SOCCER GAME

Bilkent University Electrical and Electronics Engineering Department
EEE102 Term Project

HİDAYET SELİM ASAN
22103812
SECTION 01

Youtube:

https://youtu.be/xjNQx9v85Pc

Objective:

This project aimed to implement a basic soccer game via Basys3. In this game, there are two players in front of two their goals. The aim is to prevent the ball from entering your goal and score to the opponent's goal. Ball bounces as it hits either sides or players. Players are allowed to move only vertically and they can be controlled by the buttons on Basys3.

Methodology:

The initial step was to desing a vga controller clockule. This clockule will control hscync and vscync signals, scan the screen in x and y coordinates, and indicate signal_on areas on the screen. Having completed vga controller clockule, Top clockule for the gameplay controls was designed. Finally, design clockifications were done.

Design Specifications:

There are seven inputs and five outputs as listed below:

i_clk : in std_logic

bP1_d : in std_logic

bP1_u : in std_logic

bP2_d : in std_logic

bP2_u : in std_logic

hscync : out std_logic

vscync : out std_logic

o_red : out std_logic_vector(3 downto 0)

o_green : out std_logic_vector(3 downto 0)

o_blue : out std_logic_vector(3 downto 0)

i_continue : in std_Logic

i_start: inout std_logic

There are six processes for the gameplay as listed below:

1. Game_Screens:

   Controls the transitions between game screens with if statements depending on signals *i_start* and *i_continue*.

2. Movements_of_Gkeepers1

   If the button controlling signal *bP1_d* is pressed, it decreases the position of Player1 by one. If the button controlling signal *bP1_u* is pressed, it increments the position of Player1 by one.

3. Movements_of_Gkeepers2

   If the button controlling signal *bP2_d* is pressed, it decreases the position of Player2 by one. If the button controlling signal *bP2_u* is pressed, it increments the position of Player2 by one.

4. Ball_movement

   It keeps the previous position of the ball in x and y coordinates. If ball does not hit any bouncer object (sides or players), the ball continues it movement in the same direction. If it hits, it reflects with the proper angle.

5. Player1_wins

   If player1 reaches the score limit, it wins and coordinate related signal to change the game screen to *Victory of Player 1* screen. This transition is organised by *Game_screen* process.

6. Player2_wins

   7. If player2 reaches the score limit, it wins and coordinate related signal to change the game screen to *Victory of Player 2* screen. This transition is organised by *Game_screen* process.

Figure1 depicts the RTL schematic of the design.

