2019-2020 Spring Semester

Bilkent University

EE-102 Term Project Report

Instructor: Cem Tekin

Student: İsmail Görkem Yeni

ID: 21802537

Project: “Catch the Ball VGA Game”

YouTube video link: https://youtu.be/9aPnN_RB4Z4

## Abstract

This project aims to design a VHDL game with Basys-3 FPGA board for understanding and learning the VHDL language and a creating a detailed project by it. Even though the world is trying to get use to the new normal, people still spend most of their time in their houses or closed and isolated areas. Therefore, this project is made up for these people to have some fun in these hard times and forget their problems and stress for a while. While doing this game, personally, I aimed to learn a lot of things about VHDL and finish this project as good as I can manage.

## Project’s Design Specification Plan

To be able to do such a VHDL project, I searched and examined a lot of projects from the past on this course. After some research, I found out that there were a lot of VHDL projects for the ping-pong game. However, I wanted to improve this game to a further stage which the bar below does not reflect the ball, it catches it. However, the game might seem a regular game, I tried to make it harder and more competitive by adding some details. Player has 3 lives at the beginning of the game, whenever player catches the ball with the cup the score increases and after two catches, the cup decreases in size till the score becomes 80 which can be interpreted as game ends at score 80. The players can continue to play as they wish and score will be counting up but size of the cup won’t change. Hence, the difficulty won’t change further.

## Design Methodology

Necessary tools for the project are listed below:

- Basys-3 FPGA board
- VGA cable
- Computer Monitor

The gameplay of the game is as the following. User can move the cup by using left and right buttons of the Basys-3 board and follow the remaining lives by looking at the LEDs on the board. Score will be shown on the seven-segment display. There is one switch that controls the reset and start of the game and another switch to restart the VGA display, it does not change the game’s states.

To be able to use the monitor screen I had to learn about Basys-3’s VGA Port, after some research [1], I managed to adjust my screen according to the monitor I have and Basys-3. Since Basys-3 has 100MHz clock frequency, I used monitor specifications as 800x600 and changed the regarding constants (See Appendix 2.A) in VGA sync module with 50 MHz Pixel

Frequency [2]. I used my previous lab project for making the seven-segment display for the score table, however, it was more difficult and took many hours because of synchronizing it to my main code ball animation part (See Appendix 3.A Ball animation). For random number generation, I used pseudo-number generator [3] since there is not any random function in VHDL as in Python (See appendix 4.A), and also synchronized it with my main code (See appendix 3.A Ball Animation).

To sum up, I used the following modules in this project:

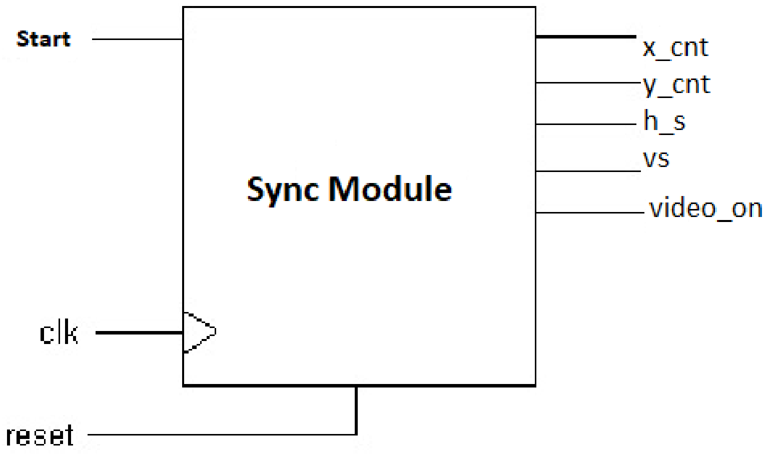
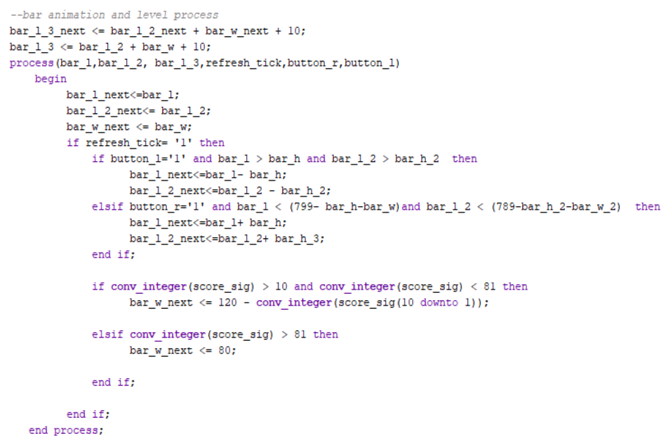


Figure 1. Sync Module Demonstration

<u>VGA sync</u>: It is the necessary module in terms of using the VGA port of the Basys-3 FPGA board. It divides the Basys's clock by two and creates a 50MHz clock frequency which is same with 800x600 50MHz pixel frequency. I adjusted its constants as I wanted to use (See Appendix 2.A).

<u>Image Generator:</u> In this module, the objects of the game and the colors are created. Moreover, the game's dynamics, its animation, score, reset and lives management are coded in this module (See appendix 3.A).

- Bar Animation: Inside the image generator module, I also control the bar animation and inside the bar animation part, I also check the proper circumstances for the level increase and downsizing the bar accordingly as shown below (Also See Appendix 3.A Bar Animation part).



```vhdl
--bar animation and level process
bar_l_3_next <= bar_l_2_next + bar_w_next + 10;
bar_l_3 <= bar_l_2 + bar_w + 10;
process(bar_l,bar_l_2, bar_l_3,refresh_tick,button_r,button_l)
    begin
         bar_l_next<=bar_l;
         bar_l_2_next<= bar_l_2;
         bar_w_next <= bar_w;
         if refresh_tick= '1' then
             if button_l='1' and bar_l > bar_h and bar_l_2 > bar_h_2  then
                   bar_l_next<=bar_l- bar_h;
                   bar_l_2_next<=bar_l_2 - bar_h_2;
             elsif button_r='1' and bar_l < (799- bar_h-bar_w)and bar_l_2 < (789-bar_h_2-bar_w_2)  then
                   bar_l_next<=bar_l+ bar_h;
                   bar_l_2_next<=bar_l_2+ bar_h_3;
             end if;

             if conv_integer(score_sig) > 10 and conv_integer(score_sig) < 81 then
                   bar_w_next <= 120 - conv_integer(score_sig(10 downto 1));

             elsif conv_integer(score_sig) > 81 then
                   bar_w_next <= 80;

             end if;

         end if;
   end process;
```

Figure 2. Bar Animation

- Ball Animation, Score and Lives Counting: Inside the image generator module, another part that generates game mechanics is ball animation part. In this part, I also gather the proper score output and lives_left output (See Appendix 3.A Ball Animation part).

