



Universidad Nacional Autónoma de México Facultad de ingeniería



Diseño Digital VLSI

Grupo : 4

Práctica: 9

DISEÑO DE UN TRANSMISOR PARA COMUNICACIÓN SERIAL

Edgar Daniel Barcenás Martínez

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:

```
1  library ieee;
2  use ieee.std_logic_1164.all;
3  use ieee.numeric_std.all;
4
5  entity vga 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 |
17 |  port (   clk50MHz: in std_logic;
18 |         red: out std_logic_vector (3 downto 0);
19 |         green: out std_logic_vector (3 downto 0);
20 |         blue: out std_logic_vector (3 downto 0);
21 |         h_sync: out std_logic;
22 |         v_sync: out std_logic);
23 |
24 |  end entity vga;
25
26  architecture behavioral of vga is
27  |
28  |  --Contadores
29  |  signal h_period : INTEGER := h_pulse + h_bp + h_pixels + h_fp;
30  |  signal v_period : INTEGER := v_pulse + v_bp + v_pixels + v_fp;
31  |  signal h_count : INTEGER RANGE 0 TO h_period - 1 := 0;
32  |  signal v_count : INTEGER RANGE 0 TO v_period - 1 := 0;
33  |  signal reloj_pixel, display_ena : std_logic;
34  |  signal column : INTEGER RANGE 0 TO h_period - 1 := 0;
35  |  signal row : INTEGER RANGE 0 TO v_period - 1 := 0;
36  |
```

```

36 L
37 begin
38
39 A: process (clk50MHz) is
40 begin
41 if rising_edge(clk50MHz) then
42 reloj_pixel <= not reloj_pixel;
43 end if;
44 end process A; -- 25mhz
45
46 contadores : process (reloj_pixel) -- H_periodo=800, v_periodo=525
47 begin
48 if rising_edge(reloj_pixel) then
49 if h_count<(h_period-1) then
50 h_count<=h_count+1;
51 else
52 h_count<=0;
53 if v_count<(v_period-1) then
54 v_count<=v_count+1;
55 else
56 v_count<=0;
57 end if;
58 end if;
59 end if;
60 end process contadores;
61
62 serial_hsync : process (reloj_pixel) --h_pixel+h_fp+h_pulse= 784
63 begin
64 if rising_edge(reloj_pixel) then
65 if h_count>(h_pixels + h_fp) or
66 h_count>(h_pixels + h_fp + h_pulse) then
67 h_sync<='0';
68 else
69 h_sync<='1';
70 end if;
71 end if;
72 end process serial_hsync;
73
74 serial_vsync : process (reloj_pixel) --vpixels+v_fp+v_pulse=525
75 begin --chechar si se en parte visible es 1 o 0
76 if rising_edge(reloj_pixel) then
77 if v_count>(v_pixels + v_fp) or
78 v_count>(v_pixels + v_fp + v_pulse) then
79 v_sync<='0';
80 else
81 v_sync<='1';
82 end if;
83 end if;
84 end process serial_vsync;
85
86 coords_pixel: process(reloj_pixel)
87 begin --asignar una coordenada en parte visible
88 if rising_edge(reloj_pixel) then
89 if (h_count < h_pixels) then
90 column <= h_count;
91 end if;
92 if (v_count < v_pixels) then
93 row <= v_count;
94 end if;
95 end if;
96 end process coords_pixel;
97
98 generador_imagen: PROCESS(display_ena, row, column)
99 variable contador:integer range 300 to 500 :=300;
100 variable contador1:integer range 350 to 600 := 350;
101
102 BEGIN
103 if rising_edge(reloj_pixel) then
104 contador:=contador +1;
105 contador1:=contador1 +1;
106 end if;

```



```

112     elsif ((row > 300 and row <350) and (column>450 and column<500)) THEN
113         red <= (OTHERS => '0');
114         green<=(OTHERS => '1');
115         blue<=(OTHERS => '0');
116     elsif ((row > 300 and row <350) and (column>550 and column<600)) THEN
117         red <= (OTHERS => '0');
118         green<=(OTHERS => '0');
119         blue<=(OTHERS => '1');
120     else
121         red <= (OTHERS => '0');
122         green <= (OTHERS => '0');
123         blue <= (OTHERS => '0');
124     end if;
125 ELSE
126     red<= (OTHERS => '0');
127     green <= (OTHERS => '0');
128     blue<= (OTHERS => '0');
129 END IF;
130 END PROCESS generador_imagen;
131
132 display_enable: process(reloj_pixel) --- h_pixels=640; y_pixeles=480
133 begin
134     if rising_edge(reloj_pixel) then
135         if (h_count < h_pixels AND v_count < v_pixels) THEN
136             display_ena <= '1';
137         else
138             display_ena <= '0';
139         end if;
140     end if;
141 end process display_enable;
142
143 end behavioral;

```

Practica Complementaria:

```

1  library ieee;
2  use ieee.std_logic_1164.all;
3  use ieee.numeric_std.all;
4  entity practica10 is
5      GENERIC( ---Constantes para monitor VGA en 640x480
6
7
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 );
17 port ( clk50MHz: in std_logic;
18       red: out std_logic_vector (3 downto 0);
19       green: out std_logic_vector (3 downto 0);
20       blue: out std_logic_vector (3 downto 0);
21       h_sync: out std_logic;
22       v_sync: out std_logic );
23 end entity practica10;
24
25 architecture Behavioral OF practica10 IS
26     component divisor is
27         Generic ( N : integer := 24);
28         Port ( clk : in std_logic;
29              div_clk : out std_logic);
30     end component;

