Lab 6

Adrian Torres and Angel Ordonez Retamar

Part 2

Modified Output:
https://youtu.be/_ZCDB9b1rqM

Explanation of Code Changes

“Bat_n_ball.vhd”:

In the port section, a *hit_counter signal* was added to count the number of hits the ball makes with the bat, along with a *SW signal* for reading the positions of the switches. The architecture section was updated to include a wider bat by doubling its width. The fixed initial value for the *ball_speed* vector was removed to allow dynamic adjustments later in the code. Additionally, a *S_hit_counter signal* was introduced to track the hit count. The *ball_speed* was updated to change with the switches that are flipped up. When the player loses, the *ball_hit signal* is reset to zero, and the *ball_hit port* is synchronized with the *ball_hit signal*. The code also ensures that the *ball_hit signal* increments and the *bat_w signal* decreases each time the ball hits the bat. Lastly, the colors for the bat and ball were changed to see them better.

“Pong.vhd”:

A *SW signal* was added to manage input from the switches. The architecture section includes the *S_hit_counter signal* for tracking hits. The bat and ball ports were updated with *hit_count* and *SW* vectors. The *SW signal* was mapped to the *SW port* so that they have the same values. The hit_count port was mapped to the display signal so the number of hits can be seen.

“pong.xdc”:

We defined the first five switches from the right in the constraints file to incorporate them into the project.

Below we have all the original and changed codes with the specific changes highlighted yellow

Original code “bat_n_ball.vhd”:

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_ARITH.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;

ENTITY bat_n_ball IS
   PORT (
      v_sync : IN STD_LOGIC;
      pixel_row : IN STD_LOGIC_VECTOR(10 DOWNTO 0);
      pixel_col : IN STD_LOGIC_VECTOR(10 DOWNTO 0);
      bat_x : IN STD_LOGIC_VECTOR (10 DOWNTO 0); -- current bat x position
      serve : IN STD_LOGIC; -- initiates serve
      red : OUT STD_LOGIC;
```

```
      green : OUT STD_LOGIC;
      blue : OUT STD_LOGIC
   );
END bat_n_ball;

ARCHITECTURE Behavioral OF bat_n_ball IS
   CONSTANT bsize : INTEGER := 8; -- ball size in pixels
   CONSTANT bat_w : INTEGER := 20; -- bat width in pixels
   CONSTANT bat_h : INTEGER := 3; -- bat height in pixels
   -- distance ball moves each frame
   CONSTANT ball_speed : STD_LOGIC_VECTOR (10 DOWNTO 0) :=
CONV_STD_LOGIC_VECTOR (6, 11);
   SIGNAL ball_on : STD_LOGIC; -- indicates whether ball is at current pixel position
   SIGNAL bat_on : STD_LOGIC; -- indicates whether bat at over current pixel position
   SIGNAL game_on : STD_LOGIC := '0'; -- indicates whether ball is in play
   -- current ball position - intitialized to center of screen
   SIGNAL ball_x : STD_LOGIC_VECTOR(10 DOWNTO 0) :=
CONV_STD_LOGIC_VECTOR(400, 11);
   SIGNAL ball_y : STD_LOGIC_VECTOR(10 DOWNTO 0) :=
CONV_STD_LOGIC_VECTOR(300, 11);
   -- bat vertical position
   CONSTANT bat_y : STD_LOGIC_VECTOR(10 DOWNTO 0) :=
CONV_STD_LOGIC_VECTOR(500, 11);
   -- current ball motion - initialized to (+ ball_speed) pixels/frame in both X and Y directions
   SIGNAL ball_x_motion, ball_y_motion : STD_LOGIC_VECTOR(10 DOWNTO 0) :=
ball_speed;
BEGIN
   red <= NOT bat_on; -- color setup for red ball and cyan bat on white background
   green <= NOT ball_on;
   blue <= NOT ball_on;
   -- process to draw round ball
   -- set ball_on if current pixel address is covered by ball position
   balldraw : PROCESS (ball_x, ball_y, pixel_row, pixel_col) IS
      VARIABLE vx, vy : STD_LOGIC_VECTOR (10 DOWNTO 0); -- 9 downto 0
   BEGIN
      IF pixel_col <= ball_x THEN -- vx = |ball_x - pixel_col|
         vx := ball_x - pixel_col;
      ELSE
         vx := pixel_col - ball_x;
      END IF;
      IF pixel_row <= ball_y THEN -- vy = |ball_y - pixel_row|
         vy := ball_y - pixel_row;
      ELSE
         vy := pixel_row - ball_y;
      END IF;
      IF ((vx * vx) + (vy * vy)) < (bsize * bsize) THEN -- test if radial distance < bsize
         ball_on <= game_on;
      ELSE
         ball_on <= '0';
      END IF;
   END PROCESS;
   -- process to draw bat
   -- set bat_on if current pixel address is covered by bat position
```

