ULTIMATE PONG

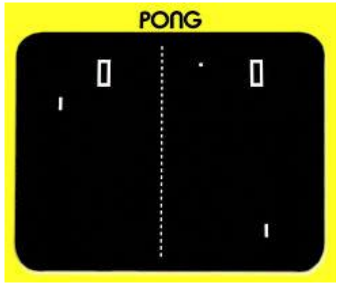
TEAM 19: PATRICIA LUIS, SUZELLE MEJIA, CHARLES MO

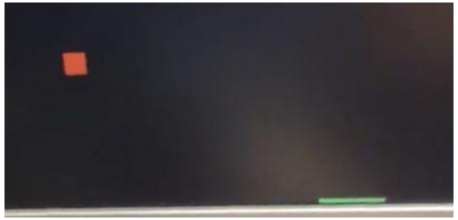
ULTIMATE PONG

TEAM 19: PATRICIA LUIS, SUZELLE MEJIA, CHARLES MO

GOAL / MOTIVATION

- ✗ To replicate the arcade game of Pong
- ✗ Single player, but stores the highest score so players can still compete with their friends
- ✗ The real life use of this is mainly entertainment





FUNCTIONALITY

- ✗ We are creating a single player pong game in which the buttons on the FPGA board slide a paddle across the bottom of the screen to catch a ball.
- ✗ As time progresses the ball gets faster to make it more challenging.
- ✗ Additionally, the current score is displayed on the 7-segment display, and the highest score is saved and also available to display on the 7-segment display.

SPECIFICATION

Requirements:

✗ Needed to use the VGA display as a screen to view the game

✗ Use buttons to move the paddle left and right and have a button that starts the game

