Department of Electric & Electronic Engineering,
Boğaziçi University

EE 244 FINAL PROJECT

VHDL SNAKE GAME

Enes Kuzuoğlu
Mehmet Emin Algül

Project Advior : Şenol Mutlu

04/06/2024

Table of Contents

1 INTRODUCTION 3
2 PROBLEM STATEMENT 3
3 RELATED BACKGROUND 3
3.1 Definitions, Acronyms and Abbreviations 4
3.2 System Architecture 4
3.3 Definitions, Acronyms and Abbreviations 5
3.4 System Architecture 5
4 DESIGN 5
4.1 Definitions, Acronyms and Abbreviations 5
4.2 System Architecture 6
4.3 Game Logic Architecture 7
5 RESULTS 9
5.1 Expected Functionality 9
5.2 Limitations and Future Improvements 10
6 CONCLUSION 11
7 REFERENCES 11

1 INTRODUCTION

This project report details the design and implementation of a classic Snake game using VHDL (VHSIC Hardware Description Language). The project aims to create a fully functional game experience on a hardware platform, specifically targeting the Nexys 3 Field-Programmable Gate Array (FPGA) board. The primary implementation medium is the Xilinx Spartan VI XC6LX16-CS324 FPGA IC chip. Players will be able to control the movement of a virtual snake on the screen, collecting food items to increase its length and score. The report will cover the design process, functionality, and implementation details of the Snake game in VHDL.

2 PROBLEM STATEMENT

This project addresses the challenge of creating a functional Snake game in hardware using VHDL. The game needs to exhibit the following functionalities:

- **Snake Movement:** The user should be able to control the snake's direction (up, down, left, right) using buttons or other input mechanisms.
- **Collision Detection:** The game logic must identify collisions between the snake and the boundaries of the playing area, as well as collisions with its own body. Upon collision, the game should end or trigger a penalty.
- **Food Generation:** Food items need to be randomly placed on the playing field for the snake to collect.
- **Scorekeeping:** The game should track the number of food items collected by the snake and display the score on an external device (e.g., seven-segment display).
- **Synchronization Signals:** Understand timing information from horizontal synchronization and vertical to determine the active display region.
- **Map Snake and Food Positions:** Translate snake body and food location (x, y coordinates) into corresponding row and column values for the VGA display.
- **Coordinate Color Generation:** Generate appropriate color signals (red, green, blue) based on mapped positions and collision detection results to represent the snake, food, and background on the screen.

3 RELATED BACKGROUND

This section delves into the key concepts behind implementing the Snake game in VHDL:

Field-Programmable Gate Arrays (FPGAs):

FPGAs are programmable logic devices containing an array of configurable logic blocks and routing resources. Unlike traditional microprocessors, FPGAs allow hardware designers to define the functionality of the circuit themselves by configuring these blocks. This makes them ideal for implementing custom digital logic circuits like the Snake game, where precise control over timing and processing is crucial.

VHDL (VHSIC Hardware Description Language):

VHDL is a hardware description language used to specify the behavior of digital circuits. It allows engineers to describe the functionality of a circuit at a high level using a text-based format. VHDL code can then be synthesized into a hardware configuration that can be programmed onto an FPGA. This project leverages VHDL to define the game logic, user interaction, and screen drawing mechanisms for the Snake game.

Game Concepts:

While the specific implementation details will be discussed later, the project utilizes established game design principles:

3.1 Collision Detection

Efficient algorithms are necessary to identify collisions between the snake and various elements:

- **Playing Field Boundaries:** The snake's movement needs to be restricted to stay within the defined play area. This can be achieved by comparing the snake's head position (x and y coordinates) against the pre-determined boundaries.
- **Snake's Own Body:** Collisions with the snake's own body signify game over or a penalty. Tracking the positions of all body segments (tail included) and comparing them against the current head position allows for detection. Common approaches involve storing the snake's body positions in a memory element and iterating through them during collision checks.

