Universidad Nacional Autonoma de Mexico Facultad de ingeniería

Diseño Digital VLSI

Grupo : 4

Práctica: 10

SIMULACIÓN DE UN LED DE 7 SEGMENTOS

Edgar Daniel Barcenas Martinez

OBJETIVO:

El alumno utilizará el controlador VGA de la práctica anterior para implementar la emulación de un display de 7 segmentos. Cada segmento es visualizado como un rectángulo.

ESPECIFICACIONES:

Utilizando un FPGA, un cable y un monitor con entrada VGA, se diseñará un sistema en el que su entrada sea un número binario de cuatro bits y su salida sea la visualización de ese número en un display de 7segmentos en de un monitor.

DIAGRAMA DE BLOQUES:

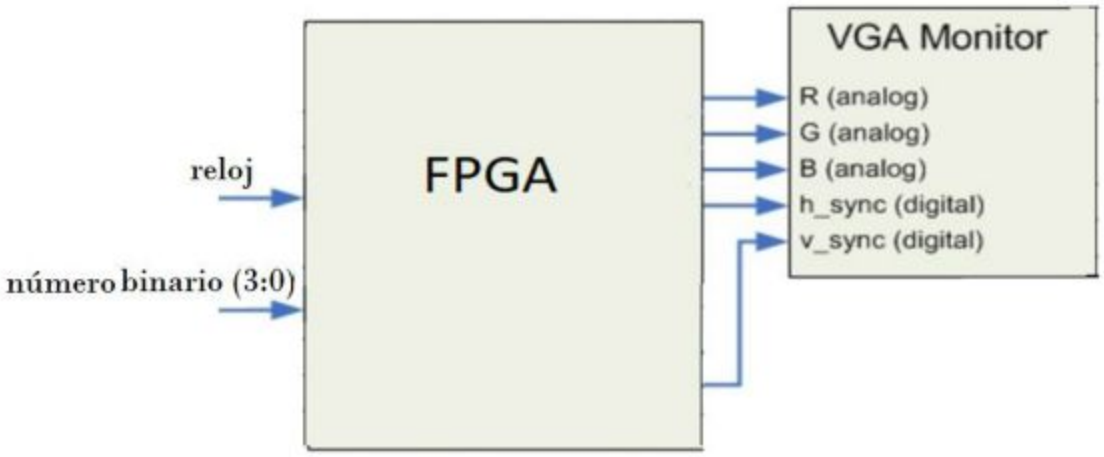

Figura 11.1. Diagrama de Bloques del sistema Emulador de Display 7 Segmentos en Monitor

DIAGRAMA DE BLOQUES FUNCIONALES:

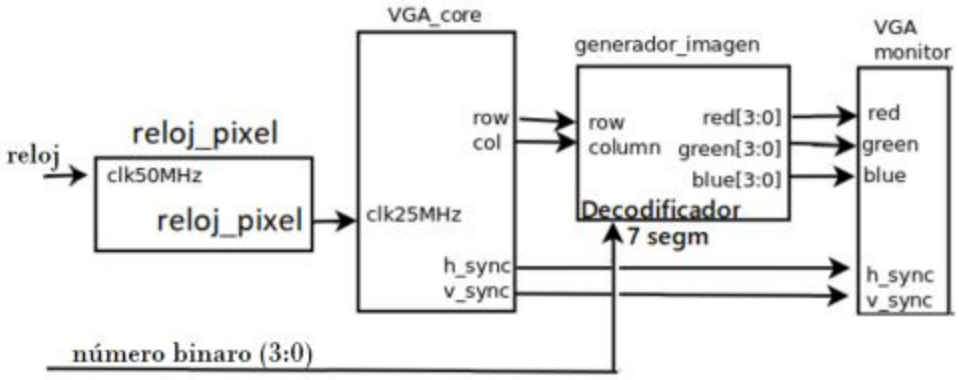

Figura 11.2. Bloques funcionales del sistema Emulador de Display 7 segmentos en Monitor

