Ege Ozan Özyedek
Lab 06
EEE102 – 02
Lab Report

- Design Methodology

For this week's lab, I decided to draw a game level and show it on the monitor via the VGA connection that exists on BASYS 3. The connection was possible by assigning the VGA input pins to the corresponding binary units on the code. The picture was shown on a 1440x900 display. The pixel count is then changed according to the front porch, back porch and sync pulse. The lab assignment was achieved by using 4 modules which have different responsibilities.

  - vga_main: The top module, which only creates components of other modules. Its inputs and outputs are
    - clk: The 100 MHz clock that exists on BASYS 3. Used to refresh the pixels on other modules
    - hsync, vsync: VGA synchronization signals which change their value to LOW on sync pulse.
    - r, g, b: The 4-bit binary color codes for red, green and blue colors.

  - vga_controller: This module is the module which controls the synchronization of VGA signals (hsync, vsync). It is responsible for the timing of front porch, back porch and sync pulse. It also changes the value of sync signals accordingly.

  - vga_draw: This module is responsible for drawing the shapes on the monitor. It uses the package draw (created in draw_packages) which has a procedure called rectangle. The module changes the r, g, b colors of the current pixel according to the output of the procedure.

  - draw_packages: This module is used to create packages with procedures in them. The procedure called rectangle for example, takes various inputs (such as side lengths, position and desired color values) and outputs its color at the desired pixels.

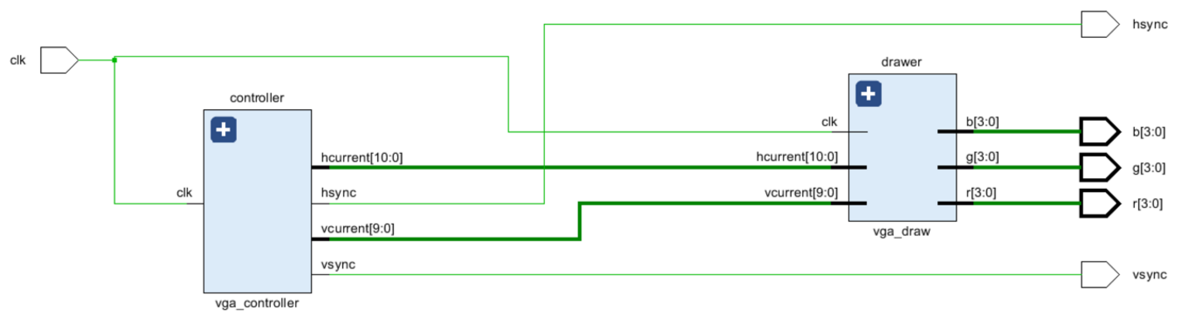

Figure 1: The RTL schematic

- Results

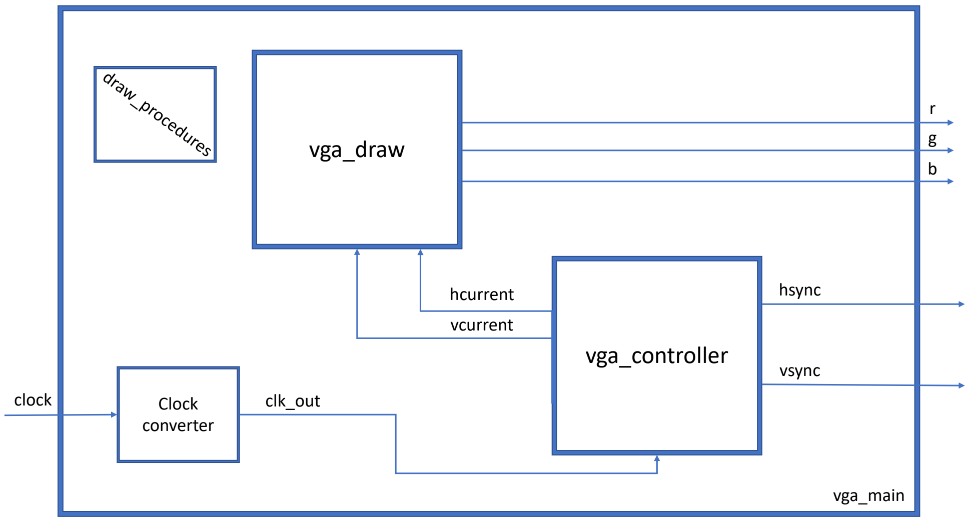

Figure 2: The expected block diagram

The results were, for the most part, as expected. The expected block diagram (which was presented in the preliminary report) and the RTL schematic can be observed in the given figures. They are the same, except for the draw_packages module which I included in the main module but since it is independent of the module it is not visible in the RTL schematic. I also did not use a clock divider since the 100MHz clock on BASYS 3 was a viable pixel clock.

