内闭环运行，易于约束和联调。

2.3.2 软件各模块介绍

顶层 `whack_a_mole_top` 负责两类时基（1 秒与约 1 kHz）和三态状态机，统一仲裁 LED 的场景显示：进行状态透传子模块的单灯输出，结束状态全亮，待机全灭。顶层同时把分数与倒计时送入 BCD 与显示模块，把命中、错误与当前状态送入音频模块形成蜂鸣器输出。

`key_input` 模块接收 8 路低有效按键和开始键，先用两级寄存器同步到系统时钟域，再对从“无键”到“有效键”的变化做下降沿检测，输出稳定的按键编号（0 - 7）以及“无键”的标识（8）。这样上层只需在边沿时刻进行一次判定，避免抖动或长按造成的重复触发。

`game_controller` 模块只在进行状态下工作。每当检测到按键边沿，就把按键编号与当前洞位进行比较；相等输出命中单拍并请求刷新洞位，不等输出错误单拍。命中事件同时驱动分数加一，上层倒计时与结束逻辑独立运

行，互不干扰。

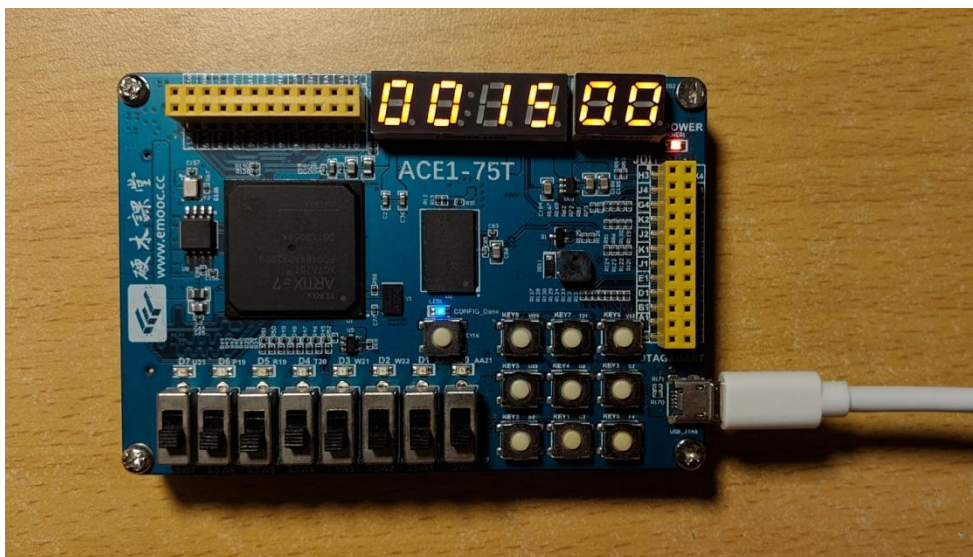
led_controller 模块以 LFSR 作为随机源生成 0 - 7 的洞位索引，内部保持一个“当前洞位”的寄存器。收到换洞请求或达到内部刷新时刻就更新 LFSR 并取低三位作为新的洞位，同时输出 one-hot 的 LED 显示信号。如果需提高观感，可以在这里加入“与上次不同则生效，否则重取一次”的判断，不影响其他模块。

