Universidad Nacional
Autonoma de Mexico
Facultad de ingeniería

Diseño Digital VLSI

Grupo : 4

Práctica: 9

DISEÑO DE UN TRANSMISOR PARA COMUNICACIÓN SERIAL

Edgar Daniel Barcenas Martinez

OBJETIVO:

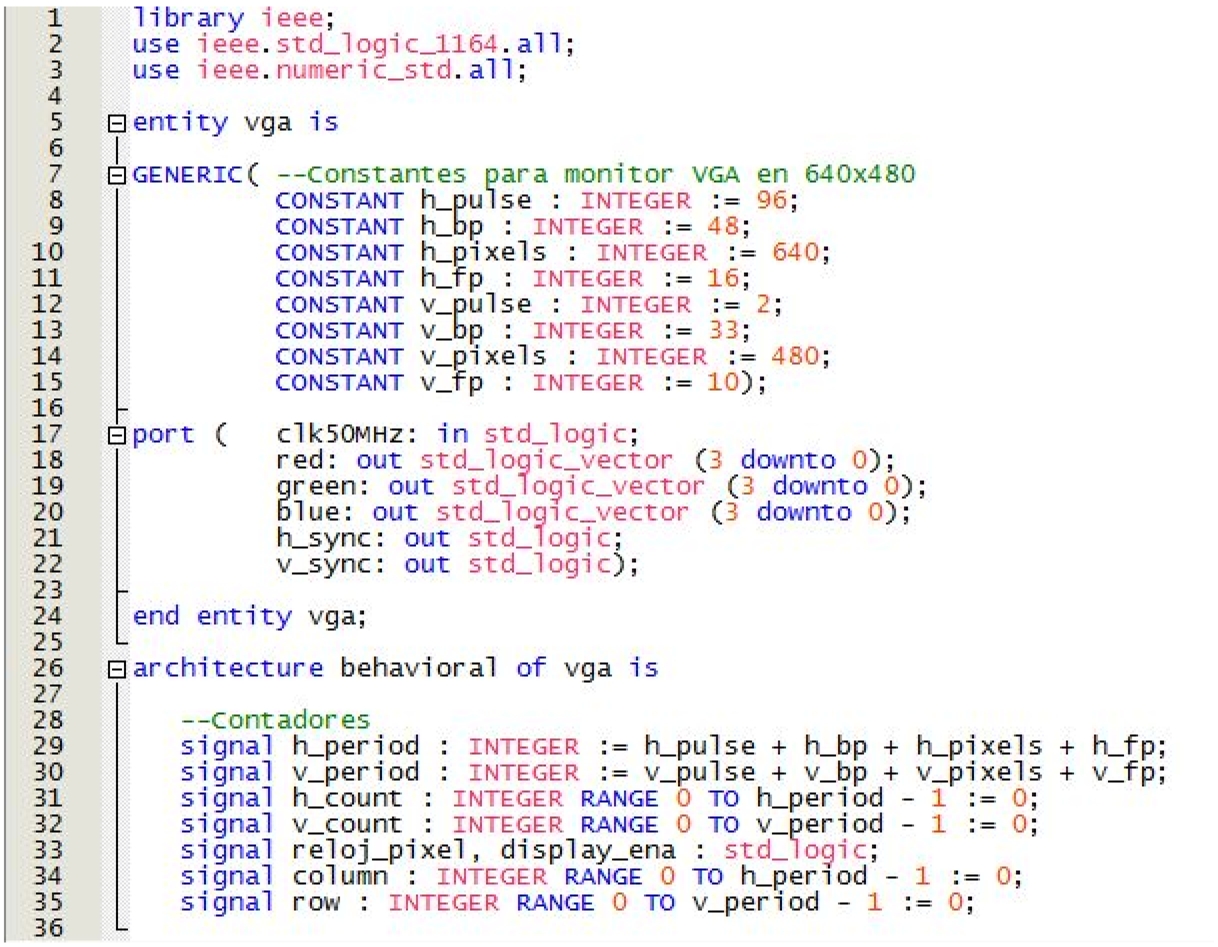
El alumno aprenderá como es la señalización para generar video VGA con el fin de comprender la funcionalidad del código que controla el mencionado tipo de señal de video.

