1.0 - Introduction

This laboratory project is done in the environment of Xilinx ISE CAD software based on VHDL. The purpose of this project is to examine the functionalities of VGA and video-output subsystem with the connections with the FPGA board. It is also designed to learn VHDL-coding for real-time applications.

2.0 - Specifications

The final result of this project is to make a simple pong game, where there are two paddles that can reflect the ball, and there are borders to keep the ball inside the platform, and two holes for the ball to hit a score. The pong game is to be displayed on the screen and controlled by the switches on the FPGA board. The platform/field of the game should be in the colour of green, the two paddles should be red and blue, and the ball should be in yellow. The entire game design is called the Simple Video-Game Processor (SVGP).

The SVGP is required to display a static video frame with dynamic elements ball and paddles. The yellow ball should move freely within the borders and reflecting when it hits the borders or the paddles, where it changes its trajectory of 90 degrees. The paddles can be controlled by the players which can move only up or down. Whenever the ball hits a goal, where the ball moves into the holes, the colour of the ball would change to red, and the position of the ball would be reset to the center of the field with the colour of yellow.

3.0 – Device Design

3.1 – Symbol of the device

Figure 1 shows the schematic symbol of the top-level design entity VHDL file, which is basically the main program.

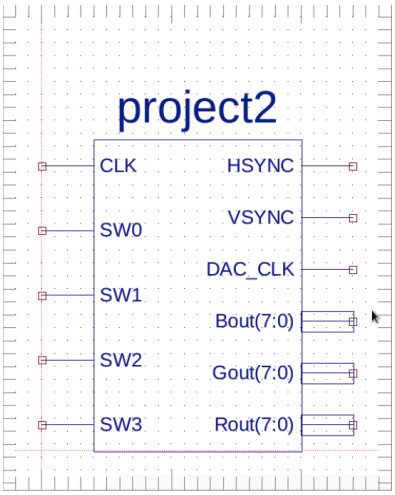

Figure 1 – Symbol of the top-level design entity

3.2 – VGA Specifications

In order to show the game on the screen, the horizontal time periods are to be multiples of the VGA pixel clock, which is 25Mhz for every 60Hz refresh. The vertical time periods are specified in multiples of VGA lines. Tables I and II shows the detailed clock cycles needed for each component.

Table I – Horizontal time periods specification

Parameter	Clock Cycles
Complete Line	800
Front Porch	16
Sync Pulse	96
Back Porch	48
Active image area	640