PseudoRNG: This module generates Pseudo random numbers. I had some difficulties founding a functional example of this code however, managed to do after a lot of internet-surfing [3]. It is not actually random numbers but since we have to deal with 1s and 0s in VHDL coding language, this module gives us some meaningless 8-bit outputs which looks like randomly generated. This module is important for the ball to spawn between some particular pixels and enhance the difficulty of the game. I assigned the output of this module Q to x_y_in input of my image generator module in the top module (See appendix 4.A).

Clock Divider: This clock divider is used for creating a proper clock for the seven-segment display that human eyes can observe the score (See appendix 5.C).

Segment Decoder: As in our previous lab4 project, this module is important for display of the numbers on the display (See appendix 5.A).

Segment Selector: Segment selector is used for turning the right diode of the seven-segment according to the number(score) wanted to be displayed (See appendix 5.B).

Top Module: It is the module where I connected all the other modules in. Moreover, I assigned the LEDs for the lives here which I thought would be more efficient (See appendix 1.A).

Constraints: This module is essential for using the VGA port, LEDs and switches of the board, without this module, connection with the computer to the board would not be provided (See appendix 6.A).

## Results

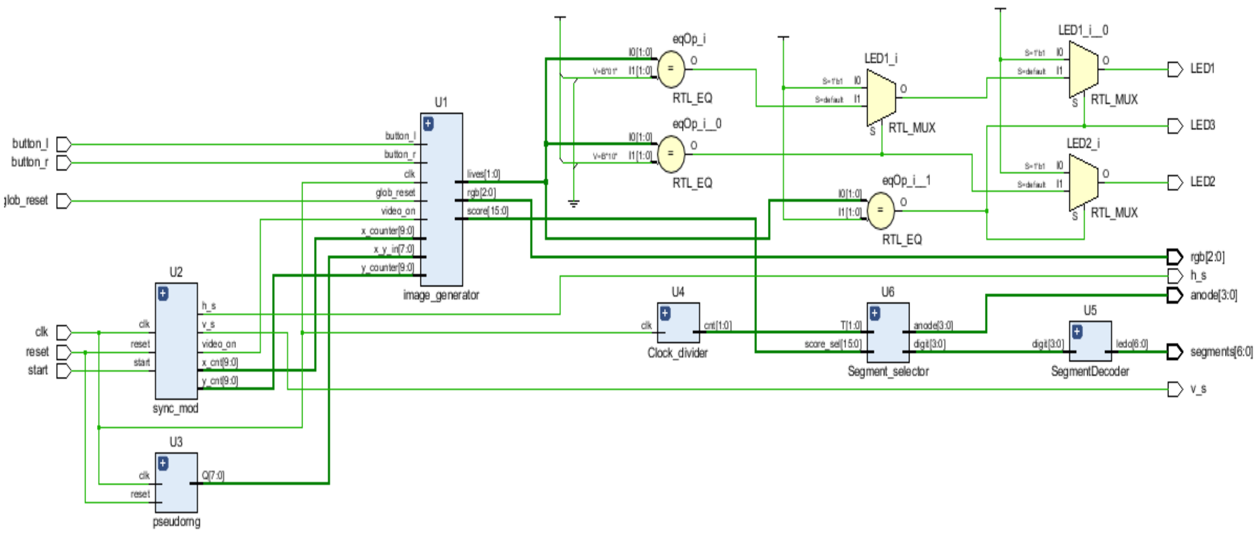


Figure 3.  RTL Schematic

RTL schematics is crucial for the lucidity of the project, understanding the project and the connections between modules is possible through examining it. Since there are many modules and connections it looks a little bit complicated and modules alone are very complicated consisting hundreds of logic gates. Image generator module can be seen as a good example of it:

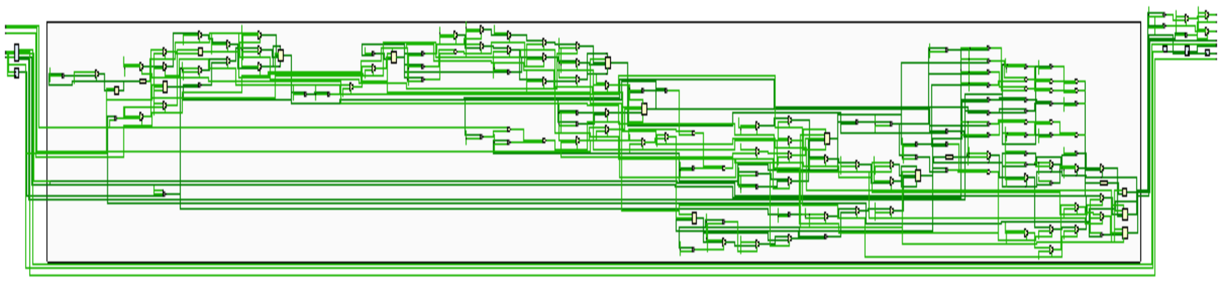


Figure 4. RTL Schematic of Image Generator

Some example situations from the game and the board.