3.2 Scorekeeping

A simple counter keeps track of the number of food items collected by the snake. This section explores potential approaches:

- **Counter Implementation:** A dedicated register or counter can be utilized to increment its value every time the snake consumes a food item. VHDL provides efficient constructs for implementing counters.
- **Score Display:** The accumulated score needs to be displayed on an external device (e.g., seven-segment display) for the user. This may involve converting the counter value into a format suitable for the chosen display interface and transmitting the data through appropriate output signals.

3.3 Random Food Generation and High Clock Speed

To create an element of surprise and challenge, the game employs a random number generator to determine the location of the food item on the playing field. Here's how it might work:

- **Food Placement:** After the snake consumes a food item, a random number generator is triggered by a high-speed clock (e.g., 25 MHz). This clock ensures frequent updates to the food's location, making it difficult for the user to predict its exact position based on the snake's movement pattern.
- **Coordinate Calculation:** Based on the random values generated, the food's x and y coordinates are calculated and added by a constant value (e.g., 8) to ensure it appears within the playable area and avoids immediate placement on top of the snake.

3.4 Efficient Body and Food Drawing based on Absolute Differences

To accurately represent the snake's body and food on the screen, the game logic utilizes absolute differences:

- **Drawing Logic:** The drawing mechanism compares the absolute values of the difference between the body/food location (x and y coordinates) and the current row and column signals from the VGA controller.
- **Collision Detection Similarity:** This approach resembles the collision detection logic. By setting a threshold value (e.g., food/body size) for the absolute difference, the drawing process can determine if a specific pixel location needs to be colored based on the snake's body, food item, or background. This ensures efficient drawing while maintaining collision detection accuracy.

4 DESIGN

This section details the design architecture of the Snake game implemented in VHDL.

4.1 Definitions, Acronyms and Abbreviations

- **VGA (Video Graphics Array):** A standard for displaying graphics on a computer monitor.
- **hsync (Horizontal Sync):** A signal that synchronizes the horizontal scanning of a video display.
- **vsync (Vertical Sync):** A signal that synchronizes the vertical scanning of a video display.
- **BCD (Binary Coded Decimal):** A method for representing decimal digits using binary numbers.
- **VHDL (VHSIC Hardware Description Language):** A language for describing the behavior of digital circuits.

4.2 System Architecture

The Snake game design utilizes a modular approach with several interconnected components, as shown in Figure 1 (**Figure to be included, showing the block diagram**):

