```
----------------------------------------------------------------------------------
-- 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.
--         EXCEPT FOR PARTS OF LINES 236 and 241 SPECIFICALLY THE COMPARISON
ORIENTATION
--         THESE PARTS WERE CREATED BY THE FOLLOWING
--         Engineer: Anindya Bal
--
----------------------------------------------------------------------------------

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

entity space_ship is
   port(
      clk, reset: in std_logic;
      btn: in std_logic_vector(3 downto 0); --R2 increasing the BTN array to 3
      sw: in std_logic_vector(0 downto 0);
      video_on: in std_logic;
      pixel_x, pixel_y: in std_logic_vector(9 downto 0);
      bar_on_out: out std_logic;
      bar_rgb_out: out std_logic_vector(2 downto 0);
      bar_x_reg_out, bar_y_reg_out: out unsigned( 9 downto 0);
      rom_selector_out: out std_logic_vector(2 downto 0);
      graph_rgb: out std_logic_vector(2 downto 0);
      fire_out,fire_btn_signal_out: out std_logic;
      fire_ready_out: in std_logic

   );
end space_ship;

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

architecture rtl of space_ship 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;
-- Bar moving velocity when a button is pressed -- the amount the bar is moved.

--
================================================================================
================
-- BAR STUFF
   -- 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.
   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_SIZE: integer := 16; --R2
   -- Reg to track top boundary (x position is fixed)
   -- signal bar_y_reg: unsigned( 9 downto 0) := "1111001110";
   -- signal bar_x_reg: unsigned( 9 downto 0) := "1111001110";
   -- signal bar_x_next, bar_y_next: unsigned( 9 downto 0);
   signal bar_x_reg, bar_y_reg, bar_x_next, bar_y_next: unsigned( 9 downto 0);
   -- signal bar_x_reg_out, bar_y_reg_out: unsigned( 9 downto 0);

-- ball movement can be pos or neg
   constant BAR_V: integer:= 3;

-- round TOP image
   type spaceship_rom_type is array(0 to 15) of std_logic_vector(0 to 15); ----------
Changed from array(0 to 7)
   constant SPACESHIP_UP_ROM: spaceship_rom_type:= (
         "0000000110000000",
         "0000000110000000",
         "0000000110000000",
         "0000001111000000",
         "0000001111000000",
         "1100001111000011",
         "1100001111000011",
         "1100011001100011",
         "1100011001100011",
         "1111111111111111",
         "1111111111111111",
         "1100011001100011",
         "1100011001100011",
         "1100001001000011",
         "0001110110111000",
         "0001011001101000");

   -- round BOTTOM image
   constant SPACESHIP_DOWN_ROM: spaceship_rom_type:= (
         "0001011001101000",
         "0001110110111000",
         "1100001001000011",
         "1100011001100011",
         "1100011001100011",
         "1111111111111111",
         "1111111111111111",
         "1100011001100011",
         "1100011001100011",
         "1100001111000011",
         "1100001111000011",
         "0000001111000000",
         "0000001111000000",
         "0000000110000000",
         "0000000110000000",
         "0000000110000000");

   constant SPACESHIP_RIGHT_ROM: spaceship_rom_type:= (
         "0011111111100000",
         "0011111111100000",
         "0000011000000000",
         "1100011000000000",
         "0100011000000000",
         "1101111110000000",
         "1011111111111000",
         "0100011001111111",
         "0100011001111111",
         "1011111111111000",
         "1101111110000000",
         "0100011000000000",
         "1100011000000000",
         "0000011000000000",
         "0011111111100000",
```

```
         "0011111111100000");

         constant SPACESHIP_LEFT_ROM: spaceship_rom_type:= (
         "0000011111111100",
         "0000011111111100",
         "0000000001100000",
         "0000000001100011",
         "0000000001100010",
         "0000000001111011",
         "0001111111111101",
         "1111111001100010",
         "1111111001100010",
         "0001111111111101",
         "0000000001111011",
         "0000000001100010",
         "0000000001100011",
         "0000000001100000",
         "0000011111111100",
         "0000011111111100");

   signal spaceship_rom_addr, spaceship_rom_col: unsigned(3 downto 0); ----------
----------Changed to 4 bits from unsigned (2 downto 0)
   signal spaceship_rom_data: std_logic_vector(15 downto 0); ----------------------
-Changed from (7 downto 0)
   signal spaceship_rom_bit: std_logic;
-- --
================================================================================
=================

   signal bar_on, rom_bar_on: std_logic;
   signal bar_rgb: std_logic_vector(2 downto 0);
   signal rom_selector: std_logic_vector(2 downto 0);
   signal fire_next, fire_reg: std_logic;
   signal fire_btn_signal_reg: std_logic := '0';
   signal fire_btn_signal_next: std_logic;