DESARROLLO:

```
1   LIBRARY IEEE;
2   USE IEEE.STD_LOGIC_1164.ALL;
3   USE IEEE.NUMERIC_STD.ALL;
4
5   ENTITY emu7Seg IS
6   port(
7       clk50MHz : IN std_logic;
8       red      : OUT std_logic_vector (3 DOWNTO 0);       -- al monitor
9       green    : OUT std_logic_vector (3 DOWNTO 0);
10      blue     : OUT std_logic_vector (3 DOWNTO 0);
11      h_sync   : OUT std_logic;
12      v_sync   : OUT std_logic;
13      dipsw    : IN std_logic_vector(3 DOWNTO 0);      -- numeros para decodificador
14      display  : OUT STD_LOGIC_VECTOR (6 DOWNTO 0) -- al display de la FPGA
15  );
16  END ENTITY emu7Seg;
17
18  ARCHITECTURE behavioral OF emu7Seg IS
19
20      CONSTANT h_pulse : INTEGER := 96;
21      CONSTANT h_bp : INTEGER := 48;
22      CONSTANT h_pixels : INTEGER := 640;
23      CONSTANT h_fp : INTEGER := 16;
24      CONSTANT v_pulse : INTEGER := 2;
25      CONSTANT v_bp : INTEGER := 33;
26      CONSTANT v_pixels : INTEGER := 480;
27      CONSTANT v_fp : INTEGER := 10;
28
29      --Contadores
30      signal h_period : INTEGER := h_pulse + h_bp + h_pixels + h_fp;
31      signal v_period : INTEGER := v_pulse + v_bp + v_pixels + v_fp;
32      signal h_count : INTEGER RANGE 0 TO h_period - 1 := 0;
33      signal v_count : INTEGER RANGE 0 TO v_period - 1 := 0;
34      signal reloj_pixel, display_ena : std_logic;
35      signal column : INTEGER RANGE 0 TO h_period - 1 := 0;
36      signal row : INTEGER RANGE 0 TO v_period - 1 := 0;
37
38      constant cero: std_logic_vector(6 downto 0):="0111111"; --gfedcba
39      constant uno: std_logic_vector(6 downto 0):="0000110";
40      constant dos: std_logic_vector(6 downto 0):="1011011";
```

```
    constant dos: std_logic_vector(6 downto 0):="1011011";
    constant tres: std_logic_vector(6 downto 0):="1001111";
    constant cuatro: std_logic_vector(6 downto 0):="1100110";
    constant cinco: std_logic_vector(6 downto 0):="1101101";
    constant seis: std_logic_vector(6 downto 0):="1111101";
    constant siete: std_logic_vector(6 downto 0):="0000111";
    constant ocho: std_logic_vector(6 downto 0):="1111111";
    constant nueve: std_logic_vector(6 downto 0):="1100111";

    --constant r1:std_logic_vector(3 downto 0):=(OTHERS => '1');
    --constant r0:std_logic_vector(3 downto 0):=(OTHERS => '0');
    --constant g1:std_logic_vector(3 downto 0):=(OTHERS => '1');
    --constant g0:std_logic_vector(3 downto 0):=(OTHERS => '0');
    --constant b1:std_logic_vector(3 downto 0):=(OTHERS => '1');
    --constant b0:std_logic_vector(3 downto 0):=(OTHERS => '0');   -- variable a,b,c,d,e,f: std_logic;

    signal conectornum:std_logic_vector(6 downto 0);            -- conexion del decodificador con image_gen

BEGIN

    divisor: process (clk50MHz) is
    begin
       if rising_edge(clk50MHz) then
          reloj_pixel <= not reloj_pixel;
       end if;
    end process divisor; -- 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;
```

```
    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)
    BEGIN
       IF(display_ena = '1') THEN

          display <= not(conectornum);

          case conectornum is

             when cero=>
                if ((row > 200 and row <210) and (column>110 and column<140)) THEN -- a azul
                   red <= (OTHERS => '0');
                   green <= (OTHERS => '0');
                   blue <= (OTHERS => '1');
                elsif ((row > 210 and row <240) and (column>140 and column<150)) THEN -- b verde
                   red <= (OTHERS => '0');
                   green <= (OTHERS => '1');
                   blue <= (OTHERS => '0');
                elsif ((row > 250 and row <280) and (column>140 and column<150)) THEN -- c rojo
                   red <= (OTHERS => '1');
                   green <= (OTHERS => '0');
                   blue <= (OTHERS => '0');
                elsif ((row > 280 and row <290) and (column>110 and column<140)) THEN -- d blanco
                   Red <= (OTHERS => '1');
                   green <= (OTHERS => '1');
                   blue <= (OTHERS => '1');
                elsif ((row > 250 and row <280) and (column>100 and column<110)) THEN -- e cian
                   Red <= (OTHERS => '0');
                   Green <= (OTHERS => '1');
                   Blue <= (OTHERS => '1');
