# Maze Runner

Jonatan Cogiel, Gabe Espinosa, Joni Llana, Roman Kulikovskiy
Electrical and Computer Engineering Department
School of Engineering and Computer Science
Oakland University, Rochester, MI
emails: jonatancogiel@oakland.edu, jonillana@oakland.edu, gabrielespinosa@oakland.edu, rkulikovskiy@oakland.edu

***Abstract* - The maze runner implements the FPGA to create an interactable and enjoyable game for users to play. Its purpose is to help show off the potential of VHDL and its interface; in conclusion the language is a useful tool for programmable design and integrated circuits.**

## *Introduction*

Our group consisted of casual gamers. During our meeting to decide our project, we brought up the idea of possibly creating a game and decide to advance with that idea. The game we decided to create is a maze runner game. The goal of the game is complete the maze without touching any of the walls before the timer runs out. To implement the game, we needed to have a good understanding of how VGA works with VHDL. The VHDL portion consisted of multiple files that included drawing different stages of the games, the player, the timer, and the winning and losing conditions. After that feature of the coding was finished, the next step was to determine the functions of each part. For instance, the players movements and the walls conditions. To determine if the code worked, we used a monitor with a VGA connector to test the code. We got a functional game to work and all that was left was to debug and optimize the game.

## *Methodology*

### *VGA Display*

The project was allocated to two major portions; coding the VGA onto a monitor to display the game and coding the functions of each element of the game.

The front porch is an interval period between the end of picture information and start of horizontal pulse. The level of front porch is a high and the purpose is to clear any signal level that remains before the horizontal pulse occurs; the duration of the front porch is very short.

The back porch is the duration between end of horizontal pulse and start of the next line with video information. The back porch lasts more than front porch and the main purpose is to give the time to beam scanning for reverse direction (right to left) to start new line.

Due to the concept being new and having no prior experience; writing the code for the initial maze map was the most problematic aspect of the VGA portion. By creating an array that was named bitMap and making x and y values 32 bits each, the map was created to display a grid of a 32x32 matrix maze. To be able to convert the pixels into a 32x32 matrix, division was required. The scaling of the map itself had a vertical integer value of 1024 pixels and a horizontal integer value 1280 pixels. To make the scaling 32x32, both integer values were divide by 32. To make further additions or levels to the game, the same theory can be applied to draw different mazes. The VHDL code stays consistent except for the bitMap for the VGA portion of the project.

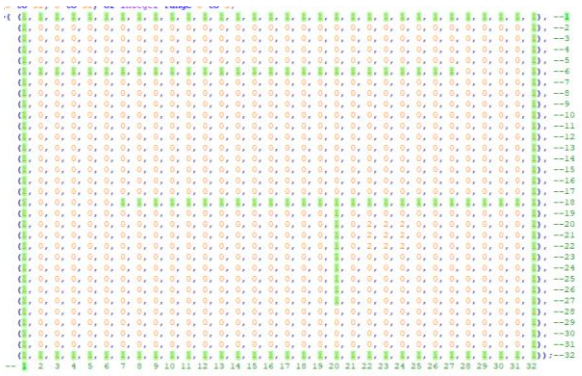


**Figure 1:** Level 1 Bitmap

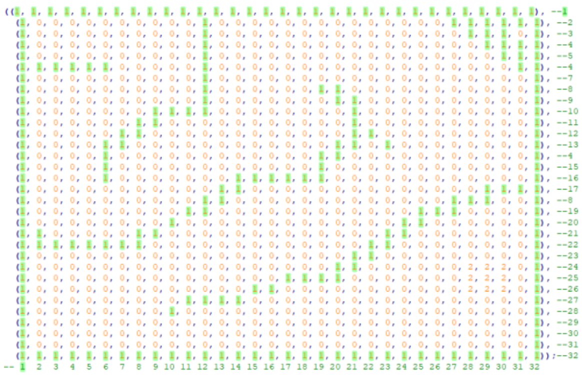


**Figure 2:** Level 2 Bitmap

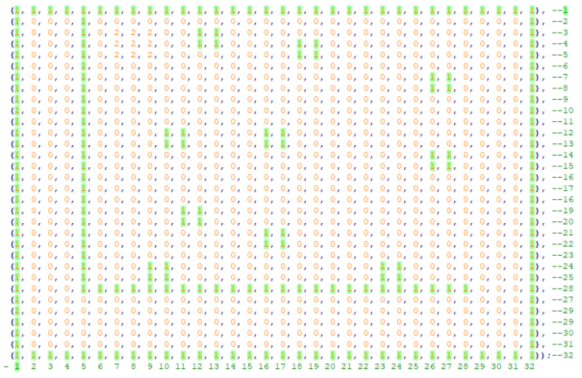


**Figure 3:** Level 3 Bitmap

Aside from the mapping, a clock will be set to time the user in completing the game; depending on completion of the game, a win or lose condition will appear on the screen. As for the player themselves, it takes up 4 squares in the matrix. There is an initial value set up at the top left corner of the map.

