



Universidad Nacional Autónoma de México Facultad de ingeniería



Diseño Digital VLSI

Grupo : 4

Práctica: 10

SIMULACIÓN DE UN LED DE 7 SEGMENTOS

Edgar Daniel Barcenás Martínez

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 7 segmentos en de un monitor.

DIAGRAMA DE BLOQUES:

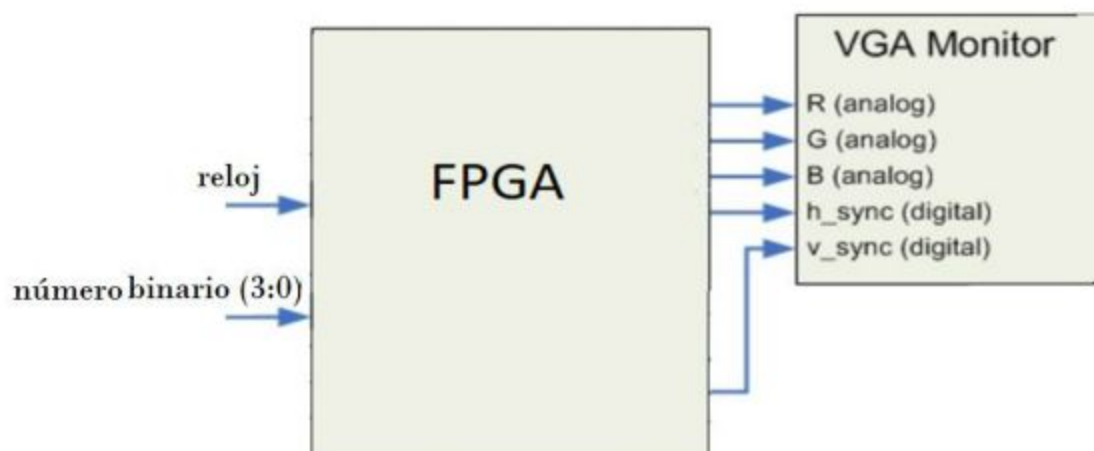


Figura 11.1. Diagrama de Bloques del sistema Emulador de Display 7 Segmentos en Monitor

DIAGRAMA DE BLOQUES FUNCIONALES:

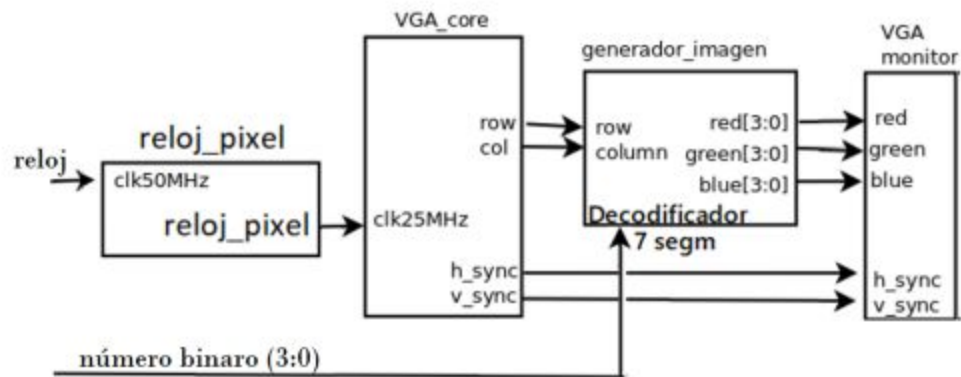


Figura 11.2. Bloques funcionales del sistema Emulador de Display 7 segmentos en Monitor

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     d1psw    : 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):="011111"; --gfedcba
39     constant uno  : std_logic_vector(6 downto 0):="0000110";
40     constant dos  : std_logic_vector(6 downto 0):="1011011";

```