Table II – Vertical time periods specification

Parameter	Clock Cycles
Complete Line	525
Front Porch	10
Sync Pulse	2

Back Porch	33
Active image area	480

3.3 – Block Diagram

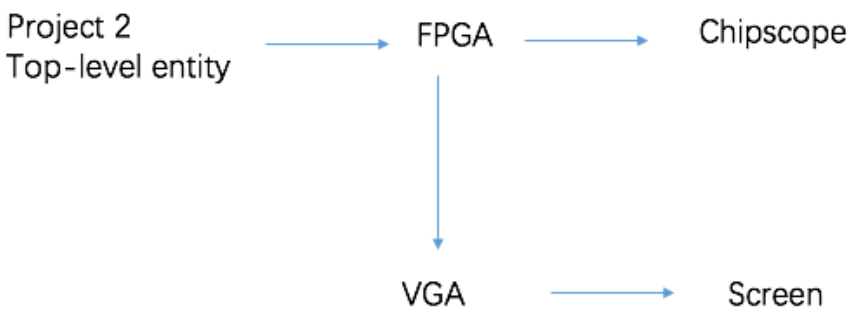

Figure 2 – Block Diagram

3.4 – Process Diagram

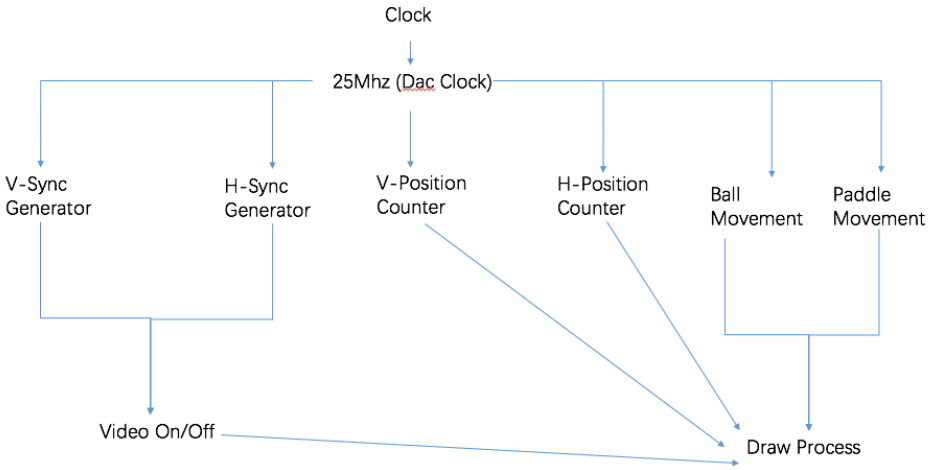

Figure 3 – Process Diagram

3.5 – Description of Results

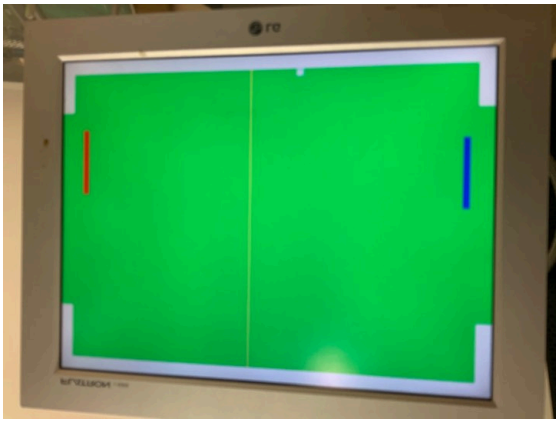

Figure 4 – Results of the Display

The VGA adapter was able to draw the game and meet the design specifications. The paddles move based on the position set in the paddle movement process. The ball movement is handled by ball movement process. The results were sent to the draw process in which it detects the pixel sequentially in the draw process. The top border, bottom border is detected first, the hole next, right border, then the left border, and then the paddle 1 and paddle 2 is next which is followed by the ball, and finally the line at the very middle, else the pixel is a background. This was obtained through and if, elsif and else statement to make sure that one-pixel is not drawn as both the background and a ball. The basic layout is there and the switches work and moves the paddle. The ball detection was very simplistic. If the ball was found to be in the region of the paddle or top border, change the ball speed based on where it wants to go i.e. right is positive x position. The absolute value was taken of the ball speed because if the ball does not exit the paddle within a frame, the ball will be detected on the next frame and it would change the speed to the initial speed which causes the ball to get stuck if the ball does not exit the area within one clock cycle.

4.0 – Conclusion

The purpose of this laboratory project is to examine the functionalities of VGA and video-output system with real-time applications. The implementation of this based on VHDL-coding in the environment of Xilinx ISE CAD system. The pong game is successfully displayed on the screen, and the paddles can move up and down by controlling the switches on the FPGA board. The colour of the fields, paddles and the ball is successfully set up by setting the RGB value of either 256 (“11111111”) or 0 (“00000000”). The synchronization is successfully implemented by detecting the front and back porch and setting the pulse to zero. The colour of the ball is changed and reset when it hits a score. The ball reflects by an angle of 90 degrees whenever it hits the border or the paddles. It is verified that the clock cycles (time periods) of the parameters given in the lab instruction matches the actual size of the screen.