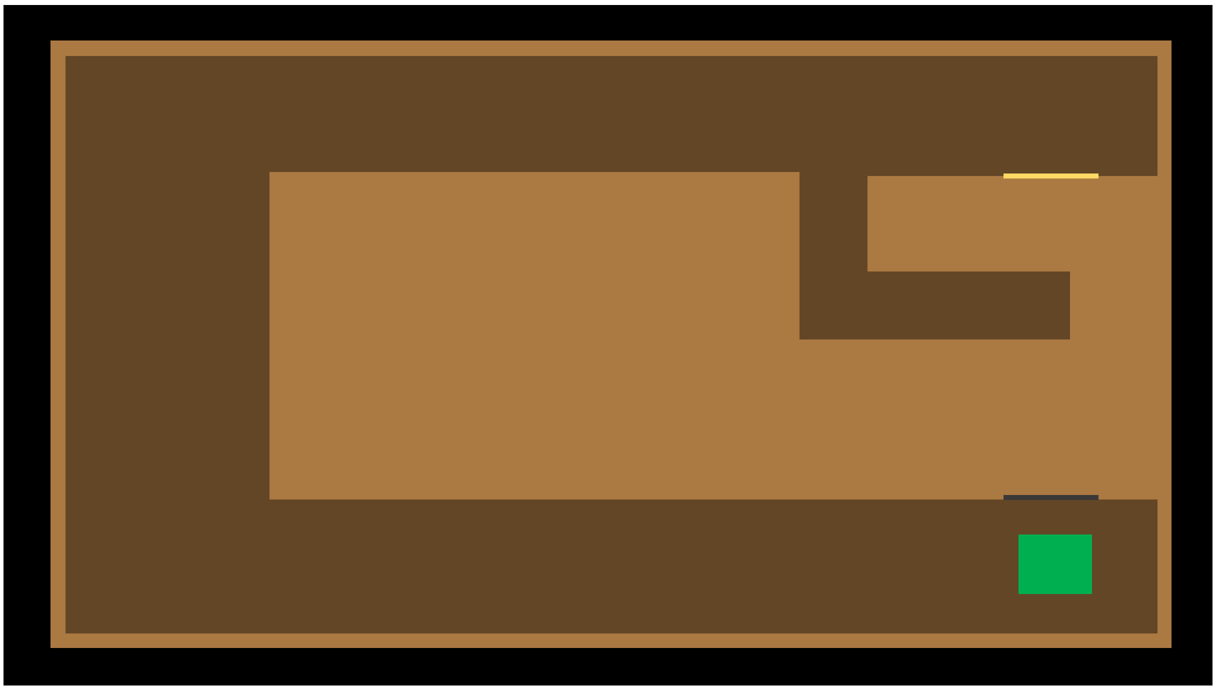

Figure 3: Expected picture to be shown on the monitor

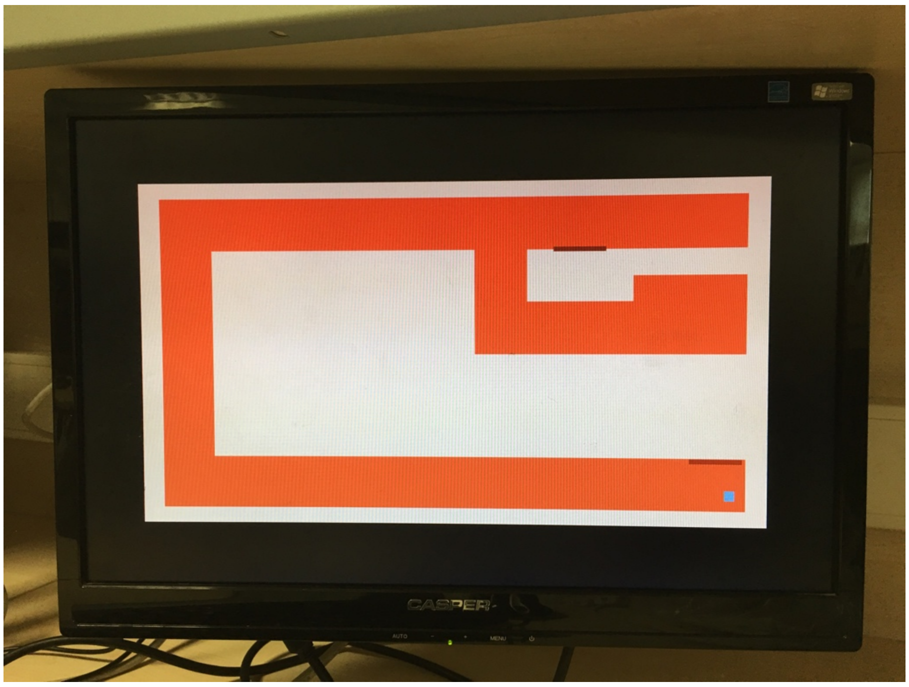
Figure 4: The picture drawn on the monitor

The picture displayed on the monitor is also very similar. However, since the RGB signals are 4-bit signals, there aren't as many color options as there are on power point (which was the program I used to draw the expected picture). I also couldn't find a source which could help me chose colors in 4-bit form, which is another reason why the colors are much different.

- Conclusion

This week's lab was a very important one for me. Like most digital labs that we are assigned to do, it has increased my comfortability on writing and reading VHDL code. I learned how to create procedures and how easy it is to use them for this lab. It was also beneficial for me to learn coding and connecting via VGA since I will be creating a game and displaying it on a screen via VGA. I learned how the VGA connection works, what certain terms mean (front porch, back porch, sync pulse) and what their purpose is. All in all, it was a beneficial assignment that helped me learn many elements of VGA and VHDL.

- Appendix

    - VHDL code for vga_main

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use ieee.numeric_std.all;
```

```
entity vga_main is
  Port ( clk : in std_logic;
  hsync, vsync : out std_logic;
  r, g, b : out std_logic_vector (3 downto 0));
end vga_main;

architecture Behavioral of vga_main is

component vga_controller is