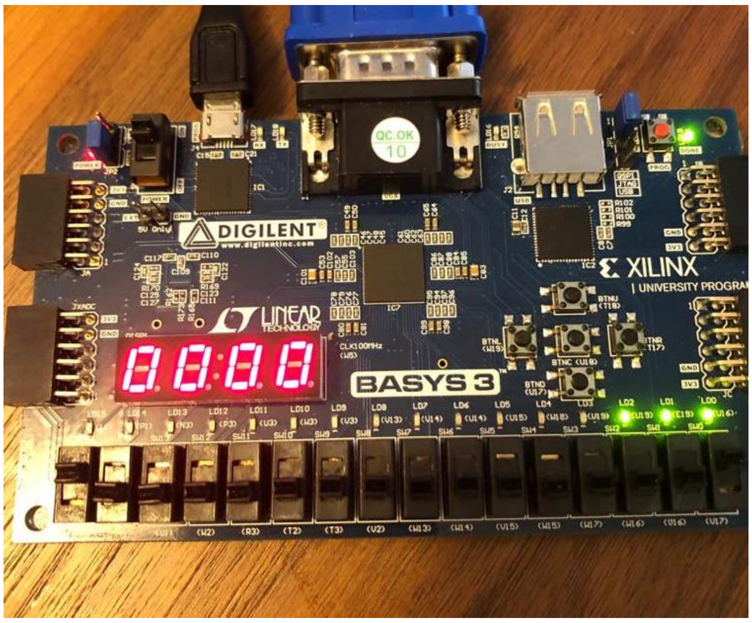


Figure 5. Initial Board State

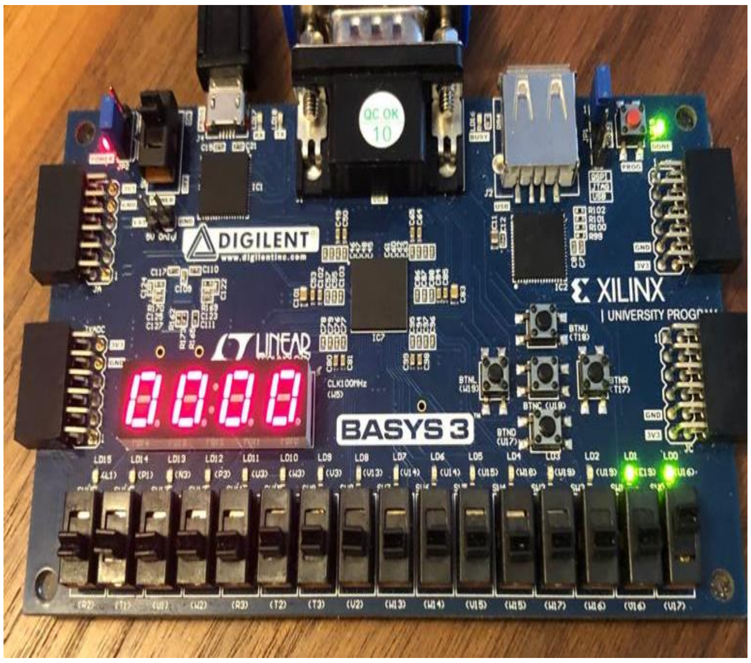


Figure 6. After player lose 1 life.

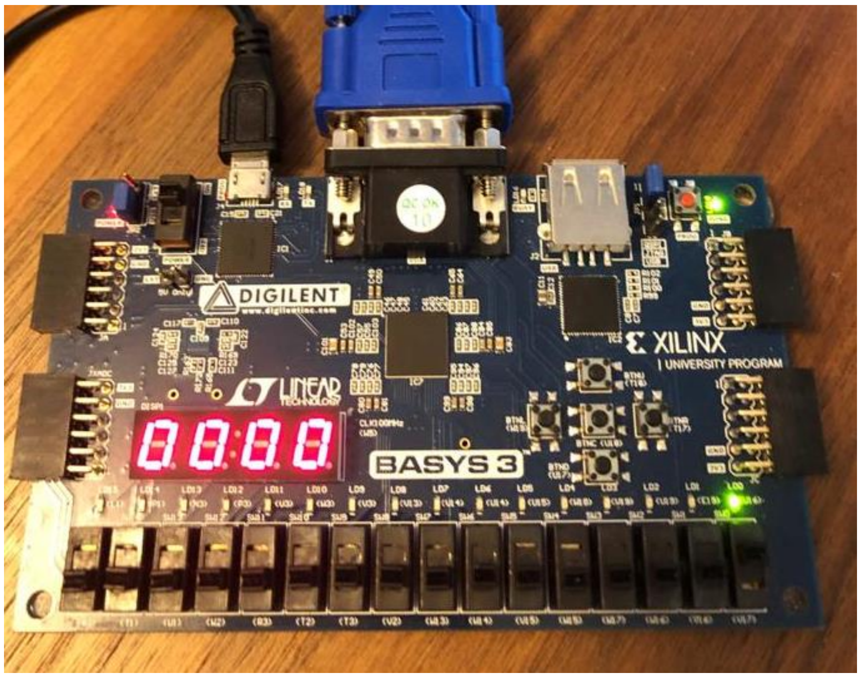


Figure 7. Player has one more life left.

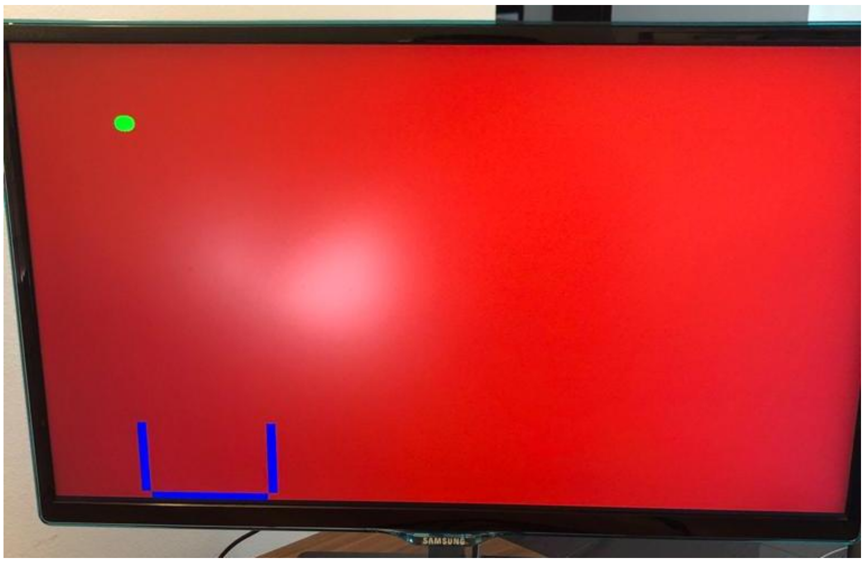


Figure 8. The initial and final positions of the game objects.

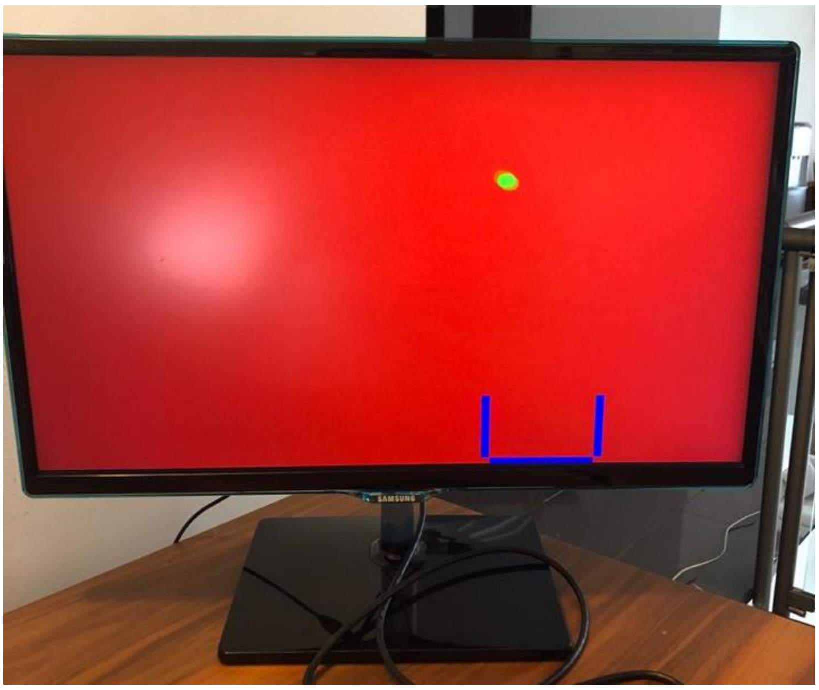


Figure 9. Randomly spawned ball is falling.

