EE 314 Digital Electronics Laboratory

Spring 2018-2019 Term Project

Fake Quidditch

Final Report

Şeyda Kaya 2166791 – Rafet Kavak 2166783
Middle East Technical University
Department of Electrical and Electronics Engineering
Ankara, Turkey
E-mail: e216679@metu.edu.tr - e216678@metu.edu.tr

Abstract— **In this report, the process of implementing 'Quidditch' game in FPGA is explained. Initially, the aim and the members of the game are described. Then, the codes that is written and some photos of the project are provided.**

Keywords— **Quidditch, boundary, play area, players, balls, bludger, circles, scoring, timing**

I. INTRODUCTION

In our Fake Quidditch game, there are two play areas which are separated by a border. In each area, there are two players. To distinguish the opponent players, players are described with different colors such that players in the top play area are blue and that in the bottom play area are red, as can be seen in the Figure 1. VGA interface is used in this project so that colorful figures can be obtained.

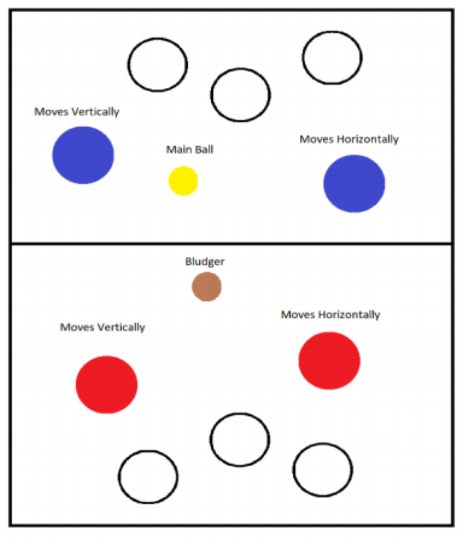

Figure 1 Orientation of the play areas

One of the players in the same color can only move horizontally while the other can only move vertically as stated in the Project Definition. In our design, players on the left side are to move vertically and players on the right side are to move vertically.

As can also be seen Figure 1, there are two other balls in the areas. Moreover, there are three empty circles in each play area, whose size is larger than the size of these two balls. One of these balls is called the main ball and the other is called bludger. By players, main ball is to go through the one of the empty circles in the other play area, in order to score goals. On the other hand, bludger punishes the players if any player hits the bludger. In other words, player which touches the bludger becomes inactive in its present place for 10 seconds. Furthermore, bludger should bounce back randomly from borders. For other cases, it should bounce back according to the physical principles, as the main ball always does.

Movement of the players are manually produced by using the switches in the FPGA since there are not enough buttons to control all players. When each player in the same color scores goal, the current scores are to be displayed over the play areas, as well as the remaining penalty time for each player that hits bludger and remaining time for the game. The duration of the game is three minutes.

II. COMPONENTS

In this project, we have used;

- the DE1-SOC FPGA board,
- VGA cable,
- monitor and
- the program Quartus II.

III. STATE DIAGRAM

In the Figure 2, the state diagram of our project can be seen, which is also the general summary of the previous part.

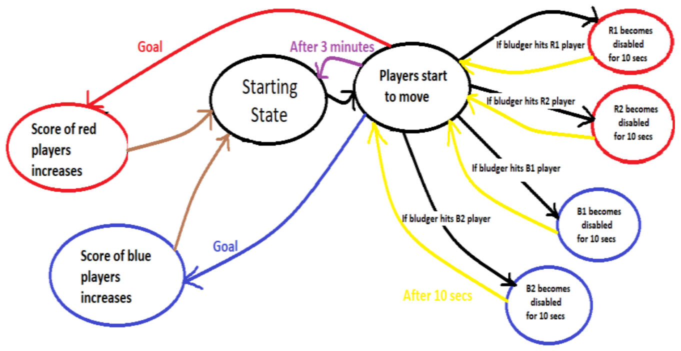
Goal
After 3 minutes
Starting State
Players start to move
If bludger hits R1 player
R1 becomes disabled for 10 secs
If bludger hits R2 player
R2 becomes disabled for 10 secs
If bludger hits B1 player
B1 becomes disabled for 10 secs
If bludger hits B2 player
B2 becomes disabled for 10 secs
After 10 secs
Score of red players increases
Score of blue players increases
Goal

Figure 2 State diagram of the project

IV. VERILOG CODING

To be able to construct necessary modules in the Verilog, we have done some research and examined some concepts. Firstly, we have searched for VGA interface and constructed the necessary submodule. Then, we have designed borders, empty circles, players, balls and we have added colors to them. After that, we have written necessary codes that control the movements of the players and balls. Finally, we have completed the modules by defining the scores, remaining time and the penalty time. In the following parts, these steps are described in detail.