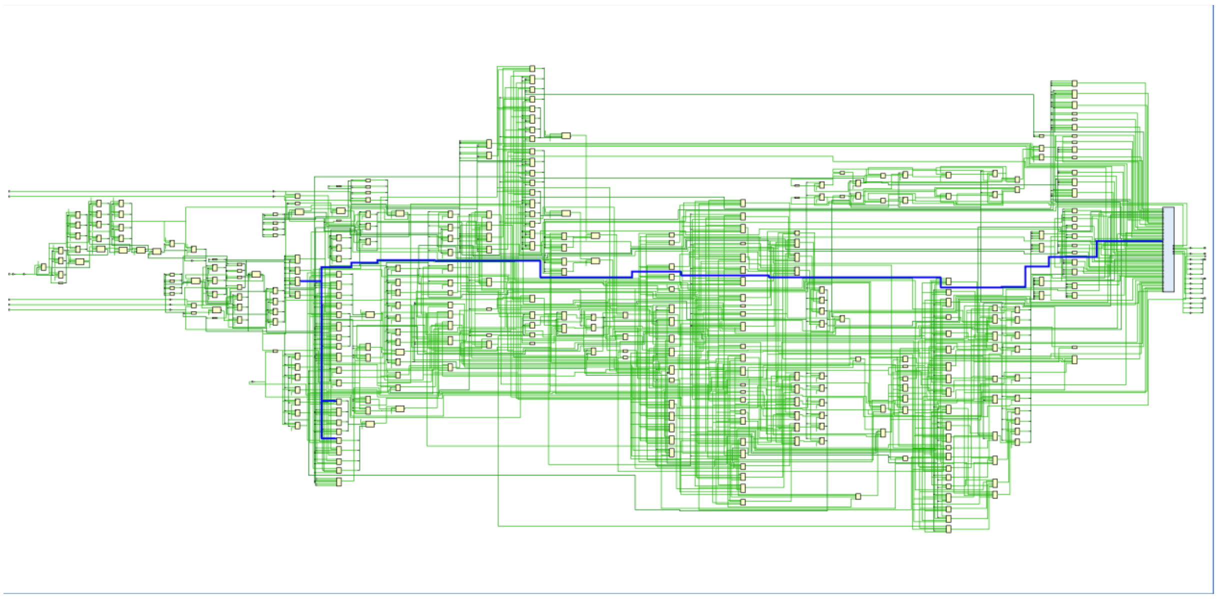

Figure 1 The RTL Schematic

Results:

The Start Menu (Figure 2)

i_start = ‘0’

i_continue = ‘x’

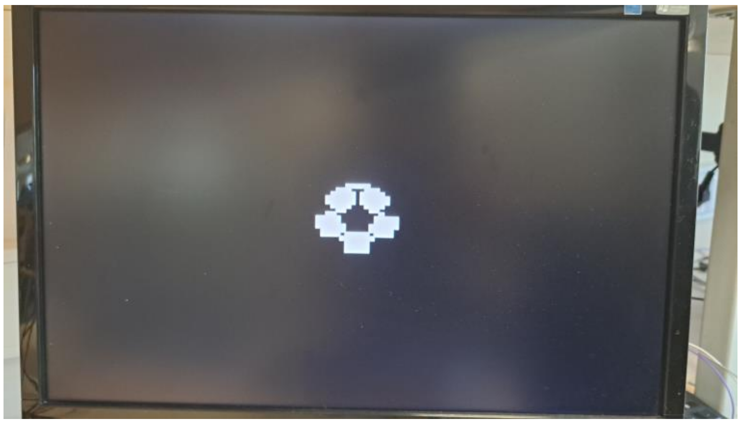

Figure 2 The Start Menu

Game Continuous (Figure 3)

i_start = ‘1’

i_continue = ‘1’

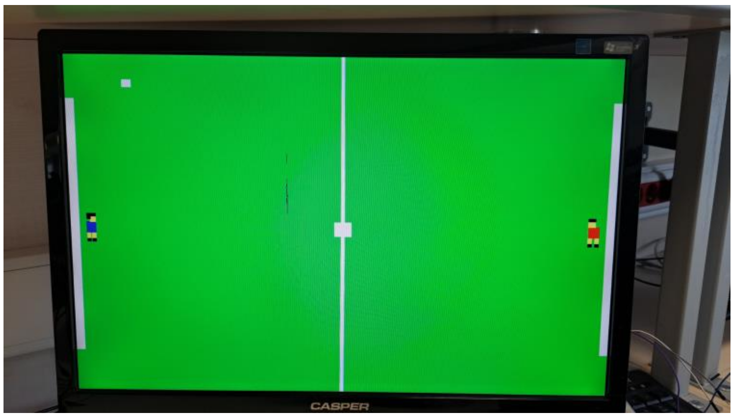

Figure 3 Game Continuous

Red Wins (Figure 4)

Red reaches the score limit.

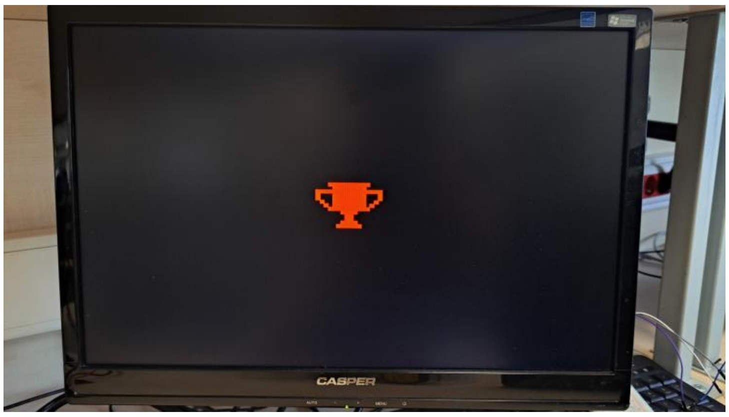

Figure 4 Red Wins

Blue Wins (Figure 5)

Blue reaches the score limit.