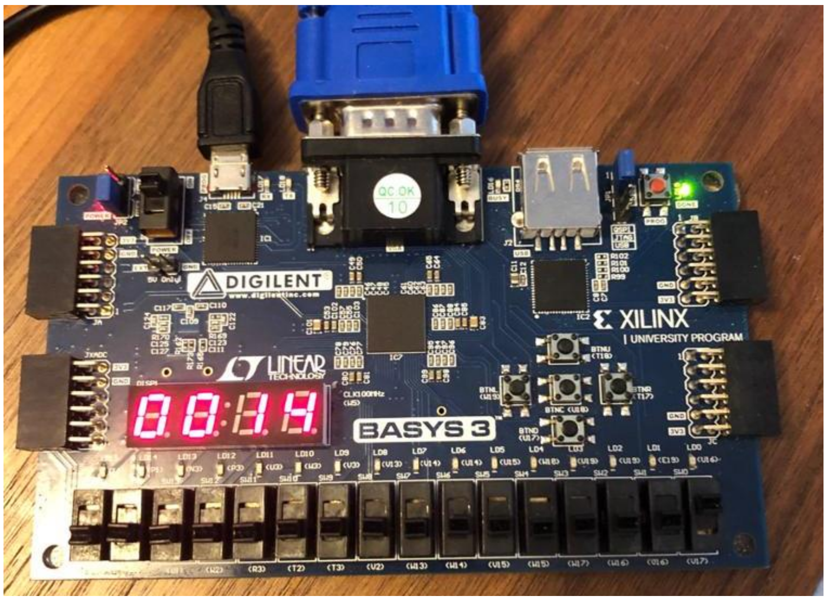


Figure 10. The game is over, player lost and scored 14.

Also it can be said by looking at the utilization table, this game is easy to implement on Basys-3 and won't cause any damage.

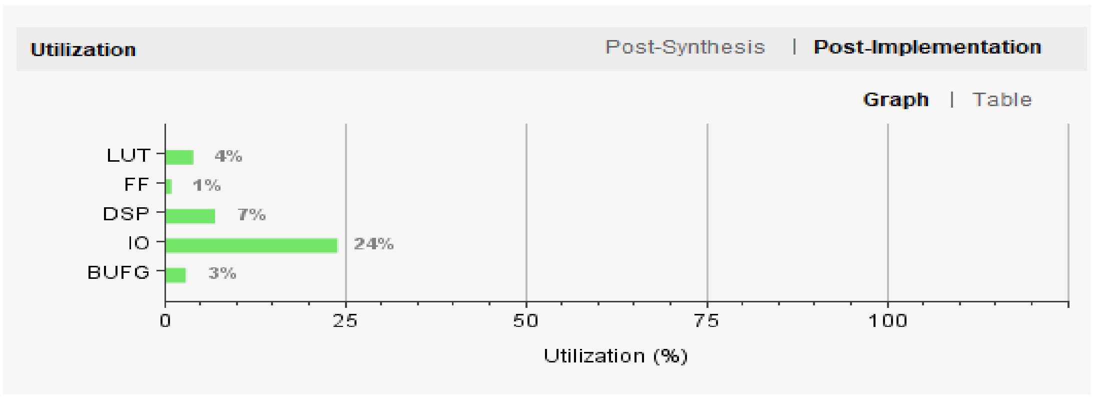


Figure 11. Utilization Table

## Conclusion

While doing this term project, I learned a lot of things about VHDL and VGA. Initially, I learnt how VGA monitor works and how does it communicate with Basys-3. I learned how to synchronize my system according to the Basys and VGA monitor. Even though I know there are many different colors that can be appliable via Basys to monitor, I only used the main colors; red, blue and green since my game is easy to perform only with these colors.

VHDL showed me that it does not matter how different the coding language is such as VHDL, you can finish a project. It took a lot of hours for me to get rid of a basic problem. For me, it was resetting the game after lives left is 0, even though I figured out what I have to do for a global reset which reset everything, my lives_left = '0' condition didn't workout. Nevertheless, I understood that I forgot one basic but most important thing which is Clk synchronization. VHDL wants you to care itself with four eyes and fully awake. But I wished that there was no such a thing called multi driven error.

## References

1- "Basys 3 FPGA Board Reference Manual". *Digilent.* Retrieved 15.06.2020, from https://reference.digilentinc.com/_media/basys3:basys3_rm.pdf.

2- "VESA Signal 800 x 600 @ 72 Hz timing". 2008. *TinyVGA.com.* Retrieved 15.06.2020 from http://tinyvga.com/vga-timing/800x600@72Hz

3- Gsm. "Pseudo Random Number Generator using LFSR in VHDL", *stackoverflow.com.* March 29, 2017. Retrieved 15.06.2020 from https://stackoverflow.com/questions/43081067/pseudo-random-number-generator-using-lfsr-in-vhdl