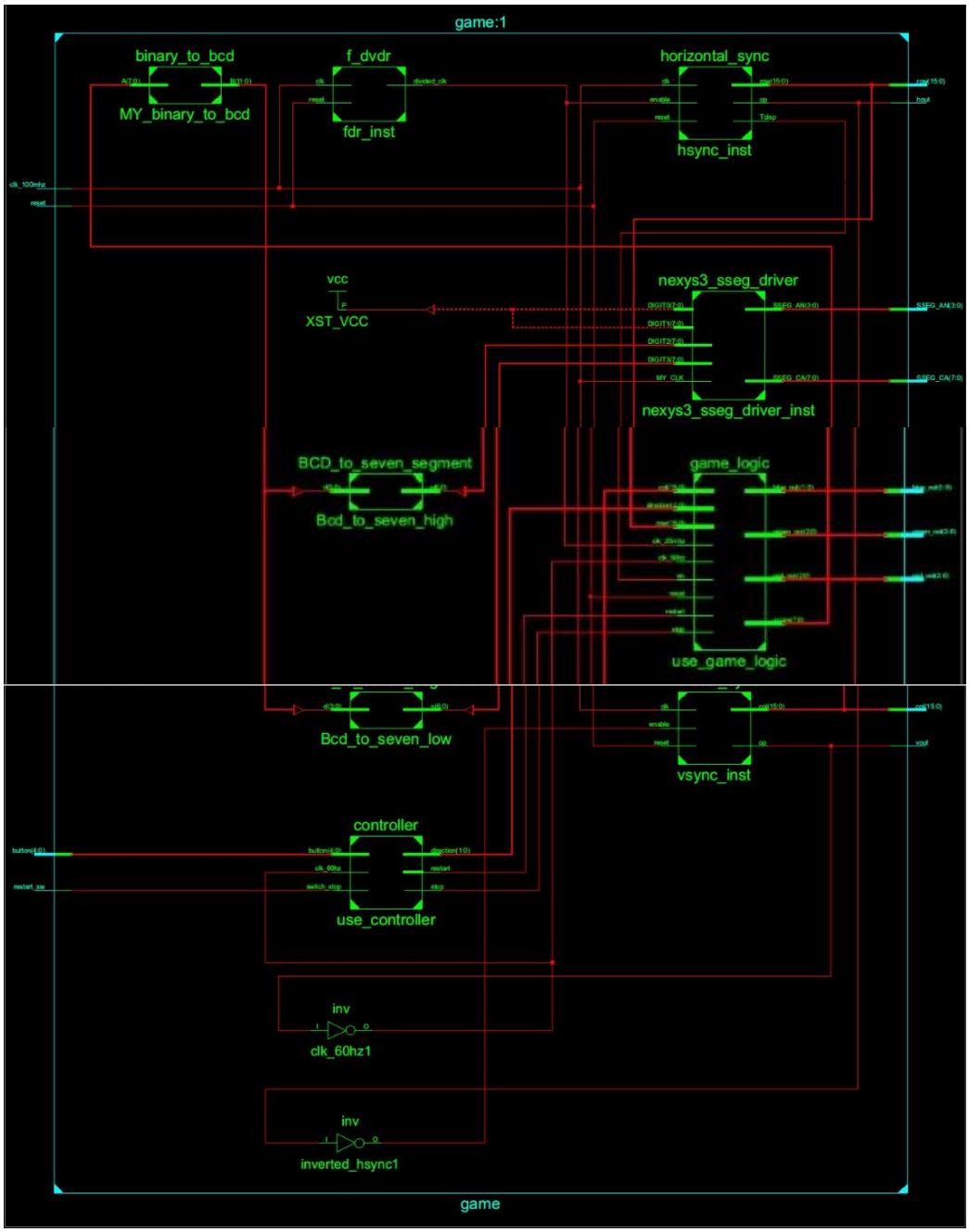

Figure 1: RTL Schematics of game entity.

- **Game Entity:** This is the top-level entity of the design and serves as the main VHDL file. It instantiates and interacts with all other components.
- **Game Logic:** This central component handles core gameplay functionalities including user input processing, snake movement, collision detection, food generation, scorekeeping, and overall game state management.
- **VGA Controllers (hsync and vsync):** This component generates the necessary signals (including hsync, vsync, row, and column) to drive a VGA display. It interfaces with the game logic to determine the appropriate colors for each pixel on the screen 640x480 display.
- **User Input Interface (Controller):** This block captures user input from the joystick or buttons and translates it into control signals for the snake's movement. Buttons for up-down-left-right movements and restart, switches for restart and pause game.
- **Frequency Divider:** This component divides the system clock (e.g., 100 MHz) down to a lower frequency (e.g., 25 MHz) using a binary divider circuit. This lower frequency may be used for specific functionalities within the game, such as controlling the food item update rate (i.e. random number generation).
- **Score Display Interface:** This block manages the output of the game score to the seven-segment displays on the Nexys 3 FPGA board. It employs sub-components:
    - **Binary-to-BCD Converter:** This component converts the binary score value from the game logic into BCD format, suitable for driving the seven-segment displays.
    - **BCD-to-Seven Segment Driver:** This component receives the BCD data and generates the appropriate control signals to activate specific segments on the seven-segment displays, displaying the score to the user.
    - **Nexys3SSegDriver:** This component is a pre-built VHDL module specific to the Nexys 3 board, providing an interface between the BCD-to-Seven Segment Driver and the physical seven-segment displays on the hardware.

These components communicate with each other through well-defined interfaces, ensuring modularity and easier development. Figure 1 provides a visual representation of this system architecture.

4.3 Game Logic Architecture

The game logic architecture for the Snake game in VHDL is composed of three main processes:

- **Snake Movement**: This process handles the movement of the snake based on user input, including direction changes (up, down, left, right), boundary checks, self-collision detection, and food consumption. The snake's head and body positions are updated accordingly, and the game state is managed (active, fail, reset, restart).
- **Random Number Generation**: This process generates random positions for the food within the screen boundaries using a simple increment-based approach, ensuring the food appears at different locations on the screen.
- **Drawing the Game**: This process determines the colors of the pixels to be displayed on the VGA screen. It checks if the current pixel belongs to the snake's body, food, or screen border and sets the corresponding color output. The process handles the visual representation of the snake, food, and boundaries on the display.

