Problem Description

The project was to utilize the Altera DE2-115 board to implement a game of two player pong. The board would create the game and display it to an external screen connected by a VGA cable. The control of the paddles in the game stem from buttons on the board as well as input from a keyboard. The project will involve researching and learning the vga output as well as keyboard inputs and getting them to interface with the image.

Background

The game of pong will be built from the ground up in verilog, meaning that the whole game will be controlled by logic statements and digital logic. The creation of the game and logic is based upon our design, however, our methods of interaction are more constricted. The FPGA contains both a PS2 input slot as well as VGA output pins. These inputs and outputs have specific hardware parameters and standards which must be met in order to operate accordingly. Therefore, research and code must be taken to make these interfaces useful.

Design Description

The project design began by splitting the project into 2 portions: visual design and keyboard interface. This was done because both peripherals needed to be implemented, and the visual aspect could be fed the keyboard inputs in order to make it operate.

Visual Design

The visual design began by researching the VGA system and learning how the screen outputs images and color. This research taught us about the “scanning” system of the screen output. The values for each pixel are assigned from the top left of the screen and written from left to right and then proceed to move a line down and write the following line of pixels, until the entire screen’s pixels are written, in which case the pixel writer returns to the top left. This occurs at a rate of 60 times per second. Bailey Hull wrote the code to get the screen to a certain color, complete with pixel writers, vertical and horizontal resets. This code was then modified to allow for shapes to be drawn on screen in different colors. We kept to a simple 2 color system in respect to the original game.

The next hurdle we had to face was getting the images on screen to change locations and simulate animation. This began with the idea of using a “ticker” which would go true when the pixel writer was just starting. This proved not to be useful as the ticker would be true at a rate of 60Hz, but the calculations that the board was doing were based on a 50MHz clock. therefore, the images would change locations sporadically and sometimes not resemble the image that needed to be portrayed . However, through this design, we learned and implemented an overflow catcher, which set bounds for our image showing up.

The next implementation of our animation included creating a timer at about 500Hz. This counter would count to 1000000, which is one five hundredth of the clock speed, and this counter as well as the value were used to control the inputs and refresh of the images. This also allowed us to get the button inputs on the DE2 board to function accordingly, because now the values added to image location could only be added at this frequency. Through this, the paddles could be operated by using 2 key buttons, which was a great stride in our project.

The next portion of the project required us to get the ball to move and bounce off of the edges as well as the paddles. This turned out to be a little challenging, especially when dealing with the ball changing directions. The initial plan was to have a velocity variable for X and Y coordinates that would be added to the ball, and reversed when the ball ran into an obstacle in the X or Y direction. This proved to fail because of verilog’s failure to handle negative numbers in an appropriate fashion. Therefore, the x and y velocity variables became a boolean, with 1 being movement in the positive direction and 0 in the negative. Therefore, getting the ball to bounce horizontally and vertically was solved. The only problem that remained was to clean up the logic with the paddles’ boundaries as well as the reset conditions. The paddle’s boundaries were easy to create and make the logic for. However, the reset conditions as well as the losing/winning conditions proved to be a bit of a challenge. The initial idea was to create a boolean with a win condition, which, when true, would fix the ball’s location when the game was won by a player. This proved to be challenging because the ball continued to move. Therefore, the logic was amended to the ball’s movement structure and would not let the ball move as long as the system sensed no inputs. Through this, we created the pong game and got it to display on screen.

Keyboard Interface

The keyboard part of the project required us to read signals from the input of PS2 keyboard. The goal was to move a paddle with the keys of a keyboard. To do this required the research of the layout of incoming signals, from the device to the fpga that is. The information comes in 8 bit packets for each key. The data all comes in like a serial stream. For example the keys “A” and “D” being pressed back and forth sends the code in the following order: 1C-23-1C-23. ..etc. For the keyboard to work effectively with the DE2-115 in retrieving the date, we needed to assign the ps2\_clock and ps2\_data pin assignments. The ps2\_clock tells if the signal coming in is valid during the falling edge of the clock.

The scan\_code is the most important part of this part of the keyboard section. This is the actual scanning of the keys. There are 3 different states that scan\_code recognizes. These are the single pressing of the key, the continuous holding down of a key and the releasing of the key which is known as the break code. The pressing down of the key as mentioned above sends a signal 8 bits wide, that is actually represented by two 4 bit hexadecimal numbers (example: A=8’h1C). We have the hex values of the keys displayed on the seven segment display just to make sure everything is being read correctly. When nothing is being pressed, an F0 shows up on the seven segment display along with the value of the last key pressed.

We used code from john loomis online to get the actual keyboard to interface with the DE2 and got rid of the elements that did not apply to our project such as the code for the inout GPI0 which is not used in our project.

Bringing it Together

To tie everything together, we call our keyboardYo module which fetches the scan code and displays the values of the key signals on the seven segment display. Due to some issues that came up in the project, we couldn’t manipulate the scan\_code in the module keyboardYo, so we did so in our top module ps2\_lab. Two dummy variables were made called, dummy0 and dummy1. We make each of those signals high when the scan\_code reads the A key or the D key which moves the paddle up and down respectively and low when they are not. These signals are then passed to the pong visual section of the project. Then depending on the signal sent over with the dummy variables, the paddle will move up and down or not at all.

Presentation Video

<https://www.youtube.com/watch?v=aoykkMRwATU&list=UUQo_7qLTe0zvgs8S51RfMfA>

Conclusion

Through this project, we learned about hardware specifications as well as how to approach and think differently to solve certain problems with both known and unknown challenges along the way.The project taught us how to break down a project into chunks to work on it more effectively as well as how to design and implement a design from start to finish, much like an actual engineer.( Not to mention how much this felt like a desk job during finals week. )

design files

Citations

John Loomis page

Bailey Hull’s project