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
    $640 \times 480$ pixel field.
  - 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.
  - 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 $3\text{pixel} \times 3\text{pixel}$
    square.
  - The ball should start moving at a 45 degree angle at a rate that would allow it to cross the screen in 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 $640\times480$ 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.

![Diagram of display](image-url)
2 Design Approach

The design approach and structure is up to you. Here is one structural 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: accepts the paddle positions and the ball position and handles the interface to the display.
- Paddle collision detector: outputs a pulse every time the ball “collides” with a paddle. Needs access to ball position and paddle positions.
- Top/bottom field detector: outputs a pulse every time the ball reaches the top or bottom of the field. Needs access to ball position.
- Paddle units (2): accepts requests to move paddle up and down. Will move paddle to top and bottom of screen but not past.
- Rotary switch interface: sends requests to user paddle unit for movement.
- CPU player: every so often (1/8th of a second) sends request to CPU paddle to move up or down. Needs access to ball position and CPU paddle position. Decrements a counter variable until it reaches zero. When it reaches zero it resets and a pulse to request paddle movement up or down is generated.
- Ball delay unit: counts down to zero. Every time zero is reached, it resets to $T_d$ and outputs a pulse.
- Ball unit: keeps track of direction (left/right, up/down). Updates position when receiving pulse from Ball delay unit. Reverses direction based on inputs from Paddle and top/bottom 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
  - Include the output of your Verilog testbench(s), with additional explanation as needed to convince someone that your design works and your simulation-based testing is sufficient

4 Bonus

For bonus points, implement the following features. Do not do these independantly of eachother; 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 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
- +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 rounded 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.