University of Essex - CESS

Assignment 2
“Basketball” Video Game

CE339 – High level digital design

Yu Wenlu 1909135
wy19403@essex.ac.uk
Word count: 1865

Content

Abstract

This project show cases the successful design and implementation of a basketball game on the Basys3 board using VHDL language. The game consists of seven modules, each serving a specific purpose. The Top_Module acts as the highest module, defining the input and output ports of VHDL. The Clock_Divider module converts the 100MHz [1] signal generated by the Basys3 crystal oscillator to 25MHz. The VGA_Controller module reads the clock module to output the coordinate value of the current pixel and VGA signal. The Graphics module outputs RGB signals to control the color of the current pixel, creating an immersive interface for the player. The Physics module calculates the current basketball coordinate state by reading the coordinate value and initial velocity. The Control module reads input from the switches and buttons on the board, translating them into the horizontal and vertical speed of the basketball. Finally, the Feedback module determines whether the basketball enters the basket and outputs the score to the seven-segment digital tube, the finally performance is shown in CE339-assignment bilibili.

Design Description and Performance

1. Top_Module

   Top_Module as the topmost module defines the input and output ports of the entire project, Table 1 shows all port input and output. Figure 2 Schematic of Top_Module shows the flow of signals throughout the project and the relationship between the various modules. Hierarchical design provides several benefits when designing FPGA include increased design efficiency, improved design reusability, easier verification and debugging, increased scalability(Figure 3 Structure of whole project).

2. Clock_Divider

   The role of Clock_Divider is to drive the 100MHz [1] high-speed crystal oscillator in the Basys3 board through the counter to output 25MHz clk to drive the VGA display to generate a VGA signal. The entity has a generic parameter "modulo" which specifies the division factor of the frequency divider. In this case, the default value of "modulo" is set to 4, which means that the output frequency will be one-fourth of the input frequency.

3. Graphics

   The main function of the graphics module is to use the received integer type coordinates CX, CY to represent the current scanned

pixel point when receiving a clk=25MHz, X and Y represent the coordinates of the current basketball ball center pixel point, and **[offset]** is the initial player displacement entered by the user via switch[7:4]. This module is divided into **[static_display]** (draw basketry and player) and **[basketball]** (draw basketball and **it's trace**) e.g. if hcount is received as 20 and vcount as 225, which means that the pixel point currently scanned corresponds to the 20th column and 225th row on a 640*480 screen, and the color of that pixel point should be red(since the distance to center of the basketball less than radius), so the module will output the corresponding RGB='11100000000'. Similarly, 1220 such signals would make up the red basketry in Figure 1 Interface.

At the same time, This module also records the coordinates of the center of the basketball circle for each input, The above colors are then output on the pixels at these coordinates.

4. Physics

The physics module plays the role of the key in this project, simulating the movement of the ball under gravity and collisions with the walls of the virtual environment. The simulation is implemented using a state machine with two states: [debug] and [play]. This state machine is used to control the behaviour of the simulation and enables the user to switch between the two modes. This module has two modes [debug_mode] and [play_mode], shown in Figure 4 Graphics Module.

In **[debug_mode]**, the ball is at rest at the initial position specified by the user and the value of switch[15:4] is converted to the horizontal initial velocity array vx, the vertical initial velocity array vy and the initial displacement value of the player, which means the initial displacement value of the basketball.

In **[play_mode]** the module combines vx, vy, gravitational acceleration, cx and cy to simulate the real basketball path and adds additional collision detection and bounce feedback to simulate the real state of the basketball game. The simulation itself is implemented by a process called **[move_process]**. This process is triggered by the VS signal and calculates the current position of the ball based on its speed, gravity and collisions with walls. The process involves detecting collisions with walls and adjusting the ball's speed accordingly to simulate a bouncing effect. The speed of the ball is also adjusted to simulate the

effect of gravity on the ball. The module uses a constant 'G' to represent the gravity constant, which is subtracted from the vertical velocity of the ball to simulate the ball's descent. I have added an additional **[offset]** variable to offset the shot position by using the binary number of the switch [7:4] to allow the player to start the shot at a different distance from the basket.

5. Control

The purpose of this module is to implement an anti-shake mechanism for the button. It prevents false triggering of the **[key_out]**signal due to noise or bouncing of the button contacts. It only changes the **[key_out]** signal when the button is pressed or released for a certain duration.

The module has a process that executes on every rising edge of the clock. The process checks the value of the **[key_in]** signal. If it is '1', it means the button is pressed. Then, it increments the **[pressbt]** signal by 1 and resets the **[shakecount]** signal to 0. If the **[pressbt]** signal reaches 2000, it means the button has been pressed for 20 us. Then, it sets the **[key_out]** signal to '1'.If the **[key_in]** signal is '0', it means the button is not pressed. Then, it resets the **[pressbt]** signal to 0 and increments the **[shakecount]** signal by 1. If the **[shakecount]** signal reaches 1000, it means the button has been not pressed for 10 us. Then, it sets the **[key_out]** signal to '0'.

6. Feedback

The Feedback module is used to determine whether the basketball has been put into the basket. This module includes a **[7-segment display]** and a **[clock divider]**, shown in Figure 5 Feedback module. To simplify this logic, the code logic is that the score is considered to be scored when the coordinates of the ball's centre coincide with the red border of the basket, and the score is accumulated and fed to a seven-segment digital display that shows the current score on the seven-segment display, while the clock divider is used to control the refresh rate of the display.

The module consists of two sub-modules. The first module is used to control the refresh rate of the display. It uses a clock divider to generate a 5MHz signal from a 100MHz board clock. The output of this process is stored in the **[output]** signal. This signal is used as the clock input for the digit counter process. The **[digit_counter]** process uses the **[digit_counter]** signal to

determine which number to display on the 7-segment display. It updates the **[current_digit]** signal based on the value of **[digit_counter]** and sets the an signal to the appropriate value to drive the common anode of the display.

The second sub-module is used to update the player's score based on their position. It checks if the position of the basketball overlaps with the pixels of the basket and increases the score accordingly. The score is then used to update the **[sw]** signal, which is used to display the score on the 7-segment display.

7. VGA_Controller

The VGA_Controller module is mainly used to control the signal output of the VGA display, including output resolution, refresh rate, synchronization signal and pixel data. It receives input signals from Top_Module and generates VGA signals based on these signals to control the VGA monitor to display images on the screen. In this project, the VGA_Controller module also needs to update the pixel data according to the data output by the Physics and Graphics modules to display images such as basketball, hoop and track.

