Lab 4: Paddle Ball Game
CMPE 415
UMBC

March 10, 2016

You are going to create a paddle-ball game displayed on a monitor using the rotary switch (i.e. a video game).

1 Game description

The game should fit these descriptions:

- To start the game, press RESET/BTN_SOUTH. Upon release, a "ball" and two "paddles" will appear on the screen.
- Paddles
  - The paddles are 64 pixels tall and 1 *pixel* wide. There is one on each side with a space of 64 pixels to the sides of the 640×480 *pixel* field. The paddle should be white.
  - The user paddle should be on the left and should be controlled using the rotary switch. Each rotation increment left or right should correspond to a paddle movement of 32 *pixels*.
  - The computer paddle (CPU paddle) should be on the right. The computer should attempt to track the ball by updating its position every $1/8^{th}$ of a second, according to the ball's vertical position and the CPU paddle position the computer decides which direction the paddle should move if at all.
  - Paddles may touch the top and bottom of the field, but should not be allowed to go past the edges.
- Ball
  - The ball will be at or near the center of the screen when the game starts. It is implemented as a 3pixel $\times$ 3pixel square and should be the color green.
  - The ball should start moving at 45 degree angle a rate that would allow it to cross the screen in about 4 seconds.
  - Update the ball position by adding/subtracting 1 to its horizontal and vertical position. Do this with some intermittent delay, T_d, that will determine the ball's velocity. Implement T_d as a multiple of the master clock.
  - The game should freeze when the ball overlaps the leftmost or rightmost column of the 640x480 field screen.
  - If the ball is adjacent to a paddle, it should reverse its horizontal direction and the velocity should increase (T_d should decrease) by some amount.
  - If the ball touches the top or bottom row of the field, it should reverse it's vertical velocity.

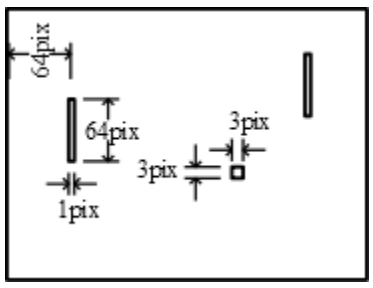

Figure 1: Depiction of display

2 Design Approach

The design approach and structure is up to you. Here is one modular approach that breaks up the design into many small pieces. I don't represent that this is the best or easiest. It just seemed like the easiest one to describe. More complex behavioral units would allow for different segmenting of the design.

- Display unit: The module would accept the paddle positions and the ball position and handles the interface to the display. The ball should should be green and the paddles should be white.
- Paddle-Bad collision detector: this module would monitor the paddle and ball positions and output a pulse every time the ball "collides" with a paddle.
- Top/bottom field detector: The module outputs a pulse every time the ball reaches the top or bottom of the field taking into account the size of the ball and its position. It needs access to ball position.
- Paddle units (2): these store and control the paddle position. They accept inputs representing requests to move paddle up and down. These will allow the paddles to move to the top and bottom of screen but not past.
- Rotary switch interface: this sends requests to the user paddle unit for movement.
- CPU player: every so often (1/8th of a second) this modules sends request to CPU paddle to move up or down. It needs access to ball position and the CPU paddle position. For timing, it should decrement a counter variable until it reaches zero. When it reaches zero it resets and generates a pulse to request that its paddle move up or down.
- Ball delay unit: This unit utilizes a count-down timer. Every time zero is reached, it resets to T_d and outputs a pulse. The value of T_d will determine the rate of the pulses which in turn control the ball movement rate.
- Ball unit: This unit stores the the ball position. It also keeps track of direction (left/right, up/down). It will updates the position when receiving a pulse signal from the Ball delay unit. It reverses direction based on inputs from Paddle and top/bottom collision detection units.
- End game unit..detects end of game based on ball position.
- other pieces as need...

3 What to turn in

- Compiled bit file and all source files used to generate it (YOUR COMMENTING OF CODE WILL BE GRADED)
- Create and hand in one or multiple Verilog testbench modules that test your design
- Create a report that briefly explains your design and your testing. You must have one testbench for each module.
  - Include the output of your Verilog testbench(s), with additional explanation as needed to convince someone that each part of your design works and your simulation-based testing of each module is sufficient.

4 Bonus

For bonus points, implement the following features. Do not do these independently of each other; you may only turn in one implementation of the game. You MUST document that you have implemented these so that the grader will know to check for them.

- +5% Initialize the ball with a pseudo random position and direction.
- +5% Implement ball directions and initial directions other than NW,NE,SE,SW. still only moving the ball by 1 pixel at a time in order to not complicate collision detection.
- +5% Implement more complex collision with with paddle. Collide with side of paddle and the ball reverses vertical direction instead.
- +5% Implement a paddle with a round shape. Ball should bounce off at interesting angles accordingly.
- +5% Implement smoother paddle steps so that paddle doesn't jump by 32 and instead smoothly slides.
- +5% Implement faster paddle movement if rotary switch send two or more steps quickly.
- +5% Implement ball deflection based on recent or present movement of paddle. If the paddle is moving quickly up during the collision, then the ball would take on a more northerly trajectory.