2024 Hardware Design and Lab. Final Project Report

It’s On Heat :

An FPGA-Based Interactive Sports Gaming Device

Combining Hardware, Software, and Thematic Design for an Immersive Experience

Group 35

110000229 余佳軒、110000224 張舜涵

National Tsing Hua University, Computer Science Department

A. Introduction

In this project, we present an innovative sports gaming device combining FPGA technology with interactive hardware, inspired by the magical world of Harry Potter. Our device aims to provide a dynamic and engaging user experience through the integration of game logic implemented in Verilog and physical interaction facilitated by custom-designed game tiles, designed to detect player input. This creates an immersive gaming environment by enhancing interactivity.

Titled "It's On Heat," the game challenges players to avoid fire tiles and capture the Golden Snitch, with gameplay mechanics reflecting the Harry Potter theme. Progression is guided by INIT, PLAY, and FINISH states.

By merging hardware and software, this project highlights a holistic approach to developing interactive gaming devices, focusing on engagement and innovation.

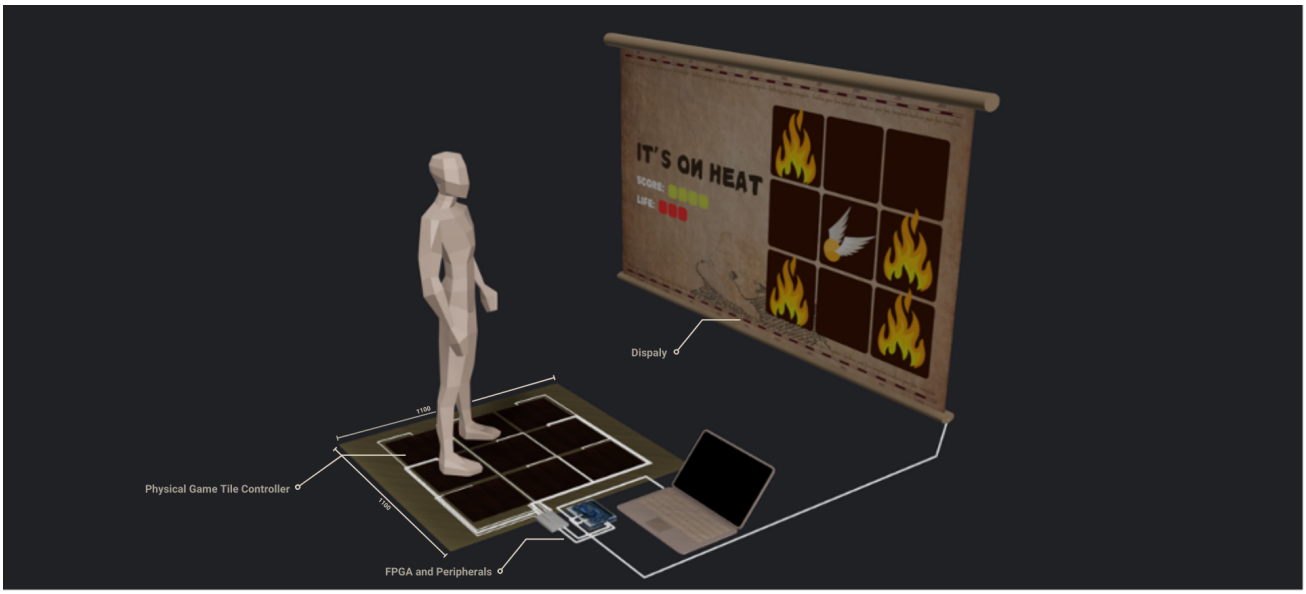

Fig.1 Game Hardware Design Overview

B. System Design

a. System Architecture Overview

The system integrates an FPGA as the core processing unit, orchestrating complex game logic and rendering visual outputs through a VGA display module. Supporting hardware includes an Arduino Nano 33 IoT, dedicated to managing sound playback and enhancing the audio experience. The physical game tiles, acting as the primary input mechanism, are designed with custom circuits to ensure accurate signal detection and reliable player interaction. These tiles incorporate durable materials and optimized designs to handle repeated physical stress, while their connection to the FPGA ensures seamless communication and responsive gameplay.