5.0 – Appendix

```
----------------------------------------------------
-- Company:
-- Engineer:
--
-- Create Date:        17:41:57 11/25/2019
-- Design Name:
-- Module Name:        project2 - Behavioral
-- Project Name:
-- Target Devices:
-- Tool versions:
-- Description:
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
```

```
-- Additional Comments:
--
----------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity project2 is
  Port ( CLK : in    STD_LOGIC;
                SW0: in STD_logic;
                        SW1: in STD_logic;
                        SW2: in std_logic;
                        SW3: in std_logic;
                HSYNC : out    STD_LOGIC;
                VSYNC : out    STD_LOGIC;
                        DAC_CLK: out STD_LOGIC;

                Bout: out STD_logic_vector(7 downto 0);
                        Gout: out STD_logic_vector(7 downto 0);
                        Rout: out STD_logic_vector(7 downto 0));
end project2;
```

```
architecture Behavioral of project2 is

    signal clk25 : std_logic := '0';

    constant HD : integer := 639;    --    639      Horizontal Display (640)
    constant HFP : integer := 16;              --      16      Right border (front porch)
    constant HSP : integer := 96;            --      96      Sync pulse (Retrace)
    constant HBP : integer := 48;             --      48      Left boarder (back porch)

    constant VD : integer := 479;     --    479      Vertical Display (480)
    constant VFP : integer := 10;               --      10      Right border (front porch)
    constant VSP : integer := 2;                        --       2      Sync pulse (Retrace)
    constant VBP : integer := 33;            --      33      Left boarder (back porch)

    signal hPos : integer := 0;
    signal vPos : integer := 0;

    signal ball_pos_h1          : integer := 320;
    signal ball_pos_v1          : integer:= 240;

    signal paddle1_pos_h1          : integer := 35;
    signal paddle1_pos_v1          : integer:= 300;
    signal paddle1_pos_length_h          : integer := 10;
    signal paddle1_pos_length_v          : integer := 100;

    signal paddle2_pos_h1          : integer := 600;
    signal paddle2_pos_v1          : integer:= 300;
```

```
        signal paddle2_pos_length_h          : integer := 10;
        signal paddle2_pos_length_v          : integer := 100;



        signal newframe: std_logic :='0';
        signal ballcolor: std_logic:= '0';
        signal ball_speed_h     :    integer range -3 to 3 := 1;
        signal ball_speed_v     :    integer range -3 to 3:= 1;

        constant ballsize: integer:= 10;
        constant topborder: integer:= 0;
        constant topborderlength: integer:= 20;
        constant botborder:integer:= 479;
        constant botborderlength: integer:= 20;
        constant leftborder:integer:= 0;
        constant leftborderlength: integer:=20;
        constant rightborder:integer:= 639;
        constant rightborderlength: integer := 20;

        constant hole_left_v1: integer:= 100;
        constant hole_left_length: integer:= 300;

        constant hole_right_v1: integer:= 100;
        constant hole_right_length: integer:= 300;
        constant strip:integer:= 300;

        signal videoOn : std_logic := '0';

begin
clk_div:process(CLK)
```

```
begin
        if(CLK'event and CLK = '1')then
                clk25 <= not clk25;
        end if;
end process;

Horizontal_position_counter:process(clk25)
begin

        if(clk25'event and clk25 = '1')then
                if (hPos = (HD + HFP + HSP + HBP)) then
                        newframe <= '0' ;
                        hPos <= 0;
                        if (vPos = (VD + VFP + VSP + VBP)) then
                                newframe <= '1';

                        end if;
                else
                        newframe <= '0' ;
                        hPos <= hPos + 1;

                end if;
        end if;
end process;

Vertical_position_counter:process(clk25, hPos)
begin

        if(clk25'event and clk25 = '1')then
                if(hPos = (HD + HFP + HSP + HBP))then
```

```
                if (vPos = (VD + VFP + VSP + VBP)) then
                        vPos <= 0;
                else
                        vPos <= vPos + 1;

                end if;
        end if;
    end if;
end process;

Horizontal_Synchronisation:process(clk25, hPos)
begin

    if(clk25'event and clk25 = '1')then
        if((hPos <= (HD + HFP)) OR (hPos > HD + HFP + HSP))then
                HSYNC <= '1';
        else
                HSYNC <= '0';
        end if;
    end if;
end process;

Vertical_Synchronisation:process(clk25, vPos)
begin

    if(clk25'event and clk25 = '1')then
        if((vPos <= (VD + VFP)) OR (vPos > VD + VFP + VSP))then
                VSYNC <= '1';
        else
                VSYNC <= '0';
```

```
            end if;
        end if;
end process;

video_on:process(clk25, hPos, vPos)
begin

        if(clk25'event and clk25 = '1')then
            if(hPos <= HD and vPos <= VD)then
                videoOn <= '1';
            else
                videoOn <= '0';
            end if;
        end if;
end process;

ball_move:process(clk25, newframe)
begin
        if(clk25'event and clk25 = '1' and newframe = '1')then

            ballcolor <= '0';
            if (ball_pos_v1< topborder+topborderlength) then