# Appendix

## 1.A: Top Module

```
library IEEE;

use IEEE.STD_LOGIC_1164.ALL;

use IEEE.STD_LOGIC_ARITH.ALL;

use IEEE.STD_LOGIC_UNSIGNED.ALL;


entity top_module is

   Port (     clk : in  STD_LOGIC;

              start      : in  STD_LOGIC;

              glob_reset : in std_logic;

              reset       : in  STD_LOGIC;

              button_l   : IN std_logic;

              button_r   : IN std_logic;

              rgb           : out  STD_LOGIC_VECTOR (2 downto 0);

              h_s          : out  STD_LOGIC;

              v_s           : out  STD_LOGIC;

              segments : out std_logic_vector(6 downto 0);

              anode : out std_logic_vector(3 downto 0);

              LED1: out std_logic;

              LED2: out std_logic;

              LED3: out std_logic);

    end top_module;


architecture Behavioral of top_module is


COMPONENT image_generator
```

```
  PORT(       clk : IN std_logic;
              glob_reset: in std_logic;
              x_counter    : IN std_logic_vector(9 downto 0);
              button_l     : IN std_logic;
              button_r     : IN std_logic;
              y_counter     : IN std_logic_vector(9 downto 0);
              video_on   : IN std_logic;
              rgb             : OUT std_logic_vector(2 downto 0);
              x_y_in :     in std_logic_vector(7 downto 0);
              score:      out std_logic_vector(15 downto 0);
              lives: out std_logic_vector(1 downto 0));
 END COMPONENT;

COMPONENT sync_mod
PORT(       clk  : IN std_logic;
            reset       : IN std_logic;
            start      : IN std_logic;
            y_cnt   : OUT std_logic_vector(9 downto 0);
            x_cnt   : OUT std_logic_vector(9 downto 0);
            h_s          : OUT std_logic;
            v_s           : OUT std_logic;
            video_on   : OUT std_logic );
END COMPONENT;

component pseudorng
port ( clk : in STD_LOGIC;
      reset : in STD_LOGIC;
      Q : out STD_LOGIC_VECTOR (7 downto 0)
      );
```

```
end component;

component Clock_divider is
   Port ( clk : in STD_LOGIC;
          cnt : out STD_LOGIC_vector (1 downto 0));
end component;

component SegmentDecoder is
   Port ( digit : in STD_LOGIC_VECTOR (3 downto 0);
          ledo : out STD_LOGIC_VECTOR (6 downto 0));
end component;

component Segment_selector is
   Port ( score_sel : in STD_LOGIC_VECTOR (15 downto 0);
          T : in STD_LOGIC_VECTOR (1 downto 0);
          anode : out STD_LOGIC_VECTOR(3 downto 0);
          digit : out STD_LOGIC_VECTOR (3 downto 0));
end component;

signal x,y:std_logic_vector(9 downto 0);
signal video:std_logic;
signal x_y_sig : std_logic_vector(7 downto 0);
signal T : std_logic_vector(1 downto 0);
signal digit: std_logic_vector ( 3 downto 0);
signal scoresig: std_logic_vector (15 downto 0);
signal lives: std_logic_vector(1 downto 0);


begin
```

```
U1: image_generator PORT MAP( clk =>clk ,  x_counter => x, button_l =>button_l  , button_r
=> button_r, y_counter => y,
                                        video_on =>video , rgb => rgb, x_y_in => x_y_sig, score => scoresig,
lives => lives, glob_reset => glob_reset );


U2: sync_mod PORT MAP( clk => clk, reset => reset, start => start, y_cnt => y, x_cnt =>x , h_s
=> h_s ,
                                        v_s => v_s, video_on =>video );


U3: pseudorng PORT MAP (clk => clk, reset => reset, Q => x_y_sig);


U4:Clock_divider PORT MAP( clk => clk, cnt => T);


U5: SegmentDecoder PORT MAP (digit => digit, ledo => segments);


U6:Segment_selector PORT MAP( score_sel => scoresig, T => T, anode => anode, digit =>
digit);



lives_with_leds: process(lives)
begin
   if lives = "11" then
      LED1 <= '1';
      LED2 <= '1';
      LED3 <= '1';
   elsif lives = "10" then
      LED1 <= '1';
      LED2 <= '1';
      LED3 <= '0';
   ELSIF lives = "01" then
      LED1 <= '1';
```

```
        LED2 <= '0';
        LED3 <= '0';
    ELSE
        LED1 <= '0';
        LED2 <= '0';
        LED3 <= '0';
    end if;
end process;

end Behavioral;
```

## 2A. SYNC Module

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity sync_mod is
      Port ( clk        : in  STD_LOGIC;
             reset      : in  STD_LOGIC;
             start     : in  STD_LOGIC;
             y_cnt : out  STD_LOGIC_VECTOR (9 downto 0);
             x_cnt  : out  STD_LOGIC_VECTOR (9 downto 0);
             h_s          : out  STD_LOGIC;
             v_s          : out  STD_LOGIC;
            video_on : out  STD_LOGIC);
end sync_mod;

architecture Behavioral of sync_mod is
```

```
-- specifications
constant HGO:integer:=800;--Yatay görüntü
constant HFP:integer:=56;--Horizontal Front Porch
constant HBP:integer:=64;--Horizontal Back Porch
constant HGK:integer:=120;--Yatay geri kayma
constant VGO:integer:=600;--Düşey görüntü
constant VFP:integer:=37;--Vertical Front Porch
constant VBP:integer:=23;--Vertical Back Porch
constant VGK:integer:=6;--Düşey geri kayma
--sync cnters
signal count_h,count_h_next: integer range 0 to 1039;
signal count_v,count_v_next: integer range 0 to 665;
--mod 2 counter
signal counter_mod2,counter_mod2_next: std_logic:='0';
--state sigs
signal h_end, v_end:std_logic:='0';
--out signals -- buffers
signal hs_buffer,hs_buffer_next:std_logic:='0';
signal vs_buffer,vs_buffer_next:std_logic:='0';
--pixel counters
signal x_cn, x_cn_next:integer range 0 to 900;
signal y_cn, y_cn_next:integer range 0 to 900;
--video_on_of
signal video:std_logic;
begin
--register
  process(clk,reset,start)
     begin
        if reset ='1' then
```

```
            count_h<=0;

            count_v<=0;

            hs_buffer<='0';

            hs_buffer<='0';

            counter_mod2<='0';

         elsif clk'event and clk='1' and start = '1' then

            count_h<=count_h_next;

            count_v<=count_v_next;

             x_cn<=x_cn_next;

             y_cn<=y_cn_next;

             hs_buffer<=hs_buffer_next;

             vs_buffer<=vs_buffer_next;

             counter_mod2<=counter_mod2_next;

        end if;

end process;

--video on/off

      video <= '1' when  (count_v >= VBP) and (count_v < VBP + VGO) and (count_h >=HBP)
and (count_h < HBP + HGO)

                          else  '0';


--mod 2 cnter

      counter_mod2_next<=not counter_mod2;

-- horizontal cnt end

      h_end<= '1' when count_h=1039 else --(HGO+HFP+HBP+HGK-1)

                       '0';

-- vertical cnt end

     v_end<= '1' when count_v=665 else --(VGO+VFP+VBP+VGK-1)

                    '0';

-- horizontal counter

process(count_h,counter_mod2,h_end)
```

```
    begin
       count_h_next<=count_h;
       if  counter_mod2= '1' then
           if h_end='1' then
               count_h_next<=0;
           else
               count_h_next<=count_h+1;
          end if;
     end if;
  end process;