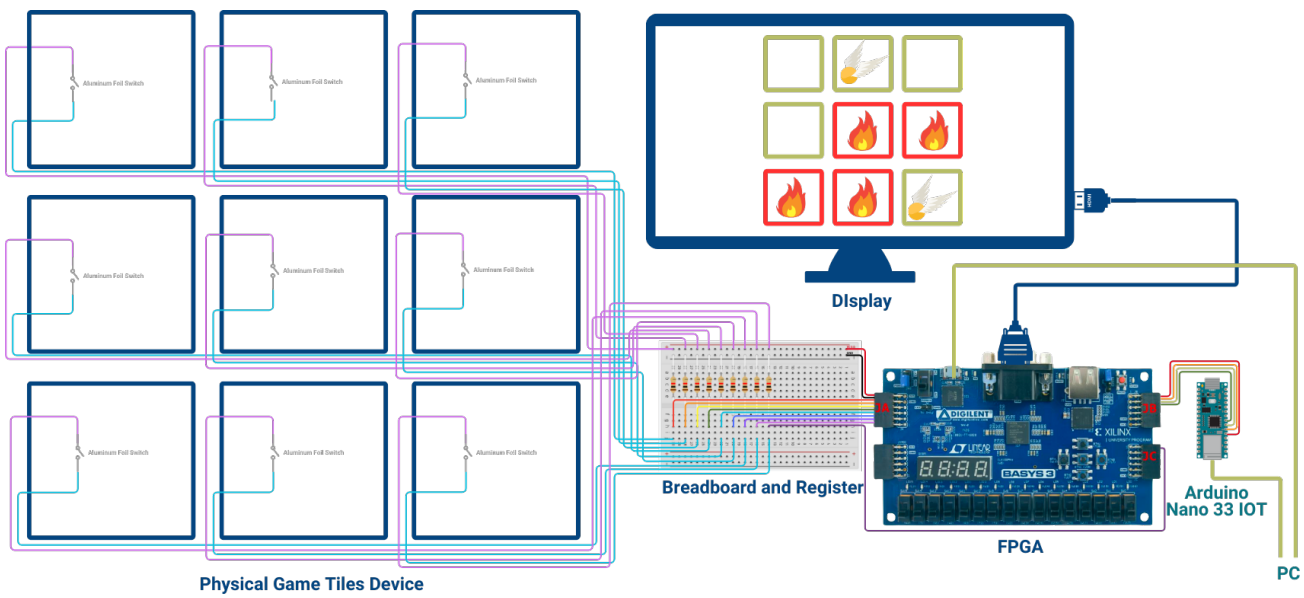

Fig.2 Overview of the Hardware Connection

b. Hardware Design

The hardware setup includes custom-designed game tiles composed of conductive materials to detect player input. Initially, aluminum foil was used for connections, but stability issues led to the adoption of jump wires for reliable signal transmission. The FPGA is connected to a VGA display for visual output, while the Arduino Nano 33 IoT interfaces with the FPGA to manage background music. The overall design prioritizes durability and responsiveness.