```

40 constant dos: std_logic_vector(6 downto 0):="1011011";
41 constant tres: std_logic_vector(6 downto 0):="1001111";
42 constant cuatro: std_logic_vector(6 downto 0):="1100110";
43 constant cinco: std_logic_vector(6 downto 0):="1101101";
44 constant seis: std_logic_vector(6 downto 0):="1111101";
45 constant siete: std_logic_vector(6 downto 0):="0000111";
46 constant ocho: std_logic_vector(6 downto 0):="1111111";
47 constant nueve: std_logic_vector(6 downto 0):="1100111";
48
49 --constant r1:std_logic_vector(3 downto 0):=(OTHERS => '1');
50 --constant r0:std_logic_vector(3 downto 0):=(OTHERS => '0');
51 --constant g1:std_logic_vector(3 downto 0):=(OTHERS => '1');
52 --constant g0:std_logic_vector(3 downto 0):=(OTHERS => '0');
53 --constant b1:std_logic_vector(3 downto 0):=(OTHERS => '1');
54 --constant b0:std_logic_vector(3 downto 0):=(OTHERS => '0'); -- variable a,b,c,d,e,f: std_logic;
55
56 signal conectornum:std_logic_vector(6 downto 0); -- conexion del decodificador con image_gen
57
58 BEGIN
59
60 divisor: process (clk50MHZ) is
61 begin
62 if rising_edge(clk50MHZ) then
63 reloj_pixel <= not reloj_pixel;
64 end if;
65 end process divisor; -- 25mhz
66
67 contadores : process (reloj_pixel) -- H_periodo=800, V_periodo=525
68 begin
69 if rising_edge(reloj_pixel) then
70 if h_count<(h_period-1) then
71 h_count<=h_count+1;
72 else
73 h_count<=0;
74 if v_count<(v_period-1) then
75 v_count<=v_count+1;
76 else
77 v_count<=0;
78
79 end if;
80 end if;
81 end process contadores;
82
83 senial_hsync : process (reloj_pixel) --h_pixel+h_fp+h_pulse= 784
84 begin
85 if rising_edge(reloj_pixel) then
86 if h_count>(h_pixels + h_fp) or
87 h_count>(h_pixels + h_fp + h_pulse) then
88 h_sync<='0';
89 else
90 h_sync<='1';
91 end if;
92 end if;
93 end process senial_hsync;
94
95 senial_vsync : process (reloj_pixel) --vpixels+v_fp+v_pulse=525
96 begin --chechar si se en parte visible es 1 o 0
97 if rising_edge(reloj_pixel) then
98 if v_count>(v_pixels + v_fp) or
99 v_count>(v_pixels + v_fp + v_pulse) then
100 v_sync<='0';
101 else
102 v_sync<='1';
103 end if;
104 end if;
105 end process senial_vsync;

```



```

108 begin --asignar una coordenada en parte visible
109 if rising_edge(reloj_pixel) then
110 if (h_count < h_pixels) then
111 column <= h_count;
112 end if;
113 if (v_count < v_pixels) then
114 row <= v_count;
115 end if;
116 end if;
117 end process coords_pixel;
118
119 generador_imagen: PROCESS(display_ena, row, column)
120 BEGIN
121 IF(display_ena = '1') THEN
122 display <= not(conectornum);
123
124 case conectornum is
125
126 when cero=>
127 if ((row > 200 and row <210) and (column>110 and column<140)) THEN -- a azul
128 red <= (OTHERS => '0');
129 green <= (OTHERS => '0');
130 blue <= (OTHERS => '1');
131
132 elsif ((row > 210 and row <240) and (column>140 and column<150)) THEN -- b verde
133 red <= (OTHERS => '0');
134 green <= (OTHERS => '1');
135 blue <= (OTHERS => '0');
136
137 elsif ((row > 250 and row <280) and (column>140 and column<150)) THEN -- c rojo
138 red <= (OTHERS => '1');
139 green <= (OTHERS => '0');
140 blue <= (OTHERS => '0');
141
142 elsif ((row > 280 and row <290) and (column>110 and column<140)) THEN -- d blanco
143 red <= (OTHERS => '1');
144 green <= (OTHERS => '1');
145 blue <= (OTHERS => '1');
146
147 elsif ((row > 250 and row <280) and (column>100 and column<110)) THEN -- e cian
148 red <= (OTHERS => '0');
149 green <= (OTHERS => '1');
150 blue <= (OTHERS => '1');
151
152 else -- f amarillo
153 red <= (OTHERS => '1');
154 green <= (OTHERS => '1');
155 blue <= (OTHERS => '0');
156 end if;
157
158 when uno=>
159 if ((row > 210 and row <240) and (column>140 and column<150)) THEN -- b verde
160 red <= (OTHERS => '0');
161 green <= (OTHERS => '1');
162 blue <= (OTHERS => '0');
163
164 elsif ((row > 250 and row <280) and (column>140 and column<150)) THEN -- c rojo
165 red <= (OTHERS => '1');
166 green <= (OTHERS => '0');
167 blue <= (OTHERS => '0');
168
169 else -- fondo
170 red <= (OTHERS => '0');
171 green <= (OTHERS => '0');
172 blue <= (OTHERS => '0');
173 end if;
174
175 when dos=>
176 if ((row > 200 and row <210) and (column>110 and column<140)) THEN -- a azul
177 red <= (OTHERS => '0');
178 green <= (OTHERS => '0');
179 blue <= (OTHERS => '1');
180
181 elsif ((row > 210 and row <240) and (column>140 and column<150)) THEN -- b verde
182 red <= (OTHERS => '0');
183 green <= (OTHERS => '1');
184 blue <= (OTHERS => '0');
185
186 end if;
187
188 end process generador_imagen;
189
190 end entity;

```