```
batdraw : PROCESS (bat_x, pixel_row, pixel_col) IS
   VARIABLE vx, vy : STD_LOGIC_VECTOR (10 DOWNTO 0); -- 9 downto 0
BEGIN
   IF ((pixel_col >= bat_x - bat_w) OR (bat_x <= bat_w)) AND
    pixel_col <= bat_x + bat_w AND
       pixel_row >= bat_y - bat_h AND
       pixel_row <= bat_y + bat_h THEN
          bat_on <= '1';
   ELSE
      bat_on <= '0';
   END IF;
END PROCESS;
-- process to move ball once every frame (i.e., once every vsync pulse)
mball : PROCESS
   VARIABLE temp : STD_LOGIC_VECTOR (11 DOWNTO 0);
BEGIN
   WAIT UNTIL rising_edge(v_sync);
   IF serve = '1' AND game_on = '0' THEN -- test for new serve
      game_on <= '1';
      ball_y_motion <= (NOT ball_speed) + 1; -- set vspeed to (- ball_speed) pixels
   ELSIF ball_y <= bsize THEN -- bounce off top wall
      ball_y_motion <= ball_speed; -- set vspeed to (+ ball_speed) pixels
   ELSIF ball_y + bsize >= 600 THEN -- if ball meets bottom wall
      ball_y_motion <= (NOT ball_speed) + 1; -- set vspeed to (- ball_speed) pixels
      game_on <= '0'; -- and make ball disappear
   END IF;
   -- allow for bounce off left or right of screen
   IF ball_x + bsize >= 800 THEN -- bounce off right wall
      ball_x_motion <= (NOT ball_speed) + 1; -- set hspeed to (- ball_speed) pixels
   ELSIF ball_x <= bsize THEN -- bounce off left wall
      ball_x_motion <= ball_speed; -- set hspeed to (+ ball_speed) pixels
   END IF;
   -- allow for bounce off bat
   IF (ball_x + bsize/2) >= (bat_x - bat_w) AND
    (ball_x - bsize/2) <= (bat_x + bat_w) AND
       (ball_y + bsize/2) >= (bat_y - bat_h) AND
       (ball_y - bsize/2) <= (bat_y + bat_h) THEN
          ball_y_motion <= (NOT ball_speed) + 1; -- set vspeed to (- ball_speed) pixels
   END IF;
   -- compute next ball vertical position
   -- variable temp adds one more bit to calculation to fix unsigned underflow problems
   -- when ball_y is close to zero and ball_y_motion is negative
   temp := ('0' & ball_y) + (ball_y_motion(10) & ball_y_motion);
   IF game_on = '0' THEN
      ball_y <= CONV_STD_LOGIC_VECTOR(440, 11);
   ELSIF temp(11) = '1' THEN
      ball_y <= (OTHERS => '0');
   ELSE ball_y <= temp(10 DOWNTO 0); -- 9 downto 0
   END IF;
   -- compute next ball horizontal position
   -- variable temp adds one more bit to calculation to fix unsigned underflow problems
   -- when ball_x is close to zero and ball_x_motion is negative
   temp := ('0' & ball_x) + (ball_x_motion(10) & ball_x_motion);
```