✗ Have a collision detector that senses when the ball hits the paddle

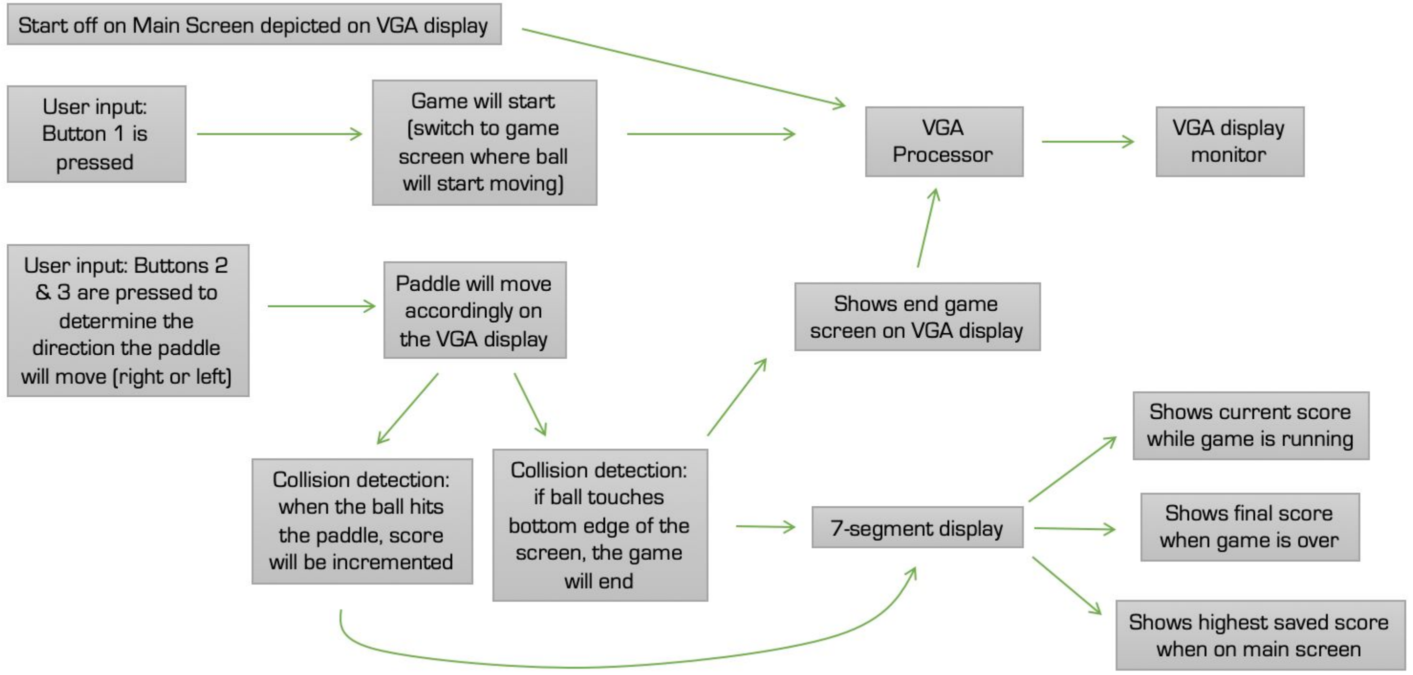
✗ Keep track of the score

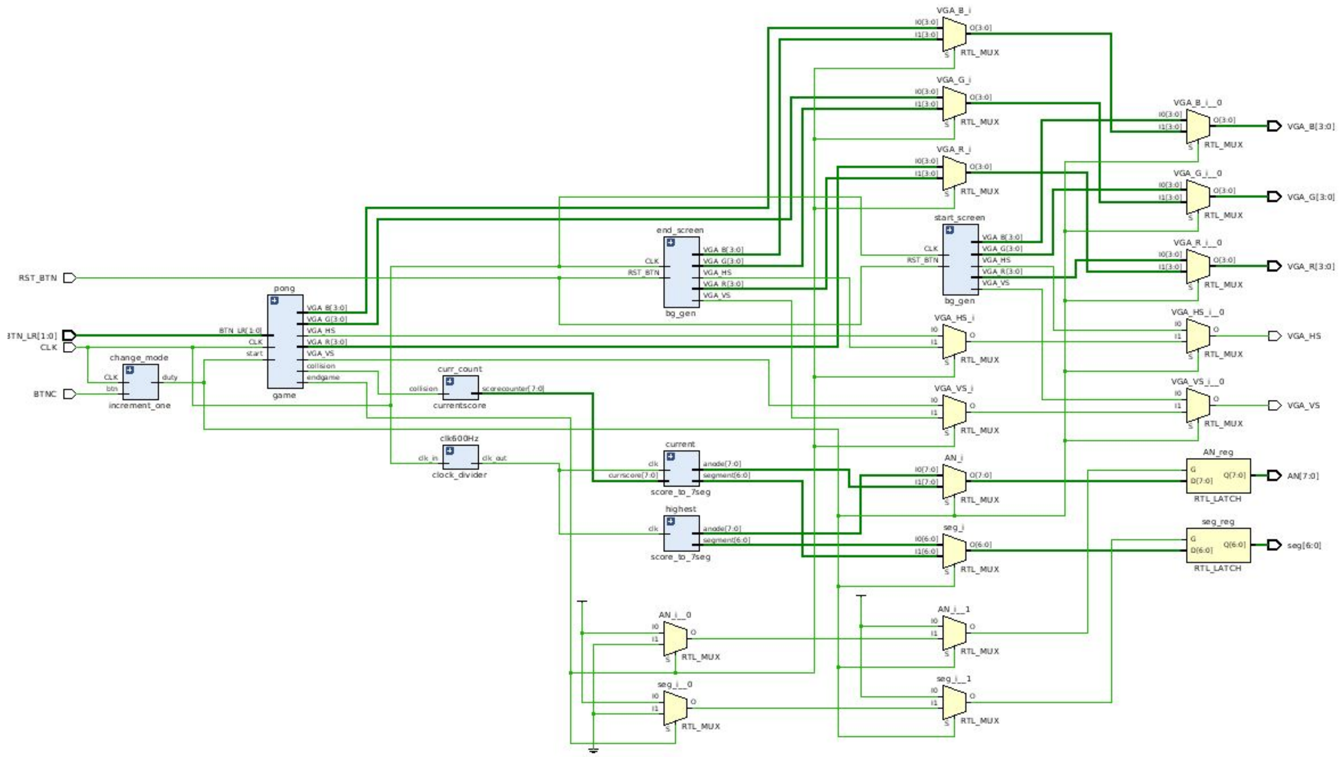
Constraints:

✗ How quickly we could make the ball go for the game to remain functional.

✗ Making the ball a circle.

BLOCK DIAGRAM

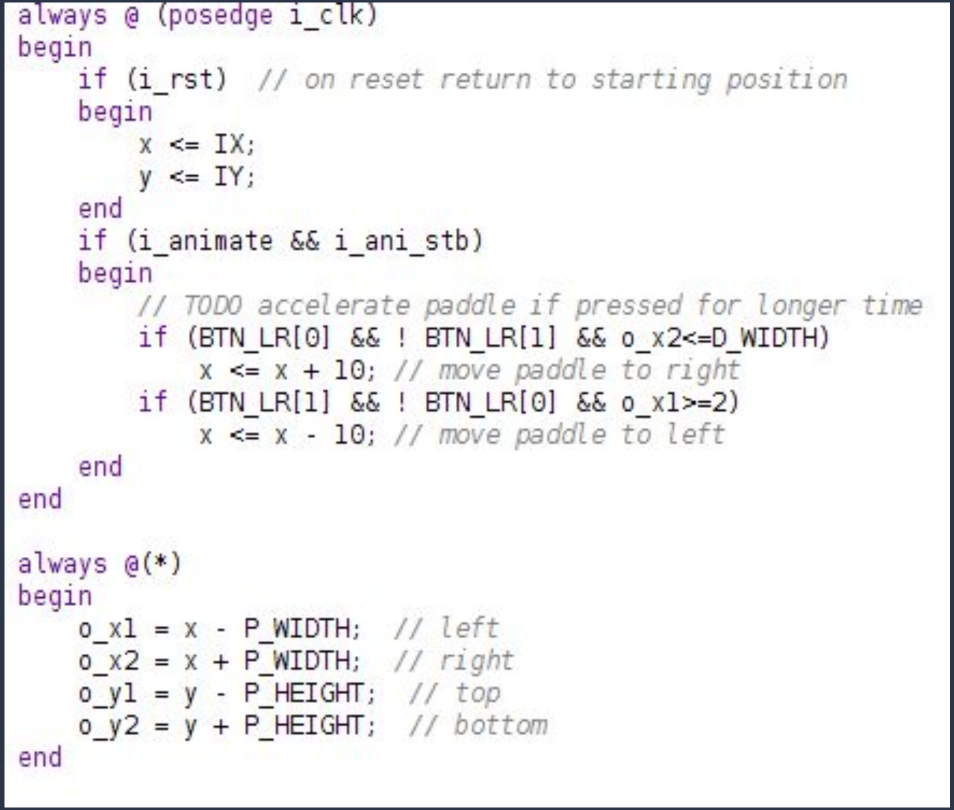

Start off on Main Screen depicted on VGA display
User input: Button 1 is pressed
Game will start (switch to game screen where ball will start moving)
VGA Processor
VGA display monitor
User input: Buttons 2 & 3 are pressed to determine the direction the paddle will move (right or left)
Paddle will move accordingly on the VGA display
Shows end game screen on VGA display
Collision detection: when the ball hits the paddle, score will be incremented
Collision detection: if ball touches bottom edge of the screen, the game will end
7-segment display
Shows current score while game is running
Shows final score when game is over
Shows highest saved score when on main screen