ESPECIFICACIONES:

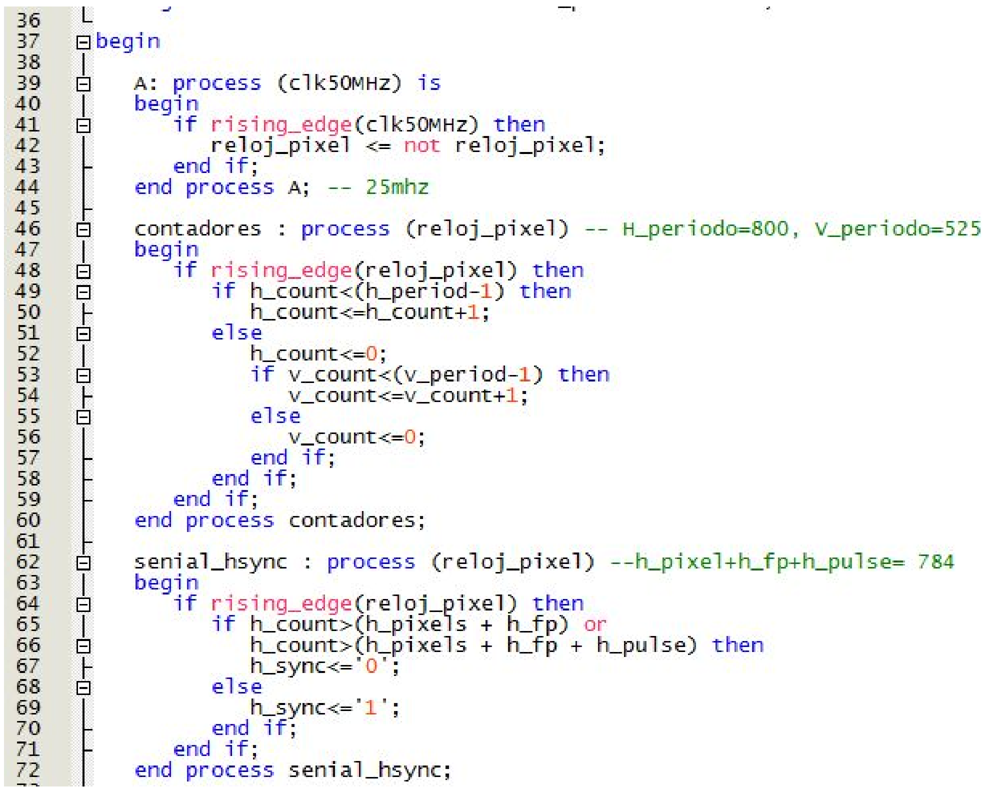
Utilizando un FPGA, un cable y pantalla VGA, se programará el controlador de video VGA, con la finalidad de proyectar una imagen estática.

Como se observa en el diagrama de bloques de la figura 10.1, el sistema tiene una entrada de reloj, y cinco salidas HS, VS, R, G y B.

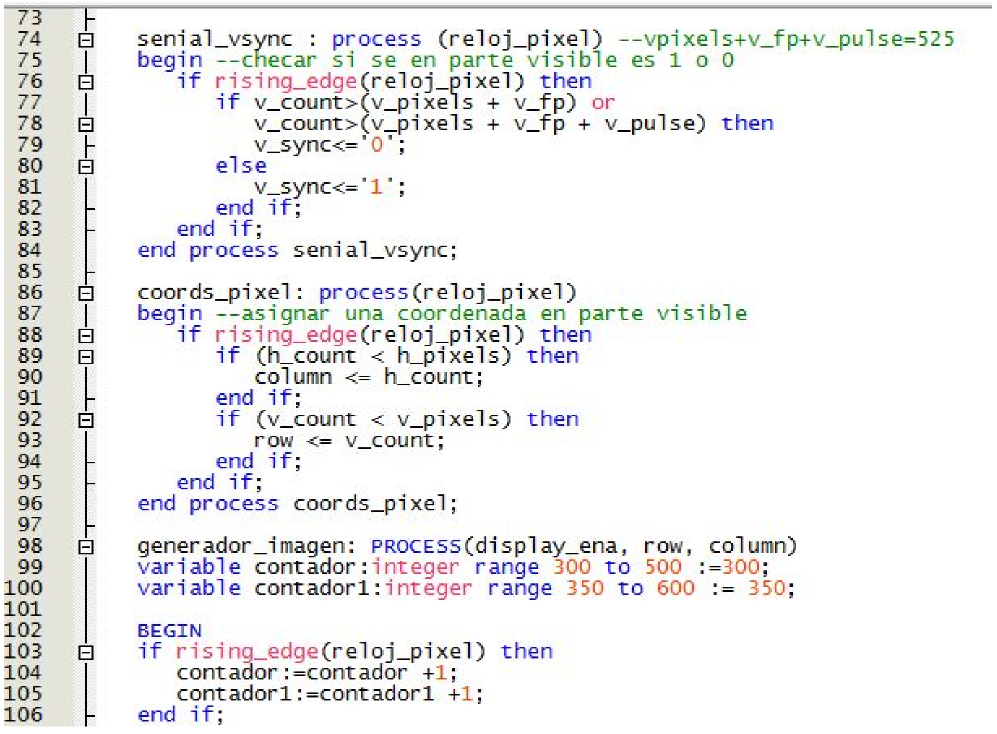
DESARROLLO:



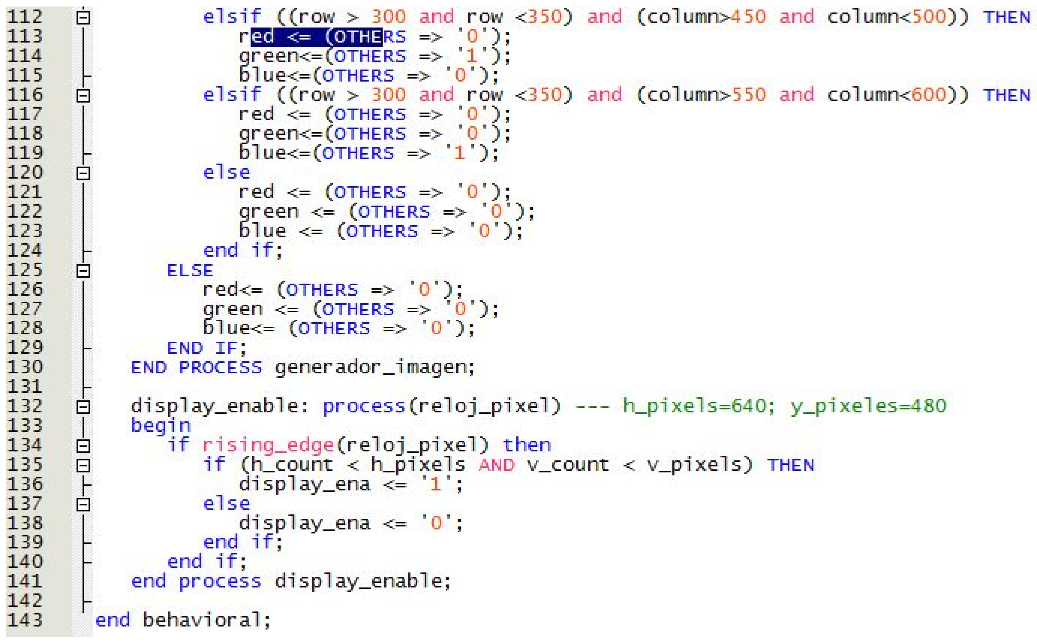
```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity vga is

GENERIC( --Constantes para monitor VGA en 640x480
         CONSTANT h_pulse : INTEGER := 96;
         CONSTANT h_bp : INTEGER := 48;
         CONSTANT h_pixels : INTEGER := 640;
         CONSTANT h_fp : INTEGER := 16;
         CONSTANT v_pulse : INTEGER := 2;
         CONSTANT v_bp : INTEGER := 33;
         CONSTANT v_pixels : INTEGER := 480;
         CONSTANT v_fp : INTEGER := 10);

port (   clk50MHz: in std_logic;
         red: out std_logic_vector (3 downto 0);
         green: out std_logic_vector (3 downto 0);
         blue: out std_logic_vector (3 downto 0);
         h_sync: out std_logic;
         v_sync: out std_logic);

end entity vga;

architecture behavioral of vga is

   --Contadores
   signal h_period : INTEGER := h_pulse + h_bp + h_pixels + h_fp;
   signal v_period : INTEGER := v_pulse + v_bp + v_pixels + v_fp;
   signal h_count : INTEGER RANGE 0 TO h_period - 1 := 0;
   signal v_count : INTEGER RANGE 0 TO v_period - 1 := 0;
   signal reloj_pixel, display_ena : std_logic;
   signal column : INTEGER RANGE 0 TO h_period - 1 := 0;
   signal row : INTEGER RANGE 0 TO v_period - 1 := 0;

```



```vhdl
begin

   A: process (clk50MHz) is
   begin
      if rising_edge(clk50MHz) then
         reloj_pixel <= not reloj_pixel;
      end if;
   end process A; -- 25mhz

   contadores : process (reloj_pixel) -- H_periodo=800, V_periodo=525
   begin
      if rising_edge(reloj_pixel) then
         if h_count<(h_period-1) then
            h_count<=h_count+1;
         else
            h_count<=0;
            if v_count<(v_period-1) then
               v_count<=v_count+1;
            else
               v_count<=0;
            end if;
         end if;
      end if;
   end process contadores;

   senial_hsync : process (reloj_pixel) --h_pixel+h_fp+h_pulse= 784
   begin
      if rising_edge(reloj_pixel) then
         if h_count>(h_pixels + h_fp) or
            h_count>(h_pixels + h_fp + h_pulse) then
            h_sync<='0';
         else
            h_sync<='1';
         end if;
      end if;
   end process senial_hsync;
```