```
        IF temp(11) = '1' THEN
            ball_x <= (OTHERS => '0');
        ELSE ball_x <= temp(10 DOWNTO 0);
        END IF;
    END PROCESS;
END Behavioral;
```

Modified code “bat_n_ball.vhd”:

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_ARITH.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;

ENTITY bat_n_ball IS
    PORT (
        v_sync : IN STD_LOGIC;
        pixel_row : IN STD_LOGIC_VECTOR(10 DOWNTO 0);
        pixel_col : IN STD_LOGIC_VECTOR(10 DOWNTO 0);
        bat_x : IN STD_LOGIC_VECTOR (10 DOWNTO 0); -- current bat x position
        serve : IN STD_LOGIC; -- initiates serve
        red : OUT STD_LOGIC;
        green : OUT STD_LOGIC;
        blue : OUT STD_LOGIC;
        hit_count : OUT STD_LOGIC_VECTOR(15 DOWNTO 0);
        SW : IN STD_LOGIC_VECTOR(4 DOWNTO 0)
    );
END bat_n_ball;

ARCHITECTURE Behavioral OF bat_n_ball IS
    CONSTANT bsize : INTEGER := 8; -- ball size in pixels
    SIGNAL bat_w : INTEGER := 40; -- bat width in pixels (adjustable)
    CONSTANT bat_w_start : INTEGER := 40; -- starting bat width
    CONSTANT bat_h : INTEGER := 4; -- bat height in pixels
    -- distance ball moves each frame
    SIGNAL ball_speed : STD_LOGIC_VECTOR (10 DOWNTO 0);
    SIGNAL ball_on : STD_LOGIC; -- indicates whether ball is at current pixel position
    SIGNAL bat_on : STD_LOGIC; -- indicates whether bat at over current pixel position
    SIGNAL game_on : STD_LOGIC := '0'; -- indicates whether ball is in play
    SIGNAL ball_x : STD_LOGIC_VECTOR(10 DOWNTO 0) :=
CONV_STD_LOGIC_VECTOR(400, 11);
    SIGNAL ball_y : STD_LOGIC_VECTOR(10 DOWNTO 0) :=
CONV_STD_LOGIC_VECTOR(300, 11);
    CONSTANT bat_y : STD_LOGIC_VECTOR(10 DOWNTO 0) :=
CONV_STD_LOGIC_VECTOR(500, 11);
    SIGNAL ball_x_motion, ball_y_motion : STD_LOGIC_VECTOR(10 DOWNTO 0) :=
ball_speed;
    SIGNAL S_hit_counter : STD_LOGIC_VECTOR (15 DOWNTO 0);
BEGIN
    -- Color setup for red ball and cyan bat on white background
    red <= NOT bat_on;
    green <= NOT ball_on;
    blue <= NOT bat_on;
```

```
   -- Process to draw the ball
   balldraw : PROCESS (ball_x, ball_y, pixel_row, pixel_col) IS
      VARIABLE vx, vy : STD_LOGIC_VECTOR (10 DOWNTO 0);
   BEGIN
      IF pixel_col <= ball_x THEN vx := ball_x - pixel_col; ELSE vx := pixel_col - ball_x; END
IF;
      IF pixel_row <= ball_y THEN vy := ball_y - pixel_row; ELSE vy := pixel_row - ball_y;
END IF;
      IF ((vx * vx) + (vy * vy)) < (bsize * bsize) THEN ball_on <= game_on;
      ELSE ball_on <= '0';
      END IF;
   END PROCESS;

   -- Process to draw the bat
   batdraw : PROCESS (bat_x, pixel_row, pixel_col) IS
   BEGIN
      IF ((pixel_col >= bat_x - bat_w) AND pixel_col <= bat_x + bat_w) AND
        pixel_row >= bat_y - bat_h AND pixel_row <= bat_y + bat_h THEN
         bat_on <= '1';
      ELSE
         bat_on <= '0';
      END IF;
   END PROCESS;

   -- Process to move the ball and manage hit/miss detection
   mball : PROCESS
      VARIABLE temp : STD_LOGIC_VECTOR (11 DOWNTO 0);
   BEGIN
      ball_speed <= (10 DOWNTO SW'length => '0') & SW;
      WAIT UNTIL rising_edge(v_sync);

      IF serve = '1' AND game_on = '0' THEN
         game_on <= '1';
         ball_y_motion <= (NOT ball_speed) + 2;
      ELSIF ball_y <= bsize THEN
         ball_y_motion <= ball_speed;
      ELSIF ball_y + bsize >= 600 THEN
         ball_y_motion <= (NOT ball_speed) + 2;
         game_on <= '0';
         bat_w <= bat_w_start; -- Reset bat width after miss
         S_hit_counter <= (others => '0');
         hit_count <= S_hit_counter;
 -- Reset hit counter
      END IF;

      IF ball_x + bsize >= 800 THEN
         ball_x_motion <= (NOT ball_speed) + 1;
      ELSIF ball_x <= bsize THEN
         ball_x_motion <= ball_speed;
      END IF;

      IF (ball_x + bsize/2) >= (bat_x - bat_w) AND
```

```
            (ball_x - bsize/2) <= (bat_x + bat_w) AND
            (ball_y + bsize/2) >= (bat_y - bat_h) AND
            (ball_y - bsize/2) <= (bat_y + bat_h) THEN
             ball_y_motion <= (NOT ball_speed) + 1;

            IF bat_w > 1 THEN bat_w <= bat_w - 1; S_hit_counter <= S_hit_counter + 1;
hit_count <= S_hit_counter; END IF;

        END IF;

        temp := ('0' & ball_y) + (ball_y_motion(10) & ball_y_motion);
        IF game_on = '0' THEN
            ball_y <= CONV_STD_LOGIC_VECTOR(440, 11);
        ELSIF temp(11) = '1' THEN
            ball_y <= (OTHERS => '0');
        ELSE
            ball_y <= temp(10 DOWNTO 0);
        END IF;

        temp := ('0' & ball_x) + (ball_x_motion(10) & ball_x_motion);
        IF temp(11) = '1' THEN
            ball_x <= (OTHERS => '0');
        ELSE
            ball_x <= temp(10 DOWNTO 0);
        END IF;
    END PROCESS;
END Behavioral;
```