   -- signal rom_selector_out:std_logic_vector(1 downto 0);
--
================================================================================
=================
   begin
   bar_on_out <= rom_bar_on;
   bar_rgb_out <= bar_rgb;
   bar_x_reg_out <= bar_x_reg;
   bar_y_reg_out <= bar_y_reg;
   rom_selector_out <= rom_selector;
   fire_out <= fire_next;
   fire_btn_signal_out <= fire_btn_signal_reg;

   process (clk, reset)
      begin
      if (reset = '1') then
         -- bar_y_reg <= (others => '0');
         -- bar_x_reg <= (others => '0');--R2
         bar_y_reg <= (to_unsigned(240,10));
         bar_x_reg <= (to_unsigned(320,10));--R2
         fire_reg <= '0';
         fire_btn_signal_reg <= '0';
      elsif (clk'event and clk = '1') then
         bar_y_reg <= bar_y_next;
         bar_x_reg <= bar_x_next;--R2
         fire_reg <= fire_next;
         fire_btn_signal_reg <= fire_btn_signal_next;
      end if;
   end process;
--
================================================================================
=================
   --FIRE CONTROL
   process (btn, fire_reg , fire_ready_out, fire_btn_signal_reg)
   begin
      fire_next <= fire_reg;
      fire_btn_signal_next <= fire_btn_signal_reg;
      if (btn(0) = '1' and fire_btn_signal_reg = '0') then
      -- if (btn(0) = '1' ) then
         fire_next <= '1';
         fire_btn_signal_next <= '1';
      elsif (fire_ready_out = '1') then
         fire_btn_signal_next <= '0';
         fire_next <= '0';
```

```
215
216       end if;
217    end process;
218
219
220 -- Process bar movement requests
221    process( bar_y_reg, bar_y_b, bar_y_t,bar_x_reg,bar_x_L,bar_x_R, refr_tick, btn,
rom_selector)
222    begin
223    bar_y_next <= bar_y_reg; -- no move
224    bar_x_next <= bar_x_reg; -- no move--R2
225
226    if ( refr_tick = '1' ) then
227       --SW0 DOWN and BTN2 1 --FIRE DOWN NOT AT EDGE
228       if ( btn(2) = '1' and sw(0) = '0' and bar_y_b < (WALL3_X_T - 1 - BAR_V)) then -
-CHANGE TO WALL PARAMETERS
229          bar_y_next <= bar_y_reg + BAR_V; -- move down
230          rom_selector <= "000";
231
232       --SW0 UP and BTN2 1 --FIRE UP NOT AT EDGE
233       elsif ( btn(2) = '1' and sw(0) = '1' and  bar_y_t > (WALL4_X_B - 1 - BAR_V))
then
234          bar_y_next <= bar_y_reg - BAR_V; -- move up
235          rom_selector <= "110";
236
237       -- if btn 0 pressed and bar not at RIGHT yet --R2
238       elsif (btn(1) = '1' and bar_x_R < (WALL2_X_L - 1 - BAR_V)) then
239          bar_x_next <= bar_x_reg + BAR_V; -- move RIGHT--R2
240          rom_selector <= "100";
241
242       -- if btn 0 NOT pressed and bar not at  yet
243       elsif (btn(3) = '1' and bar_x_L > (WALL1_X_R - 1 - BAR_V)) then
244          bar_x_next <= bar_x_reg - BAR_V;--move LEFT
245          rom_selector <= "010";
246
247       end if;
248    end if;
249    end process;
250
251    with rom_selector select
252       spaceship_rom_data <= SPACESHIP_UP_ROM(to_integer(spaceship_rom_addr)) when
"110",
253                             SPACESHIP_DOWN_ROM(to_integer(spaceship_rom_addr)) when
"000",
254                             SPACESHIP_LEFT_ROM(to_integer(spaceship_rom_addr)) when
"100",
255                             SPACESHIP_RIGHT_ROM(to_integer(spaceship_rom_addr)) when
others;
256
257    spaceship_rom_bit <= spaceship_rom_data(to_integer(spaceship_rom_col));-- Get
column bit
258 --
================================================================================
================
259    pix_x <= unsigned(pixel_x);
260    pix_y <= unsigned(pixel_y);
261
262 -- Refr_tick: 1-clock tick asserted at start of v_sync, e.g., when the screen is
refreshed -- speed is 60 Hz
263    refr_tick <= '1' when (pix_y = 1) and (pix_x = 1) else '0';
264
265 --
================================================================================
================
266 -- pixel within paddle
267
268    bar_x_L <= bar_x_reg;--R2
269    bar_y_t <= bar_y_reg;
270    bar_x_R <= bar_x_L + BAR_SIZE - 1;--R2
271    bar_y_b <= bar_y_t + BAR_SIZE - 1;
272    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';
273    bar_rgb <= "100"; -- Red color
274    -- Map scan coord to ROM addr/col -- use low order three bits of pixel and ball
positions. ROM row
275       spaceship_rom_addr <= pix_y(3 downto 0) - bar_y_t(3 downto 0); ---------------
---- CHANGED TO 4 BITS
276    -- ROM column
```

```
277       spaceship_rom_col <= pix_x(3 downto 0) - bar_x_L(3 downto 0);------------------
  -- CHANGED TO 4 BITS
278    -- Get row data
279    --    spaceship_rom_data <= SPACESHIP_UP_ROM(to_integer(spaceship_rom_addr));
280    -- -- Get column bit
281    --    spaceship_rom_bit <= spaceship_rom_data(to_integer(spaceship_rom_col));
282    -- Turn ball on only if within square and the ROM bit is 1.
283       rom_bar_on <= '1' when (bar_on = '1') and (spaceship_rom_bit = '1') else '0';
284       bar_rgb <= "111"; -- WHITE BALL COLOR
285
286
287
288
289 end rtl;
```