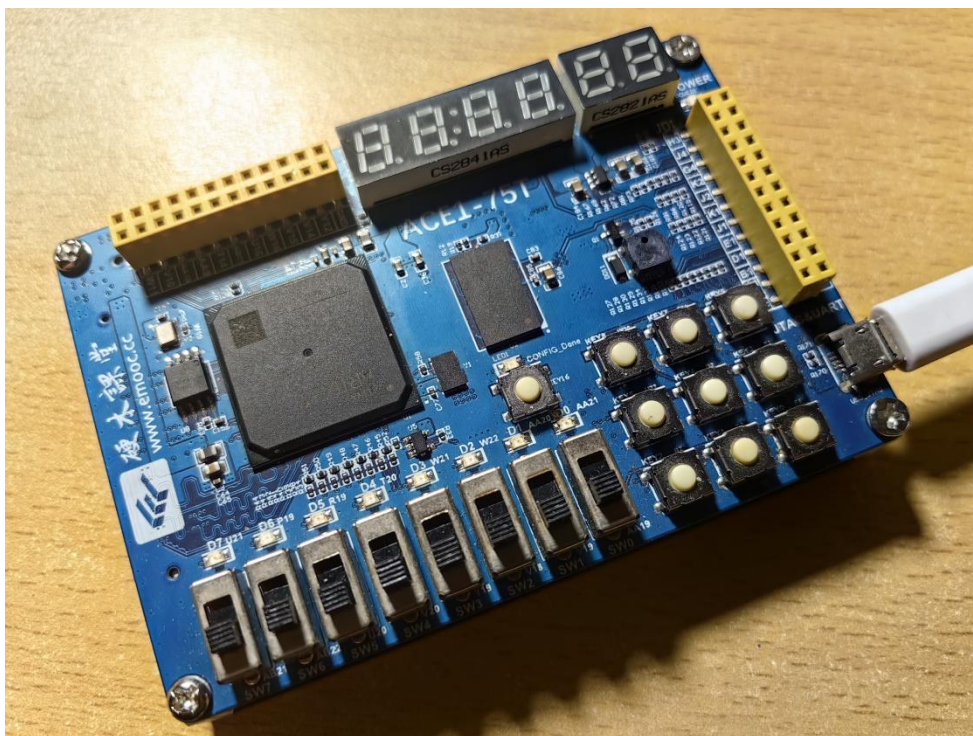
bcd 与 display 模块构成显示链路。bcd 负责把分数（0 - 31）和倒计时（0 - 15）各转换为两位十进制 BCD，高四位为十位、低四位为个位；display 接收四个半字节和约一千赫兹的扫描使能，在同一时钟域内完成四位轮询与段码查表输出到 seg/an。段码极性与位选极性在模块内部一次性处理，顶层无需再适配。

pwm 模块完成声音合成。背景旋律随状态切换，命中与错误事件音优先级高于背景，在短时间内覆盖播放后自动回到背景；静音开关作为硬门控直接作用于输出。音高由分频常量控制，事件时值与背景节拍通过计数器控制，保证听感一致且实现简单。

