Project 2 Report

Repository Link: https://github.com/iihakk/PongGame-HDL.git

Team : Abdulaziz Al-Haidary, Koussay Jaballah, Yara Abdelkader

Modules:-

Game (Top Module)

Description:
The game module integrates all the submodules to manage the gameplay. It coordinates the scoring system, paddle movements, VGA display, and audio feedback. It takes inputs for reset, paddle movements, and clock signals, and outputs the game visuals, score display, and sound.

Responsibilities:

- Instantiates the VGAControl, Score, and segmentDecoder modules.
- Connects outputs from the VGAControl (e.g., goal_player1, goal_player2) to update scores in the Score module.
- Passes score data to the segmentDecoder for display.

Paddle Module

This module controls the vertical position of a paddle on the screen.

- **Inputs**:
    - clk: The clock signal for timing.
    - reset: Resets the paddle position to the center.
    - move_up and move_down: Signals to move the paddle up or down.
- **Output**:
    - y: The vertical position of the paddle.
- Implements movement with a delay to prevent rapid updates. Ensures paddles stay within screen boundaries and only move a limited distance with a single button press

Score Module

The **Score** module tracks the players' scores and updates them when a goal is detected.

- **Inputs**:
    - clk: Clock signal for synchronization.

    - reset: Resets both players' scores to zero.
    - goal_player1 and goal_player2: Signals indicating a score by Player 1 or Player 2.
- **Outputs**:
    - player1_units, player1_tens, player2_units, player2_tens: Scores in BCD format.
- It uses a pulse-based system to avoid multiple score increments for a single goal.

segmentDecoder Module

The **segmentDecoder** drives a 7-segment display to show the scores of both players.

- **Inputs**:
    - clk and reset: Clock and reset signals.
    - player1_units, player1_tens, player2_units, player2_tens: Scores in BCD format from the **Score** module.
- **Outputs**:
    - anode: Controls which segment is active.
    - segments: The 7-segment data to display digits.
- Uses a multiplexing technique to display each player's scores alternately on a single 7-segment display.

VGAControl Module

The **VGAControl** module generates VGA signals and renders the game on the screen. It also handles ball movement, paddle collisions, and sound generation.

- **Inputs**:
    - clk100MHz: System clock.
    - reset: Resets the game state.
    - move_up1, move_down1, move_up2, move_down2: Paddle movement signals for both players.
- **Outputs**:
    - hsync, vsync: Horizontal and vertical sync signals for VGA display.
    - red, green, blue: RGB signals for pixel coloring.
    - goal_player1, goal_player2: Signals to update the score when a goal is detected.
    - sound: Sound output for paddle hits and goals.
- **Key Features**:
    - Implements VGA timing for 640x480 resolution.
    - Handles ball collision with walls and paddles, resetting its position after goals.
    - Includes sound generation for paddle hits and goals.
    - Controls the areas of all entities on the screen.

Tone_gen Module

Purpose: Generate a square wave signal (sound) with a frequency controlled by the counter input value.

Inputs:

- clk: Clock signal that drives the module.
- enable: Control signal to enable or disable the sound output.
- counter: A 22-bit value that sets the period for toggling the sound output, indirectly controlling the frequency.