BLOCK DIAGRAM

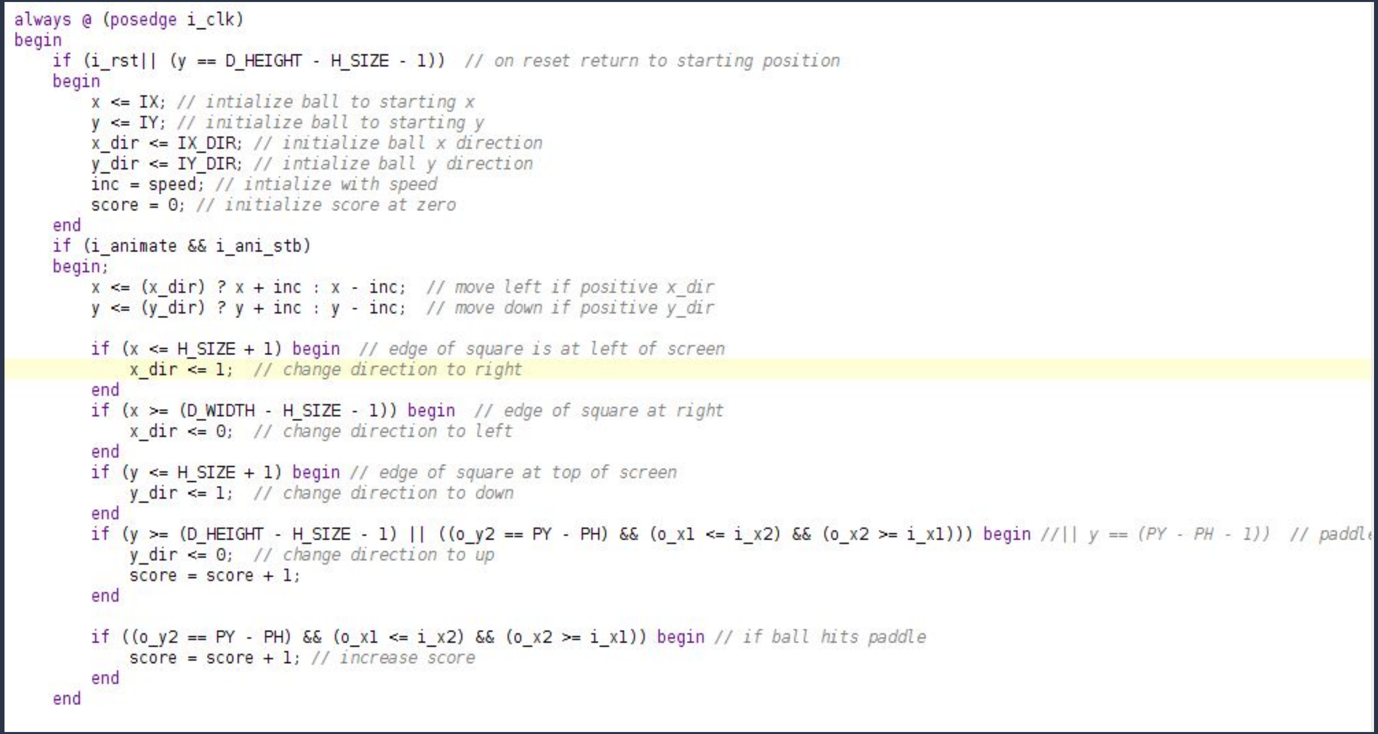
CODE SNIPPET #1 :

Buttons to move the paddle left and right



```
always @ (posedge i_clk)
begin
    if (i_rst)  // on reset return to starting position
    begin
        x <= IX;
        y <= IY;
    end
    if (i_animate && i_ani_stb)
    begin
        // TODO accelerate paddle if pressed for longer time
        if (BTN_LR[0]  && ! BTN_LR[1]  && o_x2<=D_WIDTH)
            x <= x + 10; // move paddle to right
        if (BTN_LR[1]  && ! BTN_LR[0]  && o_x1>=2)
            x <= x - 10; // move paddle to left
    end
end

always @(*)
begin
    o_x1 = x - P_WIDTH;  // left
    o_x2 = x + P_WIDTH;  // right
    o_y1 = y - P_HEIGHT;  // top
    o_y2 = y + P_HEIGHT;  // bottom
end
```