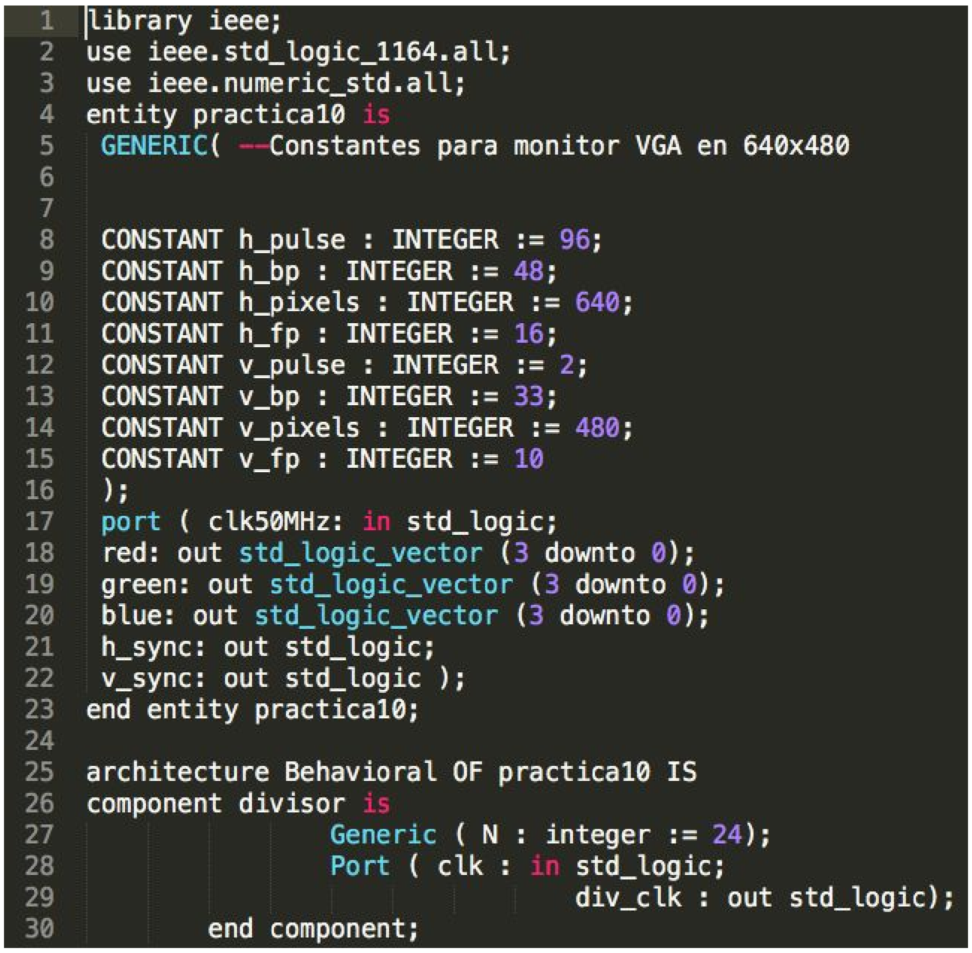


```vhdl

   senial_vsync : process (reloj_pixel) --vpixels+v_fp+v_pulse=525
   begin --checar si se en parte visible es 1 o 0
      if rising_edge(reloj_pixel) then
         if v_count>(v_pixels + v_fp) or
            v_count>(v_pixels + v_fp + v_pulse) then
            v_sync<='0';
         else
            v_sync<='1';
         end if;
      end if;
   end process senial_vsync;

   coords_pixel: process(reloj_pixel)
   begin --asignar una coordenada en parte visible
      if rising_edge(reloj_pixel) then
         if (h_count < h_pixels) then
            column <= h_count;
         end if;
         if (v_count < v_pixels) then
            row <= v_count;
         end if;
      end if;
   end process coords_pixel;

   generador_imagen: PROCESS(display_ena, row, column)
   variable contador:integer range 300 to 500 :=300;
   variable contador1:integer range 350 to 600 := 350;

   BEGIN
   if rising_edge(reloj_pixel) then
      contador:=contador +1;
      contador1:=contador1 +1;
   end if;
```