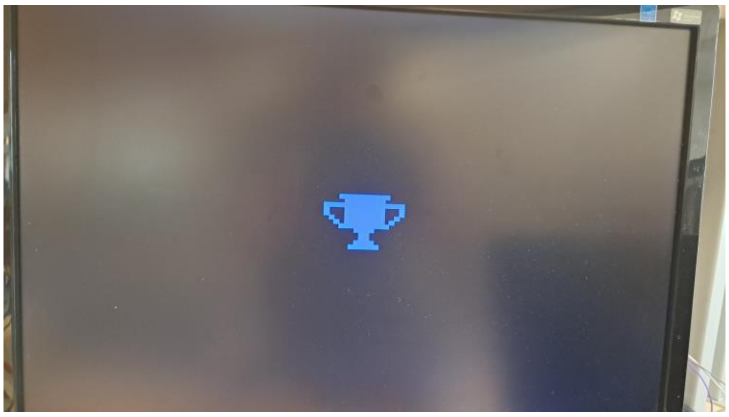

Figure 5 Blue Wins

Conclusion:

In this project, I designed a basic soccer game. It is a two-player local co-op game. Players can move their goalkeepers with the buttons on Basys3. Game can be paused and continued with the switch on Basys3. There were lots of errors during the design process. With appropriate troubleshooting they all are fixed.

Appendix:

Top_clockule.vhd

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;



entity Top_clockule is
   Port (i_clk : in  std_logic;
        bP1_d  : in  std_logic;
        bP1_u  : in  std_logic;
        bP2_d  : in  std_logic;
        bP2_u  : in  std_logic;
        hscync  : out std_logic;
        vscync : out std_logic;
        o_red   : out std_logic_vector(3 downto 0);
        o_green : out std_logic_vector(3 downto 0);
        o_blue  : out std_logic_vector(3 downto 0);
        i_continue  : in  std_Logic;
        i_start: inout std_logic
);
end Top_clockule;

architecture Behavioral of Top_clockule is

component vga_controller is
```

```
port(

     clk: in std_logic;

     hscync, vscync: out std_logic;

     signal_on : out std_logic;

     pixel_x,pixel_y: out unsigned(9 downto 0)


);

end component;


signal r_x, r_y : unsigned(9 downto 0); -- row and column of the vga controller scanner

signal Gkeeper1, Gkeeper2 : unsigned(9 downto 0) := to_unsigned(240,10); -- Positions of
goolkeepers in y direction

signal signal_on  : std_logic; -- signal_on signal coming from vga controller

signal r_hscync, r_vscync: std_logic;

signal clk_Gkeeper : std_logic; -- slower clock for the movements of Goalkeepers

signal red, blue : std_logic_vector(3 downto 0) := (others => '0');

signal green    : std_logic_vector(3 downto 0)  := "0011"; -- the court will be green

signal clk_slower  : unsigned(17 downto 0) := (others => '0'); -- refresher for slower clock cycles

signal ball_x : unsigned(9 downto 0) := to_unsigned(320,10); -- Position of the ball in both x directions

signal ball_y : unsigned(9 downto 0) := to_unsigned(240,10); -- Position of the ball in both y
directions

signal ball_x_prev: unsigned(9 downto 0) := to_unsigned(320,10) ;

signal ball_y_prev: unsigned(9 downto 0) := to_unsigned(240,10);

signal direction  : std_logic_vector(1 downto 0) := "00"; -- it goes left up

signal clk_ball   : std_logic;