Challenge and Improvement

1. Alternative Options

a) When first starting to implement the Graphics module, I was inspired by the GitHub code [2] I received and decided to draw a basketball, a player, and a basket using a bitmap pattern shown in Figure 6 Try process 1. However, as I progressed with the implementation, it became apparent that drawing a dynamic basketball model using a bitmap would be challenging. Drawing a moving basketball requires creating and rendering numerous circles, each with different positions as the ball moves. This would necessitate drawing an extensive number of circles, which can easily reach 307,200 (640 * 480), with each pixel point serving as the center of a circle. I realized that this process would be highly inefficient, and it would be better to use a different approach to render dynamic images. Consequently, I determined that bitmap graphics are more suitable for generating static images than for dynamic ones.

b) When working with the graphics module, I intended to include a function that would generate the basketball's trajectory on the screen. However, I am having problems tracking the position of the basketball in previous clock cycles [3]. Specifically, I

needed to store the position of the basketball at each rising edge of the clock signal in order to calculate its trajectory over time. In other words, I needed many registers to store the coordinates of the basketball, and I was going to use a two-dimensional integer array to store the X and Y values of each input, but this would undoubtedly use up all of the FPGA's RAM resources [4](Figure 7 Try process 2). I also need to consider how to display the stored X and Y values on the screen.

2. Trying Process and Restoration

a) To solve above 1.a) I use flags as symbols at the centre of the circle and where the radius is radiated, and when flag=1, the output of RGB is specified. As a result, my workload is reduced to a large extent. When I was adding the player offset function, w I found that if I moved both player and basketball, I would need to modify all four projects [player], [basketball], [static display] and [graphics] at the same time, which is quite a huge amount of work for each modification. So I started by getting [basketball] and [graphics] moving(Figure 6 Try process 1), and when everything was debugged, I just needed to add declarations for the corresponding ports in [player] and [static display].

b) To solve above 1.b), I used a one-dimensional array: Considering that the basketball coordinates calculated by the Physics module are discrete, in order to save RAM resources and to facilitate indexing, I use the Array(X)=Y method to store the input coordinate signal [5], in other words, I record the X coordinate and the corresponding height of the basketball in each frame, and process the input pixel coordinate signal in the same way, for example, I record the coordinates of the basketball in a certain For example, if the basketball's coordinates are (100, 200) at a certain frame, and if the pixel's X coordinate is equal to 100, and if the Y value is exactly equal to 200, then the pixel is filled with the corresponding colour. This creates a trajectory of discrete points.

Reference

[1] I. Copyright Digilent, Basys 3™ FPGA Board Reference Manual, 2016.

[2] efeacer, "PongGameVHDL," [Online]. Available: https://github.com/efeacer/PongGameVHDL. [Accessed 17 3 2023].

[3] nandland, "lfsr," [Online]. Available: https://github.com/nandland/lfsr. [Accessed 11 3 2023].

[4] "Two dimensional array in VHDL," xilinx, [Online]. Available: https://support.xilinx.com/s/question/0D52E00006iHqzbSAC/two-dimensional-array-in-vhdl?language=en_US. [Accessed 24 3 2023].

[5] vhdl-how-to-create-a-memory-block-with-some-elements-being-constant, electronics.stackexchange, [Online]. Available: https://electronics.stackexchange.com/questions/521985/vhdl-how-to-create-a-memory-block-with-some-elements-being-constant. [Accessed 31 3 2023].

Appendix

Table

Table 1 Module Input Output Table

Module name	Input port	Output port
Top_Module	switch[15:8]	an[3:0]
	clk	seg[6:0]
	btnC	vgaBlue[3:0]
		vgaGreen[3:0]
		vgaRed[3:0]
Clock_Divider	clk100	clk25
Graphics	clk	graBlue[3:0]
	X[10:0]	graGreen[3:0]
	Y[10:0]	graRed[3:0]
	CX[31:0]	
	CY[31:0]	
Physics	clk	CX[31:0]
	vx[3:0]	CY[31:0]
	vy[3:0]	
	clk	
	VS	
Control	clk100	key_out
	key_in	
Feedback	clk	an[3:0]
	hcount[10:0]	seg[6:0]
	vcount[10:0]	
VGA_Controller	clk25	VS
	rst	hcount[10:0]
		vcount[10:0]

Figure

Figure 1 Interface

Figure 2 Schematic of Top_Module

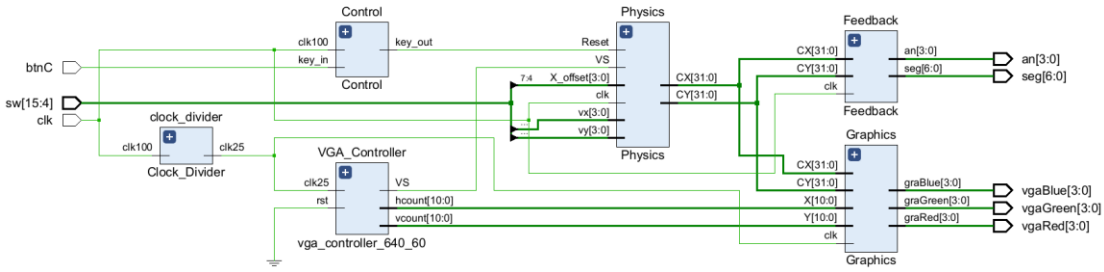


Figure 3 Structure of whole project

- **Top_Module**(Behavioral) (Top_Module.vhd) (6)
  - clock_divider : Clock_Divider(Behavioral) (Clock_Divider.vhd)
  - Control : Control(Behavioral) (Control.vhd)
  - VGA_Controller : vga_controller_640_60(Behavioral) (vga_cor
  - Physics : Physics(Behavioral) (Physics.vhd)
  - Graphics : Graphics(Behavioral) (Graphics.vhd) (2)
    - Basketball : Basketball(Behavioral) (Basketball.vhd)
    - Static_display : Static_display(Behavioral) (Static_display.
      - Player : Player(Behavioral) (Player.vhd)
      - basketry : basketry(Behavioral) (basketry.vhd)
  - Feedback : Feedback(Behavioral) (Feedback.vhd) (2)
    - Score : score(Behavioral) (Score.vhd)
    - four_digits_unit : four_digits(Behavioral) (four_digits.vhd)
      - one_digit_unit : one_digit(Behavioral) (one_digit.vhd)
      - ck_5mhz : Clock_Divider(Behavioral) (Clock_Divider.v