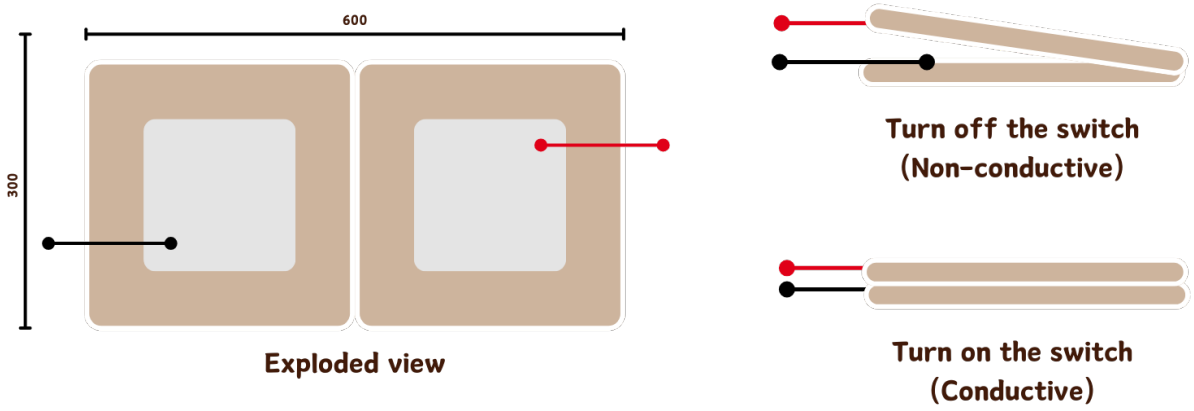

Fig.3 Single Tile Design

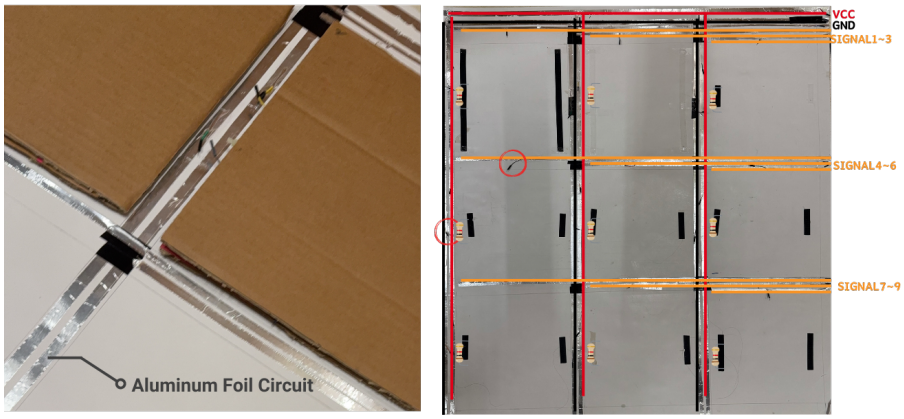

Fig.4 Aluminum Foil Circuit Version

Fig.5 Jump Wire Version

c. Software Design

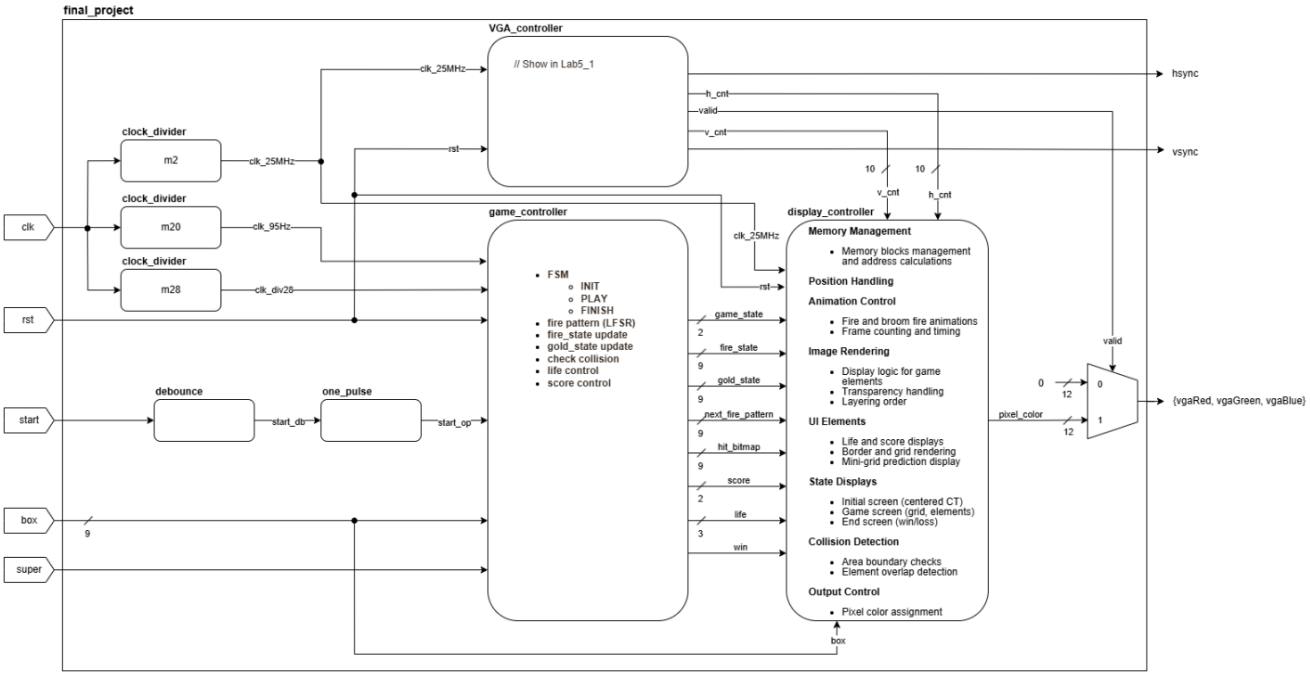

Fig6: Whole picture Block diagram

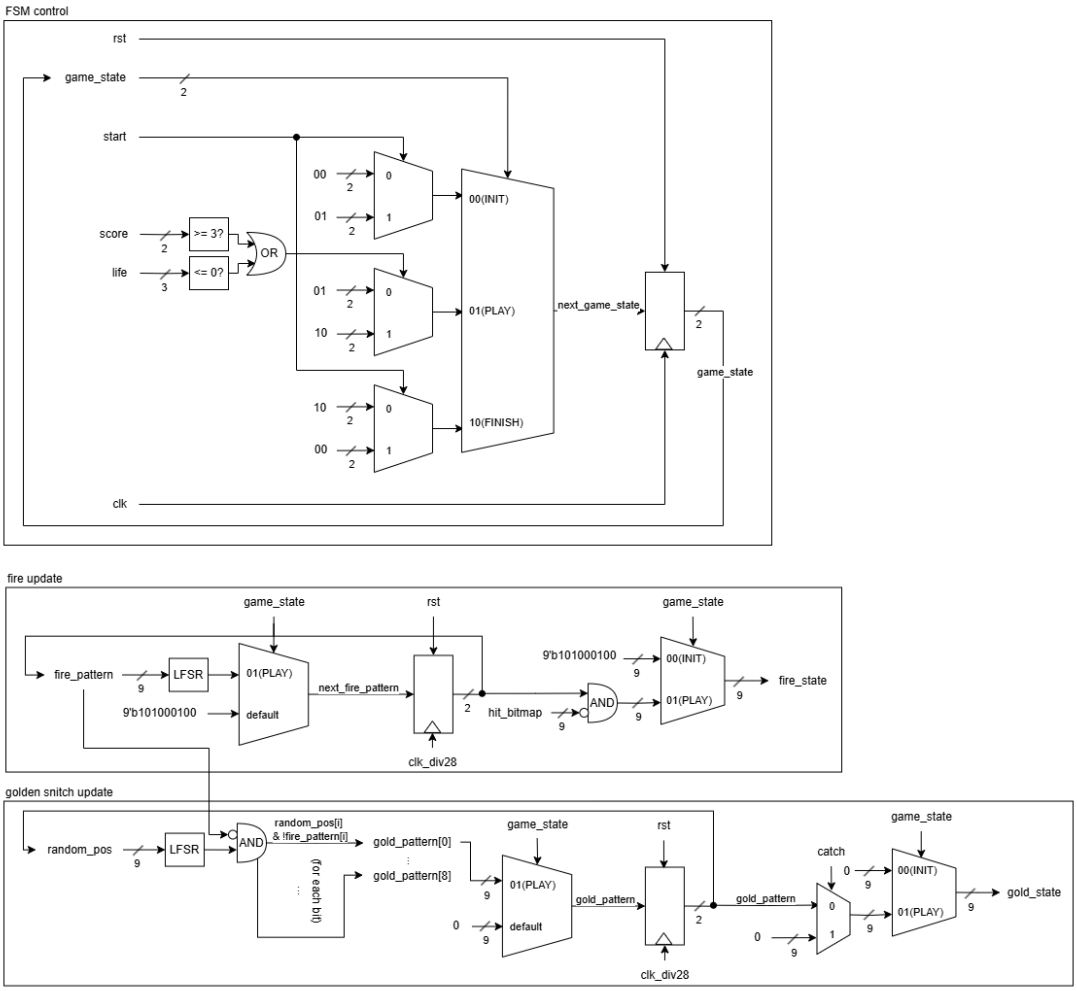

Fig7: FSM control, fire update, and golden snitch update Block diagram

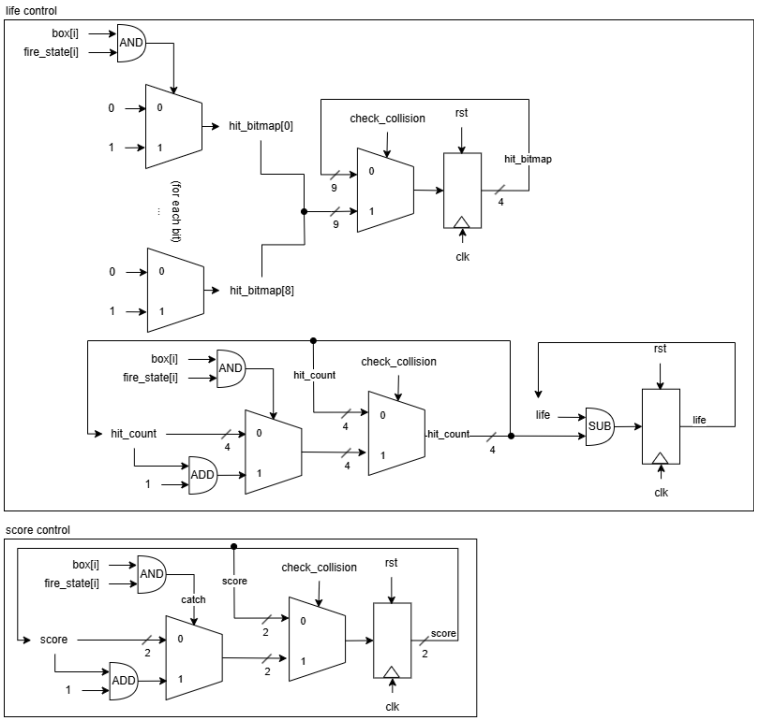

Fig8: Life control and score control Block diagram

Verilog is used to implement the logic of the game. The FPGA processes inputs from the game tiles, updates the game state, and renders outputs on the VGA display. Additional modules handle collision detection, score calculation, and fire pattern generation, ensuring a seamless gaming experience.

✧ **Game_controller module**

The game controller serves as the central processing unit of our Harry Potter themed game.