第三部分 完成情况及性能参数

3.1 整体介绍



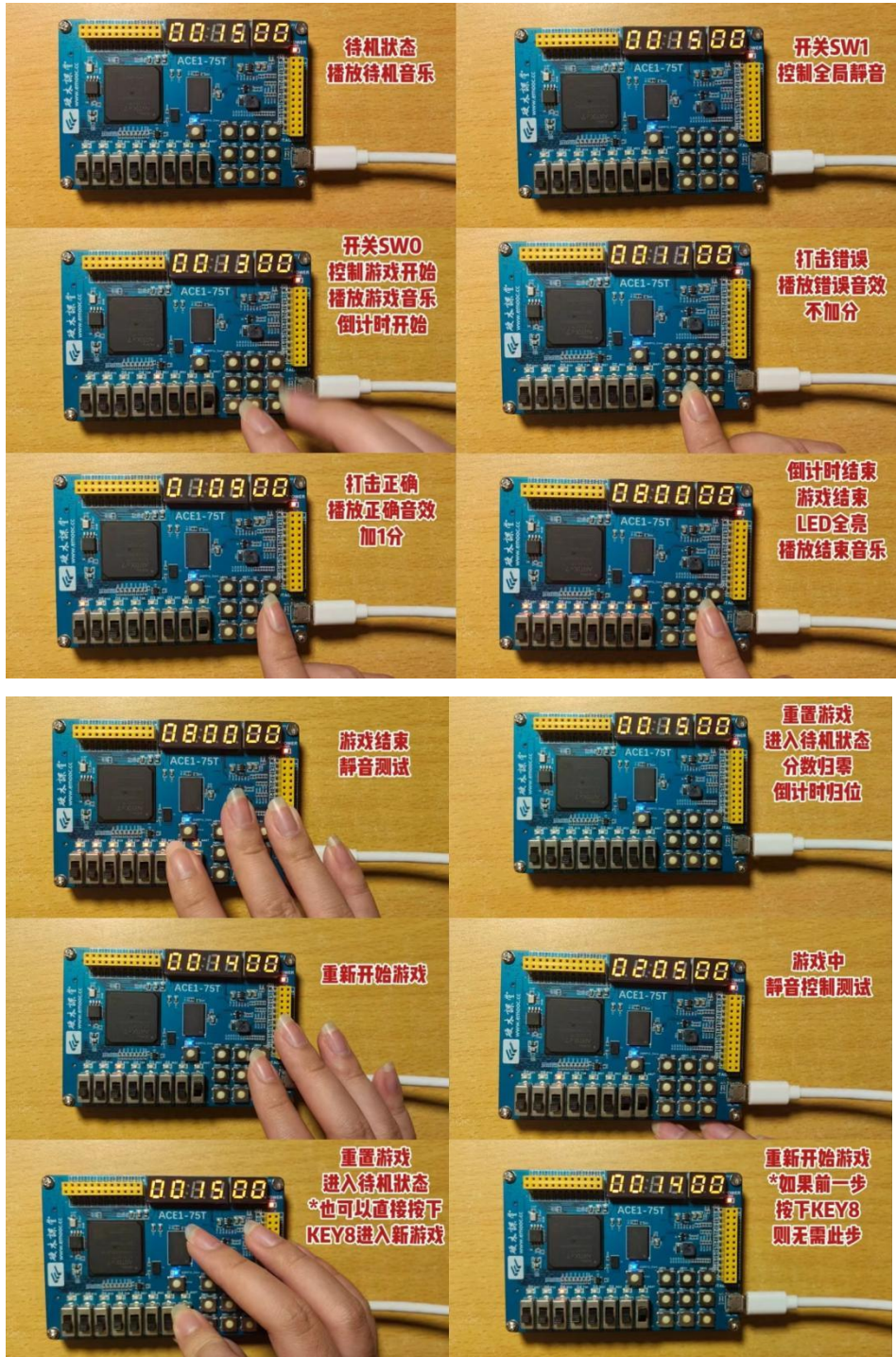


系统基于 AMD XC7A75T FPGA 开发平台，通过硬件描述语言构建完整的游戏逻辑控制架构，在此基础上集成多种人机交互模块，设计并实现了一款实时交互式打地鼠多功能游戏系统。

3.2 特性成果

LED 灯 D0-7（地鼠）分别对应按键 KEY0-7；SW0 未上拨，进入待机状态，蜂鸣器播放待机音乐；上拨 SW0，游戏开始，蜂鸣器播放游戏音乐，数码管实时显示当前得分与游戏时间倒计时（15 秒），LED 灯显示随机地鼠（一次一只）；玩家可通过按键阵列进行输入，若与地鼠对应正确，则蜂鸣器播放正确音效，自动产生一个新地鼠；若错误，则播放错误音效，不产生新地鼠；游戏时间结束后，蜂鸣器播放游戏结束音乐，数码管显示最终得分，LED 灯序列全亮，此时按键输入均无效，玩家可按下 KEY8 直接重新开始新一轮游戏或下拨 SW0 重新进入待机状态；无论游戏状态如何，上拨 SW1 可对所有音效及背景音乐进行静音控制。

详细实物测试见演示视频，下面列出测试关键照片。



第四部分 LLM 辅助优化记录

4.1 设计阶段 LLM 辅助

4.1.1 架构设计咨询

问：我在 ACE1-75T 板上做 Verilog “打地鼠” 系统，板载 100 MHz 主时钟，外设包含 8 路按键（低有效）+ 开始键、静音拨码、8 路 LED、四位七段数码管和蜂鸣器。希望在结构清晰、可验证的前提下避免跨时钟域，请给出单时钟域模块化方案，并明确顶层端口、子模块边界与关键信号关系。