CODE SNIPPET #2 : Collision detection



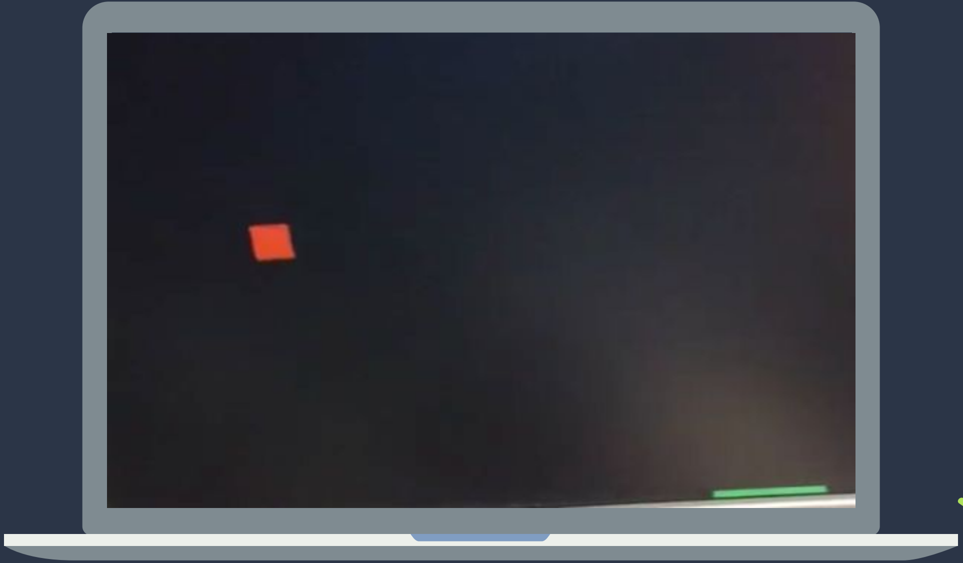
```
always @ (posedge i_clk)
begin
    if (i_rst|| (y == D_HEIGHT - H_SIZE - 1))  // on reset return to starting position
    begin
        x <= IX; // intialize ball to starting x
        y <= IY; // initialize ball to starting y
        x_dir <= IX_DIR; // initialize ball x direction
        y_dir <= IY_DIR; // intialize ball y direction
        inc = speed; // intialize with speed
        score = 0; // initialize score at zero
    end
    if (i_animate && i_ani_stb)
    begin;
        x <= (x_dir) ? x + inc : x - inc;  // move left if positive x_dir
        y <= (y_dir) ? y + inc : y - inc;  // move down if positive y_dir

        if (x <= H_SIZE + 1) begin  // edge of square is at left of screen
            x_dir <= 1;  // change direction to right
        end
        if (x >= (D_WIDTH - H_SIZE - 1)) begin  // edge of square at right
            x_dir <= 0;  // change direction to left
        end
        if (y <= H_SIZE + 1) begin // edge of square at top of screen
            y_dir <= 1;  // change direction to down
        end
        if (y >= (D_HEIGHT - H_SIZE - 1) || ((o_y2 == PY - PH) && (o_x1 <= i_x2) && (o_x2 >= i_x1))) begin //|| y == (PY - PH - 1))  // paddl
            y_dir <= 0;  // change direction to up
            score = score + 1;
        end

        if ((o_y2 == PY - PH) && (o_x1 <= i_x2) && (o_x2 >= i_x1)) begin // if ball hits paddle
            score = score + 1; // increase score
        end
    end
```

SIMULATION

ULTIMATE PONG DEMO

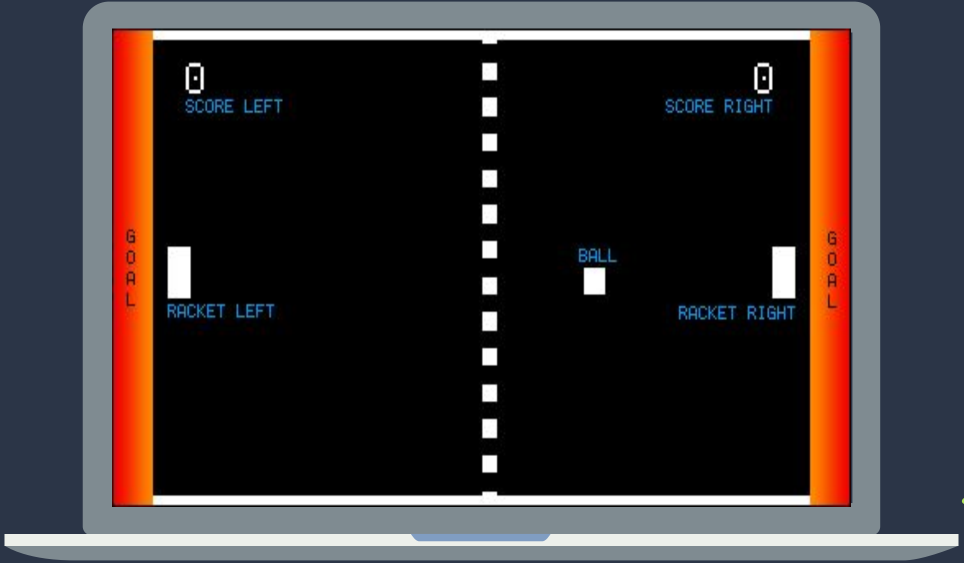
SUCCESSES

- ✗ The collision detection.
- ✗ The ball and paddle size are realistic and proportional.
- ✗ It saves the highest score and displays.
- ✗ Displays current score while playing on 7-seg.