I. **State Control:**

The module employs a 3-state FSM (INIT, PLAY, FINISH) synchronized with clk input, transitioning based on player status and start signal.

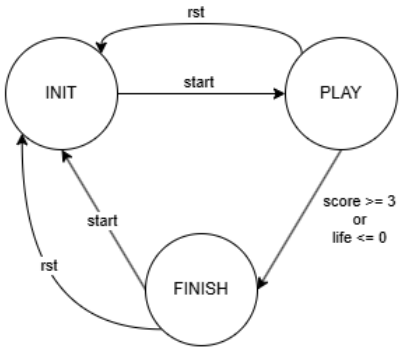

Fig.9 Finite State Machine

II. **Fire Pattern Generation:**

Fire hazards move in a deterministic pattern using LFSR-based logic executed on clk_div28 cycles (every 2 seconds). The system maintains balance by limiting active fires to a specific threshold, applying a mask (9'b110011000) when exceeded to reduce pattern complexity, ensuring fair gameplay.

III. **Gold Management:**

Employs a synchronized counter and LFSR-based position generator on clk_div28 (every 2 seconds). Gold placement occurs after verifying fire_pattern absence:

```
if (random_pos[i] && !fire_pattern[i]) gold_pattern[i] <= 1;
```

IV. **Collision System:**

Edge detection monitors state changes through prev_fire_pattern and prev_box registers. When fire_pattern differs from prev_fire_pattern or box differs from prev_box, check_collision flag triggers collision detection. On hit detection, hit_bitmap tracks collided fires while catch flag indicates gold collection. The system updates fire visibility through dynamic masking (fire_state = fire_pattern & ~hit_bitmap) and gold visibility through catch status (gold_state = catch ? 9'b0 : gold_pattern). Both hit_bitmap and catch reset on fire_update signal, enabling new collision detection for the next fire pattern cycle.

```
if (fire_pattern != prev_fire_pattern)
    fire_update <= 1;
else fire_update <= 0;
if (fire_pattern != prev_fire_pattern || box != prev_box )
    check_collision <= 1;
else check_collision <= 0;
```