RGB lighting was implemented during the process of the game to create a vibrant display. The maze, player, and titles all have specific colors given to show diversity between each component. RGB color coding constructs different combinations of red, green, and blue using 12 bits. For example, the color code below displays the color red.

```
rgb <= "1111 0000 0000"
```

***Figure 4:*** The code above represents an example of RGB color coding.

***Functionality***

The game consists of three different levels that were each drawn using a bitmap of a 32x32 matrix shown in Figures 1 through 3. For levels 1 and 2 we implemented two different game physics into the game. In level 1, the player uses the buttons as inputs to accelerate and decelerate the controlled object; while level 2's game physics consist of moving a block per button click. Level 3 consists of a different map but the same functionality as level 1.

Level switches occur when the states change in the finite state machine (FSM). The FSM not only changes levels but it also determines the surroundings of the game and the conditions. These include the open spaces, walls, and the win zone that will direct you to the next level which in turn will lead to the next state. Open spaces are indicated by 0's on the bitmap figures, walls are indicated by 1's, and the win zones are 2's. In our case, when any part of the controlled objects hits a 1 (wall) on the map, they get sent back to starting position. The other conditions of the game involve the loss and win result. The player needs to complete all three levels of the game to win, while on the other hand they simply need to let the timer run out to lose. Once these conditions are met, a green or red image appears depending on the result of the game.

Some improvements that could be made include more game physics such as a jumping mechanic, adding images to the player figures, and creating more levels for a more competitive aspect.

## *Experimental Setup*

When testing the code, the first step was to make sure the monitor can receive and transmit the data coming from the FPGA. Once the code was completed, a test was ran proving the functionality of drawing images on VHDL and displaying them onto the monitor. Next, the game logic was implemented onto the monitor, and to make sure the controller inputs functioned properly.

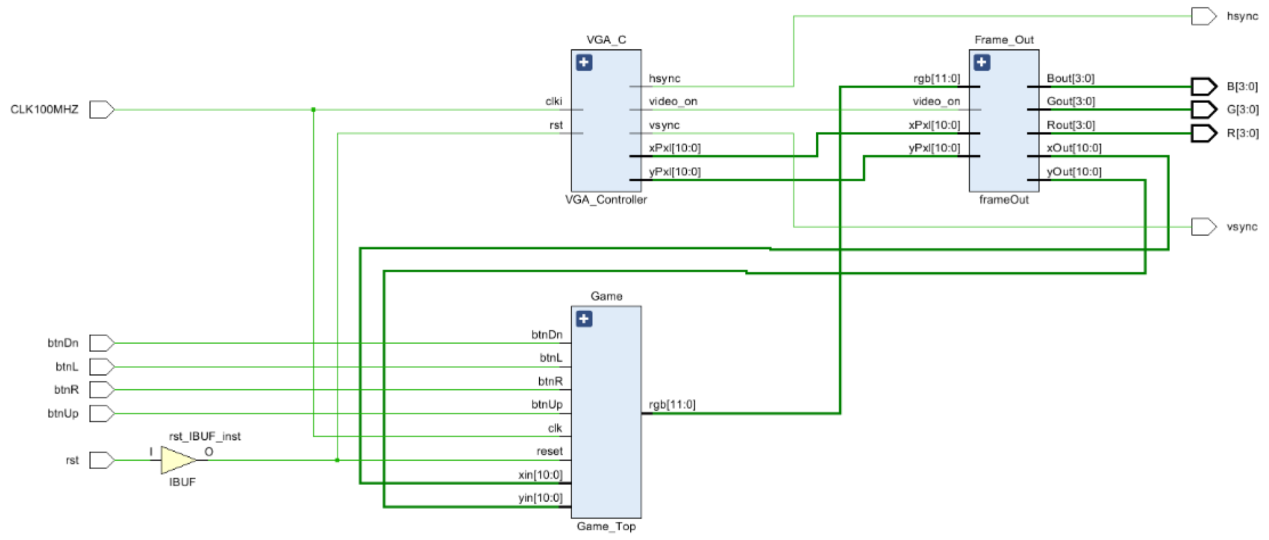
## *Results*

The results that were obtained in this project were from a visual and an I/O correspondence. When testing the maze game, the player would play through the different levels, in which case they all had different pathways as well as movement for the player themselves. What this did in turn was allow different forms of implementation to be accessed for the player, and it showed how different forms of "physics" could be applied to the movement of the player.