A. VGA Interface

To construct images on a screen, firstly necessary codes should be written. After completing codes, an interface is needed to transmit video signal generated by codes into a monitor. In this project, as preferred in the project definition manual, Video Graphics Array (VGA) interface is used for that purpose. In industry, it is also widely preferred. In VGA interface, a frame is defined as 640x480 image [1]. A frame consists of 480 lines and each line consists of 640 pixels. During writing codes, pixel and line numbers are to be taken into account. To simplify, a frame can be thought as a matrix. In that simplification, columns correspond to pixels and rows correspond to lines. Each frame is displayed on a monitor by starting from the first row-first column element. Firstly, whole first row elements from left to right are displayed, then monitor displays the second row elements. Therefore, in each frame the beginning pixel is at the top left and the finishing pixel is at the bottom right.

In VGA interface, there are also two synchronization signals, which are needed to be able to produce a video on the monitor. These are called the Horizontal Synchronization (HSync) and the Vertical Synchronization (VSync). These signals can be seen in Figure 3.

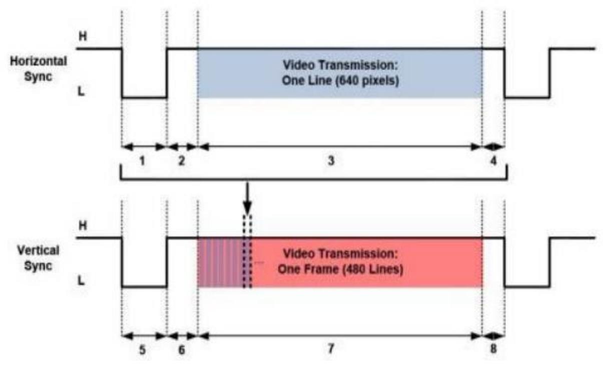
Horizontal Sync
H
L
Video Transmission: One Line (640 pixels)
1
2
3
4
Vertical Sync
H
L
Video Transmission: One Frame (480 Lines)
5
6
7
8

Figure 3 Horizontal and Vertical Synchronization [1]

These two synchronization signals are needed to communicate with the monitor. After a line or a frame is finished to write, information should be given to the monitor. The synchronization signals give the necessary information, and then monitor can start to write next line or next frame. From the Figure 3, also it can be understood that Horizontal Synchronization (HSync) is for synchronizing one line in a frame and the Vertical Synchronization is for synchronizing each frame [1]. Other needed specifications that are described in the manual are also taken into account.

In Verilog, these signals can be formed easily since counter concept is very similar to the digital blocks which are needed to construct the signals. With the help of the given information and timing table in the project definition manual, it is easily constructed.

Moreover, to be able to get an image on the screen, necessary connections should be established. There are three connections to which we need to provide digital data. These connections are called R, G, B, that can be seen in the Figure 4.

Last but the most important specification about VGA interface is that 25 MHz clock signal should be used in VGA other than the internal clock of the FPGA. For that purpose, frequency divider is constructed in top module to obtain 25 MHz from the internal clock, and the resultant clock is used only for VGA. VGA module is also inserted into the top module. Only for ModelSim simulation, it is constructed as different module.

