Ressa Reneth Sarreal
CMPE415_HW6
Professor Mohsenin
Due: 04/27/2017

INTRODUCTION

The objective of Homework 6 was to utilize our knowledge on interfacing the FPGA with VGA while being able to implement a game of pong. The game is completely functional. The project is stored in HW6_pong folder.

MODULES & FUNCTIONALITY

The following modules were developed to implement the game:

- p1_paddle
- cpu_paddle
- cpu_delay
- ball
- ball_delay
- collision_detector
- rotary_oneshot
- display

The following approaches were taken to implement the modules:
NOTE: I did not implement the rotary_oneshot module so I will not describe it below

p1_paddle:

1. Wait for the clock's positive edge
2. Check if there is a signal from reset
    a. If so, send the paddle to its initial position
3. Check if there is input from the rotary knob
    a. If it is clockwise, move up 32 pixels unless that location puts the paddle off screen
    b. If it is counter- clockwise, move down 32 pixels unless that location puts the paddle off screen

cpu_paddle:

1. Wait for the clock's positive edge (the clock will so so ever 1/8 of a second
2. Check if there is a signal from reset
    a. If so, send the paddle to its initial position
3. Check the ball's y-position & match it

cpu_delay:

1. Take in a 20ns period clock signal
2. Assign a 6240000ns period so that the new period clock signal is 1/8 second. Assign a 40% pulse width (2500000ns)
3. Count to the pulse width value with output register set to high
4. When the pulse width value is hit, set the output register to low until the period value is hit
5. When the period value is hit, set the counter back to 0 and repeat steps 3-5

ball:

1. Start the ball at its initial position and velocity
2. At reset the ball will go back to its initial position and velocity
3. If the ball collides with a paddle, reverse its x-velocity
4. If the collides with the upper or lower wall of the screen, reverse its y-velocity
5. The game is over when the ball passes behind the x positions of either of the paddles

Ressa Reneth Sarreal
CMPE415_HW6
Professor Mohsenin
Due: 04/27/2017

ball_delay:

1. Take in a 20ns period clock signal
2. Assign a 3125000ns period so that ball travels across the screen in 4 seconds. Assign a 40% pulse width (125000ns)
3. Count to the pulse width value with output register set to high
4. When the pulse width value is hit, set the output register to low until the period value is hit
5. When the period value is hit, set the counter back to 0 and repeat steps 3-5

display:

1. Take all positions of the paddles and the ball
2. Assign each item a width and height
    a. Paddles are 64x1 pixels
    b. The ball is 3x3 pixels
    c. The field is 480x640 pixels
3. Overlay color over top of the appropriate locations depending on the positions provided, and the shapes' height and width properties

METHODS & TROUBLESHOOTING

When troubleshooting, I ran my code and saw what the output was on the screen. I decided that I could visually determine what was going correctly or incorrectly in my program. I began coding the display until I could see one paddle (p1). I then coded the p1_paddle module until I could move the paddle correctly and did not cross the boundary. I then coded the ball's behavior. Once the ball could move and bounce off the walls, I then coded the cpu_paddle. I then added the delay clocks for both the ball and the cpu_paddle. Afterwards, I put all of it together and then coded the collisions. I was then able to see the ball collide with the walls and paddles appropriately.

Outside of visually testing my code, I also ran a test bench on my collision_detector module. I wanted to see if collisions were detected at the correct times. The output is shown below:

```
ball_pos(320, 240) p1[ 80, 144] cpu[240, 304] collision: 0
ball_pos(64,  240) p1[240, 304] cpu[240, 304] collision: 0
ball_pos(64,  240) p1[240, 304] cpu[240, 304] collision: 1
ball_pos(64,    0) p1[400, 464] cpu[240, 304] collision: 0
ball_pos(576, 240) p1[ 80, 144] cpu[240, 304] collision: 1
ball_pos(576, 240) p1[ 80, 144] cpu[240, 304] collision: 0
ball_pos(576,   0) p1[ 80, 144] cpu[400, 464] collision: 0
```

CONCLUSION

During this homework assignment, I learned how to implement more complex systems on the FPGA that have VGA display capabilities. I have become somewhat comfortable with implementing multi-module systems with a VGA. I also learned about how to implement several

modules into a cohesive game that can be adapted in many various ways. For example, in addition to the requested design, my design supports a change in color when the game is over. The p1 paddle is magenta during the game and the cpu paddle is cyan, while the ball is white. When game over condition is satisfied, the background turns white, from black, then the paddles and ball turn black.

APPENDIX | TOP MODULE SCHEMATIC