This modular approach allows for efficient handling of the game's core functionalities, ensuring clarity and maintainability in the VHDL design.

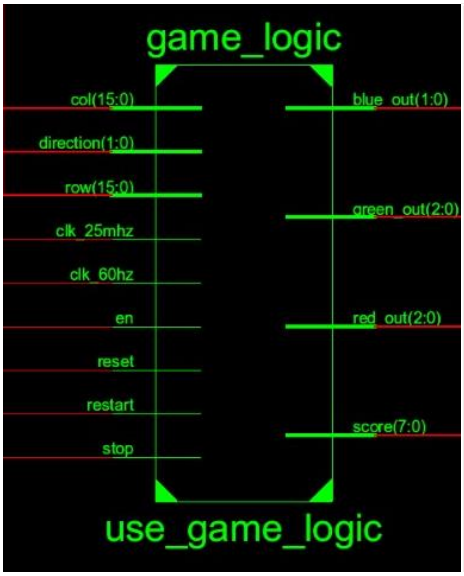

Figure 2: RTL Schematics of game logic.

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity game_logic is
    generic(
        screen_width        : integer := 640;
        screen_height       : integer := 480;
        food_width          : integer := 20;
        head_width          : integer := 20;
        initial_x           : integer := 100;
        initial_y           : integer := 240;
        base_lenght         : integer := 2;
        max_lenght        : integer := 5;
        food_initial_x        : integer := 320;
        food_initial_y        : integer := 240;
        initial_speed       : integer := 5);
    port(
        clk_60hz            : in  std_logic;
        direction           : in  std_logic_vector(1 downto 0);
        stop                : in  std_logic;
        reset               : in  std_logic;
      restart             : in  std_logic;
        clk_25mhz           : in  std_logic;
        en                  : in  std_logic;
        row, col            : in  std_logic_vector(15 downto 0);
        red_out, green_out          : out std_logic_vector(2 downto 0);
        score               : out std_logic_vector(7 downto 0);
        blue_out                : out std_logic_vector(1 downto 0)
        );
end entity;
```

Figure 3: RTL Schematics of game entity.

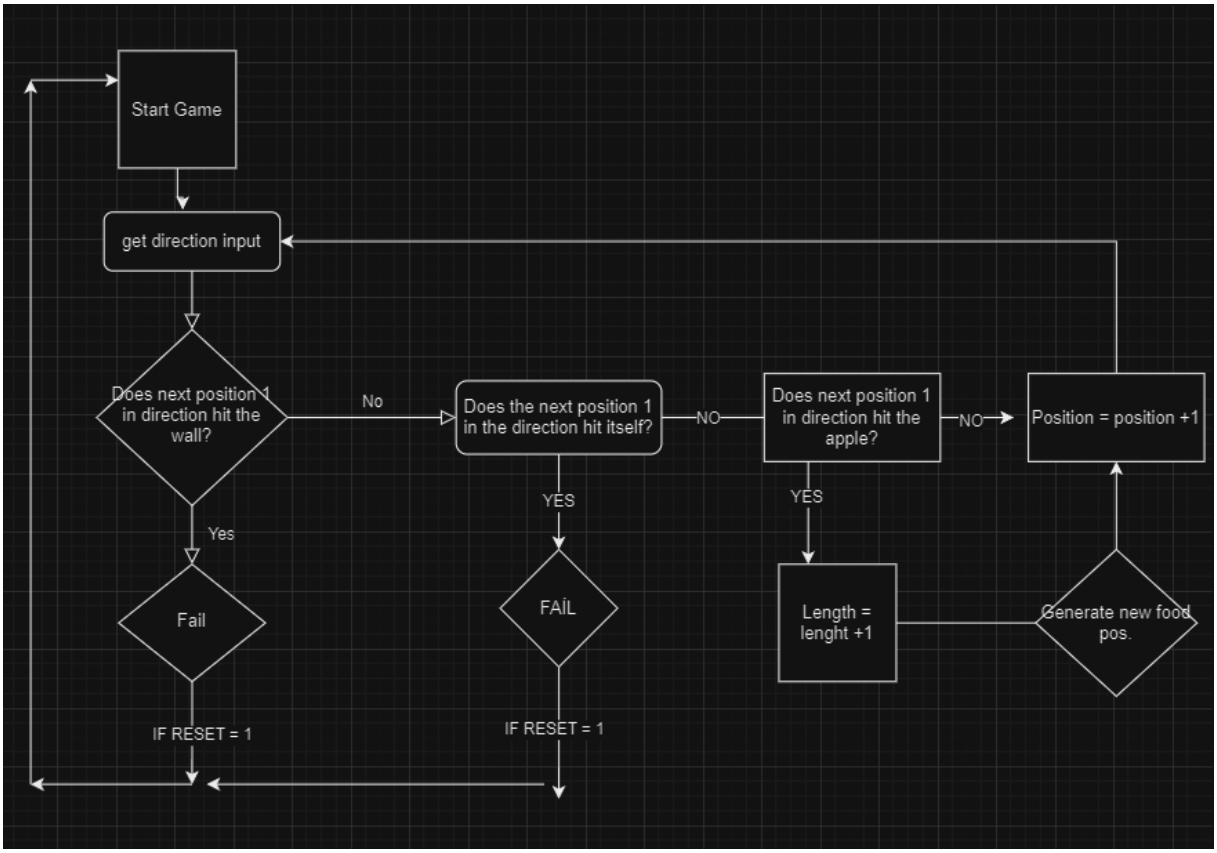

Figure 5: Game Logic Flow Diagram.

5 RESULTS

This section outlines the expected behavior and potential outcomes of the Snake game implemented in VHDL:

5.1 Expected Functionality

Based on the design architecture, the Snake game should exhibit the following functionalities:

- **User Control and Gameplay:** Players can control the snake's movement (up, down, left, right) using the joystick or buttons on the Nexys 3 board.