-- vertical counter
process(count_v,counter_mod2,h_end,v_end)
   begin
      count_v_next <= count_v;
      if  counter_mod2= '1' and h_end='1'  then
         if v_end='1' then
             count_v_next<=0;
         else
              count_v_next<=count_v+1;
         end if;
       end if;
   end process;

--pixel x counter
process(x_cn,counter_mod2,h_end,video)
    begin
       x_cn_next<=x_cn;
       if video = '1' then
```

```
          if  counter_mod2= '1' then
              if x_cn= 799 then
                  x_cn_next<=0;
              else
                 x_cn_next<=x_cn + 1;
            end if;
        end if;
    else
       x_cn_next<=0;
   end if;
end process;

--pixel y scounter
process(y_cn,counter_mod2,h_end,count_v)
    begin
       y_cn_next<=y_cn;
       if  counter_mod2= '1' and h_end='1' then
          if count_v >22 and count_v <622  then
              y_cn_next<=y_cn + 1;
        else
            y_cn_next<=0;
        end if;
     end if;
end process;

--buffer
hs_buffer_next<= '1' when count_h < 920 else --(HBP+HGO+HFP)
                              '0';
vs_buffer_next<='1' when count_v < 660 else --(VBP+VGO+VFP)
```

```
                      '0';

 --outs
y_cnt <= conv_std_logic_vector(y_cn,10);
x_cnt <= conv_std_logic_vector(x_cn,10);
h_s<= hs_buffer;
v_s<= vs_buffer;
video_on<=video;

end Behavioral;
```

## 3.A: Image Generator

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity image_generator is
    Port ( clk            : in  STD_LOGIC;
            glob_reset       : in std_logic; -- also start-stop input from user
            x_counter    : in  STD_LOGIC_VECTOR(9 downto 0);
            button_l    :in STD_LOGIC;
            button_r    :in STD_LOGIC;
            y_counter    : in STD_LOGIC_VECTOR(9 downto 0);
            video_on  : in  STD_LOGIC;
            rgb       : out  STD_LOGIC_VECTOR(2 downto 0);
            x_y_in :     in std_logic_vector(7 downto 0); -- random number input from
pseudorng module
            score: out std_logic_vector(15 downto 0);
            lives: out std_logic_vector(1 downto 0));
```

```
 end image_generator;

architecture Behavioral of image_generator is

--bottom bar
signal bar_l,bar_l_next:integer :=100; --bar's x
constant bar_t:integer :=590;--bar's y
constant bar_k:integer :=10;--barın thickness
signal bar_w, bar_w_next:integer:=120;--bar width
constant bar_h:integer:=7;--bar speed
signal bar_on:std_logic;
signal rgb_bar:std_logic_vector(2 downto 0);

--left bar
signal bar_l_2,bar_l_2_next:integer :=90; --bar's x
constant bar_t_2:integer :=490;--bar's y
constant bar_k_2:integer :=100;--bar height
constant bar_w_2:integer:=10;--bar width
constant bar_h_2:integer:=7;--bar speed
signal bar_on_2:std_logic;
signal rgb_bar_2:std_logic_vector(2 downto 0);
--right bar
signal bar_l_3,bar_l_3_next:integer :=220; --bar's x
constant bar_t_3:integer :=490;--bar's y
constant bar_k_3:integer :=100;--barın thickness
constant bar_w_3:integer:=10;--bar widht
constant bar_h_3:integer:=7;--bar's speed
signal bar_on_3:std_logic;
signal rgb_bar_3:std_logic_vector(2 downto 0);

--ball
signal ball_l,ball_l_next:integer :=100;--ball x
signal ball_t,ball_t_next:integer :=100; --ball y
signal ball_r:integer :=10; -- ball radius
constant x_v,y_v:integer:=2;-- ball's x-y velocity
signal ball_on:std_logic;
signal rgb_ball:std_logic_vector(2 downto 0);

--refresh(1/72)
signal refresh_reg,refresh_next:integer;
constant refresh_constant:integer:=691666;
signal refresh_tick:std_logic;

--top animasyon
signal xv_reg,xv_next:integer:=2;--horizontal speed variable
signal yv_reg,yv_next:integer:=2;--vertical speed variable

--x,y pixel signal
signal x,y:integer range 0 to 810;

--mux
signal mux:std_logic_vector(4 downto 0);

--score
signal score_reg, score_next: std_logic_vector(15 downto 0);
signal score_sig, score_sig_next : std_logic_vector(10 downto 0);
-- lives
```

```
signal  lives_reg, lives_left_next: std_logic_vector(1 downto 0) := "11";
-- if lives left = 0
signal lose_state : std_logic := '0';

--buffer
signal rgb_reg,rgb_next:std_logic_vector(2 downto 0);

begin

--x,y pixel marker
x <=conv_integer(x_counter);
y <=conv_integer(y_counter );

--refresh time
process(clk)
   begin
      if clk'event and clk='1' then
         refresh_reg<=refresh_next;
      end if;
   end process;

refresh_next<= 0 when refresh_reg= refresh_constant else
                    refresh_reg+1;
refresh_tick<= '1' when refresh_reg = 0 else '0';

--register
process(clk)
   begin
```

```
if clk'event and clk='1' then
    if glob_reset = '1' then


          ball_l <= 100;
          ball_t <= 100;
          bar_l <= 100;
          bar_l_2 <= 90;
         -- bar_l_3 <= 220;
          bar_w <= 120;
          score_sig <= "00000000000";
          score_reg <= "0000000000000000";
          lives_reg <= "11";
    else
        if lose_state = '0' then
              ball_l<=ball_l_next;
              ball_t<=ball_t_next;
              xv_reg<=xv_next;
              yv_reg<=yv_next;
              bar_l<=bar_l_next;
              bar_w <= bar_w_next;
              bar_l_2 <= bar_l_2_next;
             -- bar_l_3 <= bar_l_3_next;
              lives_reg <= lives_left_next;
              score_reg <= score_next;
              score_sig <= score_sig_next;
        else


              ball_l <= 100;
```

```
                ball_t <= 100;
                bar_l <= 100;
                bar_l_2 <= 90;

               -- bar_l_3 <= 220;
                bar_w <= 120;
                score_sig <= score_sig_next;
                score_reg <= score_next;
            end if;
         end if;
   end if;
end process;



--bar animation and level process
bar_l_3_next <= bar_l_2_next + bar_w_next + 10;
bar_l_3 <= bar_l_2 + bar_w + 10;
process(bar_l,bar_l_2, bar_l_3,refresh_tick,button_r,button_l)
   begin
      bar_l_next<=bar_l;
      bar_l_2_next<= bar_l_2;
      bar_w_next <= bar_w;
      if refresh_tick= '1' then
         if button_l='1' and bar_l > bar_h and bar_l_2 > bar_h_2  then
            bar_l_next<=bar_l- bar_h;
            bar_l_2_next<=bar_l_2 - bar_h_2;
         elsif button_r='1' and bar_l < (799- bar_h-bar_w)and bar_l_2 < (789-bar_h_2-
bar_w_2)  then
            bar_l_next<=bar_l+ bar_h;
            bar_l_2_next<=bar_l_2+ bar_h_3;
            end if;

            if conv_integer(score_sig) > 10 and conv_integer(score_sig) < 81 then
                 bar_w_next <= 120 - conv_integer(score_sig(10 downto 1));

            elsif conv_integer(score_sig) > 81 then
                 bar_w_next <= 80;

            end if;

          end if;
     end process;

--ball, score and,lives_left animation
process(refresh_tick,ball_l,ball_t,xv_reg,yv_reg)
     begin
          ball_l_next <=ball_l;
          ball_t_next <=ball_t;
          xv_next<=xv_reg;
          yv_next<=yv_reg;
          score_next <= score_reg;
          score_sig_next <= score_sig;
          lives_left_next <= lives_reg;
          if refresh_tick = '1' then

               if ball_l < 10 then --when ball touched the left
                    xv_next<= x_v;
               elsif ball_l> 760 then
                    xv_next<= -x_v ;        --when ball touched the right
```

```
end if;
if ball_t+(ball_r*2) > 520 then
  if ball_l > (bar_l_2+bar_w_2) and ball_l < (bar_l_3-ball_r*2) then


    ball_t_next <= conv_integer(x_y_in)/2;
    ball_l_next <= (conv_integer(x_y_in)/2)*(5);
    score_sig_next <= score_sig + 1;
    score_next(3 downto 0) <= score_reg(3 downto 0) + 1; -- ball inside the cup
      if conv_integer(score_reg(3 downto 0)) = 9 then
        score_next(3 downto 0) <= "0000";
        score_next(7 downto 4) <= score_reg(7 downto 4) +1;
        if conv_integer(score_reg(7 downto 4)) = 9 then
          score_next(7 downto 4) <= "0000";
          score_next(11 downto 8) <= score_reg(11 downto 8) +1;
          if conv_integer(score_reg(11 downto 8)) = 9 then
            score_next(11 downto 8) <= "0000";
            score_next(15 downto 12) <= score_reg(15 downto 12) +1;
          end if;
        end if;
      end if;


  else
    ball_t_next <= 50;
    ball_l_next <= 400;
    lives_left_next <= (lives_reg - 1);
  end if;


else
  ball_l_next <=ball_l +xv_next;
```

```
                ball_t_next <=ball_t+yv_next;
              end if;

              if glob_reset = '1' then
                lose_state <= '0';
              elsif lives_reg = "00" then
                lose_state <= '1';
              end if;

        end if;

    end process;
```

```
--bar object
bar_on <= '1' when x > bar_l and x < (bar_l+bar_w) and y> bar_t and y < (bar_t+ bar_k) else
'0';
rgb_bar<="001";--blue
--vertical bar 1
bar_on_2 <= '1' when x > bar_l_2 and x < (bar_l_2+bar_w_2) and y> bar_t_2 and y <
(bar_t_2+ bar_k_2) else '0';
rgb_bar_2<="001";
--vertical bar2
bar_on_3 <= '1' when x > bar_l_3 and x < (bar_l_3+bar_w_3) and y> bar_t_3 and y <
(bar_t_3+ bar_k_3) else '0';
rgb_bar_3<="001";
```

```
-- ball object
ball_on <= '1' when (x - ball_l) * (x - ball_l) + (y - ball_t) * (y - ball_t) <= ball_r * ball_r else
        '0';

rgb_ball<="010";  --green

--buffer
process(clk)
   begin
      if clk'event and clk='1' then
         rgb_reg<=rgb_next;
      end if;
   end process;

--mux
mux <= video_on & bar_on & ball_on & bar_on_2 & bar_on_3;
with mux select
                 rgb_next <="100"  when "10000",--red, background
                 rgb_bar       when "11000",
                 rgb_bar       when "11100",
                 rgb_bar_2       when "10010",
                 rgb_bar_2       when "10110",
                 rgb_bar_3       when "10001",
                 rgb_bar_3       when "10101",
                 rgb_ball        when "10100",
                 "000"           when others;
--output
rgb<=rgb_reg;
score <= score_reg;
```

```
lives <= lives_reg;
end Behavioral;
```