Figure 4 Graphics Module

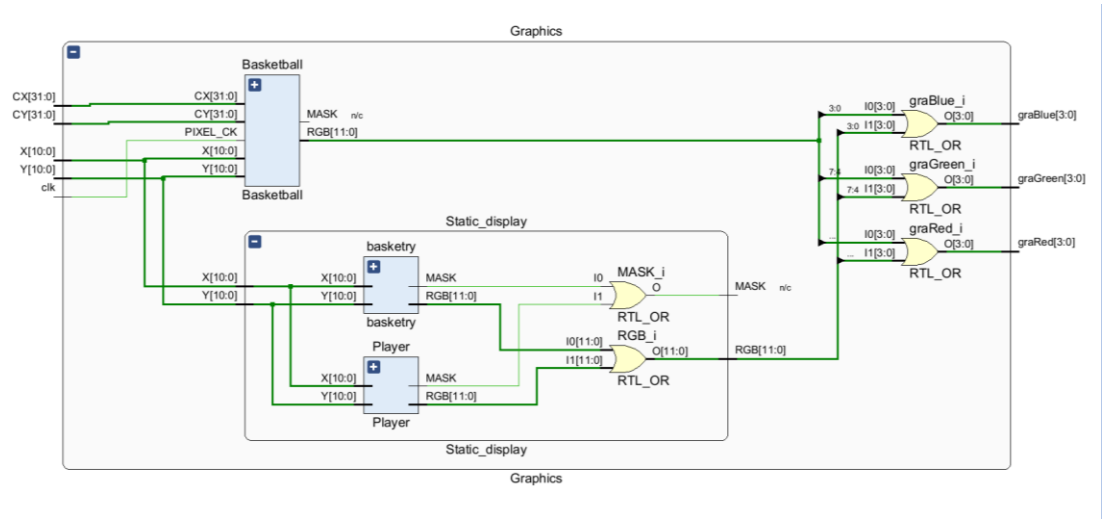


Figure 5 Feedback module

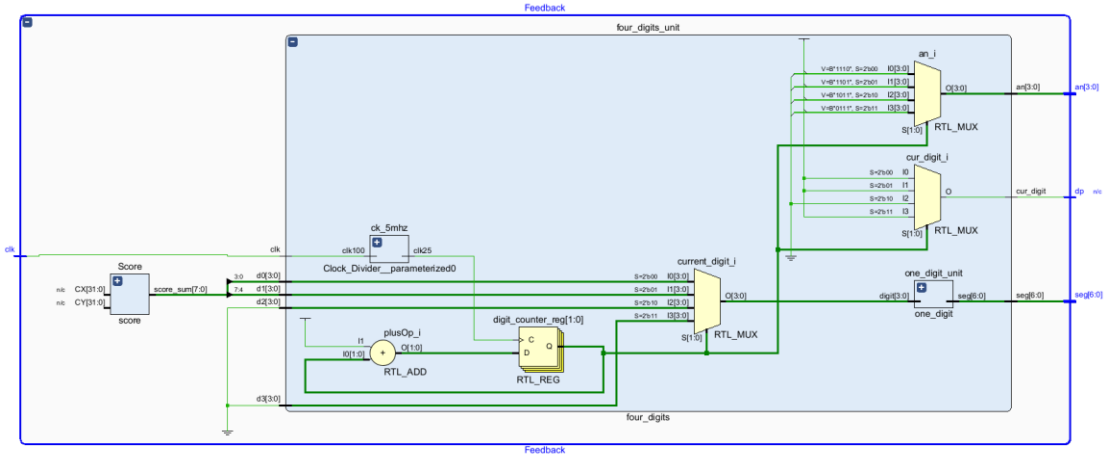


Figure 6 Try process 1

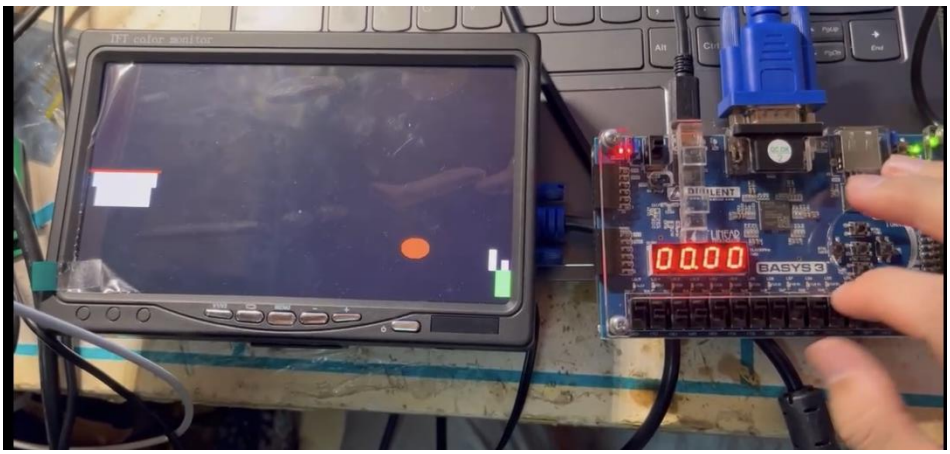
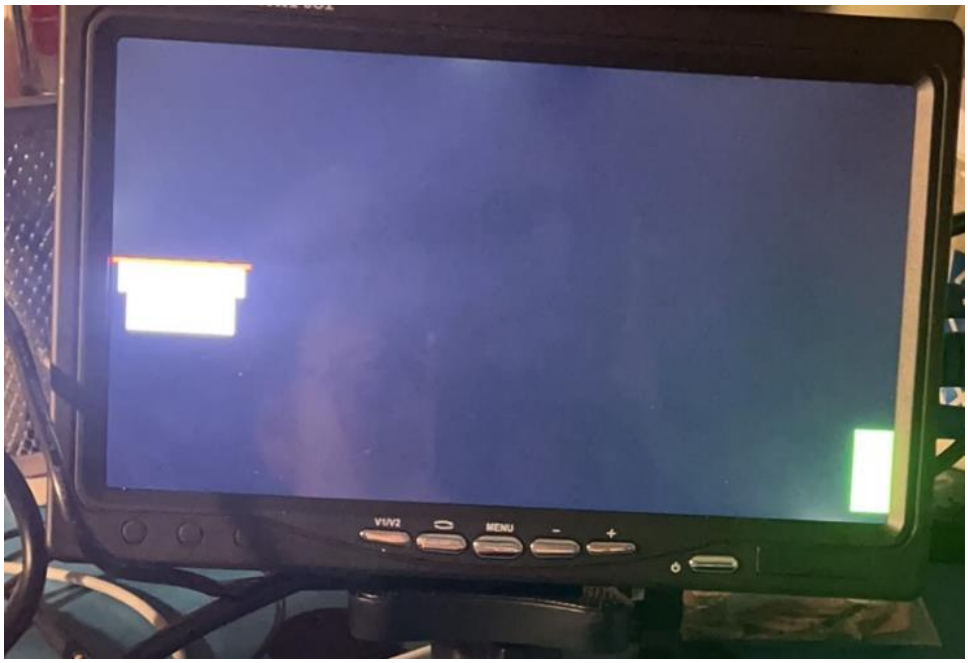