signal score_P1, score_P2 : unsigned(3 downto 0) := (others => '0'); -- Score count for players

signal out_P1, out_P2 : integer range 0 to 3 := 0; -- Out scores of player

signal P1_win, P2_win : std_logic := '0';

signal Game_state     : std_logic_vector(1 downto 0) :="00"; -- Game state: 00--> start screen, 01-->
player2 win, 10 --> player 1 win, 11 --> game continuous

constant c_scoreLimit : integer := 3; -- Score Limit
```

```
begin

vga: vga_controller
port map(clk => i_clk,
        hscync => hscync,
        vscync => vscync,
        pixel_x => r_x,
        pixel_y => r_y,
        signal_on => signal_on
);

New_clock: process(i_clk) is
begin
    if rising_edge(i_clk) then
        clk_slower <= clk_slower +1;
    end if;
end process;

clk_Gkeeper <= clk_slower(17);
clk_Ball    <= clk_slower(17);

Game_Screens: process(r_y, r_x) is
begin
      -- Start Menu
      if i_start = '0' then
           if ((((r_x <= 315) and (r_x >= 300))or ((r_x <= 345) and (r_x >= 330)))and (r_y <= 210) and (r_y
>= 205)) or
              ((((r_x <= 320) and (r_x >= 295))or ((r_x <= 350) and (r_x >= 325)))and (r_y <= 215) and (r_y
>= 210)) or
              ((((r_x <= 320) and (r_x >= 290))or ((r_x <= 355) and (r_x >= 325)))and (r_y <= 225) and (r_y
>= 215)) or
              ((((r_x <= 315) and (r_x >= 295))or ((r_x <= 350) and (r_x >= 330)))and (r_y <= 230) and (r_y
>= 225)) or
```

```
            ((((r_x <= 310) and (r_x >= 300))or ((r_x <= 345) and (r_x >= 335)))and (r_y <= 235) and (r_y
>= 230)) or
            ((((r_x <= 300) and (r_x >= 290))or ((r_x <= 350) and (r_x >= 345)))and (r_y <= 240) and (r_y
>= 235)) or
            ((((r_x <= 305) and (r_x >= 285))or ((r_x <= 360) and (r_x >= 340)))and (r_y <= 265) and (r_y
>= 240)) or
            ((((r_x <= 285) and (r_x >= 280))or ((r_x <= 365) and (r_x >= 360)))and (r_y <= 255) and (r_y
>= 240)) or
            ((((r_x <= 310) and (r_x >= 305))or ((r_x <= 340) and (r_x >= 335)))and (r_y <= 260) and (r_y
>= 250)) or
            ((((r_x <= 310) and (r_x >= 305))or ((r_x <= 340) and (r_x >= 335)))and (r_y <= 270) and (r_y
>= 265)) or
            ((r_x <= 335) and (r_x >= 310) and (r_y <= 285) and (r_y >= 260)) or
            ((r_x <= 335) and (r_x >= 310) and (r_y <= 205) and (r_y >= 200))then


            red  <= (others => '1');
            blue  <= (others => '1');
            green <= (others => '1');
          else
            red  <= (others => '0');
            blue  <= (others => '0');
            green <= (others => '0');
          end if;
      -- Player 1 win
      elsif (P1_win = '1') and (P2_win = '0') then
        if ((r_x <= 360) and (r_x >= 280) and (r_y <= 225) and (r_y >= 220)) or
        ((r_x <= 345) and (r_x >= 295) and (r_y <= 215) and (r_y >= 210)) or
        ((r_x <= 340) and (r_x >= 300) and (r_y <= 250) and (r_y >= 215)) or
        ((((r_x <= 285) and (r_x >= 280))or ((r_x <= 360) and (r_x >= 355)))and (r_y <= 235) and (r_y >=
225)) or
        ((((r_x <= 290) and (r_x >= 285))or ((r_x <= 355) and (r_x >= 350)))and (r_y <= 240) and (r_y >=
235)) or
        ((((r_x <= 295) and (r_x >= 290))or ((r_x <= 350) and (r_x >= 345)))and (r_y <= 245) and (r_y >=
240)) or
        ((((r_x <= 300) and (r_x >= 295))or ((r_x <= 345) and (r_x >= 340)))and (r_y <= 250) and (r_y >=
245)) or
```