                   Blue <= (OTHERS => '1');
                elsif ((row > 210 and row <240) and (column>100 and column<110)) THEN -- f amarillo
                   red <= (OTHERS => '1');
                   green <= (OTHERS => '1');
                   blue <= (OTHERS => '0');
                else -- fondo
                   red <= (OTHERS => '0');
                   green <= (OTHERS => '0');
                   blue <= (OTHERS => '0');
                end if;

             when uno=>
                if ((row > 210 and row <240) and (column>140 and column<150)) THEN -- b verde
                   red <= (OTHERS => '0');
                   green <= (OTHERS => '1');
                   blue <= (OTHERS => '0');
                elsif ((row > 250 and row <280) and (column>140 and column<150)) THEN -- c rojo
                   red <= (OTHERS => '1');
                   green <= (OTHERS => '0');
                   blue <= (OTHERS => '0');
                else -- fondo
                   red <= (OTHERS => '0');
                   green <= (OTHERS => '0');
                   blue <= (OTHERS => '0');
                end if;

             when dos=>
                if ((row > 200 and row <210) and (column>110 and column<140)) THEN -- a azul
                   red <= (OTHERS => '0');
                   green <= (OTHERS => '0');
                   blue <= (OTHERS => '1');
                elsif ((row > 210 and row <240) and (column>140 and column<150)) THEN -- b verde
                   Red <= (OTHERS => '0');
                   green <= (OTHERS => '1');
```

```
                green <= (OTHERS => '1');
                blue <= (OTHERS => '0');
            elsif ((row > 280 and row <290) and (column>110 and column<140)) THEN -- d blanco
                Red <= (OTHERS => '1');
                green <= (OTHERS => '1');
                blue <= (OTHERS => '1');
            elsif ((row > 250 and row <280) and (column>100 and column<110)) THEN -- e cian
                Red <= (OTHERS => '0');
                Green <= (OTHERS => '1');
                Blue <= (OTHERS => '1');
            elsif ((row > 240 and row <250) and (column>110 and column<140)) THEN -- g violeta
                red <= (OTHERS => '1');
                green <= (OTHERS => '0');
                blue <= (OTHERS => '1');
            else -- fondo
                red <= (OTHERS => '0');
                green <= (OTHERS => '0');
                blue <= (OTHERS => '0');
            end if;

         when tres=>
            if ((row > 200 and row <210) and (column>110 and column<140)) THEN -- a azul
                red <= (OTHERS => '0');
                green <= (OTHERS => '0');
                blue <= (OTHERS => '1');
            elsif ((row > 210 and row <240) and (column>140 and column<150)) THEN -- b verde
                Red <= (OTHERS => '0');
                green <= (OTHERS => '1');
                blue <= (OTHERS => '0');
            elsif ((row > 250 and row <280) and (column>140 and column<150)) THEN -- c rojo
                red <= (OTHERS => '1');
                green <= (OTHERS => '0');
                blue <= (OTHERS => '0');
            elsif ((row > 280 and row <290) and (column>110 and column<140)) THEN -- d blanco
                Red <= (OTHERS => '1');
                green <= (OTHERS => '1');
                blue <= (OTHERS => '1');
```

```
            elsif ((row > 240 and row <250) and (column>110 and column<140)) THEN -- g violeta
                red <= (OTHERS => '1');
                green <= (OTHERS => '0');
                blue <= (OTHERS => '1');
            else -- fondo
                red <= (OTHERS => '0');
                green <= (OTHERS => '0');
                blue <= (OTHERS => '0');
            end if;

         when cuatro=>
            if ((row > 210 and row <240) and (column>140 and column<150)) THEN -- b verde
                Red <= (OTHERS => '0');
                green <= (OTHERS => '1');
                blue <= (OTHERS => '0');
            elsif ((row > 250 and row <280) and (column>140 and column<150)) THEN -- c rojo
                red <= (OTHERS => '1');
                green <= (OTHERS => '0');
                blue <= (OTHERS => '0');