Figure 7 Try process 2



Code

```
----------------------------------------------------------------------------
-------------
-- Company: University of Essex
-- Engineer: Yu Wenlu
-- Create Date: 2023/03/16 17:06:15
-- Design Name:  Assignment 2
-- Module Name: Top_Module - Behavioral
-- Project Name:
-- Target Devices:
-- Tool Versions:
-- Description:  Top level module
--  This module is the top level module of the project. It is
responsible for
--  connecting all the submodules together.
-- Dependencies:    Clock_Divider
--                  Control
--                  vga_controller_640_60
--                  Physics
--                  Graphics
--                  Feedback

-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
```

```
----------------------------------------------------------------------------
-------------


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


entity Top_Module is
    Port ( clk : in STD_LOGIC;
         --Vsync : out STD_LOGIC;
           vgaRed : out STD_LOGIC_VECTOR (3 downto 0);
           vgaBlue : out STD_LOGIC_VECTOR (3 downto 0);
           vgaGreen : out STD_LOGIC_VECTOR (3 downto 0);
           sw : in unsigned(15 downto 4);
           seg  : out STD_LOGIC_VECTOR (6 downto 0);
           --dp  : out STD_LOGIC;
           an   : out STD_LOGIC_VECTOR (3 downto 0);
           btnC : in STD_LOGIC);

end Top_Module;

architecture Behavioral of Top_Module is

    signal hcount,vcount : unsigned(10 downto 0);
    SIGNAL Graphics_MASK,clk_25MHz: STD_LOGIC := '0';
    signal Hsync,Vsync,key_out_C, dp ,blank :STD_LOGIC;
    signal Horizontal_speed, Vertical_speed : STD_LOGIC_VECTOR (3
downto 0);
    signal CX,CY : integer;

begin
    clock_divider: ENTITY work.Clock_Divider(Behavioral) GENERIC
MAP(modulo =>4)
      port map ( clk100 => clk,
                 clk25 =>clk_25MHz);
    Control: entity work.Control (Behavioral)
       port map( key_in => btnC,
                 clk100 => clk,
                 key_out => key_out_C);
    VGA_Controller : ENTITY work.vga_controller_640_60(Behavioral)
       port map ( rst        => '0',
                  clk25   => clk_25MHz,
```

```
                HS          => Hsync,
                VS          => Vsync,
                hcount      => hcount,
                vcount      => vcount,
                blank       => blank);
    Physics : ENTITY work.Physics(Behavioral)
        port map( Reset => key_out_C,
                clk => clk,
                CX => CX,
                CY => CY,
                VS => Vsync,
                vx=> sw(15 downto 12),
                vy => sw(11 downto 8),
                X_offset => sw(7 downto 4));
   Graphics: ENTITY work.Graphics(Behavioral)
        port map (  clk => clk_25MHz,
                  X => hcount,
                  Y => vcount,
                  CX => CX,
                  CY => CY,
                 graRED =>  vgaRed,
                 graBlue => vgaBlue,
                 graGreen => vgaGreen );
   Feedback: ENTITY work.Feedback (Behavioral)
    port map(
        CX => CX,
        CY => CY,
        clk => clk,
        seg => seg,
        an => an,
        dp =>dp
    );

end Behavioral;

----------------------------------------------------------------------------
-------------
-- Company: University of Essex
-- Engineer: Yu Wenlu
--
-- Create Date: 2023/03/30 23:41:19
-- Design Name: Graphics
-- Module Name: Graphics - Behavioral
-- Project Name: Basketball Video Game
```

```
-- Target Devices: Basys 3 Artix-7 FPGA Trainer Board
-- Tool Versions: Vivado 2018.3
-- Description:  This module is the top module of the graphics
system. It is responsible for
--           combining the static display and the basketball display.
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
----------------------------------------------------------------------
-------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;
entity Graphics is
Port (
          clk : in STD_LOGIC;
          X, Y : in UNSIGNED(10 downto 0);
          CX : in integer ;
          CY : in integer;
          graRed : out STD_LOGIC_VECTOR (3 downto 0);
          graBlue : out STD_LOGIC_VECTOR (3 downto 0);
          graGreen : out STD_LOGIC_VECTOR (3 downto 0)
);
end Graphics;

architecture Behavioral of Graphics is
    signal clk_25MHz: STD_LOGIC :='0';
    signal hc,vc : unsigned(10 downto 0);
    SIGNAL  Static_display_MASK,Basketball_MASK: STD_LOGIC := '0';
    signal Static_display_RGB,Basketball_RGB: STD_LOGIC_VECTOR (11
downto 0);
    signal x_asix : integer;
    signal y_asix: integer;

begin
 Basketball : ENTITY work.Basketball(Behavioral)
       port map ( PIXEL_CK =>clk,
                  X => X,
                  Y => Y,
                  CX => CX,
                  CY => CY,
```

```
                RGB(11 downto 8) => Basketball_RGB(11 downto 8),
                RGB(7 downto 4) => Basketball_RGB(7 downto 4),
                RGB(3 downto 0) => Basketball_RGB(3 downto 0),
                MASK => Basketball_MASK);
  Static_display : ENTITY work.Static_display(Behavioral)
       port map ( X => X,
                Y => Y,
                RGB(11 downto 8) => Static_display_RGB(11 downto 8),
                RGB(7 downto 4) => Static_display_RGB(7 downto 4),
                RGB(3 downto 0) => Static_display_RGB(3 downto 0),
                MASK => Static_display_MASK);

    graRed <= Basketball_RGB(11 downto 8) or Static_display_RGB(11
downto 8);
    graGreen <= Basketball_RGB(7 downto 4) or Static_display_RGB(7
downto 4);
    graBlue <= Basketball_RGB(3 downto 0) or Static_display_RGB(3
downto 0);

end Behavioral;
----------------------------------------------------------------------
-------------
-- Company: University of Essex
-- Engineer: Yu Wenlu
--
-- Create Date: 2023/03/20 12:44:32
-- Design Name:
-- Module Name: Physics - 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;
use IEEE.NUMERIC_STD.ALL;

entity Physics is
    port( clk: in std_logic;
          CX : out integer;
          CY : out integer;
          VS : in std_logic;
          Reset : in STD_LOGIC;
          vx: in unsigned(3 downto 0);
          vy: in unsigned(3 downto 0);
          X_offset : in unsigned(3 downto 0)

          );
end Physics;

architecture Behavioral of Physics is

    signal xAxis : integer := 570 - 3 * to_integer(X_offset);
    signal yAxis : integer := 370;
    -- xAxis and yAxis are the initial position of the ball including
the offset

    signal x_Asix: integer := xAxis;
    signal y_Asix : integer := yAxis;
    -- x_Asix and y_Asix are the current position of the ball

    signal G : integer := 2;
    -- G is the gravity constant to simulate the gravity effect

    type STATE is (debug, play);
    signal CUR_STATE, NX_STATE: STATE;
    signal start: std_logic := '1';
    signal vHorizontal : integer :=  3 * to_integer(vx);
    signal vVertical : integer := 3 * to_integer(vy);
    signal sw : unsigned(7 downto 0):= (others =>'0');
    -- refence github.com/alexforencich/verilog-
axi/blob/master/axi_pkg.vhd

    constant collision : integer := 2;
     -- collision is the constant to simulate the collision effect

begin
```

```
    state_registers : process (reset, clk)
    begin
        if start = '1' then
            start <= '0';
            CUR_STATE <= play;
        elsif rising_edge(clk) then
            CUR_STATE <= NX_STATE;
        end if;
    end process;
    state_transition : process (CUR_STATE, clk)
    begin
        case CUR_STATE is