```
    ((r_x <= 330) and (r_x >= 310) and (r_y <= 255) and (r_y >= 250)) or
    ((r_x <= 325) and (r_x >= 315) and (r_y <= 265) and (r_y >= 255)) or
    ((r_x <= 330) and (r_x >= 310) and (r_y <= 270) and (r_y >= 265)) or
    ((r_x <= 335) and (r_x >= 305) and (r_y <= 275) and (r_y >= 270)) then
        red  <= (others => '0');
        blue  <= (others => '1');
        green <= (others => '0');
    else
        red  <= (others => '0');
        blue  <= (others => '0');
        green <= (others => '0');
    end if;

  -- Player 2 win
  elsif (P1_win = '0') and (P2_win = '1') then
        if ((r_x <= 360) and (r_x >= 280) and (r_y <= 225) and (r_y >= 220)) or
    ((r_x <= 345) and (r_x >= 295) and (r_y <= 215) and (r_y >= 210)) or
    ((r_x <= 340) and (r_x >= 300) and (r_y <= 250) and (r_y >= 215)) or
    ((((r_x <= 285) and (r_x >= 280))or ((r_x <= 360) and (r_x >= 355)))and (r_y <= 235) and (r_y >=
225)) or
    ((((r_x <= 290) and (r_x >= 285))or ((r_x <= 355) and (r_x >= 350)))and (r_y <= 240) and (r_y >=
235)) or
    ((((r_x <= 295) and (r_x >= 290))or ((r_x <= 350) and (r_x >= 345)))and (r_y <= 245) and (r_y >=
240)) or
    ((((r_x <= 300) and (r_x >= 295))or ((r_x <= 345) and (r_x >= 340)))and (r_y <= 250) and (r_y >=
245)) or
    ((r_x <= 330) and (r_x >= 310) and (r_y <= 255) and (r_y >= 250)) or
    ((r_x <= 325) and (r_x >= 315) and (r_y <= 265) and (r_y >= 255)) or
    ((r_x <= 330) and (r_x >= 310) and (r_y <= 270) and (r_y >= 265)) or
    ((r_x <= 335) and (r_x >= 305) and (r_y <= 275) and (r_y >= 270)) then
        red  <= (others => '1');
        blue  <= (others => '0');
        green <= (others => '0');
        else
              red  <= (others => '0');
              blue  <= (others => '0');
              green <= (others => '0');
        end if;
      -- Game Contunious
      else
           if ((r_x >= ball_x - 5) and (r_x <= ball_x + 5) and (r_y >= ball_y -5) and (r_y <= ball_y +5)) or --
Ball
            ((r_x >= 0) and (r_x <= 10) and (r_y >= 60) and (r_y <= 419)) or ((r_x >= 630) and (r_x <= 640)
and (r_y >= 60) and (r_y <= 419)) or -- Goals
            ((r_x >= 318) and (r_x <= 322)) or
            ((r_x >= 310) and (r_x <= 330) and (r_y <= 250) and (r_y >= 230))  then
              red  <= (others => '1');
              blue  <= (others => '1');
              green <= (others => '1');
           -- Goal Keeper 2 red
           elsif ((r_x >= 609) and (r_x <= 617) and (r_y <= Gkeeper2 -16 ) and (r_y >= Gkeeper2 - 20 )) or
               ((r_x >= 621) and (r_x <= 617) and (r_y <= Gkeeper2 -12 ) and (r_y >= Gkeeper2 - 20 )) or
               ((((r_x >= 609) and (r_x <= 613)) or ((r_x >= 617) and (r_x <= 621))) and (r_y <= Gkeeper2
+20) and (r_y >= Gkeeper2 +16 )) then -- hair and shoes
              red  <= (others => '0');
              blue  <= (others => '0');
              green <= (others => '0');
           elsif ((r_x >= 609) and (r_x <= 617) and (r_y <= Gkeeper2 -8 ) and (r_y >= Gkeeper2 - 16 )) or
               ((((r_x >= 609) and (r_x <= 613)) or ((r_x >= 617) and (r_x <= 621))) and (r_y <= Gkeeper2
+16) and (r_y >= Gkeeper2 +8 )) then -- Face and legs
               red  <= (others => '1');
               blue  <= (others => '0');
               green <= (others => '1');
           elsif  ((r_x >= 609) and (r_x <= 621) and (r_y <= Gkeeper2 +8 ) and (r_y >= Gkeeper2 - 8 )) then -
- body
              red  <= (others => '1');
              blue  <= (others => '0');
```