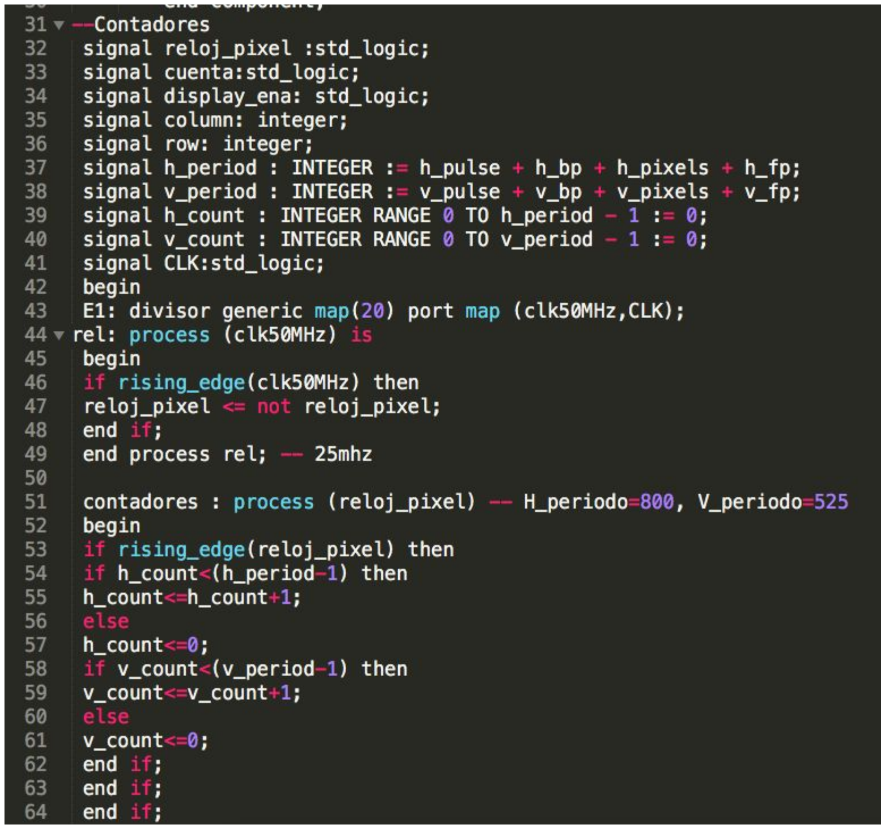


```
         elsif ((row > 300 and row <350) and (column>450 and column<500)) THEN
            red <= (OTHERS => '0');
            green<=(OTHERS => '1');
            blue<=(OTHERS => '0');
         elsif ((row > 300 and row <350) and (column>550 and column<600)) THEN
            red <= (OTHERS => '0');
            green<=(OTHERS => '0');
            blue<=(OTHERS => '1');
         else
            red <= (OTHERS => '0');
            green <= (OTHERS => '0');
            blue <= (OTHERS => '0');
         end if;
      ELSE
         red<= (OTHERS => '0');
         green <= (OTHERS => '0');
         blue<= (OTHERS => '0');
      END IF;
   END PROCESS generador_imagen;

   display_enable: process(reloj_pixel) --- h_pixels=640; y_pixeles=480
   begin
      if rising_edge(reloj_pixel) then
         if (h_count < h_pixels AND v_count < v_pixels) THEN
            display_ena <= '1';
         else
            display_ena <= '0';
         end if;
      end if;
   end process display_enable;

end behavioral;
```

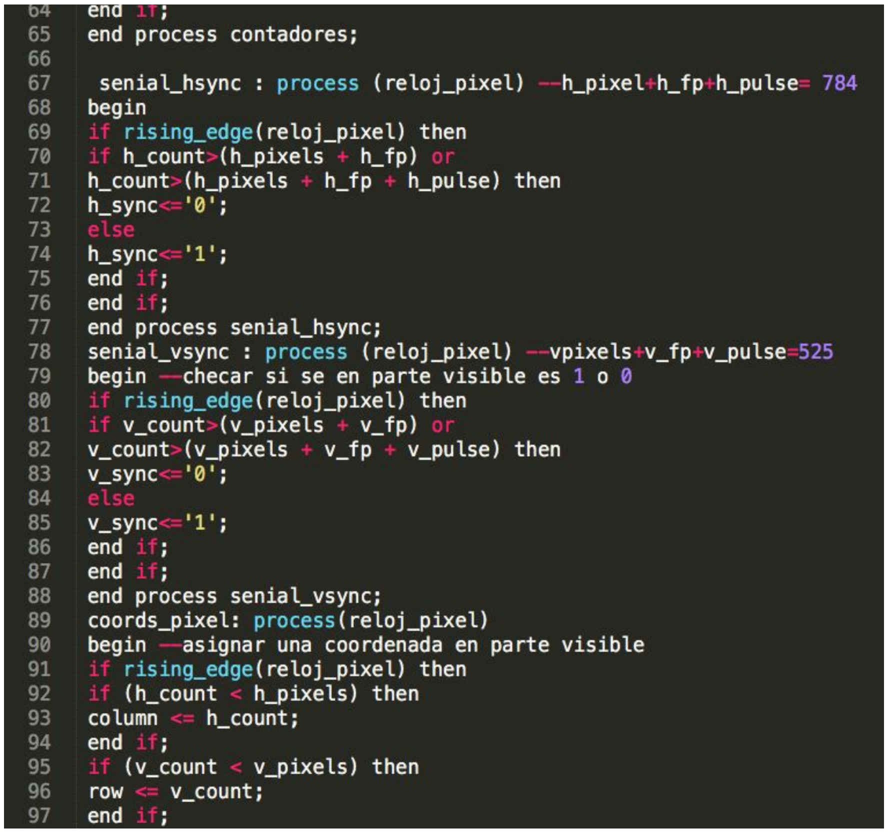
Practica Complementaria:



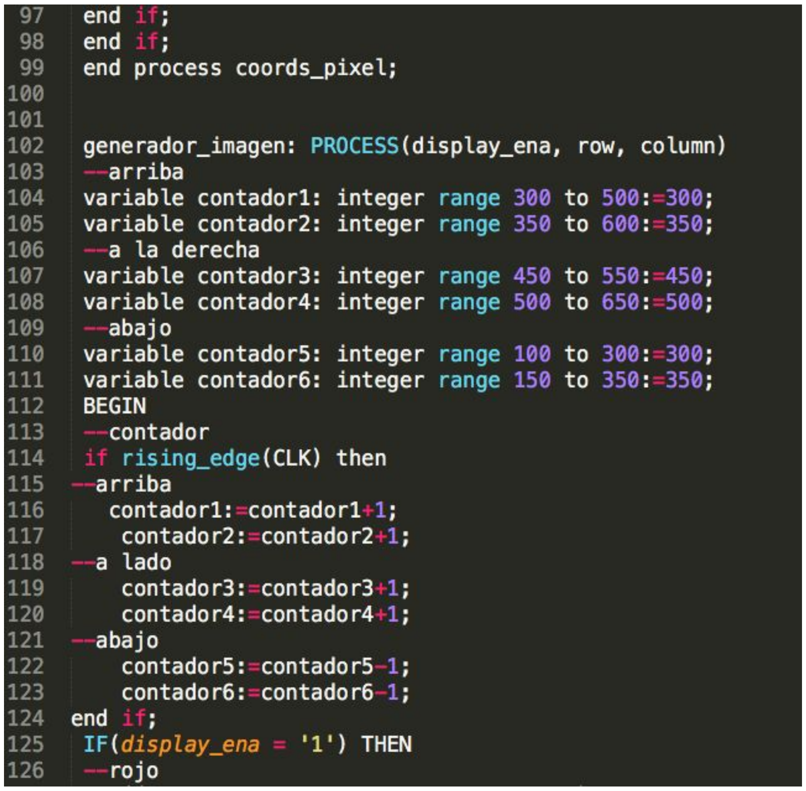
```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
entity practica10 is
 GENERIC( --Constantes para monitor VGA en 640x480


 CONSTANT h_pulse : INTEGER := 96;
 CONSTANT h_bp : INTEGER := 48;
 CONSTANT h_pixels : INTEGER := 640;
 CONSTANT h_fp : INTEGER := 16;
 CONSTANT v_pulse : INTEGER := 2;
 CONSTANT v_bp : INTEGER := 33;
 CONSTANT v_pixels : INTEGER := 480;
 CONSTANT v_fp : INTEGER := 10
 );
 port ( clk50MHz: in std_logic;
 red: out std_logic_vector (3 downto 0);
 green: out std_logic_vector (3 downto 0);
 blue: out std_logic_vector (3 downto 0);
 h_sync: out std_logic;
 v_sync: out std_logic );
end entity practica10;

architecture Behavioral OF practica10 IS
component divisor is
        Generic ( N : integer := 24);
        Port ( clk : in std_logic;
                div_clk : out std_logic);
    end component;
```