            when debug =>
            if rising_edge(clk) then
                if (Reset = '1') then
                    NX_STATE <= play;
                else
                    NX_STATE <= debug;
                end if;
            end if;
            when play =>
            if rising_edge(clk) then
                if (Reset = '1') then
                    NX_STATE <= debug;
                else
                    NX_STATE <= play;
                end if;
            end if;
        end case;
    end process;
    move_process : process(VS)
    --this is Collision detection
    --the ball will bounce back when it hits the wall
    -- we can change the consta of vHorizontal and vVertical to
change the speed of the ball
    begin
        if (CUR_STATE = play) then
            if (falling_edge(VS)) then
                if (x_Asix >= 8 and y_Asix <= 457) then
                    x_Asix <= x_Asix - vHorizontal;
                    y_Asix <= y_Asix - vVertical;
                    vVertical <= vVertical - G;
                    -- if the ball hits the left wall
                else
```

```
                if ( y_Asix > 457 ) then
                  vHorizontal <= (vHorizontal/collision);
                  vVertical <= -(vVertical)/collision/2;
                  y_Asix <= 457;
                  -- if the ball hits the bottom wall
                 end if;
                if (x_Asix <8) then
                    vHorizontal <= (vHorizontal/collision);
                    vVertical <= -(vVertical)/collision/2;
                    x_Asix <= 8;
                 -- if the ball hits the top wall
               end if;
            end if;
         end if;
       elsif (CUR_STATE = debug) then
          vHorizontal <= 3 * to_integer(vx);
          vVertical <=3 * to_integer(vy);
          x_Asix <= xAxis;
          y_Asix <= yAxis;
      end if;
   end process;
   CX <= x_Asix;
   CY <= y_Asix;

end Behavioral;

----------------------------------------------------------------------------
-- Company: University of Essex
-- Engineer: Yu Wenlu
--
-- Create Date: 2023/03/20 12:44:32
-- Design Name: assignment2
-- Module Name: Physics - Behavioral
-- Project Name:
-- Target Devices: basys3 board
-- Tool Versions:
-- Description: this is the physics module
-- simulation the gravity and collision effect of the ball

-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
```

```
-- Additional Comments:
--
----------------------------------------------------------------------------
-------------


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

entity Physics is
    port( clk: in std_logic;
          CX : out integer;
          CY : out integer;
          VS : in std_logic;
          Reset : in STD_LOGIC;
          vx: in unsigned(3 downto 0);
          vy: in unsigned(3 downto 0);
          X_offset : in unsigned(3 downto 0)

          );
end Physics;

architecture Behavioral of Physics is

    signal xAxis : integer := 570 - 3 * to_integer(X_offset);
    signal yAxis : integer := 370;
    -- xAxis and yAxis are the initial position of the ball including
the offset

    signal x_Asix: integer := xAxis;
    signal y_Asix : integer := yAxis;
    -- x_Asix and y_Asix are the current position of the ball

    signal G : integer := 2;
    -- G is the gravity constant to simulate the gravity effect

    type STATE is (debug, play);
    signal CUR_STATE, NX_STATE: STATE;
    signal start: std_logic := '1';
    signal vHorizontal : integer :=  3 * to_integer(vx);
    signal vVertical : integer := 3 * to_integer(vy);
    signal sw : unsigned(7 downto 0):= (others =>'0');
    -- refence github.com/alexforencich/verilog-
```

```
axi/blob/master/axi_pkg.vhd

    constant collision : integer := 2;
     -- collision is the constant to simulate the collision effect

begin

    state_registers : process (reset, clk)
    begin
        if start = '1' then
            start <= '0';
            CUR_STATE <= play;
        elsif rising_edge(clk) then
            CUR_STATE <= NX_STATE;
        end if;
    end process;
    state_transition : process (CUR_STATE, clk)
    begin
        case CUR_STATE is
            when debug =>
            if rising_edge(clk) then
                if (Reset = '1') then
                    NX_STATE <= play;
                else
                    NX_STATE <= debug;
                end if;
            end if;
            when play =>
            if rising_edge(clk) then
                if (Reset = '1') then
                    NX_STATE <= debug;
                else
                    NX_STATE <= play;
                end if;
            end if;
        end case;
    end process;
    move_process : process(VS)
    --this is Collision detection
    --the ball will bounce back when it hits the wall
    -- we can change the consta of vHorizontal and vVertical to
change the speed of the ball
    begin
        if (CUR_STATE = play) then
```

```
            if (falling_edge(VS)) then
                if (x_Asix >= 8 and y_Asix <= 457) then
                    x_Asix <= x_Asix - vHorizontal;
                    y_Asix <= y_Asix - vVertical;
                    vVertical <= vVertical - G;
                    -- if the ball hits the left wall
                else
                    if ( y_Asix > 457 ) then
                      vHorizontal <= (vHorizontal/collision);
                      vVertical <= -(vVertical)/collision/2;
                      y_Asix <= 457;
                      -- if the ball hits the bottom wall
                     end if;
                    if (x_Asix <8) then
                       vHorizontal <= (vHorizontal/collision);
                       vVertical <= -(vVertical)/collision/2;
                       x_Asix <= 8;
                     -- if the ball hits the top wall
                   end if;
                end if;
            end if;
         elsif (CUR_STATE = debug) then
            vHorizontal <= 3 * to_integer(vx);
            vVertical <=3 * to_integer(vy);
            x_Asix <= xAxis;
            y_Asix <= yAxis;
        end if;
    end process;
    CX <= x_Asix;
    CY <= y_Asix;

end Behavioral;

----------------------------------------------------------------------------
-------------
-- Company: University of Essex
-- Engineer: Yu Wenlu
-- Create Date: 2023/03/31 08:52:14
-- Design Name:
-- Module Name: Feedback - Behavioral
-- Project Name:
-- Target Devices:
-- Tool Versions:
-- Description: this module is used to show the score of the player
```

```
--  shown in segment display

-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
----------------------------------------------------------------------------------


library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;
entity Feedback is
Port (     CX : in integer;
          CY : in integer;
           clk : in STD_LOGIC; -- 100MHz board clock
          seg  : out STD_LOGIC_VECTOR (6 downto 0);
          dp  : out STD_LOGIC;
          an   : out STD_LOGIC_VECTOR (3 downto 0)
);
end Feedback;

architecture Behavioral of Feedback is
    signal x_asix : integer;
    signal y_asix: integer;
    signal player_score : unsigned(7 downto 0);
begin
    Score : ENTITY work.score(Behavioral)
       port map( CX => x_asix,
                 CY => y_asix,
                 score_sum => player_score);

    four_digits_unit : entity work.four_digits(Behavioral)
    Port map (d3 => "0000",
              d2 => "0000",
              d1 => player_score(7 downto 4),