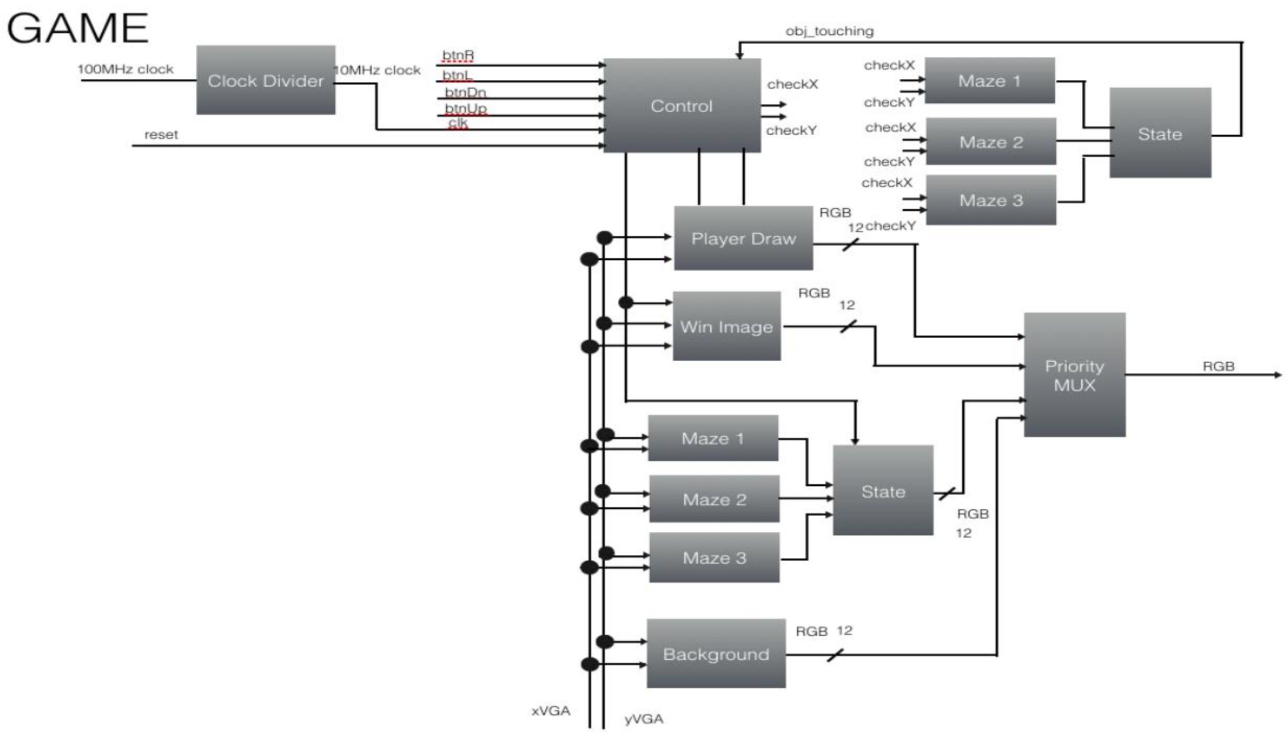
The portion from class that this project related to was from unit 7, where the usability of the VGA connector was discussed.

## *Appendix*

http://ece-research.unm.edu/jimp/vhdl_fpgas/slides/VGA.pdf



VGA_C
VGA_Controller
clki
rst
hsync
video_on
vsync
xPxl[10:0]
yPxl[10:0]
Frame_Out
frameOut
rgb[11:0]
Bout[3:0]
Gout[3:0]
Rout[3:0]
xOut[10:0]
yOut[10:0]
hsync
B[3:0]
G[3:0]
R[3:0]
vsync
CLK100MHZ
Game
Game_Top
btnDn
btnL
btnR
btnUp
clk
reset
xin[10:0]
yin[10:0]
rgb[11:0]
rst
rst_IBUF_inst
I
O
IBUF



GAME
100MHz clock
Clock Divider
10MHz clock
reset
btnR
btnL
btnDn
btnUp
clk
Control
checkX
checkY
obj_touching
Maze 1
Maze 2
Maze 3
State
Player Draw
RGB
12
Win Image
Priority MUX
RGB
Background
xVGA
yVGA