Original code “pong.vhd”:

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_ARITH.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;

ENTITY pong IS
    PORT (
        clk_in : IN STD_LOGIC; -- system clock
        VGA_red : OUT STD_LOGIC_VECTOR (3 DOWNTO 0); -- VGA outputs
        VGA_green : OUT STD_LOGIC_VECTOR (3 DOWNTO 0);
        VGA_blue : OUT STD_LOGIC_VECTOR (3 DOWNTO 0);
        VGA_hsync : OUT STD_LOGIC;
        VGA_vsync : OUT STD_LOGIC;
        btnl : IN STD_LOGIC;
        btnr : IN STD_LOGIC;
        btn0 : IN STD_LOGIC;
        SEG7_anode : OUT STD_LOGIC_VECTOR (7 DOWNTO 0); -- anodes of four 7-seg
displays
        SEG7_seg : OUT STD_LOGIC_VECTOR (6 DOWNTO 0)
    );
END pong;

ARCHITECTURE Behavioral OF pong IS
    SIGNAL pxl_clk : STD_LOGIC := '0'; -- 25 MHz clock to VGA sync module
```

```
    -- internal signals to connect modules
    SIGNAL S_red, S_green, S_blue : STD_LOGIC; --_VECTOR (3 DOWNTO 0);
    SIGNAL S_vsync : STD_LOGIC;
    SIGNAL S_pixel_row, S_pixel_col : STD_LOGIC_VECTOR (10 DOWNTO 0);
    SIGNAL batpos : STD_LOGIC_VECTOR (10 DOWNTO 0); -- 9 downto 0
    SIGNAL count : STD_LOGIC_VECTOR (20 DOWNTO 0);
    SIGNAL display : std_logic_vector (15 DOWNTO 0); -- value to be displayed
    SIGNAL led_mpx : STD_LOGIC_VECTOR (2 DOWNTO 0); -- 7-seg multiplexing clock
    COMPONENT bat_n_ball IS
       PORT (
          v_sync : IN STD_LOGIC;
          pixel_row : IN STD_LOGIC_VECTOR(10 DOWNTO 0);
          pixel_col : IN STD_LOGIC_VECTOR(10 DOWNTO 0);
          bat_x : IN STD_LOGIC_VECTOR (10 DOWNTO 0);
          serve : IN STD_LOGIC;
          red : OUT STD_LOGIC;
          green : OUT STD_LOGIC;
          blue : OUT STD_LOGIC
       );
    END COMPONENT;
    COMPONENT vga_sync IS
       PORT (
          pixel_clk : IN STD_LOGIC;
          red_in    : IN STD_LOGIC_VECTOR (3 DOWNTO 0);
          green_in  : IN STD_LOGIC_VECTOR (3 DOWNTO 0);
          blue_in   : IN STD_LOGIC_VECTOR (3 DOWNTO 0);
          red_out   : OUT STD_LOGIC_VECTOR (3 DOWNTO 0);
          green_out : OUT STD_LOGIC_VECTOR (3 DOWNTO 0);
          blue_out  : OUT STD_LOGIC_VECTOR (3 DOWNTO 0);
          hsync : OUT STD_LOGIC;
          vsync : OUT STD_LOGIC;
          pixel_row : OUT STD_LOGIC_VECTOR (10 DOWNTO 0);
          pixel_col : OUT STD_LOGIC_VECTOR (10 DOWNTO 0)
       );
    END COMPONENT;
    COMPONENT clk_wiz_0 is
       PORT (
          clk_in1  : in std_logic;
          clk_out1 : out std_logic
       );
    END COMPONENT;
    COMPONENT leddec16 IS
       PORT (
          dig : IN STD_LOGIC_VECTOR (2 DOWNTO 0);
          data : IN STD_LOGIC_VECTOR (15 DOWNTO 0);
          anode : OUT STD_LOGIC_VECTOR (7 DOWNTO 0);
          seg : OUT STD_LOGIC_VECTOR (6 DOWNTO 0)
       );
    END COMPONENT;

BEGIN
    pos : PROCESS (clk_in) is
    BEGIN
```

```
    if rising_edge(clk_in) then
        count <= count + 1;
        IF (btnl = '1' and count = 0 and batpos > 0) THEN
            batpos <= batpos - 10;
        ELSIF (btnr = '1' and count = 0 and batpos < 800) THEN
            batpos <= batpos + 10;
        END IF;
    end if;
  END PROCESS;
  led_mpx <= count(19 DOWNTO 17); -- 7-seg multiplexing clock
  add_bb : bat_n_ball
  PORT MAP(--instantiate bat and ball component
    v_sync => S_vsync,
    pixel_row => S_pixel_row,
    pixel_col => S_pixel_col,
    bat_x => batpos,
    serve => btn0,
    red => S_red,
    green => S_green,
    blue => S_blue
  );