```
             green <= (others => '0');


          -- Goal Keeper 1 blue

          elsif ((r_x >= 23) and (r_x <= 29) and (r_y <= Gkeeper1  -16 ) and (r_y >= Gkeeper1  - 20 )) or

              ((r_x >= 19) and (r_x <= 23) and (r_y <= Gkeeper1  -12 ) and (r_y >= Gkeeper1  - 20 )) or

              ((((r_x  >= 19) and (r_x <= 23)) or ((r_x >= 25) and (r_x <= 29)))  and (r_y <= Gkeeper1  +20)
and (r_y >= Gkeeper1  +16 )) then -- hair and shoes

             red  <= (others => '0');

             blue  <= (others => '0');

             green <= (others => '0');

          elsif ((r_x >= 23) and (r_x <= 29) and (r_y <= Gkeeper1  -8 ) and (r_y >= Gkeeper1  - 16 )) or

              ((((r_x  >= 19) and (r_x <= 23)) or ((r_x >= 25) and (r_x <= 29)))  and (r_y <= Gkeeper1  +16)
and (r_y >= Gkeeper1  +8 )) then -- Face and legs

              red  <= (others => '1');

              blue  <= (others => '0');

              green <= (others => '1');

          elsif  ((r_x >= 19) and (r_x <= 29) and (r_y <= Gkeeper1 +8 ) and (r_y >= Gkeeper1  - 8 )) then --
body

             red  <= (others => '0');

             blue  <= (others => '1');

             green <= (others => '0');


         else -- The court

             red  <= (others => '0');

             blue  <= (others => '0');

             green <= (others => '1');

          end if;

     end if;

end process;


Movements_of_Gkeepers1:  process(clk_Gkeeper)

begin

   if rising_edge(clk_Gkeeper)  and (i_continue  = '1') then
```

```
          if bP1_d = '1' and bP1_u = '0' then
              Gkeeper1  <= Gkeeper1  +1 ;
          elsif bP1_d = '0' and bP1_u  = '1' then
              Gkeeper1  <= Gkeeper1  - 1 ;
          end if;
    end if;

end Process;

Movements_of_Gkeepers2:  process(clk_Gkeeper)
begin
    if rising_edge(clk_Gkeeper)  and (i_continue  = '1') then
       if (Gkeeper1  /= 20) or (Gkeeper1  /= 460) then
          if bP2_d = '1' and bP2_u = '0' then
              Gkeeper2  <= Gkeeper2  +1 ;
          elsif bP2_d = '0' and bP2_u  = '1' then
              Gkeeper2  <= Gkeeper2  - 1 ;
          end if;
       end if;
    end if;

end Process;