            elsif ((row > 210 and row <240) and (column>100 and column<110)) THEN -- f amarillo
                red <= (OTHERS => '1');
                green <= (OTHERS => '1');
                blue <= (OTHERS => '0');
            elsif ((row > 240 and row <250) and (column>110 and column<140)) THEN -- g violeta
                red <= (OTHERS => '1');
                green <= (OTHERS => '0');
                blue <= (OTHERS => '1');
            else -- fondo
                red <= (OTHERS => '0');
                green <= (OTHERS => '0');
                blue <= (OTHERS => '0');
            end if;

         when cinco=>
            if ((row > 200 and row <210) and (column>110 and column<140)) THEN -- a azul
                red <= (OTHERS => '0');
                green <= (OTHERS => '0');
                blue <= (OTHERS => '1');
```

```
254                 blue <= (OTHERS => '1');
255             elsif ((row > 250 and row <280) and (column>140 and column<150)) THEN -- c rojo
256                 red <= (OTHERS => '1');
257                 green <= (OTHERS => '0');
258                 blue <= (OTHERS => '0');
259             elsif ((row > 280 and row <290) and (column>110 and column<140)) THEN -- d blanco
260                 Red <= (OTHERS => '1');
261                 green <= (OTHERS => '1');
262                 blue <= (OTHERS => '1');
263             elsif ((row > 210 and row <240) and (column>100 and column<110)) THEN -- f amarillo
264                 red <= (OTHERS => '1');
265                 green <= (OTHERS => '1');
266                 blue <= (OTHERS => '0');
267             elsif ((row > 240 and row <250) and (column>110 and column<140)) THEN -- g violeta
268                 red <= (OTHERS => '1');
269                 green <= (OTHERS => '0');
270                 blue <= (OTHERS => '1');
271             else -- fondo
272                 red <= (OTHERS => '0');
273                 green <= (OTHERS => '0');
274                 blue <= (OTHERS => '0');
275             end if;
276 
277         when seis=>
278             if ((row > 200 and row <210) and (column>110 and column<140)) THEN -- a azul
279                 red <= (OTHERS => '0');
280                 green <= (OTHERS => '0');
281                 blue <= (OTHERS => '1');
282             elsif ((row > 250 and row <280) and (column>140 and column<150)) THEN -- c rojo
283                 red <= (OTHERS => '1');
284                 green <= (OTHERS => '0');
285                 blue <= (OTHERS => '0');
286             elsif ((row > 280 and row <290) and (column>110 and column<140)) THEN -- d blanco
287                 Red <= (OTHERS => '1');
```

```
287                 Red <= (OTHERS => '1');
288                 green <= (OTHERS => '1');
289                 blue <= (OTHERS => '1');
290             elsif ((row > 250 and row <280) and (column>100 and column<110)) THEN -- e cian
291                 Red <= (OTHERS => '0');
292                 Green <= (OTHERS => '1');
293                 Blue <= (OTHERS => '1');
294             elsif ((row > 210 and row <240) and (column>100 and column<110)) THEN -- f amarillo
295                 red <= (OTHERS => '1');
296                 green <= (OTHERS => '1');
297                 blue <= (OTHERS => '0');
298             elsif ((row > 240 and row <250) and (column>110 and column<140)) THEN -- g violeta
299                 red <= (OTHERS => '1');
300                 green <= (OTHERS => '0');
301                 blue <= (OTHERS => '1');
302             else -- fondo
303                 red <= (OTHERS => '0');
304                 green <= (OTHERS => '0');
305                 blue <= (OTHERS => '0');
306             end if;
307 
308          when siete=>
309             if ((row > 200 and row <210) and (column>110 and column<140)) THEN -- a azul
310                 red <= (OTHERS => '0');
311                 green <= (OTHERS => '0');
312                 blue <= (OTHERS => '1');
313             elsif ((row > 210 and row <240) and (column>140 and column<150)) THEN -- b verde
314                 red <= (OTHERS => '0');
```

```
313             elsif ((row > 210 and row <240) and (column>140 and column<150)) THEN -- b verde
314                red <= (OTHERS => '0');
315                green <= (OTHERS => '1');
316                blue <= (OTHERS => '0');