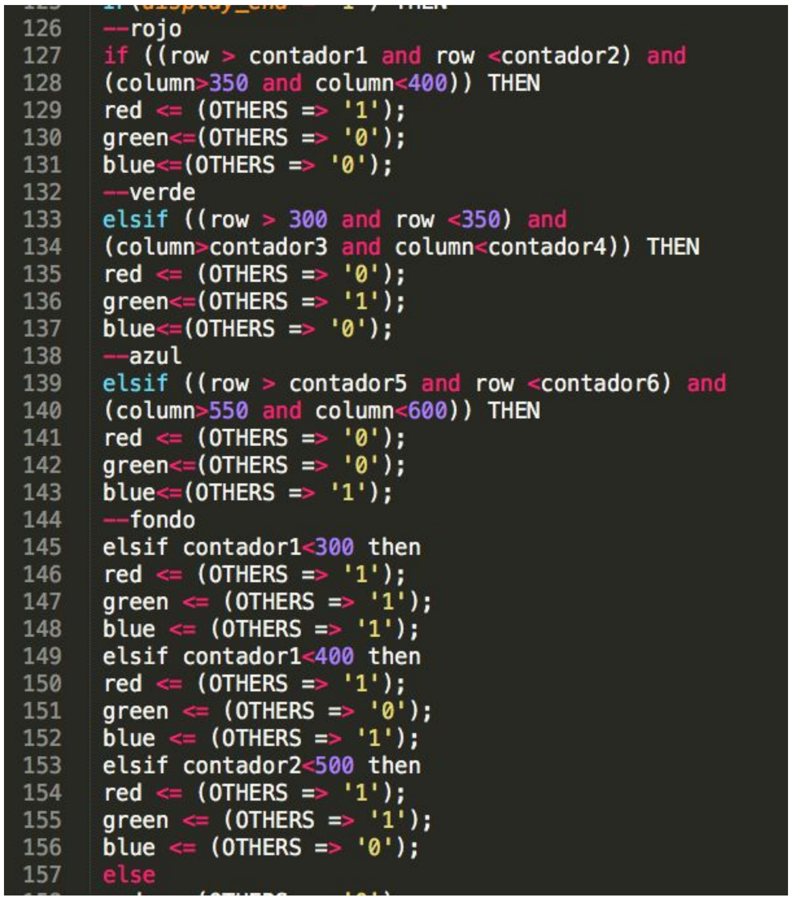
```
--Contadores
 signal reloj_pixel :std_logic;
 signal cuenta:std_logic;
 signal display_ena: std_logic;
 signal column: integer;
 signal row: integer;
 signal h_period : INTEGER := h_pulse + h_bp + h_pixels + h_fp;
 signal v_period : INTEGER := v_pulse + v_bp + v_pixels + v_fp;
 signal h_count : INTEGER RANGE 0 TO h_period - 1 := 0;
 signal v_count : INTEGER RANGE 0 TO v_period - 1 := 0;
 signal CLK:std_logic;
 begin
 E1: divisor generic map(20) port map (clk50MHz,CLK);
rel: process (clk50MHz) is
 begin
 if rising_edge(clk50MHz) then
 reloj_pixel <= not reloj_pixel;
 end if;
 end process rel; -- 25mhz

 contadores : process (reloj_pixel) -- H_periodo=800, V_periodo=525
 begin
 if rising_edge(reloj_pixel) then
 if h_count<(h_period-1) then
 h_count<=h_count+1;
 else
 h_count<=0;
 if v_count<(v_period-1) then
 v_count<=v_count+1;
 else
 v_count<=0;
 end if;
 end if;
 end if;
```



```
 end process contadores;

  senial_hsync : process (reloj_pixel) --h_pixel+h_fp+h_pulse= 784
 begin
 if rising_edge(reloj_pixel) then
 if h_count>(h_pixels + h_fp) or
 h_count>(h_pixels + h_fp + h_pulse) then
 h_sync<='0';
 else
 h_sync<='1';
 end if;
 end if;
 end process senial_hsync;
 senial_vsync : process (reloj_pixel) --vpixels+v_fp+v_pulse=525
 begin --checar si se en parte visible es 1 o 0
 if rising_edge(reloj_pixel) then
 if v_count>(v_pixels + v_fp) or
 v_count>(v_pixels + v_fp + v_pulse) then
 v_sync<='0';
 else
 v_sync<='1';
 end if;
 end if;
 end process senial_vsync;
 coords_pixel: process(reloj_pixel)
 begin --asignar una coordenada en parte visible
 if rising_edge(reloj_pixel) then
 if (h_count < h_pixels) then
 column <= h_count;
 end if;
 if (v_count < v_pixels) then
 row <= v_count;
 end if;
```

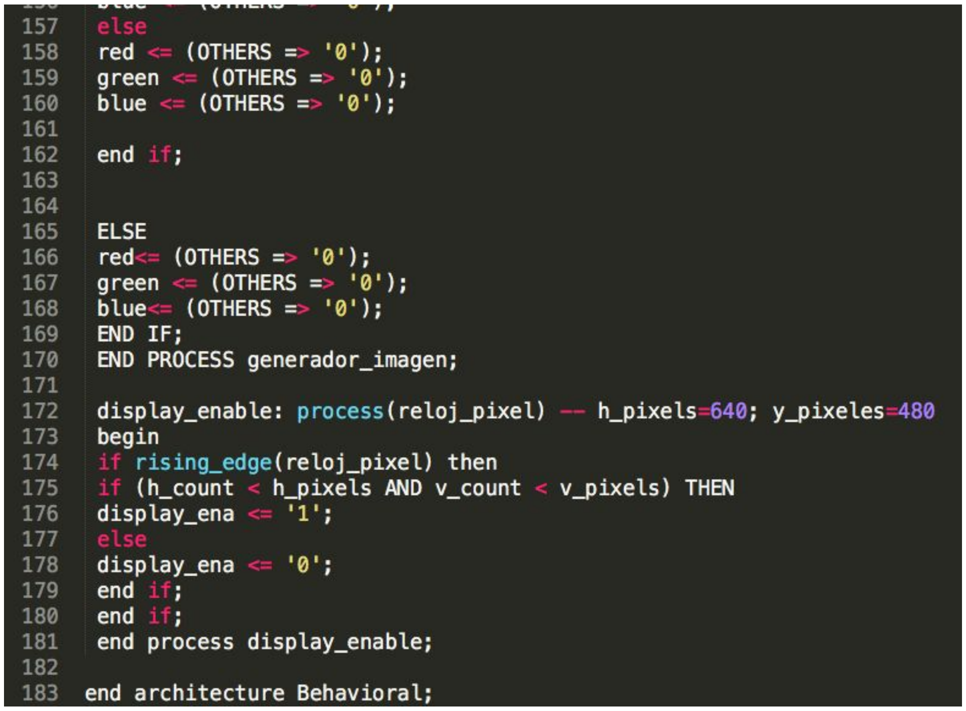


```
 end if;
 end if;
 end process coords_pixel;


 generador_imagen: PROCESS(display_ena, row, column)
 --arriba
 variable contador1: integer range 300 to 500:=300;
 variable contador2: integer range 350 to 600:=350;
 --a la derecha
 variable contador3: integer range 450 to 550:=450;
 variable contador4: integer range 500 to 650:=500;
 --abajo
 variable contador5: integer range 100 to 300:=300;
 variable contador6: integer range 150 to 350:=350;
 BEGIN
 --contador
 if rising_edge(CLK) then
--arriba
   contador1:=contador1+1;
    contador2:=contador2+1;
--a lado
    contador3:=contador3+1;
    contador4:=contador4+1;
--abajo
    contador5:=contador5-1;
    contador6:=contador6-1;
end if;
 IF(display_ena = '1') THEN
 --rojo
```