Ball_movement:  process(clk_ball)  is
begin

    if rising_edge(clk_ball)  and (i_continue  = '1') and (i_start = '1') then

           ball_x_prev <= ball_x;
           ball_y_prev <= ball_Y;
```

```
        -- hit the up wall
        if (ball_y_prev  = to_unsigned(5,10))  and (direction = "00") then
          direction <= "01";
          ball_y <= ball_y_prev  +1;
          ball_x <= ball_x_prev  -1;
        elsif (ball_y_prev  = to_unsigned(5,10))  and (direction = "10") then
          direction <= "11";
          ball_y <= ball_y_prev  +1;
          ball_x <= ball_x_prev  +1;
        -- hit the down wall
        elsif (ball_y_prev  = to_unsigned(475,10))  and (direction  = "11") then
          direction <= "10";
          ball_y <= ball_y_prev  -1;
          ball_x <= ball_x_prev  +1;
        elsif (ball_y_prev  = to_unsigned(475,10))  and (direction  = "01") then
          direction <= "00";
          ball_y <= ball_y_prev  -1;
          ball_x <= ball_x_prev  -1;
        -- hit the player 1
        elsif (ball_x_prev  <= to_unsigned(29,10))  and (ball_y_prev  <= Gkeeper1  + 20) and (ball_y_prev
>= Gkeeper1  - 20) and (direction  = "01") then
          direction <= "11";
          ball_y <= ball_y_prev  +1;
          ball_x <= ball_x_prev  +1;
        elsif (ball_x_prev  = to_unsigned(29,10))  and (ball_y_prev  <= Gkeeper1 + 20) and (ball_y_prev
>= Gkeeper1  - 20) and (direction  = "00") then
          direction <= "10";
          ball_y <= ball_y_prev  -1;
          ball_x <= ball_x_prev  +1;
        -- hit the player 2
        elsif (ball_x_prev  = to_unsigned(609,10))  and (ball_y_prev  <= Gkeeper2  + 20) and (ball_y_prev
>= Gkeeper2  - 20) and (direction  = "10") then
          direction <= "00";
          ball_y <= ball_y_prev +1;

          ball_x <= ball_x_prev +1;

        elsif (ball_x_prev = to_unsigned(609,10))  and (ball_y_prev <= Gkeeper2 + 20) and (ball_y_prev
>= Gkeeper2 - 20) and (direction = "11") then

          direction <= "01";

          ball_y <= ball_y_prev -1;

          ball_x <= ball_x_prev +1;

        -- Ball goes back Player 2

        elsif (ball_x_prev = to_unsigned(639,10))  then

          ball_y <= to_unsigned(240,10);

          ball_x <= to_unsigned(340,10);

          if (r_y >= 60) and (r_y <= 419) then -- Player 1 scores

            score_P1 <= score_P1 +1;


          end if;

        -- Ball goes back Player 1

        elsif (ball_x_prev = to_unsigned(0,10))  then

          ball_y <= to_unsigned(240,10);

          ball_x <= to_unsigned(340,10);

          if (r_y >= 60) and (r_y <= 419) then -- Player 2 scores

            score_P2 <= score_P2 + 1;


          end if;

        -- Keep moving in the same direction otherwise

        elsif direction = "00" then

          direction <= "00";

          ball_y <= ball_y_prev -1;

          ball_x <= ball_x_prev -1;

       elsif direction = "01" then

          direction <= "01";

          ball_y <= ball_y_prev +1;

          ball_x <= ball_x_prev -1;
```

```
            elsif direction = "10" then
                direction <= "10";
                ball_y <= ball_y_prev -1;
                ball_x <= ball_x_prev +1;
            elsif direction = "11" then
                direction <= "11";
                ball_y <= ball_y_prev +1;
                ball_x <= ball_x_prev +1;
            end if;

    end if;
end process;




Player1_wins: process(score_P1) is
begin
    if to_integer(score_P1) = c_scoreLimit then
        P1_win <= '1';

    end if;
end process;