              d0 => player_score(3 downto 0),
              clk => clk, seg => seg, an => an, cur_digit =>dp);

end Behavioral;
----------------------------------------------------------------------------------
```

```
-------------
-- Company: University of Essex
-- Engineer: Yu Wenlu
--
-- Create Date: 2023/03/17 15:51:45
-- Design Name:
-- Module Name: Basketball - Behavioral
-- Project Name:
-- Target Devices:
-- Tool Versions:
-- Description: this is the top level module of the basketball
-- inculde the trajectory of the circle use index to store the Y
coordinate of the circle
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
----------------------------------------------------------------------
-------------

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;
entity Basketball is
    generic (
            R : natural := 20; -- radius of a circle
            COLOUR : std_logic_vector(11 downto 0):=
"111100110000" ); -- color of a circle
    port ( PIXEL_CK : in std_logic; -- oixel clock : 25MHz
          X, Y : in  UNSIGNED (10 downto 0); -- X,Y coordinates of
the current circle
          RGB : out STD_LOGIC_VECTOR (11 downto 0); -- Color bits of
circle
          MASK : out STD_LOGIC; -- Display mask of circle
          CX : in integer ; -- X-axis coordinate of center of circle
          CY : in integer); -- Y-axis coordinate of center of circle

end Basketball;

architecture Behavioral of Basketball is
    constant    matrix_num:     integer := 600;
```

```
    TYPE       matrix_index is array (matrix_num downto 0) of integer;
    --store the Y coordinate of the circle and draw the trajectory of
the circle
    signal     Y_G:    matrix_index;
    signal DX, DY : unsigned (X'range); -- DX = |(X-CX)|; DY = |(Y -
CY)|
    signal DX2, -- (X-CX)^2
          DY2 : unsigned ((2*X'high+1) downto 0); -- (Y -CY)^2
    signal FLAG : std_logic; -- Determine when to display a circle.
    constant R2 : unsigned(DX2'range):= to_unsigned(R * R,
DX2'length); -- R^2
begin
--    G_clear:process(Clear)
--    variable i: integer := 0;
--    begin
--       if rising_edge(Clear) then
--          i := 0;
--          while(i<=matrix_num) loop
--             Y_G(i) <= 0;
--             i := i+1;
--          end loop;
--       end if;
--    end process;
    process
    begin
       wait until rising_edge(PIXEL_CK);
       Y_G(CX)<=CY;
       if X <= CX - R or X >= CX + R then
          DX2 <= R2;
       else
          DX2 <= DX2 + (X * 2) - (CX * 2) - 1;
       end if;
       if X = 0 then
          if (Y <= CY - R) or (Y >= CY + R) then
             DY2 <= R2;
          else
             DY2 <= DY2 + (Y * 2) - (CY * 2) - 1;
          end if;
       end if;
       FLAG<='0';
       if Y_G(TO_INTEGER(X)) = TO_INTEGER(Y) or (DX2 + DY2 < R2) then
          FLAG<='1';
       end if;
    end process;
```

```
    RGB <= COLOUR when FLAG = '1'else (OTHERS => '0');
    MASK <= FLAG;
end Behavioral;
----------------------------------------------------------------------------
-- Company: University of Essex
-- Engineer: Yu Wenlu
--
-- Create Date: 2023/03/25 21:23:23
-- Design Name:
-- Module Name: basketry - 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;


entity basketry is
    port ( X, Y : in  UNSIGNED (10 downto 0);
           RGB : out STD_LOGIC_VECTOR (11 downto 0);
           MASK : out STD_LOGIC );
end basketry;

architecture Behavioral of basketry is

signal TOP_FLAG, MIDDLE_FLAG, BOTTOM_FLAG : STD_LOGIC;
begin
```

```
    TOP_FLAG <='1' when (X >= 8) and (X < 130)
              and(Y >= 220) and (Y < 230) else '0'; --flag goes high
when the rectangle is being drawn
    MIDDLE_FLAG <='1' when (X >= 13) and (X < 110)
              and(Y >= 230) and (Y < 265) else '0'; --flag goes high
when the rectangle is being drawn
    BOTTOM_FLAG <='1' when (X >= 20) and (X < 95)
              and(Y >= 265) and (Y < 300) else '0'; --flag goes high
when the rectangle is being drawn
    RGB <= "111100000000" when TOP_FLAG = '1' else
          "111111111111" when MIDDLE_FLAG = '1' or BOTTOM_FLAG = '1'
else
          (OTHERS => '0');
    MASK <= TOP_FLAG or MIDDLE_FLAG or BOTTOM_FLAG;


end Behavioral;
----------------------------------------------------------------------------
-------------
-- Company: University of Essex
-- Engineer: Yu Wenlu
--
-- Create Date: 2023/03/17 15:51:45
-- Design Name:
-- Module Name:
-- Project Name:
-- Target Devices:
-- Tool Versions:
-- Description:
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
----------------------------------------------------------------------------
-------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity Clock_Divider is
    generic(
          modulo : NATURAL := 4);
```

```
    Port (
           clk100 : in STD_LOGIC; -- The clock input from the board
           clk25 : out STD_LOGIC);--The pixel clock ( the output of
frequency divider )
end Clock_Divider;

architecture Behavioral of Clock_Divider is

signal tmp : STD_LOGIC ;

begin

    process
        variable count : INTEGER;
    begin
        wait until RISING_EDGE(clk100);
            --Once the count has been reached the counter will reset
and a ouptut high logic
        if count = (modulo - 1) then
            count := 0;
            tmp <= '1';
        else
            count := count + 1;-- increment counter
            tmp <= '0';--output low logic
        end if;

    end process ;

    clk25 <= tmp;

end Behavioral;
----------------------------------------------------------------------
-------------
-- Company: University of Essex
-- Engineer: Yu Wenlu
--
-- Create Date: 2023/03/25 21:23:23
-- Design Name:
-- Module Name:
-- Project Name:
-- Target Devices:
-- Tool Versions:
-- Description:
--
```

```
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
----------------------------------------------------------------------
-------------


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

entity Player is
    port (
           X, Y : in  UNSIGNED (10 downto 0);
           RGB : out STD_LOGIC_VECTOR (11 downto 0);
           MASK : out STD_LOGIC;
           X_offset : in unsigned(3 downto 0));
end Player;

architecture Behavioral of Player is

signal TOP_FLAG,MIDDLE_FLAG,BOTTOM_FLAG : STD_LOGIC;
begin
    TOP_FLAG <='1' when (X >= 590-3*TO_INTEGER(X_offset)) and (X <
600-3*TO_INTEGER(X_offset))
              and(Y >= 370) and (Y < 410) else '0'; --flag goes high
when arm is being drawn
    MIDDLE_FLAG <='1' when (X >= 607-3*TO_INTEGER(X_offset)) and (X <
617-3*TO_INTEGER(X_offset))
              and(Y >= 390) and (Y < 410) else '0'; --flag goes high
when hand is being drawn
    BOTTOM_FLAG <='1' when (X >=600-3*TO_INTEGER(X_offset) ) and (X <
620-3*TO_INTEGER(X_offset))
              and(Y >= 410) and (Y < 460) else '0'; --flag goes high
when body is being drawn
    RGB <= "111111111111" when TOP_FLAG = '1' else
          "111100001111" when MIDDLE_FLAG = '1' else
          "000011110000" when BOTTOM_FLAG = '1' else
          (OTHERS => '0');
    MASK <= TOP_FLAG or BOTTOM_FLAG or MIDDLE_FLAG;
```

```
end Behavioral;

----------------------------------------------------------------------------
-- Company: University of Essex
-- Engineer: Yu Wenlu
--
-- Create Date: 2023/03/30 15:28:46
-- Design Name:
-- Module Name: score - 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;
use IEEE.NUMERIC_STD.ALL;
use ieee.std_logic_unsigned.all;

entity score is
    Port ( CX : in integer;
           CY : in integer;