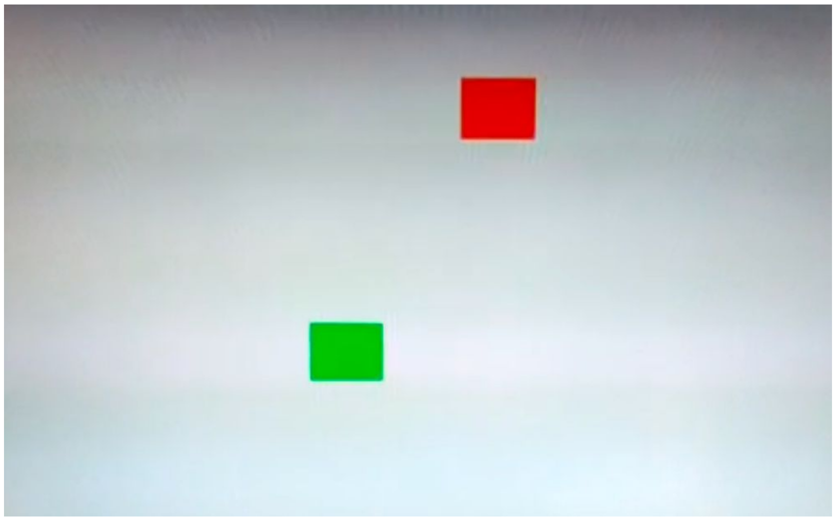
```
--rojo
```

```
if ((row > contador1 and row <contador2) and
(column>350 and column<400)) THEN
red <= (OTHERS => '1');
green<=(OTHERS => '0');
blue<=(OTHERS => '0');
--verde
elsif ((row > 300 and row <350) and
(column>contador3 and column<contador4)) THEN
red <= (OTHERS => '0');
green<=(OTHERS => '1');
blue<=(OTHERS => '0');
--azul
elsif ((row > contador5 and row <contador6) and
(column>550 and column<600)) THEN
red <= (OTHERS => '0');
green<=(OTHERS => '0');
blue<=(OTHERS => '1');
--fondo
elsif contador1<300 then
red <= (OTHERS => '1');
green <= (OTHERS => '1');
blue <= (OTHERS => '1');
elsif contador1<400 then
red <= (OTHERS => '1');
green <= (OTHERS => '0');
blue <= (OTHERS => '1');
elsif contador2<500 then
red <= (OTHERS => '1');
green <= (OTHERS => '1');
blue <= (OTHERS => '0');
else
```



```
 else
 red <= (OTHERS => '0');
 green <= (OTHERS => '0');
 blue <= (OTHERS => '0');

 end if;


 ELSE
 red<= (OTHERS => '0');
 green <= (OTHERS => '0');
 blue<= (OTHERS => '0');
 END IF;
 END PROCESS generador_imagen;

 display_enable: process(reloj_pixel) -- h_pixels=640; y_pixeles=480
 begin
 if rising_edge(reloj_pixel) then
 if (h_count < h_pixels AND v_count < v_pixels) THEN
 display_ena <= '1';
 else
 display_ena <= '0';
 end if;
 end if;
 end process display_enable;

end architecture Behavioral;
```

Resultados:



CONCLUSIONES:

En esta práctica aprendí a programar un controlador de video VGA, en el cual aprendi a generar una imagen estática. Aprendí el funcionamiento de las señales de video VGA para poder mostrar imágenes y video.