  vga_driver : vga_sync
  PORT MAP(--instantiate vga_sync component
    pixel_clk => pxl_clk,
    red_in => S_red & "000",
    green_in => S_green & "000",
    blue_in => S_blue & "000",
    red_out => VGA_red,
    green_out => VGA_green,
    blue_out => VGA_blue,
    pixel_row => S_pixel_row,
    pixel_col => S_pixel_col,
    hsync => VGA_hsync,
    vsync => S_vsync
  );
  VGA_vsync <= S_vsync; --connect output vsync

  clk_wiz_0_inst : clk_wiz_0
  port map (
    clk_in1 => clk_in,
    clk_out1 => pxl_clk
  );
  led1 : leddec16
  PORT MAP(
    dig => led_mpx, data => display,
    anode => SEG7_anode, seg => SEG7_seg
  );
END Behavioral;
```

Modified code “pong.vhd”:

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_ARITH.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;

ENTITY pong IS
   PORT (
      clk_in : IN STD_LOGIC; -- system clock
      VGA_red : OUT STD_LOGIC_VECTOR (3 DOWNTO 0); -- VGA outputs
      VGA_green : OUT STD_LOGIC_VECTOR (3 DOWNTO 0);
      VGA_blue : OUT STD_LOGIC_VECTOR (3 DOWNTO 0);
      VGA_hsync : OUT STD_LOGIC;
      VGA_vsync : OUT STD_LOGIC;
      btnl : IN STD_LOGIC;
      btnr : IN STD_LOGIC;
      btn0 : IN STD_LOGIC;
      SEG7_anode : OUT STD_LOGIC_VECTOR (7 DOWNTO 0); -- anodes of four 7-seg
displays
      SEG7_seg : OUT STD_LOGIC_VECTOR (6 DOWNTO 0);
      SW : IN STD_LOGIC_VECTOR (4 DOWNTO 0)
   );
END pong;

ARCHITECTURE Behavioral OF pong IS
   SIGNAL pxl_clk : STD_LOGIC := '0'; -- 25 MHz clock to VGA sync module
   -- internal signals to connect modules
   SIGNAL S_red, S_green, S_blue : STD_LOGIC; --_VECTOR (3 DOWNTO 0);
   SIGNAL S_vsync : STD_LOGIC;
   SIGNAL S_pixel_row, S_pixel_col : STD_LOGIC_VECTOR (10 DOWNTO 0);
   SIGNAL batpos : STD_LOGIC_VECTOR (10 DOWNTO 0); -- 9 downto 0
   SIGNAL count : STD_LOGIC_VECTOR (20 DOWNTO 0);
   SIGNAL display : std_logic_vector (15 DOWNTO 0); -- value to be displayed
   SIGNAL led_mpx : STD_LOGIC_VECTOR (2 DOWNTO 0); -- 7-seg multiplexing clock
   SIGNAL S_hit_counter : STD_LOGIC_VECTOR (15 DOWNTO 0);
   COMPONENT bat_n_ball IS
      PORT (
         v_sync : IN STD_LOGIC;
         pixel_row : IN STD_LOGIC_VECTOR(10 DOWNTO 0);
         pixel_col : IN STD_LOGIC_VECTOR(10 DOWNTO 0);
         bat_x : IN STD_LOGIC_VECTOR (10 DOWNTO 0);
         serve : IN STD_LOGIC;
         red : OUT STD_LOGIC;
         green : OUT STD_LOGIC;
         blue : OUT STD_LOGIC;
         hit_count : OUT STD_LOGIC_VECTOR (15 DOWNTO 0);
         SW : IN STD_LOGIC_VECTOR (4 DOWNTO 0)
      );
   END COMPONENT;
   COMPONENT vga_sync IS
      PORT (
         pixel_clk : IN STD_LOGIC;
         red_in    : IN STD_LOGIC_VECTOR (3 DOWNTO 0);
         green_in  : IN STD_LOGIC_VECTOR (3 DOWNTO 0);
```

```
            blue_in   : IN STD_LOGIC_VECTOR (3 DOWNTO 0);
            red_out   : OUT STD_LOGIC_VECTOR (3 DOWNTO 0);
            green_out : OUT STD_LOGIC_VECTOR (3 DOWNTO 0);
            blue_out  : OUT STD_LOGIC_VECTOR (3 DOWNTO 0);
            hsync : OUT STD_LOGIC;
            vsync : OUT STD_LOGIC;
            pixel_row : OUT STD_LOGIC_VECTOR (10 DOWNTO 0);
            pixel_col : OUT STD_LOGIC_VECTOR (10 DOWNTO 0)
        );
    END COMPONENT;
    COMPONENT clk_wiz_0 is
        PORT (
            clk_in1  : in std_logic;
            clk_out1 : out std_logic
        );
    END COMPONENT;
    COMPONENT leddec16 IS
        PORT (
            dig : IN STD_LOGIC_VECTOR (2 DOWNTO 0);
            data : IN STD_LOGIC_VECTOR (15 DOWNTO 0);
            anode : OUT STD_LOGIC_VECTOR (7 DOWNTO 0);
            seg : OUT STD_LOGIC_VECTOR (6 DOWNTO 0)
        );
    END COMPONENT;

BEGIN
    pos : PROCESS (clk_in) is
    BEGIN
        if rising_edge(clk_in) then
            count <= count + 1;
            IF (btnl = '1' and count = 0 and batpos > 0) THEN
                batpos <= batpos - 10;
            ELSIF (btnr = '1' and count = 0 and batpos < 800) THEN
                batpos <= batpos + 10;
            END IF;
        end if;
    END PROCESS;
    led_mpx <= count(19 DOWNTO 17); -- 7-seg multiplexing clock
    add_bb : bat_n_ball
    PORT MAP(--instantiate bat and ball component
        v_sync => S_vsync,
        pixel_row => S_pixel_row,
        pixel_col => S_pixel_col,
        bat_x => batpos,
        serve => btn0,
        red => S_red,
        green => S_green,
        blue => S_blue,
        hit_count => display,
        SW => SW
    );

    vga_driver : vga_sync
```

```
    PORT MAP(--instantiate vga_sync component
       pixel_clk => pxl_clk,
       red_in => S_red & "000",
       green_in => S_green & "000",
       blue_in => S_blue & "000",
       red_out => VGA_red,
       green_out => VGA_green,
       blue_out => VGA_blue,
       pixel_row => S_pixel_row,
       pixel_col => S_pixel_col,
       hsync => VGA_hsync,
       vsync => S_vsync
    );
    VGA_vsync <= S_vsync; --connect output vsync

    clk_wiz_0_inst : clk_wiz_0
    port map (
      clk_in1 => clk_in,
      clk_out1 => pxl_clk
    );
    led1 : leddec16
    PORT MAP(
      dig => led_mpx, data => display,
      anode => SEG7_anode, seg => SEG7_seg
    );
  END Behavioral;
```