V. **Game Progress Tracking:**

Life and score management systems maintain game progression. Starting with 5 lives, players lose life upon fire contact. Collecting 3 golden snitchs triggers a victory condition. The module outputs these states through life, score, and win signals, enabling the display controller to render appropriate visual feedback.

```
if(fire_update) hit_bitmap <= 9'b0;
if (check_collision && !super) begin
    hit_count <= 0;
    for (i = 0; i < 9; i = i + 1) begin
        if (box[i] & ~gold_state[i] & fire_state[i]) begin
            hit_count <= hit_count + 1;
            hit_bitmap[i] <= 1;
        end
    end
```

```
if(fire_update) catch <= 0;
if (check_collision) begin
    for (i = 0; i < 9; i = i + 1) begin
        if (box[i] & gold_state[i]) begin
            score <= score + 1;
            catch <= 1;
        end
    end
end
```

✧ **Dislpay_controller module**

The display controller manages VGA rendering with 640x480 resolution.

I. **Memory Management:**

Handles multiple image resources through Block RAM, including animated sprites (fire, gold) and static images (CT, win/loss). Each memory access is controlled by specific address calculations

II. **Position Computation**:

Implements grid-based calculations for both main game area (420x420) and mini-grid (90x90). Uses relative coordinates for efficient element placement:

```
wire [9:0] rel_x = h_cnt - GRID_START_X;
wire [9:0] rel_y = v_cnt - GRID_START_Y;
wire [1:0] grid_x = rel_x / BOX_SIZE;
wire [1:0] grid_y = rel_y / BOX_SIZE;
wire [3:0] box_index = grid_y * 3 + grid_x;
```

III. **Animation System:**

Controls sprite animations through frame counting and cycling. Both fire and golden snitch animations cycle through 6 frames synchronized with a 20-bit counter:

```
always @(posedge clk or posedge rst) begin
    if (rst) begin
        fire_counter <= 0;
        frame_counter <= 0;
    end else if (frame_counter == 20'hFFFFF) begin
        if (fire_counter == 5)
            fire_counter <= 0;
        else
            fire_counter <= fire_counter + 1;
        frame_counter <= 0;
    end else
        frame_counter <= frame_counter + 1;
end
```

```
if (fire_data[fire_counter] != TRANSPARENT_BLACK)
    pixel_color = fire_data[fire_counter];
```

The counter determines which frame to display for both types of animations, with the frame data stored in separate memory blocks

IV. **Layering System:**
Implements priority-based rendering where background renders first, followed by game elements. Transparency handling ensures proper element visibility:

```
if (box[box_index] && is_in_ct) begin
    if (is_in_broom_fire && hit_bitmap[box_index]) begin
        if (fire_data[fire_counter] != TRANSPARENT_BLACK)
            pixel_color = fire_data[fire_counter];
        else if (ct_data != TRANSPARENT_WHITE)
            pixel_color = ct_data;
    end else if (ct_data != TRANSPARENT_WHITE) begin
        pixel_color = ct_data;
    end
end
```

V. **Game State Display:**
Manages three distinct display states: INIT (centered CT), PLAY (game elements), and FINISH (win/loss screen). UI elements include life indicators, score display, and next-pattern preview grid.

d. Sound and Interaction Design

To enhance immersion, the Arduino Nano 33 IoT plays dynamic background music triggered by signals from the FPGA. Due to insufficient FPGA memory, we chose to offload music playback to an external module. Player interactions with game tiles directly affect game logic and sound effects, creating a synchronized audio-visual experience. This design emphasizes real-time feedback and user engagement, aligning with the interactive theme of the project.

