Project Proposal: Flappy Bird on FPGA

Course: CSEE W4840 - Embedded System Spring 2025
Team Members:

Sijun Li (sl5707), Yunxuan Yang(yy3526), Zidong Xu(zx2507), Tianshuo Jin(tj2591)

1. Introduction

In Flappy Bird, players must control a bird to cross obstacles made of pipes of different lengths. The core gameplay of the game is to test the player's reaction speed and the accuracy of finger clicks. As the bird continues to move forward, the difficulty of the obstacles gradually increases. Each safe passage through a pillar without hitting it is worth 1 point, and the game ends if it hits it.

This project aims to implement Flappy Bird on an FPGA system, utilizing hardware-accelerated graphics rendering and software-driven game logic. To do this, it is necessary to implement graphics rendering, such as physical simulation of gravity and collision, keyboard input controlled by the player, and a scoring system.

2. Game Features

The FPGA-based version of Flappy Bird will include the following key features:

- **Gravity Simulation:** The bird will continuously fall due to gravity, requiring the player to press a button to jump.
- **Jump Mechanic:** A button press will apply an upward velocity to simulate a flap.
- **Dynamic Difficulty Adjustment:** The game speed will gradually increase as the player progresses.
- **Random Pipe Generation:** Pipes with different heights and gaps will be generated dynamically.
- **Collision Detection:** Bounding box collision detection will determine whether the bird collides with pipes.
- **Game States:** The game will transition between different states: Start, Playing, and Game Over.
- **Scoring System:** The score will increase when the bird successfully passes through pipes.
- **Game Over Condition:** The game will end when a collision occurs or the bird falls out of bounds.

3. Game Logic

Maintain three game states: Start, Playing, Game Over. Create a score system that increases when passing obstacles, and display score and restart option upon game over.

4. Implementation Methodology

Rendering

We will use a framebuffer-based rendering system to draw the game on a HDMI display. The framebuffer will be stored in FPGA BRAM/SDRAM, where each pixel will be represented as a 24-bit RGB value.

Collision Detection

We plan to use the bounding box collision detection. Each object (bird and pipes) will have a defined hitbox. If the bird's hitbox intersects with a pipe's hitbox, the game will trigger a collision event. Checking will be performed on each frame update.

Physics Simulation

The physics system will be based on gravity acceleration and velocity updates. A button press will apply an instant upward velocity to simulate jumping.

5. Challenges and Considerations

For future optimization, in terms of graphics, for a smooth gaming experience, the frame buffer needs to be updated efficiently to ensure a reasonable frame rate. In terms of operation, gravity and jump speed need to be balanced, and collision detection must be accurate enough to achieve a natural effect. At the same time, managing FPGA RAM frame buffer processing can minimize the delay of input response and rendering updates.