```
----------------------------------------------------------------------------------
-- Engineer: Zachary Montoya
-- Submitted Date: 12-15-22
-- Module Name:    Top - Behavioral
-- Project Name:   Asteroids
-- Target Devices: Zybo Z7-10
-- Tool versions: Vivado 2020.2
--
-- Comment:
--         This file was entirely created the engineer listed above.
--
----------------------------------------------------------------------------------

-- ======================================================================================================
-- ======================================================================================================
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity asteroid is
   generic(
      asteroid_velocity: integer := 5
   );

   port(
      clk, reset: in std_logic;
      -- btn: in std_logic_vector(3 downto 0); --R2 increasing the BTN array to 3
      video_on: in std_logic;
      pixel_x, pixel_y: in std_logic_vector(9 downto 0);
      asteroid_on_out: out std_logic;
      asteroid_rgb_out: out std_logic_vector(2 downto 0)
   );
end asteroid;

-- ======================================================================================================
-- ======================================================================================================

architecture rtl of asteroid is

-- Signal used to control speed of ball and how often pushbuttons are checked for paddle movement.
   signal refr_tick: std_logic;

-- x, y coordinates (0,0 to (639, 479)
   signal pix_x, pix_y: unsigned(9 downto 0);

-- Screen dimensions
   constant MAX_X: integer := 640;
   constant MAX_Y: integer := 480;

   -- WALL1 - LEFT
   constant WALL1_X_L: integer := 0;
   constant WALL1_X_R: integer := 20;

   -- WALL2 - RIGHT
   constant WALL2_X_L: integer := 619;
   constant WALL2_X_R: integer := 639;

   -- WALL3 - BOTTOM
   constant WALL3_X_T: integer := 409;
   constant WALL3_X_B: integer := 479;

   -- WALL4 - TOP
   constant WALL4_X_T: integer := 0;
   constant WALL4_X_B: integer := 20;

-- Paddle left, right, top, bottom and height -- left & right are constant. Top & bottom are signals to allow movement. bar_y_t driven by register below.
   --constant BAR_X_L: integer := 600;--R2
   --constant BAR_X_R: integer := 603;--R2
   -- signal bar_x_L,bar_x_R: unsigned(9 downto 0); --R2
   -- signal bar_y_t, bar_y_b: unsigned(9 downto 0);
   -- constant BAR_Y_SIZE: integer := 72;
   -- constant BAR_X_SIZE: integer := 8; --R2

-- -- Reg to track top boundary (x position is fixed)
--    signal bar_y_reg, bar_y_next: unsigned( 9 downto 0);

-- -- Reg to track right boundary --R2
--    signal bar_x_reg, bar_x_next: unsigned( 9 downto 0); --R2

-- Bar moving velocity when a button is pressed -- the amount the bar is moved.
   -- constant BAR_V: integer:= 4;

-- Square ball -- ball left, right, top and bottom all vary. Left and top driven by registers below.
   constant BALL_SIZE: integer := 16; --------------------CHANGED Increased to 16
   signal ball_x_l, ball_x_r: unsigned(9 downto 0);
   signal ball_y_t, ball_y_b: unsigned(9 downto 0);

-- Reg to track left and top boundary
   signal ball_x_reg, ball_x_next: unsigned(9 downto 0);
   signal ball_y_reg, ball_y_next: unsigned(9 downto 0);

-- reg to track ball speed
   signal x_delta_reg, x_delta_next: unsigned(9 downto 0);
   signal y_delta_reg, y_delta_next: unsigned(9 downto 0);

-- ball movement can be pos or neg
   constant BALL_V_P: unsigned(9 downto 0):= to_unsigned(asteroid_velocity,10);
   constant BALL_V_N: unsigned(9 downto 0):= unsigned(to_signed(-asteroid_velocity,10));

-- round ball image
   type rom_type is array(0 to 15) of std_logic_vector(0 to 15); ----------Changed from array(0 to 7)
   constant BALL_ROM: rom_type:= (
      "0001111111110000",
      "0010110011011000",
      "0111101101110100",
      "1111011110111110",
      "1011011011011111",
      "1111101111011011",
      "1110110110111111",
      "1101111001111111",
      "1101111111011111",
      "1111011111111111",
      "1011101111111111",
      "1110110011111111",
      "0111111111111110",
      "0010111101101100",
      "0001101111111000",
      "0000111111110000");

      -- Testing
      -- 0 is to paint the background and 1 will be to use the ball
      -- Can make more than one bit to make characters and other sprites.


   -- ball is 8x8, the address only needs to be 3 bits then.