## 4.A: Pseudo Random Number Generator

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

entity pseudorng is
Port ( clk : in STD_LOGIC;
       reset : in STD_LOGIC;
       Q : out STD_LOGIC_VECTOR (7 downto 0));

end pseudorng;

architecture Behavioral of pseudorng is



signal Qt: STD_LOGIC_VECTOR(7 downto 0) := x"01";


begin


PROCESS(clk)
variable tmp : STD_LOGIC := '0';
BEGIN
```

```
IF rising_edge(clk) THEN
  IF (reset='1') THEN
    Qt <= x"01";
  ELSE
    tmp := Qt(4) XOR Qt(3) XOR Qt(2) XOR Qt(0);
    Qt <= tmp & Qt(7 downto 1);
  END IF;
END IF;
END PROCESS;
Q <= Qt;
end Behavioral;
```

## 5.A: Segment Decoder

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



entity SegmentDecoder is
   Port ( digit : in STD_LOGIC_VECTOR (3 downto 0);
          ledo : out STD_LOGIC_VECTOR (6 downto 0));
end SegmentDecoder;

architecture Behavioral of SegmentDecoder is

begin

Process(digit)
```

```
begin
        case digit is
            when "0000" => ledo <= "0000001"; --0
            when "0001" => ledo <= "1001111"; --"1"
            when "0010" => ledo <= "0010010"; --"2"
            when "0011" => ledo <= "0000110"; --"3"
            when "0100" => ledo <= "1001100"; --"4"
            when "0101" => ledo <= "0100100"; --"5"
            when "0110" => ledo <= "0100000"; --"6"
            when "0111" => ledo <= "0001111"; --"7"
            when "1000" => ledo <= "0000000"; --"8"
            when "1001" => ledo <= "0000100"; --"9"
            when others => ledo <= "0111000"; --F
--          when "1010" => ledo <= "0001000"; --A
--          when "1011" => ledo <= "1100000"; --B
--          when "1100" => ledo <= "0110001"; --C
--          when "1101" => ledo <= "1000010"; --D
--          when "1110" => ledo <= "0110000"; --E
--          when "1111" => ledo <= "0111000"; --F

        END case;
end process;
end Behavioral;
```