FAILURES

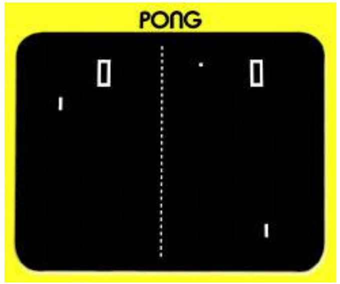
- ✗ Display the current score on the VGA as you play.
- ✗ When the speed increments it causes issues with detection.
- ✗ The ball isn't a circle, and we could've made the paddle more realistic

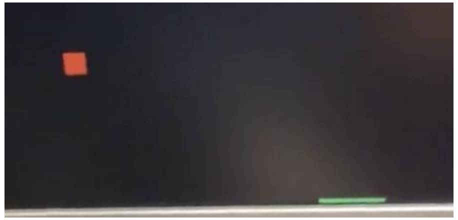


Thanks!

GOAL / MOTIVATION

- ✗ To replicate the arcade game of Pong
- ✗ Single player, but stores the highest score so players can still compete with their friends
- ✗ The real life use of this is mainly entertainment





FUNCTIONALITY

- ✗ We are creating a single player pong game in which the buttons on the FPGA board slide a paddle across the bottom of the screen to catch a ball.
- ✗ As time progresses the ball gets faster to make it more challenging.
- ✗ Additionally, the current score is displayed on the 7-segment display, and the highest score is saved and also available to display on the 7-segment display.

SPECIFICATION

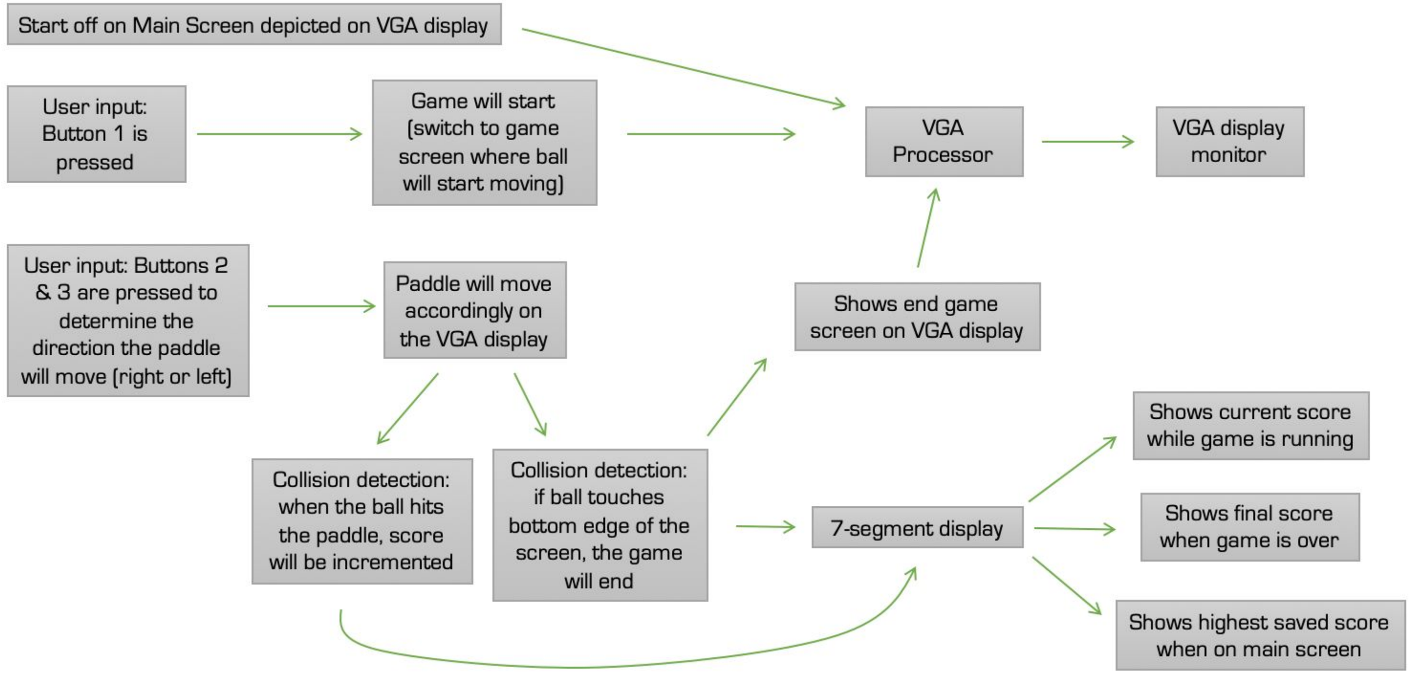
Requirements:

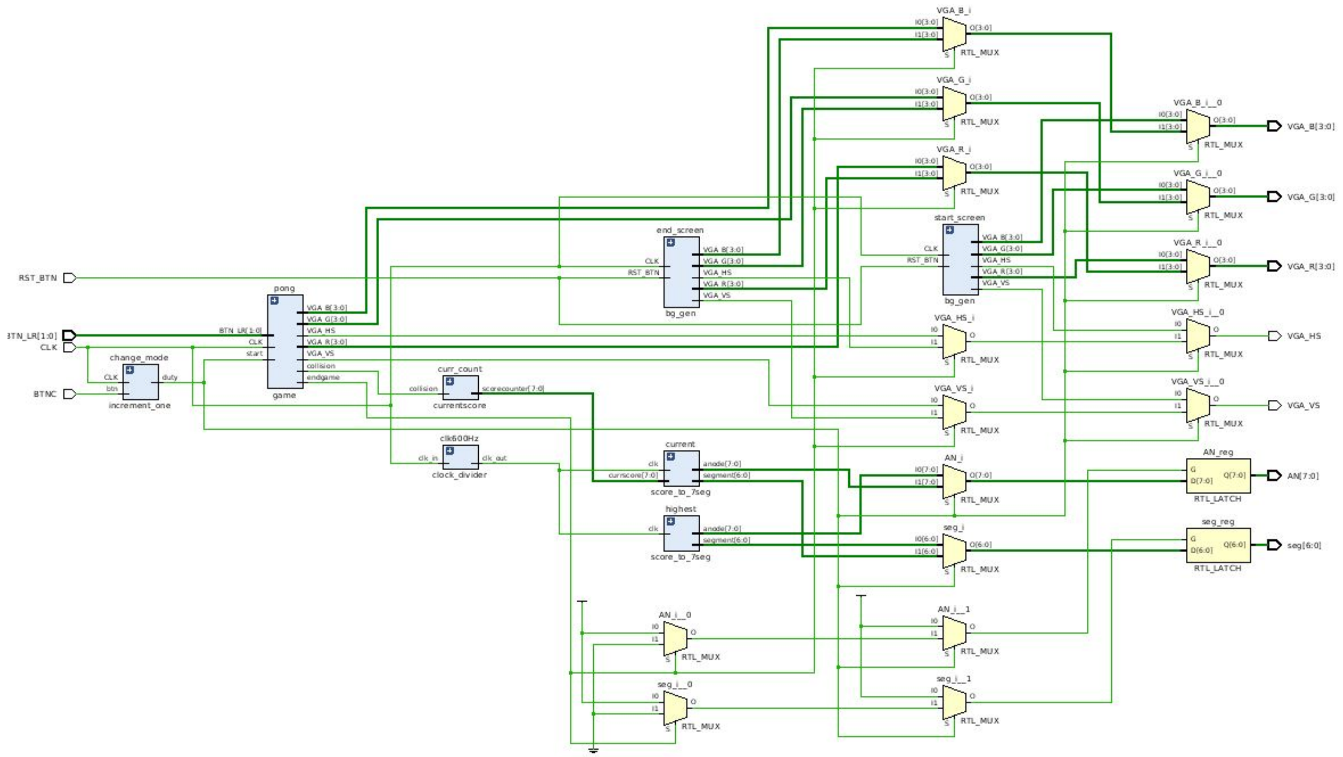
- ✗ Needed to use the VGA display as a screen to view the game
- ✗ Use buttons to move the paddle left and right and have a button that starts the game
- ✗ Have a collision detector that senses when the ball hits the paddle
- ✗ Keep track of the score

Constraints:

- ✗ How quickly we could make the ball go for the game to remain functional.
- ✗ Making the ball a circle.

BLOCK DIAGRAM

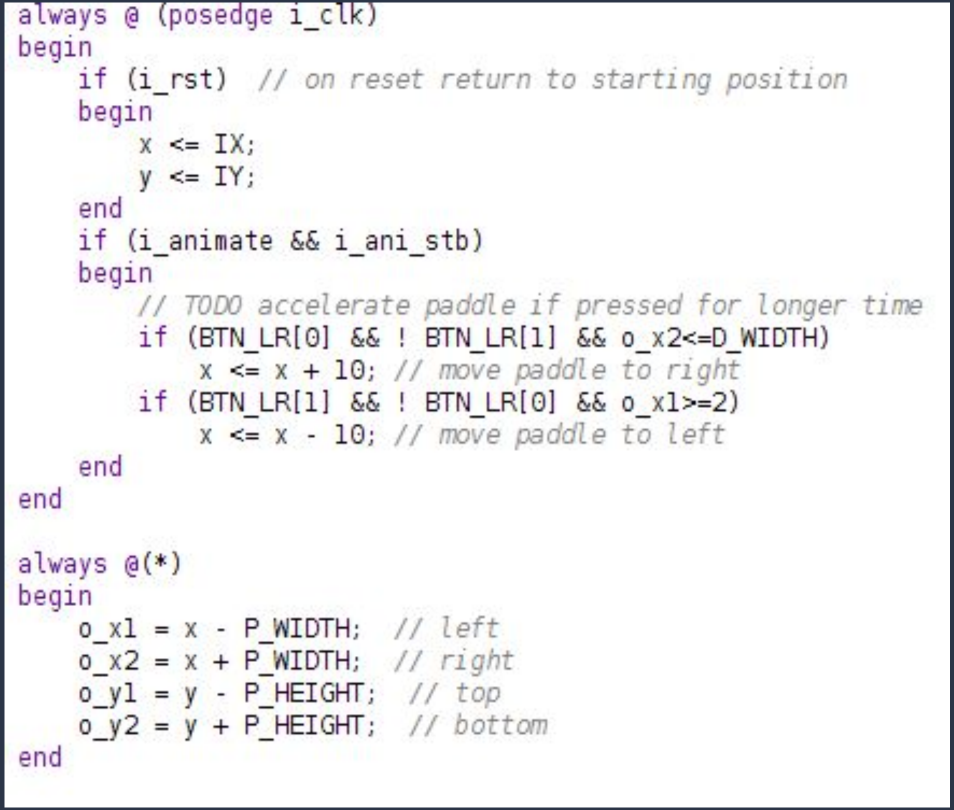

Start off on Main Screen depicted on VGA display
User input: Button 1 is pressed
Game will start (switch to game screen where ball will start moving)
VGA Processor
VGA display monitor
User input: Buttons 2 & 3 are pressed to determine the direction the paddle will move (right or left)
Paddle will move accordingly on the VGA display
Shows end game screen on VGA display
Collision detection: when the ball hits the paddle, score will be incremented
Collision detection: if ball touches bottom edge of the screen, the game will end
7-segment display
Shows current score while game is running
Shows final score when game is over
Shows highest saved score when on main screen