317             elsif ((row > 250 and row <280) and (column>140 and column<150)) THEN -- c rojo
318                red <= (OTHERS => '1');
319                green <= (OTHERS => '0');
320                blue <= (OTHERS => '0');
321             else -- fondo
322                red <= (OTHERS => '0');
323                green <= (OTHERS => '0');
324                blue <= (OTHERS => '0');
325             end if;
326
327
328          when ocho=>
329             if ((row > 200 and row <210) and (column>110 and column<140)) THEN -- a azul
330                red <= (OTHERS => '0');
331                green <= (OTHERS => '0');
332                blue <= (OTHERS => '1');
333             elsif ((row > 210 and row <240) and (column>140 and column<150)) THEN -- b verde
334                red <= (OTHERS => '0');
335                green <= (OTHERS => '1');
336                blue <= (OTHERS => '0');
337             elsif ((row > 250 and row <280) and (column>140 and column<150)) THEN -- c rojo
338                red <= (OTHERS => '1');
339                green <= (OTHERS => '0');
340                blue <= (OTHERS => '0');
341             elsif ((row > 280 and row <290) and (column>110 and column<140)) THEN -- d blanco
342                Red <= (OTHERS => '1');
343                green <= (OTHERS => '1');
344                blue <= (OTHERS => '1');
345             elsif ((row > 250 and row <280) and (column>100 and column<110)) THEN -- e cian
346                Red <= (OTHERS => '0');
347                Green <= (OTHERS => '1');
348                Blue <= (OTHERS => '1');
349             elsif ((row > 210 and row <240) and (column>100 and column<110)) THEN -- f amarillo
350                red <= (OTHERS => '1');
351                green <= (OTHERS => '1');
352                blue <= (OTHERS => '0');
353             elsif ((row > 240 and row <250) and (column>110 and column<140)) THEN -- g violeta
354                red <= (OTHERS => '1');
355                green <= (OTHERS => '0');
356                blue <= (OTHERS => '1');
357             else -- fondo
358                red <= (OTHERS => '0');
359                green <= (OTHERS => '0');
360                blue <= (OTHERS => '0');
361             end if;
362
363          when nueve=>
364             if ((row > 200 and row <210) and (column>110 and column<140)) THEN -- a azul
365                red <= (OTHERS => '0');
366                green <= (OTHERS => '0');
367                blue <= (OTHERS => '1');
368             elsif ((row > 210 and row <240) and (column>140 and column<150)) THEN -- b verde
369                red <= (OTHERS => '0');
370                green <= (OTHERS => '1');
371                blue <= (OTHERS => '0');
372             elsif ((row > 250 and row <280) and (column>140 and column<150)) THEN -- c rojo
373                red <= (OTHERS => '1');
374                green <= (OTHERS => '0');
375                blue <= (OTHERS => '0');
376             elsif ((row > 210 and row <240) and (column>100 and column<110)) THEN -- f amarillo
377                red <= (OTHERS => '1');
378                green <= (OTHERS => '1');
379                blue <= (OTHERS => '0');
380             elsif ((row > 240 and row <250) and (column>110 and column<140)) THEN -- g violeta
381                red <= (OTHERS => '1');
382                green <= (OTHERS => '0');
```

```
                blue <= (OTHERS => '1');
            else -- fondo
                red <= (OTHERS => '0');
                green <= (OTHERS => '0');
                blue <= (OTHERS => '0');
            end if;

         when others =>
            red <= (OTHERS => '0');
            green <= (OTHERS => '0');
            blue <= (OTHERS => '0');

      end case;

   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;
```

```
--****DECODIFICADOR CONCURRENTE DEL DISPLAY 7 SEGMENTOS****
with dipsw select conectornum <= --decodificador para los números
   "0111111" when "0000",
   "0000110" when "0001",
   "1011011" when "0010",
   "1001111" when "0011",
   "1100110" when "0100",
   "1101101" when "0101",
   "1111101" when "0110",
   "0000111" when "0111",
   "1111111" when "1000",
   "1100111" when "1001",
   "0000000" when others;

END behavioral;
```