Original code “pong.xdc”:

```
set_property -dict { PACKAGE_PIN E3 IOSTANDARD LVCMOS33 } [get_ports {clk_in}];
create_clock -add -name sys_clk_pin -period 10.00 -waveform {0 5} [get_ports {clk_in}];

set_property -dict { PACKAGE_PIN B11 IOSTANDARD LVCMOS33 } [get_ports { VGA_hsync }];
#IO_L4P_T0_15 Sch=vga_hs
set_property -dict { PACKAGE_PIN B12 IOSTANDARD LVCMOS33 } [get_ports { VGA_vsync }];
#IO_L3N_T0_DQS_AD1N_15 Sch=vga_vs

set_property -dict { PACKAGE_PIN B7 IOSTANDARD LVCMOS33 } [get_ports { VGA_blue[0] }];
#IO_L2P_T0_AD12P_35 Sch=vga_b[0]
set_property -dict { PACKAGE_PIN C7 IOSTANDARD LVCMOS33 } [get_ports { VGA_blue[1] }];
#IO_L4N_T0_35 Sch=vga_b[1]
set_property -dict { PACKAGE_PIN D7 IOSTANDARD LVCMOS33 } [get_ports { VGA_blue[2] }];
set_property -dict { PACKAGE_PIN D8 IOSTANDARD LVCMOS33 } [get_ports { VGA_blue[3] }];
set_property -dict { PACKAGE_PIN A3 IOSTANDARD LVCMOS33 } [get_ports { VGA_red[0] }];
#IO_L8N_T1_AD14N_35 Sch=vga_r[0]
set_property -dict { PACKAGE_PIN B4 IOSTANDARD LVCMOS33 } [get_ports { VGA_red[1] }];
#IO_L7N_T1_AD6N_35 Sch=vga_r[1]
set_property -dict { PACKAGE_PIN C5 IOSTANDARD LVCMOS33 } [get_ports { VGA_red[2] }];
#IO_L1N_T0_AD4N_35 Sch=vga_r[2]
set_property -dict { PACKAGE_PIN A4 IOSTANDARD LVCMOS33 } [get_ports { VGA_red[3] }];
set_property -dict { PACKAGE_PIN C6 IOSTANDARD LVCMOS33 } [get_ports { VGA_green[0] }];
#IO_L1P_T0_AD4P_35 Sch=vga_g[0]
```