   -- data will first be read as a row
   -- rom_bit will go to the bit in the row
   signal rom_addr, rom_col: unsigned(3 downto 0); -------------------------Changed to 4 bits from unsigned (2 downto 0)
   signal rom_data: std_logic_vector(15 downto 0); -------------------------Changed from (7 downto 0)
   signal rom_bit: std_logic;

-- object output signals -- new signal to indicate if scan coord is within ball
   -- signal wall1_on, wall2_on, wall3_on, wall4_on, bar_on, sq_ball_on, rd_ball_on: std_logic;
   -- signal wall1_rgb, wall2_rgb, wall3_rgb, wall4_rgb, bar_rgb, ball_rgb: std_logic_vector(2 downto 0);
   signal wall1_on, wall2_on, wall3_on, wall4_on, sq_ball_on, rd_ball_on: std_logic;
   signal wall1_rgb, wall2_rgb, wall3_rgb, wall4_rgb, ball_rgb: std_logic_vector(2 downto 0);

-- =====================================================================================
=================
   begin

   asteroid_on_out <= rd_ball_on;
   asteroid_rgb_out <= ball_rgb;

   process (clk, reset)
      begin
      if (reset = '1') then
         -- bar_y_reg <= (others => '0');
         -- bar_x_reg <= (others => '0');--R2
         ball_x_reg <= (others => '0');
         ball_y_reg <= (others => '0');
         x_delta_reg <= ("0000000100");
         y_delta_reg <= ("0000000100");
      elsif (clk'event and clk = '1') then
         -- bar_y_reg <= bar_y_next;
         -- bar_x_reg <= bar_x_next;--R2
         ball_x_reg <= ball_x_next;
         ball_y_reg <= ball_y_next;
         x_delta_reg <= x_delta_next;
         y_delta_reg <= y_delta_next;
      end if;
   end process;

--
=====================================================================================
=================
   pix_x <= unsigned(pixel_x);
   pix_y <= unsigned(pixel_y);

-- Refr_tick: 1-clock tick asserted at start of v_sync, e.g., when the screen is
refreshed -- speed is 60 Hz
   refr_tick <= '1' when (pix_y = 1) and (pix_x = 1) else '0';

--
=====================================================================================
=================
-- -- wall1 left vertical stripe
--    wall1_on <= '1' when (WALL1_X_L <= pix_x) and (pix_x <= WALL1_X_R) else '0'; --
convert pix_x to pix_y to make horizontal
--    wall1_rgb <= "011"; -- paddle colors blue
-- -- wall2 right vertical stripe
--    wall2_on <= '1' when (WALL2_X_L <= pix_x) and (pix_x <= WALL2_X_R) else '0';
--    wall2_rgb <= "011"; -- paddle colors blue
-- -- wall3 left vertical stripe
--    wall3_on <= '1' when (WALL3_X_T <= pix_y) and (pix_y <= WALL3_X_B) else '0'; --
convert pix_x to pix_y to make horizontal
--    wall3_rgb <= "011"; -- paddle colors blue
-- -- wall4 right vertical stripe
--    wall4_on <= '1' when (WALL4_X_T <= pix_y) and (pix_y <= WALL4_X_B) else '0';
--    wall4_rgb <= "011"; -- paddle colors blue

--
=====================================================================================
=================
-- -- pixel within paddle
--    bar_x_L <= bar_x_reg;--R2
--    bar_x_R <= bar_x_L + BAR_X_SIZE - 1;--R2

--    bar_y_t <= bar_y_reg;
--    bar_y_b <= bar_y_t + BAR_Y_SIZE - 1;
--    bar_on <= '1' when (BAR_X_L <= pix_x) and (pix_x <= BAR_X_R) and (bar_y_t <=
pix_y) and (pix_y <= bar_y_b) else '0';
--    bar_rgb <= "100"; -- Red color

--
=====================================================================================
=================
-- set coordinates of square ball.
   ball_x_l <= ball_x_reg;
   ball_y_t <= ball_y_reg;
   ball_x_r <= ball_x_l + BALL_SIZE - 1;
   ball_y_b <= ball_y_t + BALL_SIZE - 1;

-- pixel within square ball
   sq_ball_on <= '1' when (ball_x_l <= pix_x) and (pix_x <= ball_x_r) and (ball_y_t
<= pix_y) and (pix_y <= ball_y_b) else '0';

```

```
-- Map scan coord to ROM addr/col -- use low order three bits of pixel and ball positions. ROM row
   rom_addr <= pix_y(3 downto 0) - ball_y_t(3 downto 0); -------------------- CHANGED TO 4 BITS

-- ROM column
   rom_col <= pix_x(3 downto 0) - ball_x_l(3 downto 0);-------------------- CHANGED TO 4 BITS

-- Get row data
   rom_data <= BALL_ROM(to_integer(rom_addr));

-- Get column bit
   rom_bit <= rom_data(to_integer(rom_col));

-- Turn ball on only if within square and the ROM bit is 1.
   rd_ball_on <= '1' when (sq_ball_on = '1') and (rom_bit = '1') else '0';
   ball_rgb <= "111"; -- WHITE BALL COLOR

-- Update the ball position 60 times per second.
   ball_x_next <= ball_x_reg + x_delta_reg when refr_tick = '1' else ball_x_reg;
   ball_y_next <= ball_y_reg + y_delta_reg when refr_tick = '1' else ball_y_reg;

-- Set the value of the next ball position according to the boundaries.
   -- process(x_delta_reg, y_delta_reg, ball_y_t, ball_x_l, ball_x_r, ball_y_t, ball_y_b, bar_y_t, bar_y_b)
   process(x_delta_reg, y_delta_reg, ball_y_t, ball_x_l, ball_x_r, ball_y_t, ball_y_b)
      begin
      x_delta_next <= x_delta_reg;
      y_delta_next <= y_delta_reg;

-- Ball reached top, make offset positive
      if ( ball_y_t < WALL4_X_B ) then --MAKE WALL4
         y_delta_next <= BALL_V_P;

-- Reached bottom, make negative
      elsif (ball_y_b > (WALL3_X_T - 1)) then --MAKE WALL3
         y_delta_next <= BALL_V_N;

-- Reach wall1, bounce back
      elsif (ball_x_l <= WALL1_X_R ) then
         x_delta_next <= BALL_V_P;
-- Reach wall2, bounce back
      elsif (ball_x_r > WALL2_X_L) then
         x_delta_next <= BALL_V_N;

-- -- Right corner of ball inside bar
--       elsif ((BAR_X_L <= ball_x_r) and (ball_x_r <= BAR_X_R)) then

-- -- Some portion of ball hitting paddle, reverse direction
--          if ((bar_y_t <= ball_y_b) and (ball_y_t <= bar_y_b)) then
--             x_delta_next <= BALL_V_N;
--          end if;
      end if;
   end process;

--======================================================================================================


end rtl;


```