           score_sum : out unsigned (7 downto 0));
end score;

architecture Behavioral of score is
    signal score : unsigned(5 downto 0) := (others =>'0');
    signal sw : unsigned(7 downto 0):= (others =>'0');
begin
    score_display: process(score)
    variable sw_s : unsigned (5 downto 0);
    begin
        if ( score >="000000" and score <="001001") then
```

```
            sw(7 downto 4) <= "0000";
            sw(3 downto 0) <= score(3 downto 0);
        elsif (  score <= "010011") then --19
            sw(7 downto 4) <= "0001";
            sw_s := (score - "001010");
            sw(3 downto 0) <= sw_s(3 downto 0);
        elsif (  score <= "011101") then  --29
            sw(7 downto 4) <= "0010";
            sw_s := (score - "010100");
            sw(3 downto 0) <= sw_s(3 downto 0);
        elsif (  score <= "100111") then --39
            sw(7 downto 4) <= "0011";
            sw_s := (score - "011110");
            sw(3 downto 0) <= sw_s(3 downto 0);
        elsif (  score <= "110001") then --49
            sw(7 downto 4) <= "0100";
            sw_s := (score - "101000");
            sw(3 downto 0) <= sw_s(3 downto 0);
        elsif (  score <= "111011") then --59
            sw(7 downto 4) <= "0101";
            sw_s := (score - "110010");
            sw(3 downto 0) <= sw_s(3 downto 0);
        end if;
    end process;

    move_process : process(CX)
    begin
        if ((CY <= 230) and (CY >=190) and (CX <=112) and (CX >= 8))
then
            score <= score + 1;
        end if;
    end process;

    score_sum <= sw;

end Behavioral;
----------------------------------------------------------------------------
-------------
-- Company: University of Essex
-- Engineer: Yu Wenlu
-- Create Date: 2023/02/13 22:57:53
-- Design Name: Control
-- Module Name: Control - Behavioral
-- Project Name:  assignment_2
```

```
-- Target Devices: Basys 3
-- Tool Versions:  Vivado 2020.3
-- Description:    This is a behavioral model of a key debounce
circuit.
--              The circuit is designed to detect a key press and
--            prevent the key press from being registered multiple
times.
-- 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 leaf cells in this code.
--library UNISIM;
--use UNISIM.VComponents.all;


entity Control is
    Port ( key_in : in STD_LOGIC;
           clk100 : in STD_LOGIC;
           key_out : out STD_LOGIC);
end Control;

architecture Behavioral of Control is

signal pressbt : integer := 0;
signal shakecount : integer := 0;
Begin
    process(key_in, clk100)
    begin
        if rising_edge (clk100) then
```

```
            if (key_in = '1') then
                pressbt <= pressbt + 1; -- Invert if '1'
                shakecount <= 0; -- Reset shakecount if input signal
changes
                if(pressbt = 2000) then -- Threshold for anti-shake
delay
                    key_out <= '1';
                end if;
            else
                pressbt <=0;
                shakecount <= shakecount + 1; -- Increase shakecount if
input signal is stable
                if (shakecount = 1000) then -- Threshold for anti-shake
delay
                    key_out <= '0';
                end if;
            end if;
        end if;
    end process;
end Behavioral;

----------------------------------------------------------------------
------------
-- Company: University of Essex
-- Engineer: Yu Wenlu
--
-- Create Date: 2023/02/11 21:49:18
-- Design Name: main1_one_digit
-- Module Name: one_digit - Behavioral
-- Project Name: Assignment1
-- Target Devices: Basys3
-- Tool Versions: 2021.2
-- Description:
-- 7-segment decoder: the user should be able to use the rightmost
four switches, SW3-SW0, to
-- input a BCD (0-9) digit that is then displayed (duplicated) on all
four digits of the Basys3
-- display;

-- Dependencies:
--
-- Revision: 1.0
-- Revision 0.01 - File Created
-- Additional Comments:
```

```
--
----------------------------------------------------------------------------
-------------


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

entity one_digit is
    Port ( digit : in unsigned (3 downto 0);
           seg : out STD_LOGIC_VECTOR (6 downto 0));
end one_digit;

architecture Behavioral of one_digit is
    begin
        seg <= "1000000" when digit = "0000" else
               "1111001" when digit = "0001" else
               "0100100" when digit = "0010" else
               "0110000" when digit = "0011" else
               "0011001" when digit = "0100" else
               "0010010" when digit = "0101" else
               "0000010" when digit = "0110" else
               "1111000" when digit = "0111" else
               "0000000" when digit = "1000" else
               "0010000" when digit = "1001" else
               "1111111";
    end Behavioral;
----------------------------------------------------------------------------
------------
-- Company: University of Essex
-- Engineer: Yu Wenlu
--
-- Create Date: 2023/02/11 21:49:18
-- Design Name: main3_final
-- Module Name: four digits - Behavioral
-- Project Name: Assignment1
-- Target Devices: Basys3
-- Tool Versions: 2021.2
-- Description:
--The current_digit signal is updated based on the value of the
digit_counter signal,
--and the an signal is updated to enable the corresponding digit on
the display.
```

```
--The cur_digit signal is set to '1' to activate the currently
selected digit on the display.

-- Dependencies:
--
-- Revision: 1.0
-- Revision 0.01 - File Created
-- Additional Comments:
--
----------------------------------------------------------------------------
-------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;

entity four_digits is
    Port (
           clk : in STD_LOGIC;

           d3 : in UNSIGNED (3 downto 0);
           d2 : in UNSIGNED (3 downto 0);
           d1 : in UNSIGNED (3 downto 0);
           d0 : in UNSIGNED (3 downto 0);

           seg : out STD_LOGIC_VECTOR (6 downto 0);
           an : out STD_LOGIC_VECTOR (3 downto 0);
           cur_digit :out std_logic
         );
end four_digits;

architecture Behavioral of four_digits is

Signal digit_counter : Unsigned(1 downto 0);
Signal current_digit : UNSIGNED (3 downto 0);
--Signal count : Integer;
Signal action_output : STD_LOGIC:='0';

begin
    one_digit_unit : ENTITY work.one_digit(Behavioral)
       port map (digit => current_digit,
                 seg => seg);
     ck_5mhz : ENTITY work.Clock_Divider(Behavioral)
           port map ( clk100 => clk,
                clk25 =>action_output);
```

```
    process(action_output)
    begin
        if rising_edge(action_output) then
         digit_counter <= digit_counter +1;
        end if;
    end process;
process
   begin
   Case digit_counter is --Display in which segment
      When "00" => cur_digit<= '1'; current_digit <= d0; an <=
"1110"; --4th segment
      When "01" =>  cur_digit<= '1';current_digit <= d1; an <=
"1101"; --3rd segment
      When "10" =>  cur_digit<= '0';current_digit <= d2; an <=
"1011"; --2nd segment
      When "11" =>  cur_digit<= '1';current_digit <= d3; an <=
"0111"; --1st segment
   End Case;
End Process;

end Behavioral;
----------------------------------------------------------------------
---
-- vga_controller_640_60.vhd
----------------------------------------------------------------------
---
-- Author : Ulrich Zoltán
--        Copyright 2006 Digilent, Inc.
----------------------------------------------------------------------
---
-- Software version : Xilinx ISE 7.1.04i