    Port ( clk : in STD_LOGIC;
    hsync, vsync : out STD_LOGIC;
    hcurrent, vcurrent : out integer);
end component;

component vga_draw is
    Port ( clk : in STD_LOGIC;
           hcurrent, vcurrent : in integer;
           r,g,b : out STD_LOGIC_VECTOR (3 downto 0));
end component;
signal hcurrent: integer range 0 to 1904:= 0;
signal vcurrent: integer range 0 to 932:= 0;

begin

controller : vga_controller port map (clk, hsync, vsync, hcurrent, vcurrent);
drawer : vga_draw port map (clk, hcurrent, vcurrent, r, g, b);

end Behavioral;
```

- VHDL code for vga_controller

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

entity vga_controller is
    Port ( clk : in STD_LOGIC;
           hsync, vsync : out STD_LOGIC;
           hcurrent : out integer range 0 to 1904:= 0;
            vcurrent : out integer range 0 to 932:= 0);
end vga_controller;

architecture Behavioral of vga_controller is

signal hpos: integer range 0 to 1904:= 0;
signal vpos: integer range 0 to 932:= 0;

begin


process(clk) begin
```

```
  if rising_edge(clk) then
    if (hpos < 1904) then
      hpos <= hpos + 1;
    else
      hpos <= 0;
      if (vpos < 932) then
        vpos <= vpos + 1;
        else
        vpos <= 0;
      end if;
    end if;
    if (hpos > 80 and hpos < 232) then
      hsync <= '0';
    else
      hsync <= '1';
    end if;
    if (vpos > 1 and vpos < 4) then
      vsync <= '0';
    else
      vsync <= '1';
    end if;

    hcurrent <= hpos;
    vcurrent <= vpos;

  end if;
end process;

end Behavioral;
```

