Winter Climb

ECE287 Final Project Report

December 16, 2016

Group Member:

Gabe Skidmore

Shifan(Oliver) Luo

1. **Introduction**

For our project, we designed a climb game by using the knowledge of previous Digital System Design labs. Also in this project, we need to implement new hardwares and new features that we didn’t learn in those labs. So, by learning those knowledge, we did researched online and mostly got help from classmates. The project demonstrates our knowledge of the FPGA.

Our aim of the project is to design a fully functioning game with play players using the keyboard and displays the game on the VGA monitor. The whole project based on verilog. And the project relies on the integration of both software and hardware. As the software part we used the Quartus to implement and program the game. For the hardware part, we connect the VGA monitor and the keyboard to the FPGA board.

1. **The Game**

To display our game, we used VGA and on the VGA, we have a 2-D platformer. The game has two players, each player controls an object by using the keyboard. The controls for player one is the w,  a and d keys and the controls for player two are i,  j and l. The a and j key move player one and two to the left, respectively. The d and l key move player one and player two to the right, respectively. The w and i key makes player 1 and 2 jump, respectively. Both players also have an attack key that will make the other player restart from where they initially spawned. The attacking works by if there is collision between player one and two, that value is recorded. If that value is high while the attack key is pressed, the player who was attacked will go back to the beginning where they started. The attack key for player one is the f key and the attack key for player two is the h key on the keyboard. When a player starts to move in a direction, the player will continue to move in that direction until they change direction or hit a wall or the side of a platform. The goal of the game is to reach the gold rectangle on the top platform. Each player will have to jump on the platforms until one reaches the goal. Once someone reached the goal, there score is incremented by one internally but is not displayed on the VGA. The game is over when someone reaches three points and this is shown by the looser disappearing. When trying to reach for the goal, the players have to take into account falling along with jumping.  If the player misses a platform they were jumping towards or moves past the edge of the one they are currently on, they will fall off until the land on another platform or the ground. Whenever the players want to restart the game, the only someone has to do is hit the backspace key on the keyboard and the game will restart with the scores restarted as well.

When I say code, I mean it very loosely because it is not the same as high level languages like have and C++, this is a lower level language where we are designing hardware. The code for the game is long but is not hard to follow, the VGA and keyboard have their own modules. The module for the keyboard is called keyboard and has a comment above of it that says it’s the keyboard. The keyboard module is also the place the keyboard data is imputed to determine with key is being pressed or if no keys are being pressed. The keyboard module doesn’t use the keyboard clock because the keyboard clock was too error prone and would cause the keyboard data and reading of the data to go out of sync and stop working. The VGA has a few modules for it and they are all important to get the VGA to work properly. The VGA code should not be touched unless the current one does not show up. Near the top of the code is where the objects x and y coordinates are determined. If you want something to show up on the screen, this is where you want to input it's left x and top y value. The next code after this is where the objects are declared and where the sizes of the objects are determined. Each object has a corresponding top, bottom, left and right value. The top and left for each object is set to the x and y from the previous code and the bottom and right are set the top and left respectively plus a set number of pixels. The next part below this is logic of the game and it is where everything the players do is determined. Whenever a key is pressed, a register with the corresponding name of the action to be performed is set to one. When the jump key is pressed, the jump counter starts counting until the value Is above a threshold. Once the jump counter is above the threshold, the jump register is set to zero and the fall register is set to one. Unlike jumping, falling doesn’t have a counter, the player only stops falling if they land on the platform, the ground or the goal. The object's speed is based on how fast the update counter runs, if we made the update run more threshold smaller, the objects would move faster and vise versa. This was the best way to do this because the clock ticks 50 million times a second so you can't have the player move bases on the clock tick. If you do make the player move based on the clock tick, the objects looks like they teleport to the ending position. Near the top of the logic, there is a piece of logic that checks for collision between player one and two and it sets the collision register to one if there is any overlap between the players. For all the static objects, object collision is not used. Instead, the players x and y value are not allowed to be in the same range as the platforms. This is the same for the goal except that when a player reaches the goal, that player's score is incremented by one and both players are set back to start. Both players are also set back to start to start when the backspace button is pressed and there score is set to two. Instead if starting at zero, since the player doesn't see the scoring doesn't matter what the starting value is as long as it can still count up three points. Below the main logic for the code is where boxes are set and the boxes are used to change the color of the objects so they aren't the same color as the background. For each object, there is a corresponding if statement that runs if the object exists and box unique to that object is set to one. In order to say who loses, there is one line of logic that has the players showing up only when the other player has not reached the reached the goal three times. Farther done, there is a module that determines the color of the objects based on the box passed it as the parameter and is set by the RGB scale.

1. **Conclusion**

We think our project is completed successfully. Although our scoreboard is when a player win, the other player just despairs. But the scoreboard still works like track which player wins. Overall, it is a fun game, we think the attack system and it has two players really made the game fun. Before starting our project, we only learned the basic FPGA board. During our project, we got lots of help from other group. This project taught us how to make our hardware implement working through programmable means. After the project, we definitely learned more than what we learned in the previous lab.