CSEE W4840 Embedded Systems

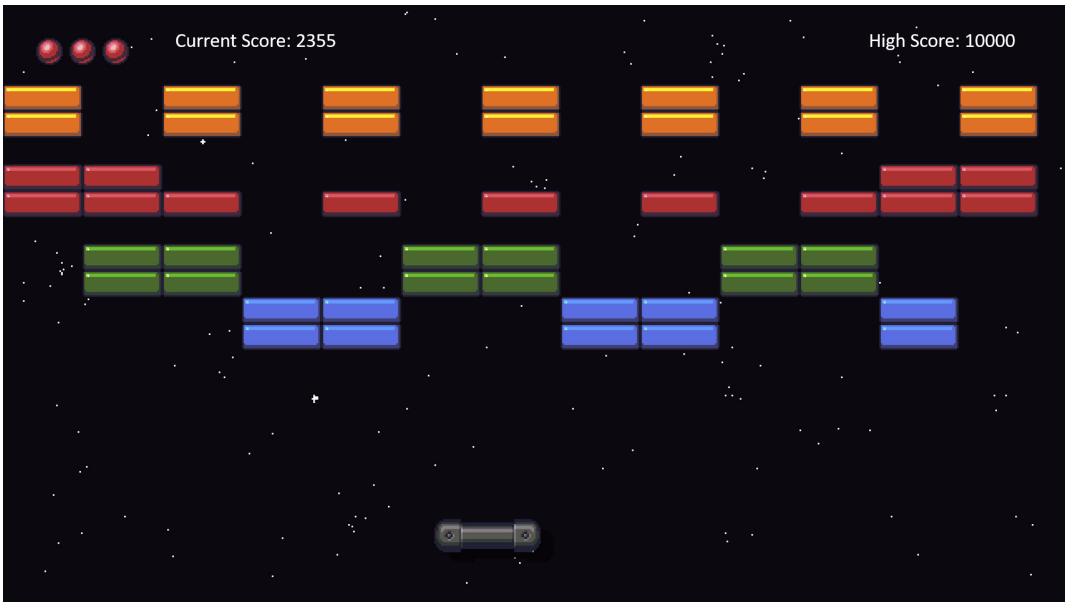
Project Proposal

Doreen Sisanalli – ds4371
Pranav Asuri – pa2708
Varsha Keshava Prasad – vk2550
Venkat Suprabath Bitra – vsb2127

March 14, 2025

Project Overview

This project aims to develop a Brick Breaker game on an FPGA board, utilizing SystemVerilog for hardware implementation and C for software components. The game will feature a single player controlling a paddle to bounce a ball and destroy bricks on the screen. To enhance gameplay, the game will include power-ups and obstacles. These elements will provide additional challenges and rewards, making the game more engaging and dynamic.



Game Design

The tool design for Brick Breaker involves three key components:

- **FPGA Board:** This is the primary hardware platform for the game. It will be programmed using SystemVerilog to handle keyboard input and VGA display output. The FPGA will manage the game logic, including ball movement, collision detection, and brick destruction.
- **VGA Monitor:** Used to display the game graphics, including the game environment, paddle, ball, and bricks. The VGA output will be designed to provide a clear and responsive visual experience.
- **Keyboard:** The input device for controlling the paddle's movement. The keyboard will allow players to move the paddle left and right, with the option to implement additional controls for launching power-ups or special moves.

Implementation

Hardware Design (FPGA & SystemVerilog)

- **Paddle Control:** Interface the FPGA with keyboard input to capture player actions, using SystemVerilog to translate these inputs into paddle movements. This involves designing a keyboard interface module that can read keyboard presses and convert them into signals that control the paddle's position on the screen.
- **VGA Display Controller:** Develop a VGA controller in SystemVerilog capable of rendering the game's graphics, including the paddle, ball, bricks, and power-ups. This controller will manage the timing and synchronization of the VGA signal to ensure a smooth and consistent display.

Software Development

- Game Logic: Implement core game logic in C, including:
  - Ball Movement: Calculate the trajectory and speed of the ball based on physics and game rules.
  - Collision Detection: Detect collisions between the ball and bricks or the paddle, handling the consequences of these collisions (e.g., brick destruction, ball bounce).
  - Power-Up Mechanics: Implement the effects of power-ups, such as increasing the paddle size or changing the ball's speed.
  - Scoring: Update the player's score based on successful brick destruction and other achievements.

- Graphics Processing: Calculate the positions of the paddle, ball, and bricks based on game logic. Generate drawing commands to send to the FPGA for rendering on the VGA display. This involves translating game state into visual representations that can be displayed on the screen.

- Game Rules: Enforce game rules in the C program, determining when a player has won or lost, and updating the game state accordingly. This includes managing player lives, level progression, and game over conditions.

Hardware-Software Communication

- **Real-Time Communication:** Dynamically update the game environment based on player actions and game events, requiring real-time communication between the FPGA and the C program. This involves designing a communication protocol that allows the FPGA to send updates to the C program and vice versa, ensuring that the game state remains consistent across both hardware and software components.

Additional Considerations

- Error Handling: Implement error handling mechanisms to ensure that the game remains stable and responsive even in unexpected situations, such as invalid user input or hardware malfunctions.

- Performance Optimization: Optimize both hardware and software components to achieve smooth gameplay and efficient resource utilization. This may involve optimizing SystemVerilog code for better FPGA performance and optimizing C code for faster execution on the software side.

Milestones

Graphics Rendering

- Verification: Verify that the VGA controller correctly renders the game graphics based on commands from the C program. This includes ensuring that all game elements such as the paddle, ball, bricks, and power-ups are displayed accurately and consistently.

- Testing Scenarios: Test various scenarios to confirm that graphics rendering works as expected, including different resolutions, colors, and frame rates.

Input Processing

- Keyboard Input Validation: Test the keyboard input processing to ensure that paddle movements are accurately captured and translated into game actions. This involves verifying that the FPGA correctly interprets keyboard presses and converts them into paddle movements on the screen.
- Responsiveness Testing: Conduct tests to ensure that the input processing is responsive and does not introduce any noticeable lag between player actions and game responses.

Gameplay

- Game Rule Validation: Conduct thorough gameplay testing to ensure that the game rules are correctly applied. This includes verifying that scoring, collision detection, power-up effects, and game over conditions are all handled as intended.
- Player Experience Evaluation: Ensure that the game provides a fun and challenging experience for the player. This involves testing different difficulty levels, ensuring that the game is neither too easy nor too hard, and evaluating overall player engagement.

Optimizing and Testing

- Performance Optimization: Optimize the game for performance by identifying bottlenecks in both hardware and software components and implementing improvements. This could involve optimizing SystemVerilog code for better FPGA performance and optimizing C code for faster execution.
- Extensive Testing: Conduct extensive testing to ensure stability and enjoyment. This includes testing for bugs, ensuring that the game runs smoothly under various conditions, and verifying that all features work as expected. Additionally, gather feedback from players to identify areas for further improvement.

Additional Milestones

- **Game Scoring and Saving:** The game will track player scores, lives, and level progress. The game logic module will manage these aspects, ensuring a smooth and engaging experience for the player.
- **Audio Integration:** To further enhance the gaming experience, sound effects and background music could be added. This would involve integrating an audio module into the FPGA design, allowing for real-time sound processing and playback.