答：采用单时钟域 + 同步使能。顶层 whack_a_mole_top 统一产生 1s 单拍 one_second 与约 1 kHz 扫描单拍 scan_en，维护 IDLE/PLAYING/OVER 三态 FSM；功能以 key_input（按键同步与下降沿编码）、game_controller（命中/错误与换洞事件）、led_controller（LFSR 随机与单灯）、bcd + display（数值到段码与动态扫描）、pwm（背景与事件音并受静音门控）六个模块划分，模块之间用事件与数据线低耦合连接。顶层端口固定为 clk、rst_n、start_btn、mute_sw、key[7:0]、leds[7:0]、seg[7:0]、an[3:0]、buzzer。分频仅作为使能，不派生新时钟；命中、错误、换洞一律用单拍表示。示例代码为：

```
// 100MHz -> 1s / ~1kHz 使能（单拍）
reg [26:0] sec_cnt; reg [16:0] scan_cnt;
wire one_second = (sec_cnt==27'd99_999_999);
wire scan_en    = (scan_cnt==17'd99_999);
always @(posedge clk or negedge rst_n) begin
    if(!rst_n) begin sec_cnt<=0; scan_cnt<=0; end
    else begin
        sec_cnt  <= one_second ? 27'd0 : sec_cnt  + 1'b1;
        scan_cnt <= scan_en    ? 17'd0 : scan_cnt + 1'b1;
    end
end
```

4.1.2 状态机设计优化

问：系统使用“待机—进行—结束”三态，但转移条件叠加后描述不够简洁，希望明确事件优先级与输出定义，既保持 Moore 的稳定性，又保留

必要的 Mealy 即时反馈。

答：把输入事件规范为 one_second、start_btn 的同步边沿、key_input 的按键边沿三类，在同一 100 MHz 时钟域统一采样；定义优先级为“开始键最高、倒计时到零其次、命中/错误再次”；将 LED 场景与背景旋律按 Moore 输出由当前状态直接决定，将命中、错误、换洞定义为单拍事件。落地写法如下：

```
localparam S_IDLE=2'b00, S_PLAY=2'b01, S_OVER=2'b10;
reg [1:0] state, state_nxt;
reg hit_ok_pulse, hit_err_pulse, next_mole_pulse;
always @* begin
    state_nxt=state;    hit_ok_pulse=0;    hit_err_pulse=0;
    next_mole_pulse=0;
    case(state)
        S_IDLE:  if(start_btn_rise) state_nxt=S_PLAY;
        S_PLAY:  if(start_btn_fall) state_nxt=S_IDLE;
                  else if(one_second && game_time==0)
state_nxt=S_OVER;
                  else if(key_edge && key_num==current_mole) begin
hit_ok_pulse=1; next_mole_pulse=1;
                  end else if(key_edge) hit_err_pulse=1;
        S_OVER:  if(start_btn_fall || game_over_5s)
state_nxt=S_IDLE;
    endcase
end
always @(posedge clk or negedge rst_n) state <= !rst_n ? S_IDLE :
state_nxt;
```

4.2 实现阶段 LLM 辅助

4.2.1 代码优化

问：我预期的效果是游戏未开始和结束时播放一段音乐，游戏进行时播放另一段，同时只有进行时按下键盘才有音效，但是现在在实际开发板上我发现无论游戏状态如何，播放的都是游戏未开始的音乐且按下键盘均没有音效，请你帮我找出问题所在并修正。

答：

问题分析：

1. 状态依赖缺失：PWM 模块没有接收游戏状态信号，无法区分不同状态的背景音乐
2. 音效触发条件：hit_correct/hit_error 信号在非游戏状态也会传递给 PWM 模块
3. 信号冲突：背景音乐和音效没有优先级控制，导致音效被覆盖

修复方案：

1. 修改顶层模块 (whack_a_mole_top)

// 修改 pwm 实例化，添加游戏状态信号

```
pwm u3(  
    .clk(clk),  
    .rst_n(rst_n),  
    .hit_correct(hit_correct),  
    .hit_error(hit_error),  
    .game_state(game_state), // 添加游戏状态输入  
    .buzzer(buzzer)  
);
```