```

```

31 --Contadores
32 signal reloj_pixel :std_logic;
33 signal cuenta:std_logic;
34 signal display_ena: std_logic;
35 signal column: integer;
36 signal row: integer;
37 signal h_period : INTEGER := h_pulse + h_bp + h_pixels + h_fp;
38 signal v_period : INTEGER := v_pulse + v_bp + v_pixels + v_fp;
39 signal h_count : INTEGER RANGE 0 TO h_period - 1 := 0;
40 signal v_count : INTEGER RANGE 0 TO v_period - 1 := 0;
41 signal CLK:std_logic;
42 begin
43 E1: divisor generic map(20) port map (clk50MHz,CLK);
44 rel: process (clk50MHz) is
45 begin
46 if rising_edge(clk50MHz) then
47 reloj_pixel <= not reloj_pixel;
48 end if;
49 end process rel; -- 25mhz
50
51 contadores : process (reloj_pixel) -- H_perodo=800, V_perodo=525
52 begin
53 if rising_edge(reloj_pixel) then
54 if h_count<(h_period-1) then
55 h_count<=h_count+1;
56 else
57 h_count<=0;
58 if v_count<(v_period-1) then
59 v_count<=v_count+1;
60 else
61 v_count<=0;
62 end if;
63 end if;
64 end if;

```

```

64 end if;
65 end process contadores;
66
67 senial_hsync : process (reloj_pixel) --h_pixel+h_fp+h_pulse= 784
68 begin
69 if rising_edge(reloj_pixel) then
70 if h_count>(h_pixels + h_fp) or
71 h_count>(h_pixels + h_fp + h_pulse) then
72 h_sync<='0';
73 else
74 h_sync<='1';
75 end if;
76 end if;
77 end process senial_hsync;
78 senial_vsync : process (reloj_pixel) --vpixels+v_fp+v_pulse=525
79 begin --chechar si se en parte visible es 1 o 0
80 if rising_edge(reloj_pixel) then
81 if v_count>(v_pixels + v_fp) or
82 v_count>(v_pixels + v_fp + v_pulse) then
83 v_sync<='0';
84 else
85 v_sync<='1';
86 end if;
87 end if;
88 end process senial_vsync;
89 coords_pixel: process(reloj_pixel)
90 begin --asignar una coordenada en parte visible
91 if rising_edge(reloj_pixel) then
92 if (h_count < h_pixels) then
93 column <= h_count;
94 end if;
95 if (v_count < v_pixels) then
96 row <= v_count;
97 end if;

```

```

97 end if;
98 end if;
99 end process coords_pixel;
100
101
102 generador_imagen: PROCESS(display_ena, row, column)
103 --arriba
104 variable contador1: integer range 300 to 500:=300;
105 variable contador2: integer range 350 to 600:=350;
106 --a la derecha
107 variable contador3: integer range 450 to 550:=450;
108 variable contador4: integer range 500 to 650:=500;
109 --abajo
110 variable contador5: integer range 100 to 300:=300;
111 variable contador6: integer range 150 to 350:=350;
112 BEGIN
113 --contador
114 if rising_edge(CLK) then
115 --arriba
116     contador1:=contador1+1;
117     contador2:=contador2+1;
118 --a lado
119     contador3:=contador3+1;
120     contador4:=contador4+1;
121 --abajo
122     contador5:=contador5-1;
123     contador6:=contador6-1;
124 end if;
125 IF(display_ena = '1') THEN
126 --rojo

```

```

127 --rojo
128 if ((row > contador1 and row < contador2) and
129 (column>350 and column<400)) THEN
130 red <= (OTHERS => '1');
131 green<=(OTHERS => '0');
132 blue<=(OTHERS => '0');
133 --verde
134 elsif ((row > 300 and row <350) and
135 (column>contador3 and column<contador4)) THEN
136 red <= (OTHERS => '0');
137 green<=(OTHERS => '1');
138 blue<=(OTHERS => '0');
139 --azul
140 elsif ((row > contador5 and row <contador6) and
141 (column>550 and column<600)) THEN
142 red <= (OTHERS => '0');
143 green<=(OTHERS => '0');
144 blue<=(OTHERS => '1');
145 --fondo
146 elsif contador1<300 then
147 red <= (OTHERS => '1');
148 green <= (OTHERS => '1');
149 blue <= (OTHERS => '1');
150 elsif contador1<400 then
151 red <= (OTHERS => '1');
152 green <= (OTHERS => '0');
153 blue <= (OTHERS => '1');
154 elsif contador2<500 then
155 red <= (OTHERS => '1');
156 green <= (OTHERS => '1');
157 blue <= (OTHERS => '0');
158 else

```



```

156 blue <= (OTHERS => '0');
157 else
158 red <= (OTHERS => '0');
159 green <= (OTHERS => '0');
160 blue <= (OTHERS => '0');
161
162 end if;
163
164
165 ELSE
166 red<= (OTHERS => '0');
167 green <= (OTHERS => '0');
168 blue<= (OTHERS => '0');
169 END IF;
170 END PROCESS generador_imagen;
171
172 display_enable: process(reloj_pixel) -- h_pixels=640; y_pixeles=480
173 begin
174 if rising_edge(reloj_pixel) then
175 if (h_count < h_pixels AND v_count < v_pixels) THEN
176 display_ena <= '1';
177 else
178 display_ena <= '0';
179 end if;
180 end if;
181 end process display_enable;
182
183 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.