Practica Complementaria:

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

entity practica11 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;
         constant cero: std_logic_vector(6 downto 0):="0111111"; --gfedcba
   constant uno:  std_logic_vector(6 downto 0):="0000110";
   constant dos:    std_logic_vector(6 downto 0):="1011011";
   constant tres:   std_logic_vector(6 downto 0):="1001111";
   constant cuatro: std_logic_vector(6 downto 0):="1100110";
   constant cinco:  std_logic_vector(6 downto 0):="1101101";
   constant seis:   std_logic_vector(6 downto 0):="1111101";
   constant siete:  std_logic_vector(6 downto 0):="0000111";
   constant ocho:   std_logic_vector(6 downto 0):="1111111";
   constant nueve:  std_logic_vector(6 downto 0):="1110011";
   constant r1:std_logic_vector(3 downto 0):=(OTHERS => '1');
   constant r0:std_logic_vector(3 downto 0):=(OTHERS => '0');
   constant g1:std_logic_vector(3 downto 0):=(OTHERS => '1');
   constant g0:std_logic_vector(3 downto 0):=(OTHERS => '0');
   constant b1:std_logic_vector(3 downto 0):=(OTHERS => '1');
```

```
30      constant b1:std_logic_vector(3 downto 0):=(OTHERS => '1');
31      constant b0:std_logic_vector(3 downto 0):=(OTHERS => '0')
32  );
33  --decodificador);
34
35  port (   clk50MHz: in std_logic;
36           red: out std_logic_vector (3 downto 0);
37           green: out std_logic_vector (3 downto 0);
38           blue: out std_logic_vector (3 downto 0);
39           h_sync: out std_logic;
40           v_sync: out std_logic;
41      --   dipsw: in std_logic_vector(3 downto 0); --numeros para
42           a,b,c,d,e,f,g: out std_logic); --decodificador
43
44  end entity practica11;
45
46  architecture behavioral of practica11 is
47      component Divisor is
48      Generic ( N : integer := 24);
49          port( Clk: in std_logic;
50                div_clk: out std_logic);
51      end component;
52
53      --Contadores
54      signal h_period : INTEGER := h_pulse + h_bp + h_pixels + h_fp;
55      signal v_period : INTEGER := v_pulse + v_bp + v_pixels + v_fp;
56      signal h_count : INTEGER RANGE 0 TO h_period - 1 := 0;
57      signal v_count : INTEGER RANGE 0 TO v_period - 1 := 0;
58      signal reloj_pixel, display_ena : std_logic;
59      signal column : INTEGER RANGE 0 TO h_period - 1 := 0;
```

```
60      signal row : INTEGER RANGE 0 TO v_period - 1 := 0;
61      signal reloj_contador, reloj_fondo : std_logic;
62      signal dip: unsigned(3 downto 0):="0000";
63      signal conectornum:std_logic_vector(6 downto 0);--coneccion del
64  --decodificador con image_gen
65
66  begin
67      U1: Divisor generic map (24) port map (Clk50MHz,reloj_contador);
68
69      A1: process (clk50MHz) is
70      begin
71          if rising_edge(clk50MHz) then
72              reloj_pixel <= not reloj_pixel;
73          end if;
74
75          if rising_edge(reloj_contador) then
76              dip<=dip+1;
77          end if;
78      end process A1; -- 25mhz
79
80
81      contadores : process (reloj_pixel) -- H_periodo=800, V_periodo=525
82      begin
83          if rising_edge(reloj_pixel) then
84              if h_count<(h_period-1) then
85                  h_count<=h_count+1;
86              else
87                  h_count<=0;
88                  if v_count<(v_period-1) then
89                      v_count<=v_count+1;
```

```
89              v_count<=v_count+1;
90           else
91              v_count<=0;
92           end if;
93        end if;
94     end if;
95  end process contadores;
96
97  senial_hsync : process (reloj_pixel) --h_pixel+h_fp+h_pulse= 784
98  begin
99     if rising_edge(reloj_pixel) then
100       if h_count>(h_pixels + h_fp) or
101          h_count>(h_pixels + h_fp + h_pulse) then
102          h_sync<='0';
103       else
104          h_sync<='1';
105       end if;
106    end if;
107 end process senial_hsync;
108
109 senial_vsync : process (reloj_pixel) --vpixels+v_fp+v_pulse=525
110 begin --checar si se en parte visible es 1 o 0
111    if rising_edge(reloj_pixel) then
112       if v_count>(v_pixels + v_fp) or
113          v_count>(v_pixels + v_fp + v_pulse) then
114          v_sync<='0';
115       else
116          v_sync<='1';
117       end if;
118    end if;
```

```
119 end process senial_vsync;
120
121 coords_pixel: process(reloj_pixel)
122 begin --asignar una coordenada en parte visible
123    if rising_edge(reloj_pixel) then
124       if (h_count < h_pixels) then
125          column <= h_count;
126       end if;
127       if (v_count < v_pixels) then
128          row <= v_count;
129       end if;
130    end if;
131 end process coords_pixel;
132
133 generador_imagen: PROCESS(display_ena, row, column)
```

Resultado:

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.