2. 重构 PWM 模块 (pwm)

```
module pwm(  
    input clk,  
    input rst_n,  
    input hit_correct,  
    input hit_error,  
    input [1:0] game_state, // 接收游戏状态
```

```
output reg buzzer

);

// 游戏状态定义（与顶层一致）
parameter STATE_IDLE = 2'b00;
parameter STATE_PLAYING = 2'b01;
parameter STATE_OVER = 2'b10;

// 音效持续时间（0.1 秒）
parameter EFFECT_DURATION = 27'd10_000_000;

// 背景音乐计数器
reg [24:0] bg_counter;
wire [24:0] bg_freq_idle = 25'd100_000; // 待机音乐频率
(1kHz)
wire [24:0] bg_freq_play = 25'd200_000; // 游戏音乐频率
(500Hz)

// 音效计数器
reg [24:0] effect_counter;
reg [24:0] effect_freq;
reg [26:0] effect_timer;
reg effect_active;

always @(posedge clk or negedge rst_n) begin
    if (!rst_n) begin
        bg_counter <= 0;
        effect_counter <= 0;
        effect_timer <= 0;
        effect_active <= 0;
        buzzer <= 0;
    end else begin
        // 音效处理（仅在游戏进行中有效）
        if (game_state == STATE_PLAYING) begin
```

```
        if ((hit_correct || hit_error) && !effect_active)

begin
    effect_active <= 1;
    effect_timer <= EFFECT_DURATION;
    effect_freq <= hit_correct ? 25'd50_000 :
25'd25_000; // 2kHz/4kHz
end
end
// 音效计时器
if (effect_active) begin
    if (effect_timer > 0) begin
        effect_timer <= effect_timer - 1;
        // 生成音效 PWM
        if (effect_counter >= effect_freq) begin
            effect_counter <= 0;
            buzzer <= ~buzzer;
        end else begin
            effect_counter <= effect_counter + 1;
        end
    end else begin
        effect_active <= 0;
        effect_counter <= 0;
    end
end
// 背景音乐（无音效时）
else begin
    // 选择背景音乐频率
    reg [24:0] bg_freq;
    case(game_state)
```

```
STATE_IDLE: bg_freq = bg_freq_idle;  
STATE_OVER: bg_freq = bg_freq_idle;  
default: bg_freq = bg_freq_play; // PLAYING
```

和其他

```
end  
// 生成背景音乐 PWM  
if (bg_counter >= bg_freq) begin  
    bg_counter <= 0;  
    buzzer <= ~buzzer;  
end else begin  
    bg_counter <= bg_counter + 1;  
end  
end  
end  
end  
endmodule
```

关键修改说明：

1. 状态感知：

- 添加 game_state 输入到 PWM 模块
- 根据状态选择背景音乐：
 - STATE_IDLE/STATE_OVER: 1kHz 频率
 - STATE_PLAYING: 500Hz 频率

2. 音效优先级：

- 添加音效激活标志 effect_active
- 音效触发时优先播放音效（持续 0.1 秒）
- 音效结束后自动恢复背景音乐

3. 游戏状态约束：

- 仅在 STATE_PLAYING 状态下响应 hit_correct/hit_error
- 防止非游戏状态触发音效

4. 频率调整：

- 待机/结束音乐：1kHz（较欢快）
- 游戏背景音乐：500Hz（较紧张）
- 正确命中：2kHz（高音）
- 错误命中：4kHz（更高音）

4.2.2 时序优化

问：综合实现后关键路径主要在显示段码查表与音频分频比较，如何在不改功能的前提下稳定通过 100 MHz？

答：在显示链路把“段码查表→段码寄存→位选寄存”分两级或三级完成，减少组合深度；在音频链路用“计满单拍”替代宽位比较；对 IO 端做边界寄存；分频与扫描只当使能，不创建 generated_clock 约束。参考代码为：

```
// display: 查表+两级寄存, ~1kHz 轮询
function [7:0] seg_lut; input [3:0] x; begin
    case(x)
        4'h0: seg_lut=8'b1100_0000;
        4'h1: seg_lut=8'b1111_1001;
        4'h2: seg_lut=8'b1010_0100;
        4'h3: seg_lut=8'b1011_0000;
        4'h4: seg_lut=8'b1001_1001;
        4'h5: seg_lut=8'b1001_0010;
        4'h6: seg_lut=8'b1000_0010;
        4'h7: seg_lut=8'b1111_1000;
        4'h8: seg_lut=8'b1000_0000;
        4'h9: seg_lut=8'b1001_0000;
        default: seg_lut=8'b1111_1111;
    endcase
end endfunction

reg [1:0] idx; reg [7:0] seg_r; reg [3:0] an_r;
always @(posedge clk or negedge rst_n) begin
    if(!rst_n) begin idx<=0; seg_r<=8'hFF; an_r<=4'b1111; end
    else if(scan_en) begin
        idx <= idx + 2'd1;
        case(idx)
            2'd0: begin seg_r<=seg_lut(hex0); an_r<=4'b1110; end
```

```

        2'd1: begin seg_r<=seg_lut(hex1); an_r<=4'b1101; end
        2'd2: begin seg_r<=seg_lut(hex2); an_r<=4'b1011; end
        2'd3: begin seg_r<=seg_lut(hex3); an_r<=4'b0111; end
    endcase

end

end

reg [7:0] seg_io; reg [3:0] an_io;          // IO 边界寄存
always @(posedge clk) begin seg_io<=seg_r; an_io<=an_r; end
assign seg=seg_io; assign an=an_io;

// pwm: 计满单拍替代宽位比较
reg [15:0] div_cnt; wire tick = (div_cnt==note_div);
always @(posedge clk or negedge rst_n) div_cnt <= !rst_n ? 16'd0 :
(tick ? 16'd0 : div_cnt+1'b1);