BLOCK DIAGRAM

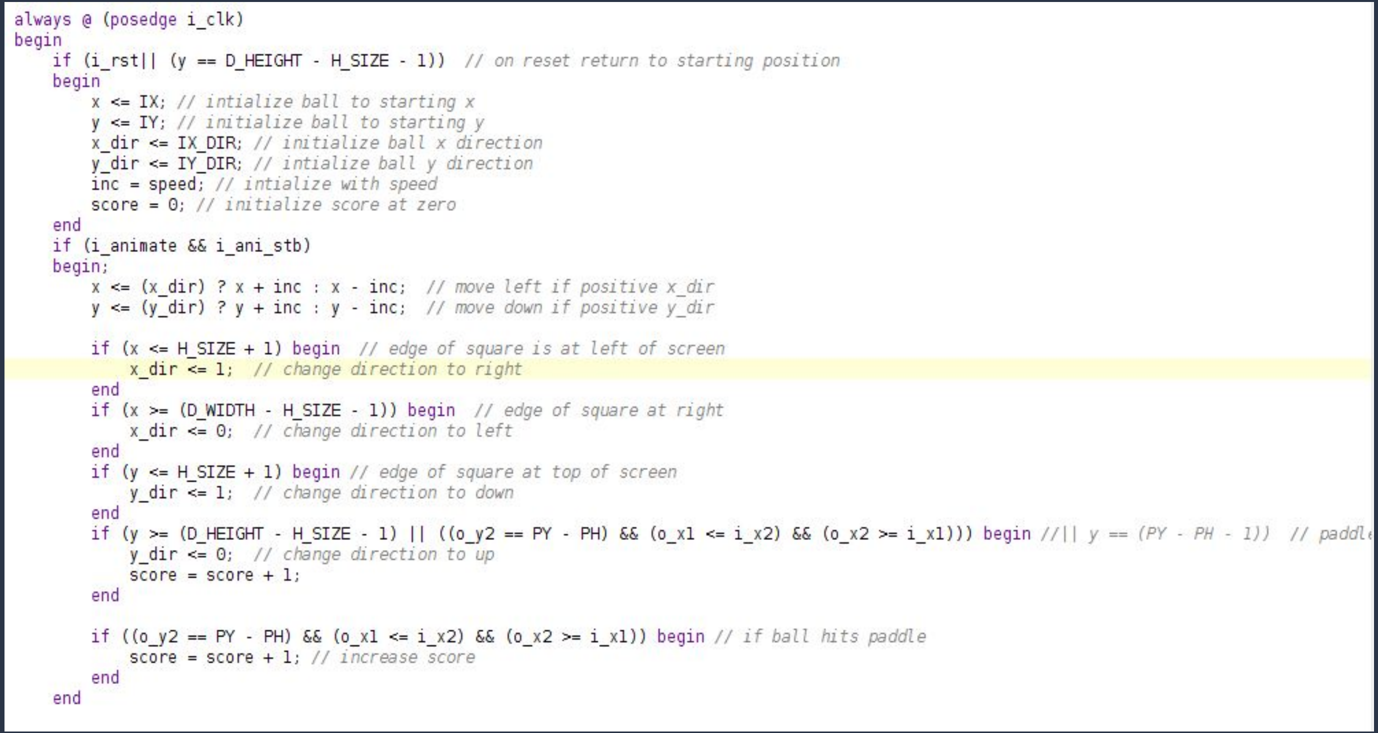
CODE SNIPPET #1 :

Buttons to move the paddle left and right



```
always @ (posedge i_clk)
begin
    if (i_rst)  // on reset return to starting position
    begin
        x <= IX;
        y <= IY;
    end
    if (i_animate && i_ani_stb)
    begin
        // TODO accelerate paddle if pressed for longer time
        if (BTN_LR[0] && ! BTN_LR[1] && o_x2<=D_WIDTH)
            x <= x + 10; // move paddle to right
        if (BTN_LR[1] && ! BTN_LR[0] && o_x1>=2)
            x <= x - 10; // move paddle to left
    end
end

always @(*)
begin
    o_x1 = x - P_WIDTH;  // left
    o_x2 = x + P_WIDTH;  // right
    o_y1 = y - P_HEIGHT;  // top
    o_y2 = y + P_HEIGHT;  // bottom
end
```