set_property -dict { PACKAGE_PIN A5 IOSTANDARD LVCMOS33 } [get_ports { VGA_green[1] }];
#IO_L3N_T0_DQS_AD5N_35 Sch=vga_g[1]
set_property -dict { PACKAGE_PIN B6 IOSTANDARD LVCMOS33 } [get_ports { VGA_green[2] }];
#IO_L2N_T0_AD12N_35 Sch=vga_g[2]
set_property -dict { PACKAGE_PIN A6 IOSTANDARD LVCMOS33 } [get_ports { VGA_green[3] }];

set_property -dict { PACKAGE_PIN N17 IOSTANDARD LVCMOS33 } [get_ports { btn0 }];
#IO_L9P_T1_DQS_14 Sch=btnc
set_property -dict { PACKAGE_PIN P17 IOSTANDARD LVCMOS33 } [get_ports { btnl }];
#IO_L12P_T1_MRCC_14 Sch=btnl
set_property -dict { PACKAGE_PIN M17 IOSTANDARD LVCMOS33 } [get_ports { btnr }];
#IO_L10N_T1_D15_14 Sch=btnr

set_property -dict {PACKAGE_PIN L18 IOSTANDARD LVCMOS33} [get_ports {SEG7_seg[0]}]
set_property -dict {PACKAGE_PIN T11 IOSTANDARD LVCMOS33} [get_ports {SEG7_seg[1]}]
set_property -dict {PACKAGE_PIN P15 IOSTANDARD LVCMOS33} [get_ports {SEG7_seg[2]}]
set_property -dict {PACKAGE_PIN K13 IOSTANDARD LVCMOS33} [get_ports {SEG7_seg[3]}]
set_property -dict {PACKAGE_PIN K16 IOSTANDARD LVCMOS33} [get_ports {SEG7_seg[4]}]
set_property -dict {PACKAGE_PIN R10 IOSTANDARD LVCMOS33} [get_ports {SEG7_seg[5]}]
set_property -dict {PACKAGE_PIN T10 IOSTANDARD LVCMOS33} [get_ports {SEG7_seg[6]}]

set_property -dict {PACKAGE_PIN U13 IOSTANDARD LVCMOS33} [get_ports {SEG7_anode[7]}]
set_property -dict {PACKAGE_PIN K2 IOSTANDARD LVCMOS33} [get_ports {SEG7_anode[6]}]
set_property -dict {PACKAGE_PIN T14 IOSTANDARD LVCMOS33} [get_ports {SEG7_anode[5]}]
set_property -dict {PACKAGE_PIN P14 IOSTANDARD LVCMOS33} [get_ports {SEG7_anode[4]}]
set_property -dict {PACKAGE_PIN J14 IOSTANDARD LVCMOS33} [get_ports {SEG7_anode[3]}]
set_property -dict {PACKAGE_PIN T9 IOSTANDARD LVCMOS33} [get_ports {SEG7_anode[2]}]
set_property -dict {PACKAGE_PIN J18 IOSTANDARD LVCMOS33} [get_ports {SEG7_anode[1]}]
set_property -dict {PACKAGE_PIN J17 IOSTANDARD LVCMOS33} [get_ports {SEG7_anode[0]}]

Modified code “pong.xdc”:

set_property -dict { PACKAGE_PIN E3 IOSTANDARD LVCMOS33 } [get_ports {clk_in}];
create_clock -add -name sys_clk_pin -period 10.00 -waveform {0 5} [get_ports {clk_in}];

set_property -dict { PACKAGE_PIN B11 IOSTANDARD LVCMOS33 } [get_ports { VGA_hsync }];
#IO_L4P_T0_15 Sch=vga_hs
set_property -dict { PACKAGE_PIN B12 IOSTANDARD LVCMOS33 } [get_ports { VGA_vsync }];
#IO_L3N_T0_DQS_AD1N_15 Sch=vga_vs