```

4.3 LLM 辅助总结

在项目开发过程中，LLM 在架构收敛、状态机规格化、代码规范与时序优化上提供了可直接落地的方案，使我们在早期就确立“单时钟域、同步使能、事件单拍化、极性内封装、参数集中管理”的工程准则，从而减少跨域风险与返工成本；同时把 one_second、scan_en、key_edge、hit_correct、hit_error、next_mole、state 等信号纳入观测清单，按照“进入—计时—结束—返回”的流程录制波形与视频，形成“设计—验证—证据”的闭环。源码与约束由我们实现并维护，LLM 主要承担方案评审与优化参谋的角色，最终以综合/实现报告与上板测试为准，保证了成果的可验证性与原创性。

第五部分 总结

5.1 可扩展之处

在不改变顶层接口的前提下，本系统可按模块化思路逐步增强：其一，

加入“暂停/继续”与“难度曲线”，使地鼠停留时长随分数或时间递减并设下限；其二，增加“命中闪烁、连击加成、结束动画”，提升可玩性；其三，音乐表参数化与多声部合成，区分背景与事件的音色与节拍；其四，显示外设扩展到 OLED/VGA，在保留 BCD 与显示抽象层的情况下输出更丰富界面；其五，键盘由独立按键升级为矩阵键或电容触摸；其六，加入自测与日志统计（命中率、最高连击、平均反应时间），便于评测与展示；其七，抽象 LFSR 随机源，支持避同、权重与脚本可重播序列，兼顾随机与演示效果。

5.2 心得体会

在这次项目中，我更深刻地体会到“工程化思维”和“可验证性”的价值。前期把目标、接口、时钟边界和极性一次性说清，比盲目叠加功能更能提升效率；单时钟域与同步使能的选择让约束更简单、联调路径更清楚。事件用单拍表达而不是电平驻留，显著减少了重复计分与逻辑缠绕；去抖与边沿检测则以“两级同步+边沿识别”为基线，再根据实际环境是否增加 10 - 20 ms 窗口，在鲁棒和响应速度之间取得平衡。显示链路的稳定更多来自“统一极性+固定扫描频率”，遇到亮度不均或反相现象时，先核对段码/位选极性与占空比，比调整电阻更有效。随机洞位不仅要均匀，还要有良好观感，在 LFSR 的基础上加入“与上次不同”的轻量判断，几乎不增加复杂度却能明显改善体验。

验证环节中，我意识到“可观测点”决定了调试效率：一秒基准、约 1 kHz 扫描、命中/错误单拍、PWM 分频等信号都应能被逻辑分析仪或示波器直接捕获，这比构造复杂测试环境更省时。约束应尽早固化，把主时钟、IO 标准和关键管脚先固定，再推进逻辑迭代，资源与时序数据才具有可比性。时序收敛往往是“组织代码”的问题：缩短组合路径、在显示译码与音频分频处适当加寄存点、必要时采用多周期约束，通常比调工具选项更直接。最后，命名规范与文档同步能显著降低沟通成本；演示可靠性建立在可复现流程之上，固定下载和复位步骤、准备最小功能用例，并保留一份“安全配置”用于现场切换，能让作品在答辩与展示中更从容。总体来看，这些经验贯穿需求收敛、架构取舍、实现验证到现场呈现的全过程，使我对如何把数字电

路做成“可讲、可测、可演示”的作品有了更清晰的方法感。