CODE SNIPPET #2 : Collision detection



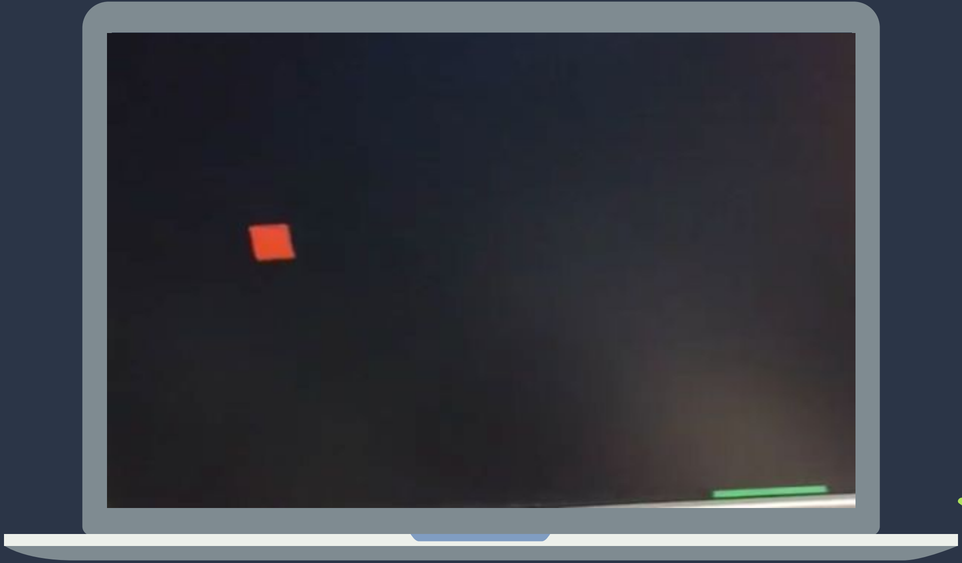
```
always @ (posedge i_clk)
begin
    if (i_rst|| (y == D_HEIGHT - H_SIZE - 1))  // on reset return to starting position
    begin
        x <= IX; // intialize ball to starting x
        y <= IY; // initialize ball to starting y
        x_dir <= IX_DIR; // initialize ball x direction
        y_dir <= IY_DIR; // intialize ball y direction
        inc = speed; // intialize with speed
        score = 0; // initialize score at zero
    end
    if (i_animate && i_ani_stb)
    begin;
        x <= (x_dir) ? x + inc : x - inc;  // move left if positive x_dir
        y <= (y_dir) ? y + inc : y - inc;  // move down if positive y_dir

        if (x <= H_SIZE + 1) begin  // edge of square is at left of screen
            x_dir <= 1;  // change direction to right
        end
        if (x >= (D_WIDTH - H_SIZE - 1)) begin  // edge of square at right
            x_dir <= 0;  // change direction to left
        end
        if (y <= H_SIZE + 1) begin // edge of square at top of screen
            y_dir <= 1;  // change direction to down
        end
        if (y >= (D_HEIGHT - H_SIZE - 1) || ((o_y2 == PY - PH) && (o_x1 <= i_x2) && (o_x2 >= i_x1))) begin //|| y == (PY - PH - 1))  // paddl
            y_dir <= 0;  // change direction to up
            score = score + 1;
        end

        if ((o_y2 == PY - PH) && (o_x1 <= i_x2) && (o_x2 >= i_x1)) begin // if ball hits paddle
            score = score + 1; // increase score
        end
    end
```

SIMULATION

ULTIMATE PONG DEMO

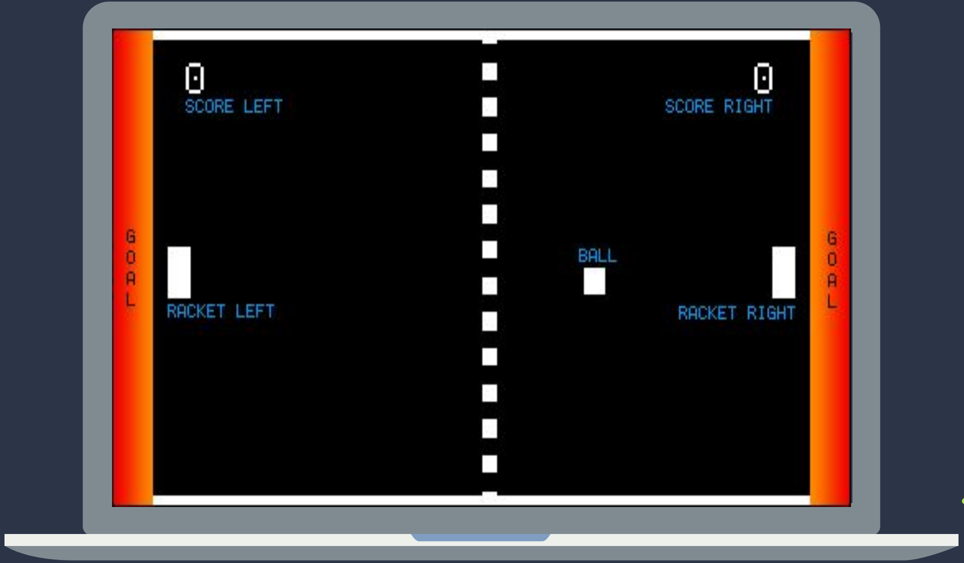
SUCCESSES

- ✗ The collision detection.
- ✗ The ball and paddle size are realistic and proportional.
- ✗ It saves the highest score and displays.
- ✗ Displays current score while playing on 7-seg.



FAILURES

- ✗ Display the current score on the VGA as you play.
- ✗ When the speed increments it causes issues with detection.
- ✗ The ball isn't a circle, and we could've made the paddle more realistic



Thanks!