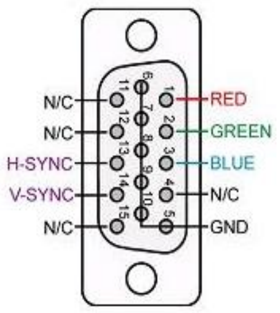

Figure 4 VGA Ports

B. *Top Module*

In this module, firstly borders, empty circles, the main ball, bludger and players are defined. Circles and players have larger diameter than the main ball and bludger. Because all players are in circular shape, the place of these are defined according to their center and the diameter. Center is characterized by its line and corresponding pixel number. For that, we have used the following notations:

```
“reg [9:0] b1x;
 reg [8:0] b1y; “
```

b1 represents the blue player that can move vertically. b1x represents the number of pixels and because pixel number is 640, it needs 10-bit register. On the other hand, b1y represents the number of lines and because pixel number is 480, it needs 9-bit register. Similar to this notation, (b2x, b2y) is used for the other blue player that can only move horizontally. Similarly, (r1x, r1y) is used for the red player that can move vertically and (r2x, r2y) is used for the other red player that can move horizontally. After images for the players are obtained, the movements are to be defined. Because of insufficient button numbers, we have used eight switches to control players; two switches to move the b1 player in the up and down directions, two switches to move the b2 player in the left and right directions, two switches to move the r1 player in the up and down directions and finally two switches to move the r2 player in the left and right directions.

Like to players, the main ball and the bludger are also defined according to their centers with the following notations:

```
“reg [9:0] mainx;
 reg [8:0] mainy;
 reg [9:0] bludgerx;
 reg [8:0] bludgery;”
```

Since the movement of these balls are not manually controlled, necessary specifications should be met by codes. For that, we have defined four integers as

```
“integer  mainx_state;
 integer  mainy_state;
 integer  bludgery_state;
 integer  bludgerx_state;”
```

Here, mainx_state represents the movement of the main ball in terms of pixel such as to right or to left and mainy_state represents the movement of the main ball in terms of line such as to up or down. Similar representation is also made for the bludger. Movement definitions for the main ball are given below:

```
// BALL MOVEMENT IN Y
if (mainy_state==3) begin mainy<=230+mainy-1; end // initial state
if (mainy_state==2) begin mainy<=mainy-1;  end // goes up
if (mainy_state==1) begin mainy<=mainy+1; end // goes down
if (mainy_state==0) begin mainy<=mainy; end // no change
```

```
// BALL MOVEMENT IN X
if (mainx_state==3) begin mainx<=275+mainx-1; end // initial state
if (mainx_state==2) begin mainx<= mainx-1; end // goes left
if (mainx_state==1) begin mainx<= mainx+1; end // goes right
if (mainx_state==0) begin mainx<= mainx; end // no change
```

After that, interactions are defined via Verilog. Necessary interactions were defined in the “Introduction” part. After examining all possible bouncing, the needed codes are written.

To finalize the project, score board, penalty timings and the remaining time of the game specifications should be completed. As can be seen in Figure 5, at the very top of the screen, all these requirements are shown. At the left, scores of blue and red players are displayed. Instead of writing “SCORE”, we have used blue and red squares to represent the scores of the blue and the red team. Then, penalty timings are shown for blue1, blue2, red1, red2 players, respectively. It counts down from 10 seconds for each player which hits the bludger and that player is made inactive during 10 seconds by using the necessary codes. And finally, remaining time is shown at the right, which counts down from 3 minutes. Necessary codes are provided when this report is delivered.

After all specifications are described, a reset button is added to the design. In case of pressing the reset button, the game should return to the beginning state. In other words, players and bludgers are again placed at the beginning locations and timing should be reset, also. Otherwise, the game continues until three-minute time finishes.

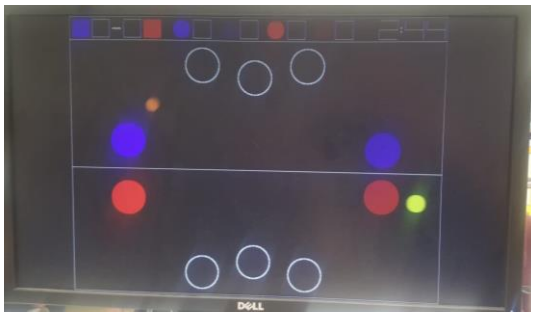

Figure 5 Our design on monitor

V. MODELSIM SIMULATION RESULTS FOR VGA

As asked from us, we have simulated the VGA module with ModelSim. In Figures 6 and 7, clock signal, VGA clock signal, HSync signal and VSync signal can be observed. To show HSync and VSync signals, figures are made very smaller, so clock signals cannot be seen exactly. To show also clock signals, other figures that are zoomed-in version of the previous figures are added. To be able to perform ModelSim simulation, necessary testbench codes have been written, which can be found in the Verilog codes.