```

180         green <= (OTHERS => '1');
181         blue <= (OTHERS => '0');
182     elsif ((row > 280 and row <290) and (column>110 and column<140)) THEN -- d blanco
183         red <= (OTHERS => '1');
184         green <= (OTHERS => '1');
185         blue <= (OTHERS => '1');
186     elsif ((row > 250 and row <280) and (column>100 and column<110)) THEN -- e cian
187         red <= (OTHERS => '0');
188         green <= (OTHERS => '1');
189         blue <= (OTHERS => '1');
190     elsif ((row > 240 and row <250) and (column>110 and column<140)) THEN -- g violeta
191         red <= (OTHERS => '1');
192         green <= (OTHERS => '0');
193         blue <= (OTHERS => '1');
194     else -- fondo
195         red <= (OTHERS => '0');
196         green <= (OTHERS => '0');
197         blue <= (OTHERS => '0');
198     end if;
199
200 when tres=>
201     if ((row > 200 and row <210) and (column>110 and column<140)) THEN -- a azul
202         red <= (OTHERS => '0');
203         green <= (OTHERS => '0');
204         blue <= (OTHERS => '1');
205     elsif ((row > 210 and row <240) and (column>140 and column<150)) THEN -- b verde
206         red <= (OTHERS => '0');
207         green <= (OTHERS => '1');
208         blue <= (OTHERS => '0');
209     elsif ((row > 250 and row <280) and (column>140 and column<150)) THEN -- c rojo
210         red <= (OTHERS => '1');
211         green <= (OTHERS => '0');
212         blue <= (OTHERS => '0');
213     elsif ((row > 280 and row <290) and (column>110 and column<140)) THEN -- d blanco
214         red <= (OTHERS => '1');
215         green <= (OTHERS => '1');
216         blue <= (OTHERS => '1');
217
218     elsif ((row > 240 and row <250) and (column>110 and column<140)) THEN -- g violeta
219         red <= (OTHERS => '1');
220         green <= (OTHERS => '0');
221         blue <= (OTHERS => '1');
222     else -- fondo
223         red <= (OTHERS => '0');
224         green <= (OTHERS => '0');
225         blue <= (OTHERS => '0');
226     end if;
227
228 when cuatro=>
229     if ((row > 210 and row <240) and (column>140 and column<150)) THEN -- b verde
230         red <= (OTHERS => '0');
231         green <= (OTHERS => '1');
232         blue <= (OTHERS => '0');
233     elsif ((row > 250 and row <280) and (column>140 and column<150)) THEN -- c rojo
234         red <= (OTHERS => '1');
235         green <= (OTHERS => '0');
236         blue <= (OTHERS => '0');
237     elsif ((row > 210 and row <240) and (column>100 and column<110)) THEN -- f amarillo
238         red <= (OTHERS => '1');
239         green <= (OTHERS => '1');
240         blue <= (OTHERS => '0');
241     elsif ((row > 240 and row <250) and (column>110 and column<140)) THEN -- g violeta
242         red <= (OTHERS => '1');
243         green <= (OTHERS => '0');
244         blue <= (OTHERS => '1');
245     else -- fondo
246         red <= (OTHERS => '0');
247         green <= (OTHERS => '0');
248         blue <= (OTHERS => '0');
249     end if;
250
251 when cinco=>
252     if ((row > 200 and row <210) and (column>110 and column<140)) THEN -- a azul
253         red <= (OTHERS => '0');
254         green <= (OTHERS => '0');
255         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 => '0');
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');

```