                ball_speed_v<= abs (ball_speed_v);
            end if;
            if (ball_pos_v1+ballsize> botborder-botborderlength) then
                ball_speed_v<= (-1)* abs(ball_speed_v);
            end if;

            if (ball_pos_h1< leftborder+leftborderlength) then
                ball_speed_h<= abs (ball_speed_h);
```

```
            end if;
            if (ball_pos_h1+ballsize> rightborder-rightborderlength) then
                ball_speed_h<= (-1)* abs(ball_speed_h);
            end if;
            if (ball_pos_h1 > paddle1_pos_h1 and ball_pos_h1 < paddle1_pos_h1+ paddle1_pos_length_h) and
(ball_pos_v1 > paddle1_pos_v1 and ball_pos_v1 < paddle1_pos_v1+ paddle1_pos_length_v) then
                ball_speed_h<=    abs(ball_speed_h);
            end if;
            if (ball_pos_h1+ballsize > paddle2_pos_h1 and ball_pos_h1+ballsize < paddle2_pos_h1+
paddle2_pos_length_h) and (ball_pos_v1 +ballsize> paddle2_pos_v1 and ball_pos_v1 < paddle2_pos_v1+
paddle2_pos_length_v) then
                ball_speed_h<= (-1)* abs(ball_speed_h);
            end if;
            ball_pos_h1 <=ball_pos_h1 +ball_speed_h;
            ball_pos_v1 <= ball_pos_v1+ball_speed_v;

            if(ball_pos_h1 <leftborder+leftborderlength and (ball_pos_v1>hole_left_v1 and ball_pos_v1
<hole_left_v1+hole_left_length)) then
                ball_pos_h1<=320;
                ball_pos_v1<=240;

            end if;
            if(ball_pos_h1+ballsize> rightborder-rightborderlength and (ball_pos_v1>hole_right_v1 and ball_pos_v1
<hole_right_v1+hole_right_length)) then
                ball_pos_h1<=320;
                ball_pos_v1<=240;
            end if;
            if(ball_pos_h1 <leftborder+leftborderlength+1 and (ball_pos_v1>hole_left_v1 and ball_pos_v1
<hole_left_v1+hole_left_length)) then
                ballcolor <= '1';
```

```
            end if;
            if(ball_pos_h1+ballsize> rightborder-rightborderlength-1 and (ball_pos_v1>hole_right_v1 and ball_pos_v1
<hole_right_v1+hole_right_length)) then
                ballcolor <= '1';
            end if;

        end if;
end process;

paddle_move:process(clk25, newframe)
begin

        if(clk25'event and clk25 = '1' and newframe = '1')then
            if (SW0 = '1' and    paddle1_pos_v1 > topborder+topborderlength) then
                paddle1_pos_v1 <= paddle1_pos_v1 - 1;
            end if;
            if (SW1 = '1' and    paddle1_pos_v1 +paddle1_pos_length_v< botborder-botborderlength) then
                paddle1_pos_v1 <= paddle1_pos_v1 + 1;
            end if;
            if (SW2 = '1' and    paddle2_pos_v1 > topborder+topborderlength) then
                paddle2_pos_v1 <= paddle2_pos_v1 - 1;
            end if;
            if (SW3 = '1' and    paddle2_pos_v1 +paddle2_pos_length_v< botborder-botborderlength) then
                paddle2_pos_v1 <= paddle2_pos_v1 + 1;
            end if;

        end if;
end process;
draw:process(clk25, hPos, vPos, videoOn)
begin
```

```
if(clk25'event and clk25 = '1') then
    if(videoOn = '1') then
        if(vpos < topborder+topborderlength or vpos> botborder-botborderlength) then
            Rout <= "11111111";
            Gout <= "11111111";
            Bout <= "11111111";
        elsif(hpos <leftborder+leftborderlength and (vpos>hole_left_v1 and vpos
<hole_left_v1+hole_left_length)) then
            Rout <= "00000000";
            Gout <= "11111111";
            Bout <= "00000000";
        elsif(hpos > rightborder-rightborderlength and (vpos>hole_right_v1 and vpos
<hole_right_v1+hole_right_length)) then
            Rout <= "00000000";
            Gout <= "11111111";
            Bout <= "00000000";
        elsif (hpos <leftborder+leftborderlength or hpos> rightborder-rightborderlength) then
            Rout <= "11111111";
            Gout <= "11111111";
            Bout <= "11111111";
        elsif ( (hpos > paddle1_pos_h1 and hpos < paddle1_pos_h1+ paddle1_pos_length_h) and (vpos >
paddle1_pos_v1 and vpos < paddle1_pos_v1+ paddle1_pos_length_v)) then
            Rout <= "11111111";
            Gout <= "00000000";
            Bout <= "00000000";
        elsif ( (hpos > paddle2_pos_h1 and hpos < paddle2_pos_h1+ paddle2_pos_length_h) and (vpos >
paddle2_pos_v1 and vpos < paddle2_pos_v1+ paddle2_pos_length_v)) then
            Rout <= "00000000";
            Gout <= "00000000";
```

```
                Bout <= "11111111";
            elsif ( (hpos >= ball_pos_h1 and hpos < ball_pos_h1 + ballsize) and (vpos >= ball_pos_v1 and vpos <
ball_pos_v1 + ballsize) ) then
                if (ballcolor = '0') then
                    Rout <= "11111111";
                    Gout <= "11111111";
                    Bout <= "11111111";
                else
                    Rout <= "11111111";
                    Gout <= "00000000";
                    Bout <= "00000000";
                end if;
            elsif( hpos > strip and hpos <strip +3) then
                Rout <= "11111111";
                Gout <= "11111111";
                Bout <= "00000000";
            else
                Rout <= "00000000";
                Gout <= "11111111";
                Bout <= "00000000";
            end if;
        else
                Rout <= "00000000";
                Gout <= "00000000";
                Bout <= "00000000";
        end if;
    end if;
end process;

DAC_CLK<=clk25;
```

end Behavioral;