- VHDL code for vga_draw

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use work.draw.all;


entity vga_draw is
  Port ( clk : in STD_LOGIC;
         hcurrent : in integer range 0 to 1904;
         vcurrent : in integer range 0 to 932;
         r,g,b : out STD_LOGIC_VECTOR (3 downto 0));
end vga_draw;

architecture Behavioral of vga_draw is

signal draw : std_logic_vector (8 downto 0);
type RGB is array (8 downto 0) of std_logic_vector (3 downto 0);
signal r_o,g_o,b_o : RGB;
```

```
begin

rectangle( 1200, 660, hcurrent, vcurrent, 584, 152, "1101", "1011", "1100", r_o(0),
g_o(0), b_o(0), draw(0));
rectangle( 1120, 600, hcurrent, vcurrent, 624, 182, "1101", "0100", "0010", r_o(1),
g_o(1), b_o(1), draw(1));
rectangle( 20, 20, hcurrent, vcurrent, 1704, 742, "0110", "1000", "1111", r_o(2), g_o(2),
b_o(2), draw(2));
rectangle( 511, 400, hcurrent, vcurrent, 724, 282, "1101", "1011", "1100", r_o(3), g_o(3),
b_o(3), draw(3));
rectangle( 510, 200, hcurrent, vcurrent, 1234, 482, "1101", "1011", "1100", r_o(4),
g_o(4), b_o(4), draw(4));
rectangle( 201, 100, hcurrent, vcurrent, 1334, 282, "1101", "1011", "1100", r_o(5),
g_o(5), b_o(5), draw(5));
rectangle( 210, 51, hcurrent, vcurrent, 1534, 282, "1101", "1011", "1100", r_o(6), g_o(6),
b_o(6), draw(6));
rectangle( 100, 10, hcurrent, vcurrent, 1384, 277, "1000", "0010", "0000", r_o(7), g_o(7),
b_o(7), draw(7));
rectangle( 99, 10, hcurrent, vcurrent, 1639, 682, "1000", "0010", "0000", r_o(8), g_o(8),
b_o(8), draw(8));


process (clk) begin

  if rising_edge(clk) then

    if (draw(8) = '1') then
      r <= r_o(8);
      g <= g_o(8);
      b <= b_o(8);

    elsif (draw(7) = '1') then
      r <= r_o(7);
      g <= g_o(7);
      b <= b_o(7);

    elsif (draw(6) = '1') then
      r <= r_o(6);
      g <= g_o(6);
      b <= b_o(6);

    elsif (draw(5) = '1') then
      r <= r_o(5);
      g <= g_o(5);
      b <= b_o(5);

    elsif (draw(4) = '1') then
      r <= r_o(4);
      g <= g_o(4);
      b <= b_o(4);
```

```
        elsif (draw(3) = '1') then
            r <= r_o(3);
            g <= g_o(3);
            b <= b_o(3);


        elsif (draw(2) = '1') then
            r <= r_o(2);
            g <= g_o(2);
            b <= b_o(2);

        elsif (draw(1) = '1') then
            r <= r_o(1);
            g <= g_o(1);
            b <= b_o(1);

        elsif (draw(0) = '1') then
            r <= r_o(0);
            g <= g_o(0);
            b <= b_o(0);

        else
            r <= (others => '0');
            g <= (others => '0');
            b <= (others => '0');
        end if;
    end if;

end process;

end Behavioral;
```

- VHDL code for draw_packages

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

package draw is
    procedure rectangle( xside, yside: in integer;
     hcurrent, vcurrent, xpos, ypos: in integer;
     r, g, b: in std_logic_vector (3 downto 0);
    signal r_o, g_o, b_o: out std_logic_vector (3 downto 0);
    signal draw: out std_logic);


end draw;

package body draw is

    procedure rectangle( xside, yside: in integer;
     hcurrent, vcurrent, xpos, ypos: in integer;
    r, g, b: in std_logic_vector (3 downto 0);
    signal r_o, g_o, b_o: out std_logic_vector (3 downto 0);
    signal draw: out std_logic) is
    begin
        if ((hcurrent > xpos and hcurrent < (xpos + xside)) and (vcurrent > ypos and vcurrent
< (ypos + yside))) then
            r_o <= r;
            g_o <= g;
            b_o <= b;
            draw <= '1';
        else
            draw <= '0';
        end if;
    end rectangle;

end draw;
```