- **Collision Detection:** The game logic effectively detects collisions between the snake and the playing field boundaries or its own body, triggering appropriate actions (game over or penalty).
- **Food Generation and Scorekeeping:** Food items are randomly placed on the playing field through the random number generator. Consuming a food item increments the score, which is displayed on the seven-segment displays.
- **Accurate Screen Drawing:** The VGA controller, in conjunction with the game logic, accurately draws the snake's body, food items, and background on the screen using hsync, vsync, row, and column signals. The snake's movement and color changes based on user input are reflected on the screen.

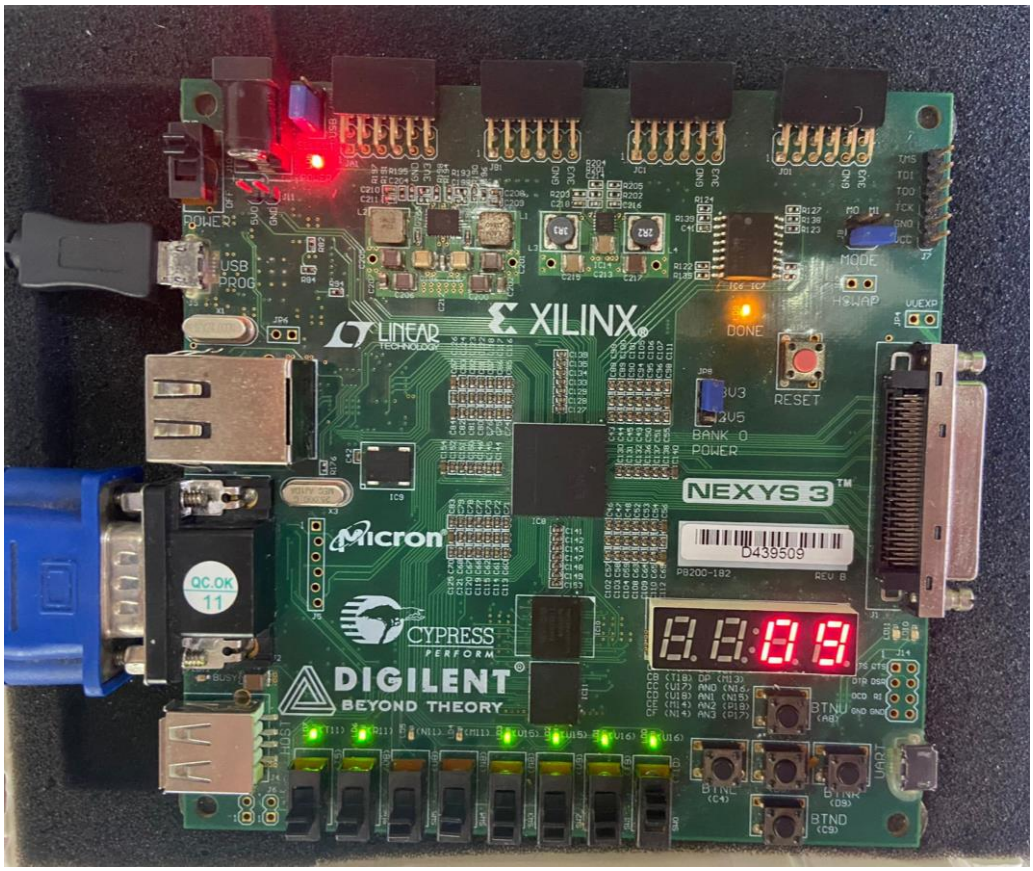

Figure 6: Nexys3 Spartan VI FPGA Board (Buttons, Switches and Seven Segment Displays).

Figure 7: Snake Gameplay on VGA.

5.2 Limitations and Future Improvements

This section might address potential limitations of the current implementation:

- **Screen Size and Complexity:** The screen size and snake speed may be limited by the available resources on the Nexys 3 board.
- **Collision Detection Optimization:** Depending on the chosen algorithm, collision detection for a longer snake may require further optimization for efficient performance.
- **Additional Features:** The current implementation might not include advanced features like sound effects or difficulty levels.

Future improvements could involve:

- **Enhanced Graphics:** Explore techniques for drawing smoother curves and more detailed snake and food representations on the VGA display.
- **Sound Integration:** Implement audio using dedicated components or libraries on the Nexys 3 board to add sound effects for gameplay events (e.g., food consumption, game over).
- **Difficulty Levels:** Introduce adjustable difficulty levels by increasing the snake speed, reducing reaction time, or introducing obstacles within the playing field.