C. Game Features

a. Game Rules and Flow

The game operates in three states: INIT for starting position, PLAY for main gameplay, and FINISH for displaying results. Starting with 5 lives and 0 points, players lose one life when touching fire but gain one point collecting golden snitches. The game enters FINISH state at 3 points (WIN) or 0 lives (LOSS).

b. Gameplay Mechanics

Fire patterns shift every 2 seconds, with upcoming patterns shown on a small grid for strategic planning. The game balances risk and reward through fire hazards and golden snitch collection opportunities, creating dynamic gameplay within the 3x3 grid system.

c. Thematic Elements

- ✧ **Animated Elements:** Six-frame animations power both fire hazards and golden snitches. Character sprites feature dynamic flame effects upon collision.
- ✧ **Game Screens:** Four distinct screens handle different phases: title screen, playing field, win screen, and loss screen.
- ✧ **Interface Design:** Essential elements include life counter, score display, next-pattern preview grid, and transparent sprite rendering for smooth background integration.

D. Results and Discussion

a. System Performance and Stability

Initially, the device was envisioned as a portable sports gaming system. Aluminum foil was chosen for the game tile design to achieve a lightweight structure. However, due to signal instability and frequent fluctuations, the design was improved by replacing the aluminum foil with jump wires, ensuring reliable signal transmission. For the tile switch mechanism, multiple durability tests involving repeated stepping were conducted. Additional enhancements, such as foam adhesive and elastic bands, were introduced to reinforce switch stability and prevent the tiles from being flattened under pressure.

b. Challenges and Solutions

During the hardware integration process, significant discussions were held to address key technical issues. One major challenge was avoiding circuit shorting when the switches were activated. To mitigate this, careful adjustments were made to increase resistance in the circuits. This solution also ensured that other signals in the system did not unintentionally reach high potential, maintaining overall stability and functionality.

For the Verilog game logic design process, one major challenge was implementing precise collision detection timing. Initial attempts using purely combinational circuits led to infinite health deduction. Our solution introduced a check_collision flag, triggered only when fire patterns change or player position updates. This flag gates collision checks for both fire damage and golden snitch collection, ensuring accurate detection without repeated damage.

Another significant challenge emerged after integrating the physical tile inputs, which exhibited signal oscillation. This oscillation caused repeated health deduction or score increments due to rapid signal fluctuations, unlike our stable switch-based testing. We resolved this by implementing a 9-bit hit_bitmap to track collided fires, immediately extinguishing them through masking (fire_state = fire_pattern & ~hit_bitmap). Similarly, for golden snitch collection, a single catch flag prevents multiple score increments, with the snitch disappearing upon collection.

The final challenge involved effectively communicating upcoming fire patterns to players. We initially tried flashing grid borders but found this unclear. The solution was implementing a mini-grid display showing the next fire pattern, significantly improving gameplay clarity and strategic planning.

c. Reflections and Insights

This sports gaming device has the potential to be adapted for various game genres. We brainstormed the possibility of integrating it with a parkour-style game in the future, where players could experience running mechanics through physical interaction with the hardware tiles. By redesigning the software interface and gameplay visuals, the device could simulate immersive running exercises, expanding its applicability and user engagement.

E. Conclusion and Future Work

This project successfully demonstrates the integration of FPGA technology and

interactive hardware to create an engaging sports gaming device. By overcoming challenges in hardware design and signal stability, we developed a robust system that combines innovative gameplay mechanics with immersive user interaction. Aside from Arduino being used to control music playback, all other functionalities were implemented using the FPGA. Future improvements, such as expanding game genres and optimizing hardware design, can further enhance the device's potential for broader applications and user engagement.

Authors

余佳軒 Yu Chia-Suan

Affiliation: National Tsing Hua University, Computer Science Department

Role: Responsible for Verilog Game Logic Design, including the development of the finite state machine (FSM), VGA display, fire pattern update, collision detection, and life, score control logic. Additionally, led the FPGA implementation to ensure seamless integration of hardware and software components.

張舜涵 Chang Shun-Han

Affiliation: National Tsing Hua University, Computer Science Department

Role: Focused on Hardware and Circuit Design, encompassing the creation and optimization of game tiles with enhanced durability and stability. Managed the physical integration of the FPGA and peripheral components, ensuring reliable signal transmission and robust interaction capabilities.

Appendices

a. Github
b. Video Link
c. PPT Link