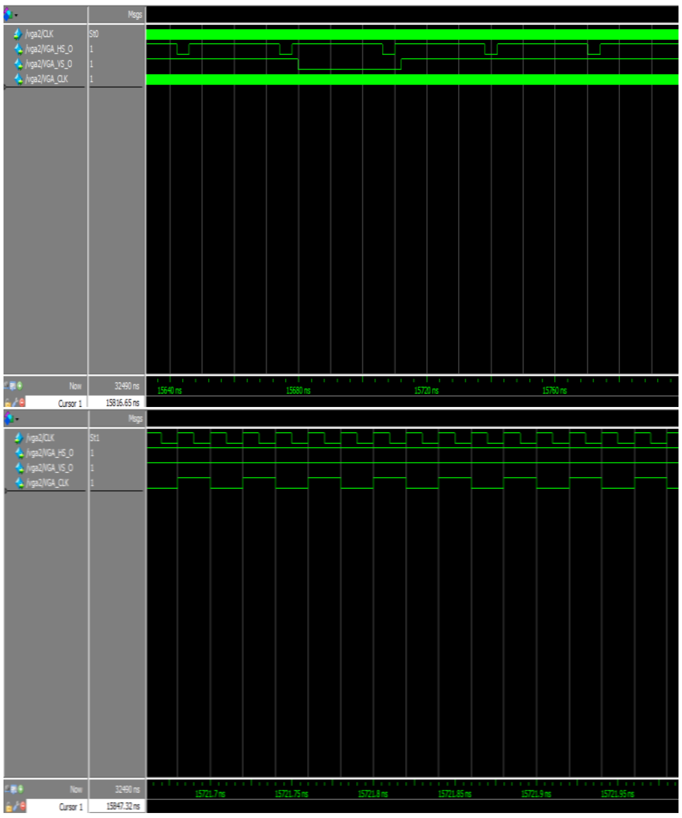

Figure 6 ModelSim simulation for VGA module-1

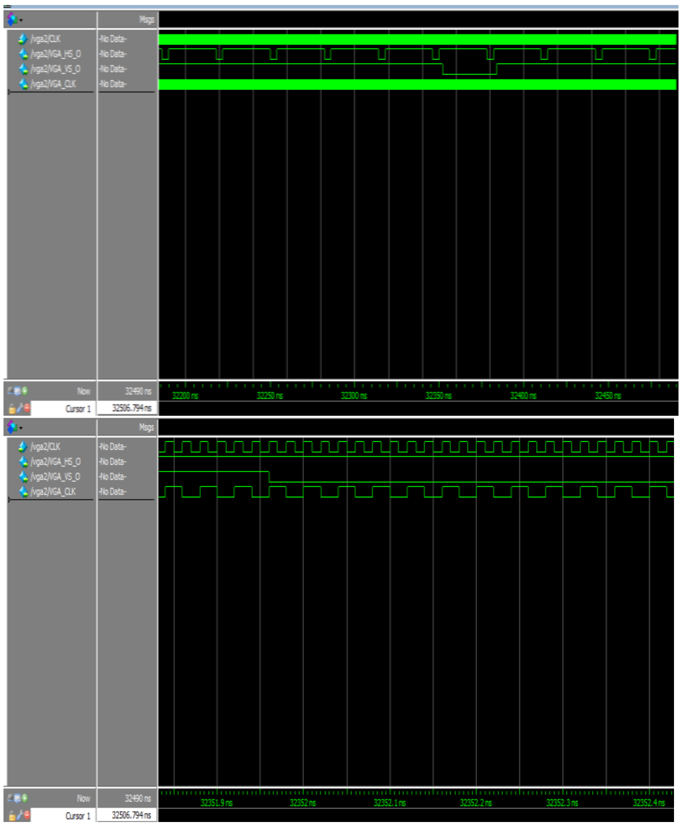

Figure 7 ModelSim simulation for VGA module-2

VI. PROJECT PHOTOS

In that part, some taken photos of our monitor display are given.