By addressing these limitations and exploring potential improvements, the Snake game can be further enhanced for a more engaging and challenging user experience.

6 CONCLUSION

This report has documented the design and implementation of a Snake game entirely in VHDL hardware description language. The project aimed to create a functional and playable game experience on the Nexys 3 FPGA board, utilizing its resources to generate visuals, handle user input, and manage game logic.

The report detailed the overall design architecture, including the central game logic entity, VGA controller, user input interface, and score display interface with its sub-components. Key game concepts like collision detection, scorekeeping, and random food generation were explored, outlining the algorithms and approaches used in the VHDL code.

The expected outcomes were discussed, highlighting user control, collision detection, food generation, scorekeeping, and accurate screen drawing using VGA signals. Additionally, potential limitations regarding screen size, snake speed, and collision detection optimization were addressed. The report also presented ideas for future improvements, such as enhanced graphics, sound integration, and adjustable difficulty levels.

Overall, this project successfully demonstrates the capabilities of VHDL for implementing interactive games on hardware platforms. The Snake game serves as a foundation for further exploration and experimentation with game design principles and hardware implementation using FPGAs.

7 REFERENCES

1. Boğaziçi University EE244 Digital System Design Lecture Slides
2. Boğaziçi University EE244 Digital System Design Lab Manuels
3. Oakland University, ECE 2700 Digital Logic Design Snake Game Final Project Presentation
4. ldm0. *VHDL-Snake-Game*. GitHub repository. https://github.com/freaktm/VHDL-Snake-Game.
5. ldm0. *FPGAsnake*. GitHub repository. https://github.com/ldm0/FPGAsnake.