You are going to extend your paddle-ball game to support multiple rounds of play using a state machine.

1 Game description

The assignment is to create a simple case-statement-based state machine to control rounds of the paddle-ball game. I’m defining what you made for the previous HW as a "round".

- The score should be displayed on the LCD display. Initially the display should be displayed as 0-0.
- You need to update the score when a point is made. You need to restart the round after each point.
- You need to modify your last HW as needed to accept control signals and output status signals to interface with your state machine (This means some of you need to finish HW4 in order to complete HW5.)
- You will need to keep score and declare a winner when the player or the computer reaches 5 points.
- You need to display a "W" in place of the numerical score of the player or CPU when there is a winner. Example W-2.
- You need to be able to restart the entire game (reset the ball and go back to a score of 0-0) when RESET is pressed.
- I will be providing the following module for you to write to the LCD display.

```verilog
module LCDDriver ( output reg [3:0] sf_d_11_8, //for LCD, see table 5–1 page 42 in user guide
                   output reg lcd_e, //for LCD, see table 5–1 page 42 in user guide
                   output reg lcd_rs, //for LCD, see table 5–1 page 42 in user guide
                   output reg lcd_rw, //for LCD, see table 5–1 page 42 in user guide
                   output reg ready, //high when module is ready for write signal
                   input wire start_write, //pulse 1 clk period to begin LCD write
                   input wire [6:0] dis_pos, //used during write strobe, fig 5–3 pg 43 in UG
                   input wire [7:0] dis_char, //used during write strobe, fig 5–4 pg 43 in UG
                   input wire rst, //must set rst for one clk cycle to init. module
                   input clk );
```

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.

- You need to create a case-statement-based state machine. You should at least create states to implement the following behaviors (some may require more than one state), though I have intentionally left the descriptions terse:
  - Initial State
  - Wait for LCDDriver (finish when LCD is ready)
  - Send CPU score 0 to LCDDriver (finish when LCD is ready again)
  - Send Player score 0 to LCDDriver (finish when LCD is ready again)
– Send - character to LCDDriver (finish when LCD is ready again)
– Start Round
– Wait for the outcome from the round (finish when a signal arrives from the round module indicating a new point)
– Update internal score and update display (wait for LCDDriver to finish)
– Finish Game (display a W)

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, the completion, and your testing
  - Be sure to 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.
- Additional clarifications to requirements may be posed on the website and should be reviewed before submitting the report.

4 Bonus
For bonus points, implement the following features. These may not be implemented independently in separate projects; 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. You may only receive extra credit for the additions in this assignment. If you received extra credit for a feature on the last assignment you do not get to reclaim the credit this time.

Note, you may change parameters of the game such as ball speed, paddle speed, CPU speed, number of points to win, etc... to make the game more playable.

The following is a list of ideas and suggested maximum points for them. The points will be assigned at the discretion of the graders. Any concern regarding this policy should be addressed by discussion and project implementation review with one of the instructors (TA or professor) prior to final submission.

- +1% Initialize the ball with a random position and direction.
- +2% Implement ball directions and initial directions other than NW,NE,SE,SW, still only moving the ball by 1 pixel at a time.
- +2% Implement more complex collision with with paddle. Collide with side of paddle and the ball reverses vertical direction instead.
- +2% Implement a paddle with a round shape. Ball should bounce off at interesting angles accordingly.
- +2% Without implementing collision detection with a rounded paddle, just make ball bounce off at interesting angles according to distance from center of paddle.
- +1% Implement smoother paddle steps so that paddle doesn’t jump by 32 and instead smoothly slides to the next position.
- +2% Implement accelerated paddle movement if rotary switch sends two or more steps quickly.
- +2% Implement ball deflection based on recent or present movement of paddle.
- +2% Implement at least 3 fixed rectangular obstacles in the field that the ball can collide with.
- +2% Implement at least 2 moving rectangular obstacles in the field that the ball can collide with.
- +2% Implement two or more balls at a time, where each ball represents a possible point.
  - +5 Implement larger, rounded balls with correct collision detection and deflections. Ball collisions should respond with proper deflection angles and preservation of momentum. (see me for details)

- I will accept proposals for other extra points.