## 5.B: Segment Selector

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

```
entity Segment_selector is
    Port ( score_sel : in STD_LOGIC_VECTOR (15 downto 0);
           T : in STD_LOGIC_VECTOR (1 downto 0);
           anode : out STD_LOGIC_VECTOR(3 downto 0);
           digit : out STD_LOGIC_VECTOR (3 downto 0));
end Segment_selector;

architecture Behavioral of Segment_selector is


begin
process(T, score_sel)
Begin
    case T is
        when "00" =>
            digit <= score_sel (3 Downto 0);
            anode <= "1110";
        when "01" =>
            digit <= score_sel (7 Downto 4);
            anode <= "1101";
        when "10" =>
            digit <= score_sel (11 Downto 8);
            anode <= "1011";
        when others =>
            digit <= score_sel (15 Downto 12);
            anode <= "0111";


     end case;
end process;
end Behavioral;
```

## 5.C: Clock Divider (For seven-segment)

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

entity Clock_divider is
   Port ( clk : in STD_LOGIC;
          cnt : out STD_LOGIC_vector (1 downto 0));
end Clock_divider;

architecture Behavioral of Clock_divider is
   signal counter : integer := 0;
   signal temp : std_logic_vector (1 downto 0);

begin
process(clk)
begin
   if rising_edge(clk) then
      if counter < 250000 then
         counter <= counter + 1;
      else
         counter <= 0;
         temp <= temp + "01";
      end if;
   end if;
end process;
cnt <= temp;
```

```
end Behavioral;
```

## 6.A: Constraints

```
## Clock signal
set_property PACKAGE_PIN W5 [get_ports clk]

        set_property IOSTANDARD LVCMOS33 [get_ports clk]


## Switches
set_property PACKAGE_PIN V17 [get_ports {start}]
        set_property IOSTANDARD LVCMOS33 [get_ports {start}]
set_property PACKAGE_PIN V16 [get_ports {reset}]
        set_property IOSTANDARD LVCMOS33 [get_ports {reset}]
set_property PACKAGE_PIN R2 [get_ports {glob_reset}]
        set_property IOSTANDARD LVCMOS33 [get_ports {glob_reset}]
## LEDs
set_property PACKAGE_PIN U16 [get_ports {LED1}]
        set_property IOSTANDARD LVCMOS33 [get_ports {LED1}]
set_property PACKAGE_PIN E19 [get_ports {LED2}]
        set_property IOSTANDARD LVCMOS33 [get_ports {LED2}]
set_property PACKAGE_PIN U19 [get_ports {LED3}]
        set_property IOSTANDARD LVCMOS33 [get_ports {LED3}]


##7 segment display
set_property PACKAGE_PIN W7 [get_ports {segments[6]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {segments[6]}]
set_property PACKAGE_PIN W6 [get_ports {segments[5]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {segments[5]}]
set_property PACKAGE_PIN U8 [get_ports {segments[4]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {segments[4]}]
set_property PACKAGE_PIN V8 [get_ports {segments[3]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {segments[3]}]
set_property PACKAGE_PIN U5 [get_ports {segments[2]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {segments[2]}]
set_property PACKAGE_PIN V5 [get_ports {segments[1]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {segments[1]}]
set_property PACKAGE_PIN U7 [get_ports {segments[0]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {segments[0]}]

#set_property PACKAGE_PIN V7 [get_ports dp]

        #set_property IOSTANDARD LVCMOS33 [get_ports dp]

set_property PACKAGE_PIN U2 [get_ports {anode[0]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {anode[0]}]
set_property PACKAGE_PIN U4 [get_ports {anode[1]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {anode[1]}]
set_property PACKAGE_PIN V4 [get_ports {anode[2]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {anode[2]}]
set_property PACKAGE_PIN W4 [get_ports {anode[3]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {anode[3]}]

##Buttons
set_property PACKAGE_PIN W19 [get_ports button_l]

        set_property IOSTANDARD LVCMOS33 [get_ports button_l]
set_property PACKAGE_PIN T17 [get_ports button_r]

        set_property IOSTANDARD LVCMOS33 [get_ports button_r]
```

```
##VGA Connector
set_property PACKAGE_PIN G19 [get_ports {rgb[2]}]
          set_property IOSTANDARD LVCMOS33 [get_ports {rgb[2]}]
set_property PACKAGE_PIN H19 [get_ports {rgb[2]}]
          set_property IOSTANDARD LVCMOS33 [get_ports {rgb[2]}]
set_property PACKAGE_PIN J19 [get_ports {rgb[2]}]
          set_property IOSTANDARD LVCMOS33 [get_ports {rgb[2]}]
set_property PACKAGE_PIN N19 [get_ports {rgb[2]}]
          set_property IOSTANDARD LVCMOS33 [get_ports {rgb[2]}]
set_property PACKAGE_PIN N18 [get_ports {rgb[0]}]
          set_property IOSTANDARD LVCMOS33 [get_ports {rgb[0]}]
set_property PACKAGE_PIN L18 [get_ports {rgb[0]}]
          set_property IOSTANDARD LVCMOS33 [get_ports {rgb[0]}]
set_property PACKAGE_PIN K18 [get_ports {rgb[0]}]
          set_property IOSTANDARD LVCMOS33 [get_ports {rgb[0]}]
set_property PACKAGE_PIN J18 [get_ports {rgb[0]}]
          set_property IOSTANDARD LVCMOS33 [get_ports {rgb[0]}]
set_property PACKAGE_PIN J17 [get_ports {rgb[1]}]
          set_property IOSTANDARD LVCMOS33 [get_ports {rgb[1]}]
set_property PACKAGE_PIN H17 [get_ports {rgb[1]}]
          set_property IOSTANDARD LVCMOS33 [get_ports {rgb[1]}]
set_property PACKAGE_PIN G17 [get_ports {rgb[1]}]
          set_property IOSTANDARD LVCMOS33 [get_ports {rgb[1]}]
set_property PACKAGE_PIN D17 [get_ports {rgb[1]}]
          set_property IOSTANDARD LVCMOS33 [get_ports {rgb[1]}]
set_property PACKAGE_PIN P19 [get_ports h_s]
          set_property IOSTANDARD LVCMOS33 [get_ports h_s]
set_property PACKAGE_PIN R19 [get_ports v_s]
        set_property IOSTANDARD LVCMOS33 [get_ports v_s]
```