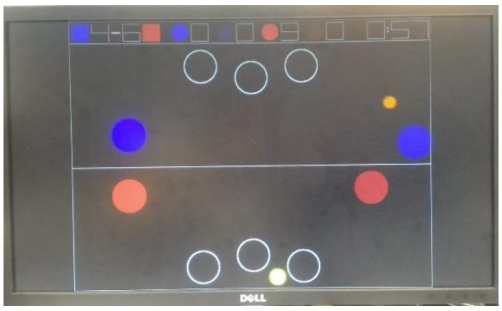

Figure 8 Display when timing is counting down

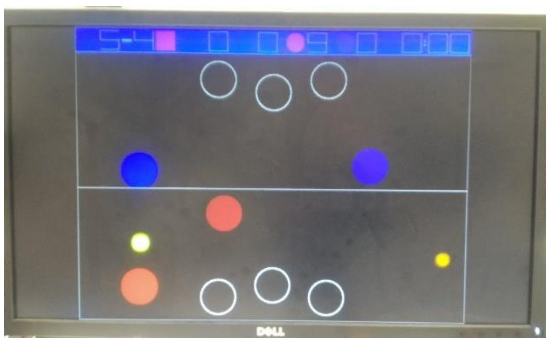

Figure 9 Display when blue team scores more goals than the red team

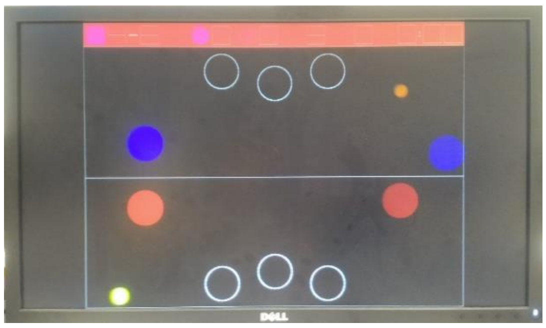

Figure 10 Display when red team scores more goals than the blue team

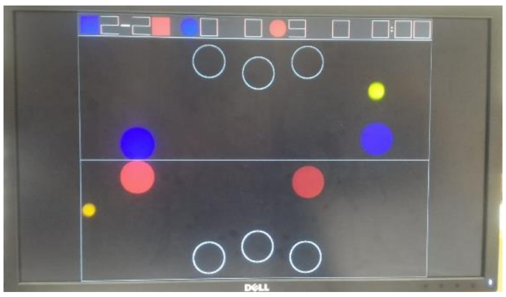

Figure 11 Display when red team's and blue team's scores are the same

VII. CONCLUSION

In this project, we have designed a game which is named as Quidditch. Firstly, we have tried to understand the aim of the game. After that, we have searched for the VGA interface. After constructing the VGA module, remaining game specifications have been examined. The most important part of the project was to construct the top module. We have defined all borders, all color assignments, all players and balls, scoring and timings in that module. Once we have constructed necessary codes for one step, we have checked the accuracy of our codes with the help of simulations and monitor displaying. After we have achieved proper results, we have continued to the other step, just like putting new blocks on main blocks.

During project sessions, we have used our Digital Electronics and Logic Design courses knowledge. Additionally, we have done some research to learn about new concepts, such as VGA interface. The process was fun and we have learned a lot about coding. We have understood that even if very easy actions should be performed in a game, deep understanding and thinking are needed for the implementation of the game; we have had great respect for the people who improves video games.

To summarize, this project has made us more familiar with the subjects which are taught during lectures. With additional concepts, we have gained insight; we have had opportunity to improve our theoretical knowledge with practical implementation.

VIII. REFERENCES

[1] Odtuclass.metu.edu.tr. (2019). *ODTUCLASS 2018-2019 SPRING: Log in to the site*. [online] Available at: https://odtuclass.metu.edu.tr/pluginfile.php/273378/mod_resource/content/2/ee314%20project%202018-2019.pdf [Accessed 26 May 2019].