EE 314
Term Project
FPGA Implementation of a 2D Strategy Game

Introduction

For decades, turn-based strategy games have been popular among people of all ages, providing a challenging gameplay experience while remaining easy to learn. Examples of such games include Tic Tac Toe and Battleships. The term project of this year project involves implementing a game called "Triangles vs. Circles" using Verilog HDL language and the Altera DE1-Soc Board. The game features a VGA interface, allowing players to see the board and their moves while the game logic operates in the background. The VGA interface provides an interactive and engaging gaming environment for players. By designing the game utilizing Verilog HDL, it will be possible to create a game that is both efficient and adaptable to potential future modifications.

We should cite the effort of one of your friends, Mr. Tuğrul Nizamioğlu, in developing the project. You will implement your project in groups of 2-3, where the group size affects nothing. However, in your demonstration and final report, you must designate which member is responsible for which part of the project. You will borrow FPGA boards to try your codes. Since the total FPGA number is limited, each FPGA will be shared by three groups. You will write a report in IEEE article format with a maximum of 4 pages. Finally, you will record a short video of up to 5 minutes to demonstrate your project.

We hope you will enjoy the project while learning Verilog HDL more thoroughly. We should emphasize that this is not a weekend task. Then, it would be best to allocate enough time for researching and implementing.

Project Summary (Game Rules)

This game takes place on a board that measures 10x10, with each square identified by a coordinate in a 2D system. The game involves two types of geometric shapes placed on the board by the player. The board has a black-and-white color scheme. Triangles and circles represent opponents, whereas color preference for players will be a part of your design.

The "triangles team" starts the first round if the score is 0-0; otherwise, the team that won the previous match starts the next round. The starting player selects two coordinates to place their team's shape in the corresponding square on the board. The coordinates are entered sequentially using two input buttons on the FPGA board. One input button is reserved for logic **0**, while the other is reserved for logic **1**. If a player tries to place a shape in an already occupied coordinate, it should be denied by the game, and the player

should be requested to enter a new coordinate. The gaming screen should indicate the player in turn.

The game continues until one side achieves a placement of four identical shapes in a row along either an orthogonal or diagonal line. A red line should indicate the winning placement, and the board should be wiped clean after 10 seconds.

If a player makes a sixth move, their oldest activity should be erased. Then, that square should be unavailable and filled with a solid red color. The same should happen at the 6^{th}, 12^{th}, … moves. The game is declared a draw if neither team achieves the winning placement until the board is 25% occupied, which basically means 25 activities.

Following a win or draw, the board is wiped clean after 10 seconds, and a new game begins.

Implementation Requirements

Input Procedure:

You will use three buttons: **`logic-0`**, **`logic-1`**, and **`activity`** button

At every turn, the code should automatically request the input. Then, the player will enter the coordinates sequentially using the **`logic-0`** and **`logic-1`** buttons. Then, the player should press the **`activity`** button.

Example: Triangles Team's move

If a player wants to place a shape to the **`4C`** coordinate, the player should press the input buttons with the **`0010 1100`** order. Notice that the order for entering a character is reversed so that the code reads **`0100 0011`**. After pressing the **`activity`** button, a triangle should be placed if the **`4C`** box is empty. Now, the circles team does the same.

Whenever new placement arrives via FPGA buttons, the empty box content on the board should be replaced on the VGA screen with the corresponding geometric shape, depending on which team is about to make its turn. If a player is about to fill an already filled box (either a solid red square or any geometrical figure), the player should be requested a new input.

Following the game rules, each team's oldest activity should be erased at every 10^{th} move, and the square should become unavailable for reuse. Therefore, for every ten actions a player makes, a box should be changed with a solid red on the VGA screen.

Screen Requirements:

The main screen should be designed using the VGA interface at a resolution of **`640×480`**. The screen should resemble the red square in Figure 1.

The following features should be presented on the main screen:

- A black and white board with an indicated coordinate system.
- Two geometrical shapes are observed on the left and right of the board. If the triangles team is about to move, the triangle should be filled with solid color, and the circle should be hollow. The vice-versa is valid when the circles team is about to move.
- The total number of moves, wins, and recent position coordinates for both teams should be displayed below the gaming area. The information should be updated with the corresponding team color in the gaming area.

If a win or draw occurs, a message should be displayed under the information in item 2 above. The red line should be drawn across four boxes indicating the winning placement. After ten seconds, the board should be cleared, and a new game shall begin. The following messages should be displayed depending on the outcome of the round:

`"Triangle Team Won! Wiping board..."`

`"Circles Team Won! Wiping board...`"

`"Draw! Wiping board...`"

All necessary deletion operations and statistical updates (total moves, recent position) should be performed on the VGA screen throughout the game.

Bonuses:

- The game might highlight the selected coordinate just before pressing the **`activity`** button. For instance, if the player specifies the **`4C`** coordinate, an indicator should appear on the **`4C`** square.
- The player can inform the game using the **`activity`** button that the player specified a wrong coordinate. Then, the game will allow the player to re-enter a coordinate. For instance, the player wanted to fill the **`4C`** square, but instead, the player pressed buttons corresponding to the **`4B`** button. In this case, the player will be allowed to change the move by using the **`activity`** button.
- Players can start a game with a screen, where some of whose coordinates are randomly occupied with circles, triangles, or squares.

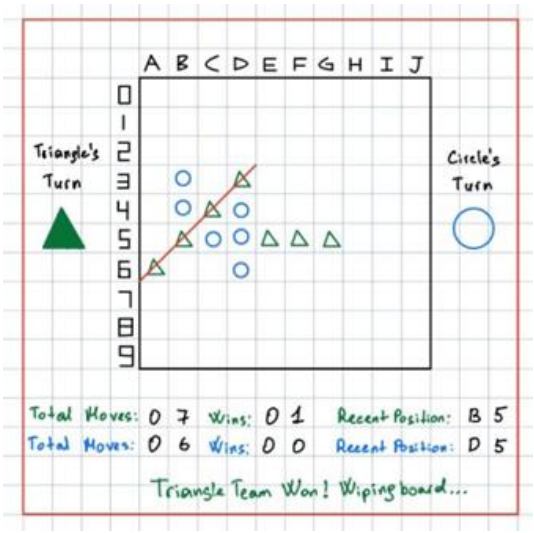

Figure 1: An Example of the Gaming Screen

Schedule

You should follow the plan below while implementing your projects:

$19^{th} - 26^{th}$ May	Registration to Project Groups via ODTUClass
$1^{st} - 23^{rd}$ June	FPGA Borrowing Period
$21^{st} - 22^{nd}$ June	Project Demonstrations
22^{nd} June	Report & Video Demonstration Submission