```

383         blue <= (OTHERS => '1');
384     else -- fondo
385         red <= (OTHERS => '0');
386         green <= (OTHERS => '0');
387         blue <= (OTHERS => '0');
388     end if;
389
390     when others =>
391         red <= (OTHERS => '0');
392         green <= (OTHERS => '0');
393         blue <= (OTHERS => '0');
394
395     end case;
396
397 ELSE
398
399     red<= (OTHERS => '0');
400     green <= (OTHERS => '0');
401     blue<= (OTHERS => '0');
402
403     END IF;
404     END PROCESS generador_imagen;
405
406     display_enable: process(reloj_pixel) --- h_pixels=640; y_pixeles=480
407     begin
408         if rising_edge(reloj_pixel) then
409             if (h_count < h_pixels AND v_count < v_pixels) THEN
410                 display_ena <= '1';
411             else
412                 display_ena <= '0';
413             end if;
414         end if;
415     end process display_enable;
416
417
418
419     --****DECODIFICADOR CONCURRENT DEL DISPLAY 7 SEGMENTOS****
420     with dipsw select conectornum <= --decodificador para los números
421         "011111" when "0000",
422         "0000110" when "0001",
423         "1011011" when "0010",
424         "1001111" when "0011",
425         "1100110" when "0100",
426         "1101101" when "0101",
427         "1111101" when "0110",
428         "0000111" when "0111",
429         "1111111" when "1000",
430         "1100111" when "1001",
431         "0000000" when others;
432
433     END behavioral;

```

Practica Complementaria:

```

1  library ieee;
2  use ieee.std_logic_1164.all;
3  use ieee.numeric_std.all;
4
5  entity practica11 is
6
7  GENERIC( --Constantes para monitor VGA en 640x480
8      CONSTANT h_pulse : INTEGER := 96;
9      CONSTANT h_bp : INTEGER := 48;
10     CONSTANT h_pixels : INTEGER := 640;
11     CONSTANT h_fp : INTEGER := 16;
12     CONSTANT v_pulse : INTEGER := 2;
13     CONSTANT v_bp : INTEGER := 33;
14     CONSTANT v_pixels : INTEGER := 480;
15     CONSTANT v_fp : INTEGER := 10;
16     constant cero: std_logic_vector(6 downto 0):="0111111"; --gfedcba
17     constant uno: std_logic_vector(6 downto 0):="0000110";
18     constant dos: std_logic_vector(6 downto 0):="1011011";
19     constant tres: std_logic_vector(6 downto 0):="1001111";
20     constant cuatro: std_logic_vector(6 downto 0):="1100110";
21     constant cinco: std_logic_vector(6 downto 0):="1101101";
22     constant seis: std_logic_vector(6 downto 0):="1111101";
23     constant siete: std_logic_vector(6 downto 0):="0000111";
24     constant ocho: std_logic_vector(6 downto 0):="1111111";
25     constant nueve: std_logic_vector(6 downto 0):="1110011";
26     constant r1:std_logic_vector(3 downto 0):=(OTHERS => '1');
27     constant r0:std_logic_vector(3 downto 0):=(OTHERS => '0');
28     constant g1:std_logic_vector(3 downto 0):=(OTHERS => '1');
29     constant g0:std_logic_vector(3 downto 0):=(OTHERS => '0');
30     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 ccolumn : INTEGER RANGE 0 TO h_period - 1 := 0;
60
61     signal row : INTEGER RANGE 0 TO v_period - 1 := 0;
62     signal reloj_contador, reloj_fondo : std_logic;
63     signal dip: unsigned(3 downto 0):="0000";
64     signal conectornum:std_logic_vector(6 downto 0);--conexion del
65     --decodificador con image_gen
66
67 begin
68     U1: Divisor generic map (24) port map (clk50MHz,reloj_contador);
69
70     A1: process (clk50MHz) is
71     begin
72         if rising_edge(clk50MHz) then
73             reloj_pixel <= not reloj_pixel;
74         end if;
75
76         if rising_edge(reloj_contador) then
77             dip<=dip+1;
78         end if;
79     end process A1; -- 25mhz
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 serial_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 serial_hsync;
108
109 serial_vsync : process (reloj_pixel) --vpixels+v_fp+v_pulse=525
110 begin --chechar 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 serial_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)
134

```


Resultado:



CONCLUSIONES:

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