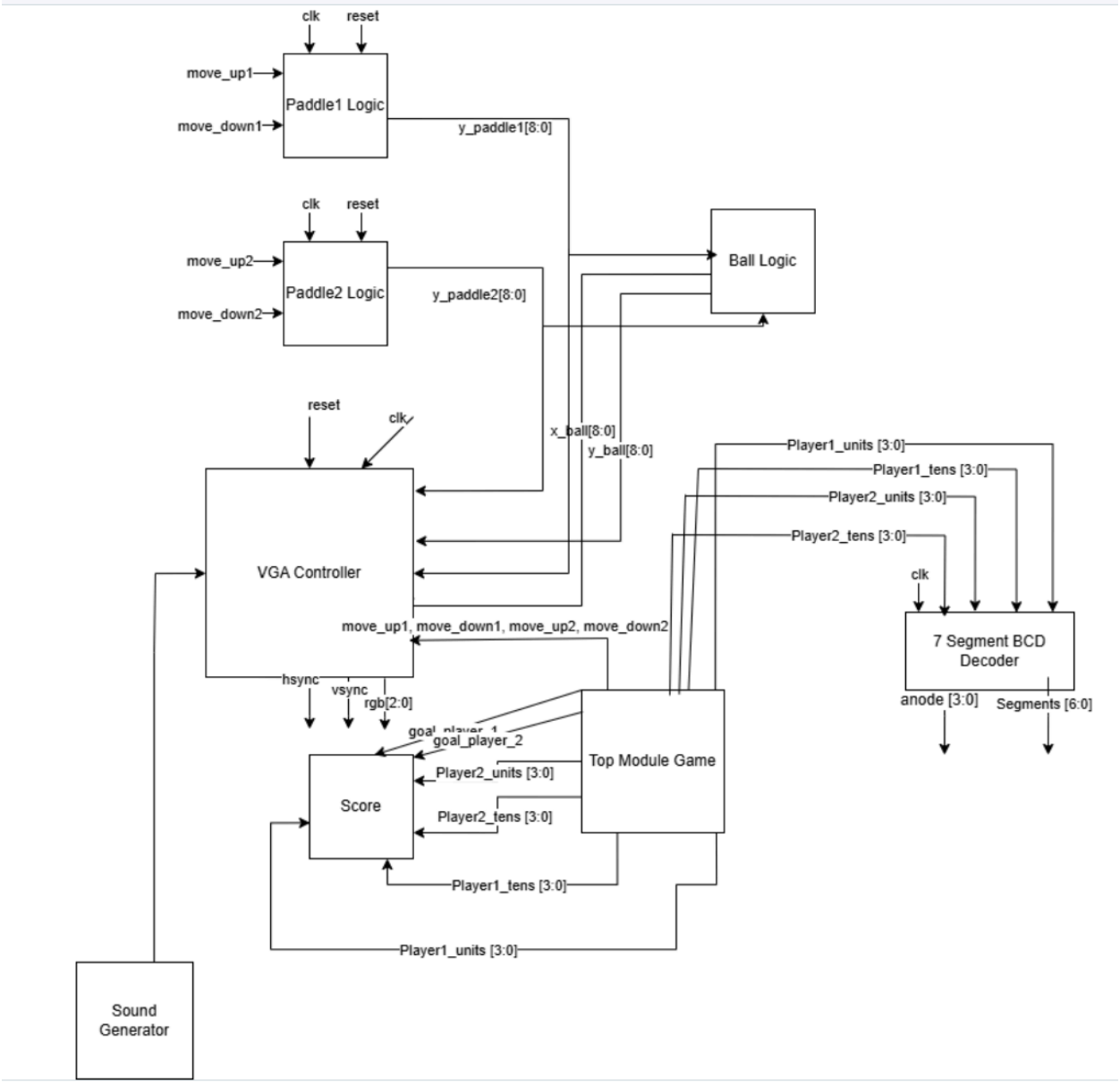
Output:

- sound: The generated square wave signal.

Contribution:-

For most of the project, the team members have worked together rather than dividing tasks.

Diagram:-


clk
reset
move_up1
move_down1
Paddle1 Logic
y_paddle1[8:0]
clk
reset
move_up2
move_down2
Paddle2 Logic
y_paddle2[8:0]
Ball Logic
reset
clk
x_ball[8:0]
y_ball[8:0]
Player1_units [3:0]
Player1_tens [3:0]
Player2_units [3:0]
Player2_tens [3:0]
clk
VGA Controller
7 Segment BCD Decoder
move_up1, move_down1, move_up2, move_down2
hsync
vsync
rgb[2:0]
anode [3:0]
Segments [6:0]
goal_player_1
goal_player_2
Top Module Game
Score
Player2_units [3:0]
Player2_tens [3:0]
Player1_tens [3:0]
Player1_units [3:0]
Sound Generator