set_property -dict { PACKAGE_PIN B7 IOSTANDARD LVCMOS33 } [get_ports { VGA_blue[0] }];
#IO_L2P_T0_AD12P_35 Sch=vga_b[0]
set_property -dict { PACKAGE_PIN C7 IOSTANDARD LVCMOS33 } [get_ports { VGA_blue[1] }];
#IO_L4N_T0_35 Sch=vga_b[1]
set_property -dict { PACKAGE_PIN D7 IOSTANDARD LVCMOS33 } [get_ports { VGA_blue[2] }];
set_property -dict { PACKAGE_PIN D8 IOSTANDARD LVCMOS33 } [get_ports { VGA_blue[3] }];
set_property -dict { PACKAGE_PIN A3 IOSTANDARD LVCMOS33 } [get_ports { VGA_red[0] }];
#IO_L8N_T1_AD14N_35 Sch=vga_r[0]
set_property -dict { PACKAGE_PIN B4 IOSTANDARD LVCMOS33 } [get_ports { VGA_red[1] }];
#IO_L7N_T1_AD6N_35 Sch=vga_r[1]
set_property -dict { PACKAGE_PIN C5 IOSTANDARD LVCMOS33 } [get_ports { VGA_red[2] }];
#IO_L1N_T0_AD4N_35 Sch=vga_r[2]
set_property -dict { PACKAGE_PIN A4 IOSTANDARD LVCMOS33 } [get_ports { VGA_red[3] }];

```
set_property -dict { PACKAGE_PIN C6 IOSTANDARD LVCMOS33 } [get_ports { VGA_green[0] }];
#IO_L1P_T0_AD4P_35 Sch=vga_g[0]
set_property -dict { PACKAGE_PIN A5 IOSTANDARD LVCMOS33 } [get_ports { VGA_green[1] }];
#IO_L3N_T0_DQS_AD5N_35 Sch=vga_g[1]
set_property -dict { PACKAGE_PIN B6 IOSTANDARD LVCMOS33 } [get_ports { VGA_green[2] }];
#IO_L2N_T0_AD12N_35 Sch=vga_g[2]
set_property -dict { PACKAGE_PIN A6 IOSTANDARD LVCMOS33 } [get_ports { VGA_green[3] }];

set_property -dict { PACKAGE_PIN N17 IOSTANDARD LVCMOS33 } [get_ports { btn0 }];
#IO_L9P_T1_DQS_14 Sch=btnc
set_property -dict { PACKAGE_PIN P17 IOSTANDARD LVCMOS33 } [get_ports { btnl }];
#IO_L12P_T1_MRCC_14 Sch=btnl
set_property -dict { PACKAGE_PIN M17 IOSTANDARD LVCMOS33 } [get_ports { btnr }];
#IO_L10N_T1_D15_14 Sch=btnr

set_property -dict { PACKAGE_PIN J15   IOSTANDARD LVCMOS33 } [get_ports { SW[0] }];
#IO_L24N_T3_RS0_15 Sch=sw[0]
set_property -dict { PACKAGE_PIN L16   IOSTANDARD LVCMOS33 } [get_ports { SW[1] }];
#IO_L3N_T0_DQS_EMCCLK_14 Sch=sw[1]
set_property -dict { PACKAGE_PIN M13   IOSTANDARD LVCMOS33 } [get_ports { SW[2] }];
#IO_L6N_T0_D08_VREF_14 Sch=sw[2]
set_property -dict { PACKAGE_PIN R15   IOSTANDARD LVCMOS33 } [get_ports { SW[3] }];
#IO_L13N_T2_MRCC_14 Sch=sw[3]
set_property -dict { PACKAGE_PIN R17   IOSTANDARD LVCMOS33 } [get_ports { SW[4] }];
#IO_L12N_T1_MRCC_14 Sch=sw[4]

set_property -dict {PACKAGE_PIN L18 IOSTANDARD LVCMOS33} [get_ports {SEG7_seg[0]}]
set_property -dict {PACKAGE_PIN T11 IOSTANDARD LVCMOS33} [get_ports {SEG7_seg[1]}]
set_property -dict {PACKAGE_PIN P15 IOSTANDARD LVCMOS33} [get_ports {SEG7_seg[2]}]
set_property -dict {PACKAGE_PIN K13 IOSTANDARD LVCMOS33} [get_ports {SEG7_seg[3]}]
set_property -dict {PACKAGE_PIN K16 IOSTANDARD LVCMOS33} [get_ports {SEG7_seg[4]}]
set_property -dict {PACKAGE_PIN R10 IOSTANDARD LVCMOS33} [get_ports {SEG7_seg[5]}]
set_property -dict {PACKAGE_PIN T10 IOSTANDARD LVCMOS33} [get_ports {SEG7_seg[6]}]

set_property -dict {PACKAGE_PIN U13 IOSTANDARD LVCMOS33} [get_ports {SEG7_anode[7]}]
set_property -dict {PACKAGE_PIN K2 IOSTANDARD LVCMOS33} [get_ports {SEG7_anode[6]}]
set_property -dict {PACKAGE_PIN T14 IOSTANDARD LVCMOS33} [get_ports {SEG7_anode[5]}]
set_property -dict {PACKAGE_PIN P14 IOSTANDARD LVCMOS33} [get_ports {SEG7_anode[4]}]
set_property -dict {PACKAGE_PIN J14 IOSTANDARD LVCMOS33} [get_ports {SEG7_anode[3]}]
set_property -dict {PACKAGE_PIN T9 IOSTANDARD LVCMOS33} [get_ports {SEG7_anode[2]}]
set_property -dict {PACKAGE_PIN J18 IOSTANDARD LVCMOS33} [get_ports {SEG7_anode[1]}]
set_property -dict {PACKAGE_PIN J17 IOSTANDARD LVCMOS33} [get_ports {SEG7_anode[0]}]
```