Lab 8

Name: Huy Huynh

Student ID: 1925597

Demo: Live demoed to Brian on Thursday 6/3

Flappy Bird block diagram:

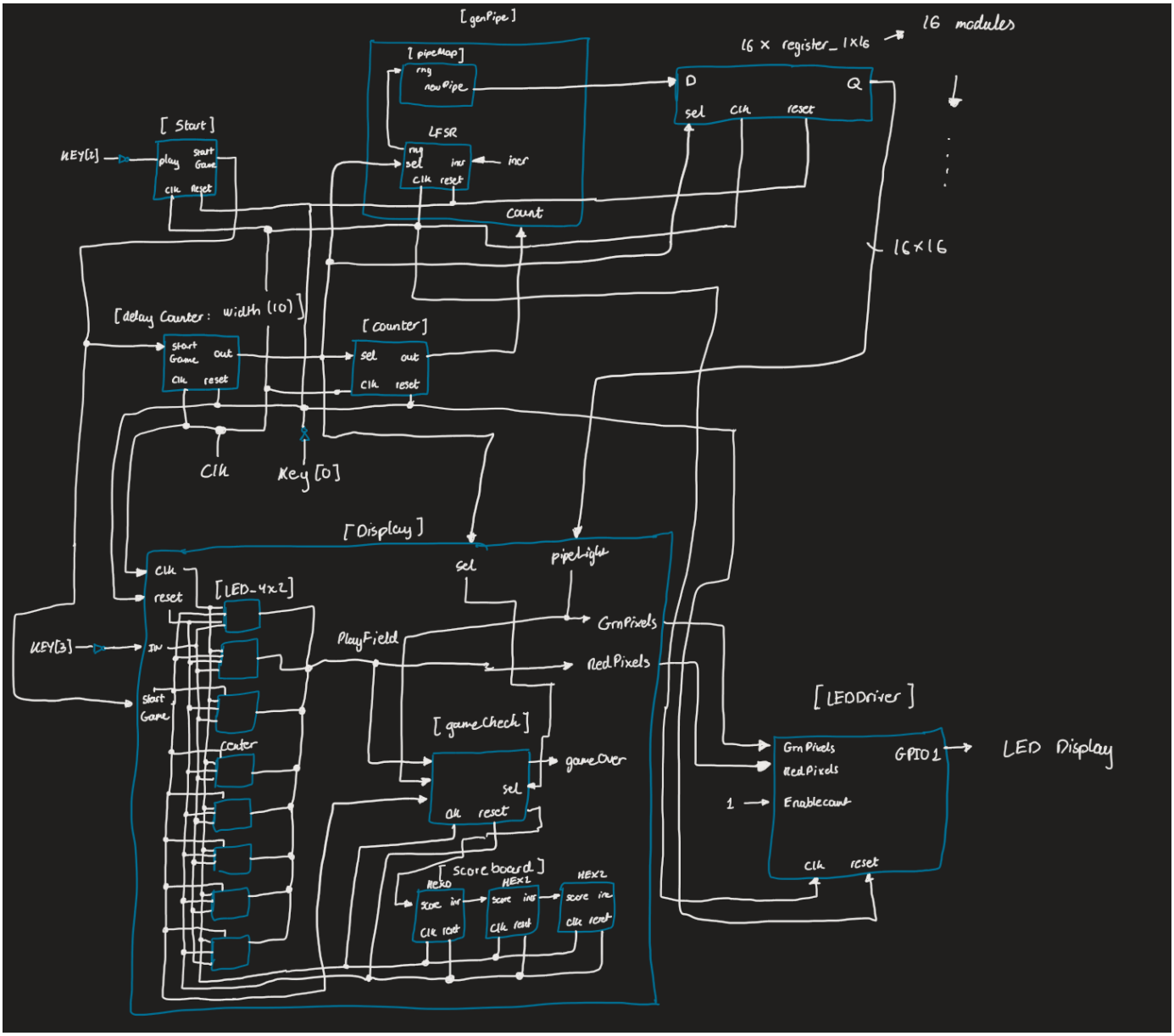

Figure 1. Flappy Bird block diagram

Description of operations:

The three controls of they game are KEY[3] (tap or hold to move the bird up otherwise the bird will fall down), KEY[2] is to start playing on initial start up or after a game over and KEY[0] is to reset the system. Note after a game over you will need to press reset and play to start the game again. On initial startup the LED will display “Play” and will also display “Play” when game over. Similar to the original Flappy Bird, the goal is to avoid the maze of pipes and gain a high score which is based on the number of pipe pass. When the bird reaches the top it will remain at the same position, but the bird will fall through the bottom of the screen and the game will end. The score is tracked on the HEX0, HEX1 and HEX2 and will update after passing each pipe. The scoreboard max score is 999 and will remain after game over until the system is reset. The game is played on the LED extension display. The bird is represented by a 2x2 red block which is situated 6 pixels to the right of the left boarder. The game is supposed to be played with the buttons and switches of the FPGA board facing toward the player.