--                 WebPack
-- Device         : 3s200ft256-4
----------------------------------------------------------------------
---
-- This file contains the logic to generate the synchronization
signals,
-- horizontal and vertical pixel counter and video disable signal
-- for the 640x480@60Hz resolution.
----------------------------------------------------------------------
---
--  Behavioral description
----------------------------------------------------------------------
---
```

```
-- Please read the following article on the web regarding the
-- vga video timings:
-- http://www.epanorama.net/documents/pc/vga_timing.html

-- This module generates the video synch pulses for the monitor to
-- enter 640x480@60Hz resolution state. It also provides horizontal
-- and vertical counters for the currently displayed pixel and a
blank
-- signal that is active when the pixel is not inside the visible
screen
-- and the color outputs should be reset to 0.

-- timing diagram for the horizontal synch signal (HS)
-- 0                     648    744          800 (pixels)
-- -------------------------|______|-----------------
-- timing diagram for the vertical synch signal (VS)
-- 0                              482    484  525 (lines)
-- ----------------------------------|______|-------

-- The blank signal is delayed one pixel clock period (40ns) from
where
-- the pixel leaves the visible screen, according to the counters, to
-- account for the pixel pipeline delay. This delay happens because
-- it takes time from when the counters indicate current pixel should
-- be displayed to when the color data actually arrives at the
monitor
-- pins (memory read delays, synchronization delays).
----------------------------------------------------------------------
---
--  Port definitions
----------------------------------------------------------------------
---
-- rst             - global reset signal
-- clk25        - input pin, from dcm_25MHz
--                 - the clock signal generated by a DCM that has
--                 - a frequency of 25MHz.
-- HS              - output pin, to monitor
--                 - horizontal synch pulse
-- VS              - output pin, to monitor
--                 - vertical synch pulse
-- hcount           - output pin, 11 bits, to clients
--                 - horizontal count of the currently displayed
--                 - pixel (even if not in visible area)
-- vcount           - output pin, 11 bits, to clients
```

```
--                - vertical count of the currently active video
--                - line (even if not in visible area)
-- blank           - output pin, to clients
--                - active when pixel is not in visible area.
----------------------------------------------------------------------
---
-- Revision History:
-- 09/18/2006(UlrichZ): created
-- 03/05/2013(LCiti): Removed reference to non-standard libraries
-- 02/06/2014(LCiti): Added subtype
----------------------------------------------------------------------
---

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

-- the vga_controller_640_60 entity declaration
-- read above for behavioral description and port definitions.
entity vga_controller_640_60 is
port(
   rst        : in std_logic;
   clk25      : in std_logic;
   HS         : out std_logic;
   VS         : out std_logic;
   hcount     : out unsigned(10 downto 0);
   vcount     : out unsigned(10 downto 0);
   blank      : out std_logic
);
end vga_controller_640_60;

architecture Behavioral of vga_controller_640_60 is

----------------------------------------------------------------------
---
-- CONSTANTS
----------------------------------------------------------------------
---
subtype HTYPE is unsigned(hcount'RANGE);
subtype VTYPE is unsigned(vcount'RANGE);

-- maximum value for the horizontal pixel counter
constant HMAX  : HTYPE := "01100100000"; -- 800
-- total number of visible columns
```

```
constant HLINES: HTYPE := "01010000000"; -- 640
-- value for the horizontal counter where front porch ends
constant HFP   : HTYPE := "01010001000"; -- 648
-- value for the horizontal counter where the synch pulse ends
constant HSP   : HTYPE := "01011101000"; -- 744
-- maximum value for the vertical pixel counter
constant VMAX  : VTYPE := "01000001101"; -- 525
-- total number of visible lines
constant VLINES: VTYPE := "00111100000"; -- 480
-- value for the vertical counter where the front porch ends
constant VFP   : VTYPE := "00111100010"; -- 482
-- value for the vertical counter where the synch pulse ends
constant VSP   : VTYPE := "00111100100"; -- 484
-- polarity of the horizontal and vertical synch pulse
-- only one polarity used, because for this resolution they coincide.
constant SPP   : std_logic := '0';

-----------------------------------------------------------------------
---
-- SIGNALS
-----------------------------------------------------------------------
---

-- horizontal and vertical counters
signal hcounter : HTYPE := (others => '0');
signal vcounter : VTYPE := (others => '0');




-- active when inside visible screen area.
signal video_enable: std_logic;

begin

   -- output horizontal and vertical counters
   hcount <= hcounter;
   vcount <= vcounter;

   -- blank is active when outside screen visible area
   -- color output should be blacked (put on 0) when blank in active
   -- blank is delayed one pixel clock period from the video_enable
   -- signal to account for the pixel pipeline delay.
   blank <= not video_enable when rising_edge(clk25);
```

```
-- increment horizontal counter at clk25 rate
-- until HMAX is reached, then reset and keep counting
h_count: process(clk25)
begin
   if(rising_edge(clk25)) then
      if(rst = '1') then
         hcounter <= (others => '0');
      elsif(hcounter = HMAX) then
         hcounter <= (others => '0');
      else
         hcounter <= hcounter + 1;
      end if;
   end if;
end process h_count;

-- increment vertical counter when one line is finished
-- (horizontal counter reached HMAX)
-- until VMAX is reached, then reset and keep counting
v_count: process(clk25)
begin
   if(rising_edge(clk25)) then
      if(rst = '1') then
         vcounter <= (others => '0');
      elsif(hcounter = HMAX) then
         if(vcounter = VMAX) then
            vcounter <= (others => '0');
         else
            vcounter <= vcounter + 1;
         end if;
      end if;
   end if;
end process v_count;

-- generate horizontal synch pulse
-- when horizontal counter is between where the
-- front porch ends and the synch pulse ends.
-- The HS is active (with polarity SPP) for a total of 96 pixels.
do_hs: process(clk25)
begin
   if(rising_edge(clk25)) then
      if(hcounter >= HFP and hcounter < HSP) then
         HS <= SPP;
      else
```

```
            HS <= not SPP;
          end if;
        end if;
    end process do_hs;

    -- generate vertical synch pulse
    -- when vertical counter is between where the
    -- front porch ends and the synch pulse ends.
    -- The VS is active (with polarity SPP) for a total of 2 video
lines
    -- = 2*HMAX = 1600 pixels.
    do_vs: process(clk25)
    begin
      if(rising_edge(clk25)) then
        if(vcounter >= VFP and vcounter < VSP) then
          VS <= SPP;
        else
          VS <= not SPP;
        end if;
      end if;
    end process do_vs;

    -- enable video output when pixel is in visible area
    video_enable <= '1' when (hcounter < HLINES and vcounter < VLINES)
else '0';

end Behavioral;
```