Player2_wins: process(score_P2) is
begin
    if to_integer(score_P2) = c_scoreLimit then
        P2_win <= '1';
    end if;
end process;



o_red <= (signal_on & signal_on & signal_on & signal_on ) and red;
```

```
o_green <= (signal_on  & signal_on & signal_on  & signal_on  ) and green;

o_blue  <=(signal_on  & signal_on  & signal_on  & signal_on  ) and blue;

vscync <= r_vscync;

hscync <= r_hscync;




end Behavioral;
```

Vga_controller.vhd

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity vga_controller is
  port(
    clk: in std_logic;
    hscync, vscync: out std_logic;
    signal_on : buffer std_logic;
    pixel_x,pixel_y: out unsigned(9 downto 0)
);
end vga_controller;

architecture arch of vga_controller is
   signal video_on:std_logic;

  -- Constant Values Of Display
  constant HD_area: integer:=640;  --horizontal visiable area
  constant HF_area: integer:=16 ; --horizontal front porch
  constant HB_area: integer:=48 ; --horizontal back porch
  constant HS_area: integer:=96 ; --horizontal sync pulse
  constant VD_area: integer:=480;  --vertical display area
  constant VF_area: integer:=10;   --vertical front porch
  constant VB_area: integer:=33;   --vertical back porch
  constant VS_area: integer:=2; --sync pulse

    signal clock2_reg, clock2_next: std_logic; -- clockd-2 counterrrrr

  signal v_count_reg, v_count_next: unsigned(9 downto 0);-- sync counters to scan
  signal h_count_reg, h_count_next: unsigned(9 downto 0);
  -- output buffer
  signal v_sync_reg, h_sync_reg: std_logic;
  signal v_sync_next, h_sync_next: std_logic;
  -- status signal
  signal h_end, v_end, pixel_tick: std_logic;
  signal clock_refresher:unsigned(1 downto 0);
  signal clk50:std_logic;



begin
  Clock_refresherrr:process(clk)
  begin
   if rising_edge(clk) then
      clock_refresher <= clock_refresher +1;
  end if;
  end process;
clk50 <= clock_refresher(0); -- 50Mhz Clok
   process (h_count_reg,h_end,pixel_tick)
  begin
    if pixel_tick='1' then -- 25 MHz tick
      if h_end='1' then
        h_count_next <= (others=>'0');
      else
        h_count_next <= h_count_reg + 1;
      end if;
    else
      h_count_next <= h_count_reg;
    end if;
  end process;

  -- clock-2 circuit to generate 25 MHz enable tick
```

```
clock2_next <= not clock2_reg;
-- 25 MHz pixel tick
pixel_tick <= '1' when clock2_reg='1' else '0';
  -- horizontal and vertical sync
h_sync_next <=
  '1' when (h_count_reg>=(HD_area+HF_area))          -- between 656 and 751
      and (h_count_reg<=(HD_area+HF_area+HS_area-1))  else
  '0';
v_sync_next <=


-- end points
h_end <=  -- warns when it comes to horizontal end
  '1' when h_count_reg=(HD_area+HF_area+HB_area+HS_area-1)  else -- clock 800
  '0';
v_end <=  -- warns when it comes to vertical end
  '1' when v_count_reg=(VD_area+VF_area+VB_area+VS_area-1)  else -- clock 525
  '0';
--horizontal scan
--vertical scan
process (v_count_reg,h_end,v_end,pixel_tick)
begin
  if pixel_tick='1' and h_end='1' then
    if (v_end='1') then
      v_count_next <= (others=>'0');
    else
      v_count_next <= v_count_reg + 1;
    end if;
  else
    v_count_next <= v_count_reg;
  end if;
end process;
```

```
    '1' when (v_count_reg>=(VD_area+VF_area))           -- between 490 and 491
         and (v_count_reg<=(VD_area+VF_area+VS_area-1))  else
    '0';
  --  SİGNAL_ON AREA
  video_on <=
    '1' when (h_count_reg<HD_area)  and (v_count_reg<VD_area)  else
    '0';
    process (clk50)
  begin
    if rising_edge(clk50)  then
      clock2_reg <= clock2_next;
      v_count_reg  <= v_count_next;
      h_count_reg  <= h_count_next;
      v_sync_reg  <= v_sync_next;
      h_sync_reg  <= h_sync_next;
    end if;
  end process;



  -- output signal
  video_on <= signal_on;
  hscync <= h_sync_reg;
  vscync <= v_sync_reg;
  pixel_x <= h_count_reg;
  pixel_y <= v_count_reg;

end architecture;
```