To generate the pipe map, I used an 8-bit LFSR to get a random 8-bit value then split the 8 bits to two 4 bits value representing top and bottom pipe. The requirements of the generated pipe are that the minimum length is 2 pixels, 3 pixels wide, the horizontal spacing between each pipe is 6 pixels and the vertical gap in between the two pipe is always 5 pixels. To calculate the length of the pipe first the two 4 bits value is compared to see which one is greater and the corresponding top or bottom pipe is calculated first. The greater pipe length is then equal to the first 3 bit of the 4-bit value which is 0-7 + 2 from the minimum length requirement. The smaller pipe length is then 11 – greater pipe length since the left-over space is 11 (16 – 5 minimum vertical space). Not all of the 8-bit value is used but the pattern generated based on my method is less repetitive than when I use a 6-bit LFSR.

The generated pipe length value is then pass through 16 1x16 shift register which represent the 16x16 LED display. The shift registers and pipe generation module update every 2^10 positive edge of the system clock. The clock of these module is slowed down so that the movement of the pipe is playable. Also, the player bird movement is also delayed and update every 2^8 positive edge of the system clock. This is the best setting for fluid movement for the player. Collision is checked by seeing if both the Green Pixels and Red Pixels at the same position is turned on since the bird remain in the same place we can just check the 2 columns where the bird is.

ModelSim Simulation:

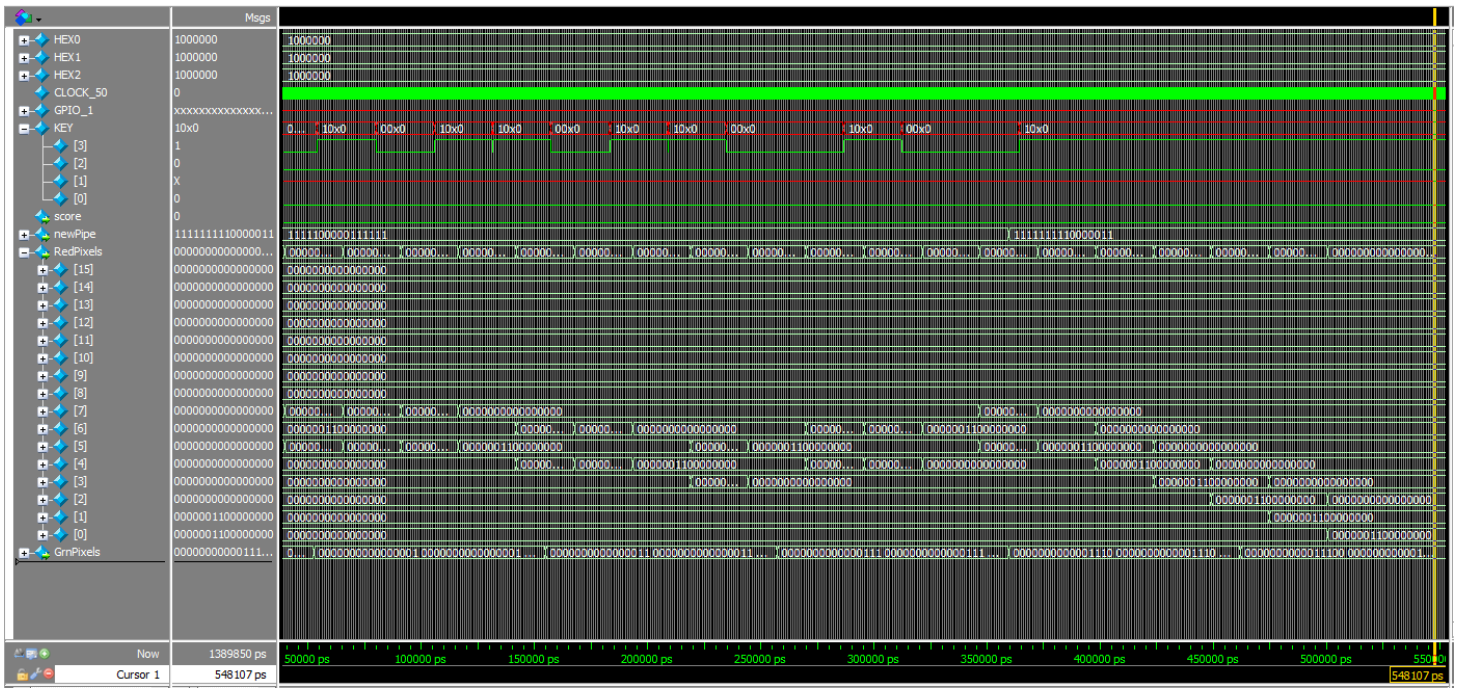

Figure 2. Flappy Bird game simulation showing Red Pixels

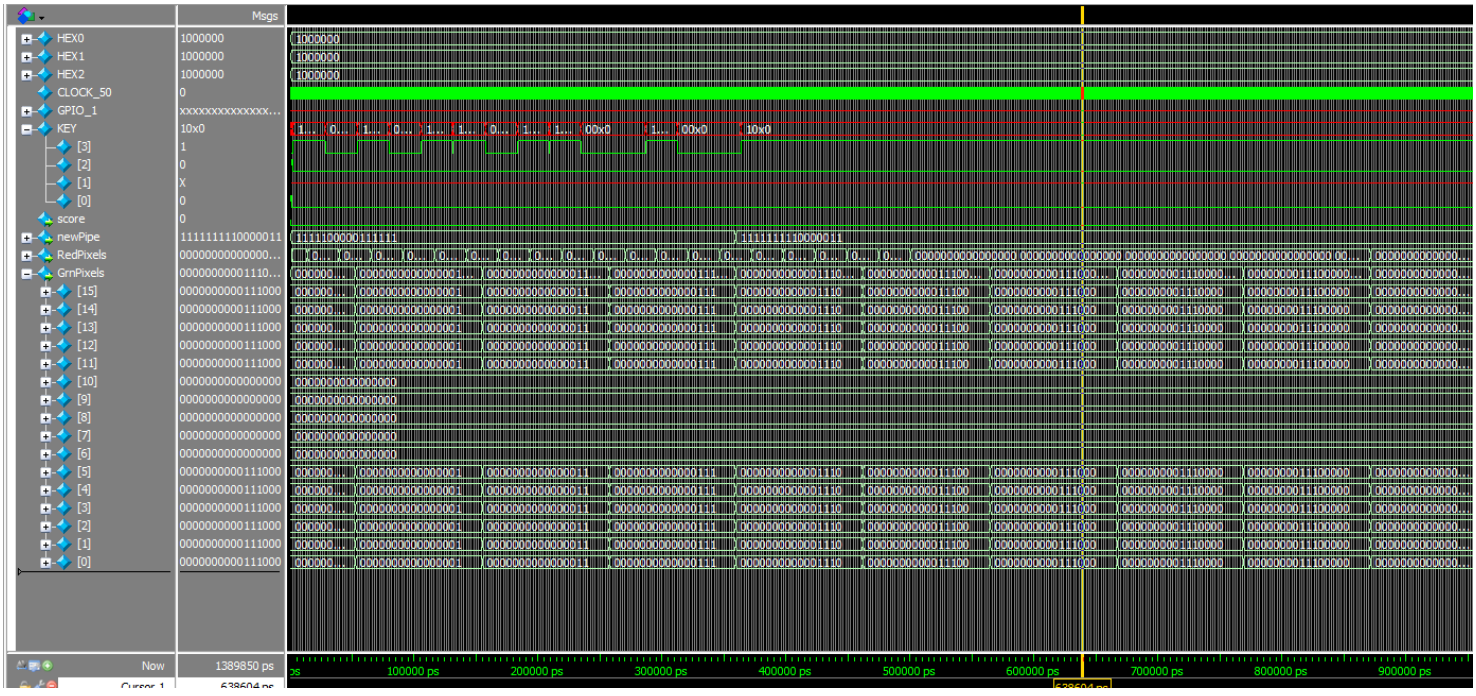

Figure 3. Flappy Bird game simulation showing Green Pixels

The system is initial reset then the play button is pressed. Controlling the bird through the simulation is kind of difficult so most of the testing was done through the individual modules and on the FPGA board itself. KEY[3] is pressed so the bird will slowly make it way to the top and from figure 2 you can see the bird moving up and down in the middle for a bit then moves to the top and stays in that position until it hits the pipe and game over. From figure 3 we can see the beginning of the pipe moving the left until it reaches the middle where it collides with the bird and result in a game over.

Time Spent: I spent a total of 4-5 days in total on this lab. The design of the collision system and using for loop took me the most time to figure out.