Escuela Superior de Ingenieros

Universidad de Sevilla
Departamento de Ingeniería Electrónica

Segunda práctica de VHDL

Microelectrónica

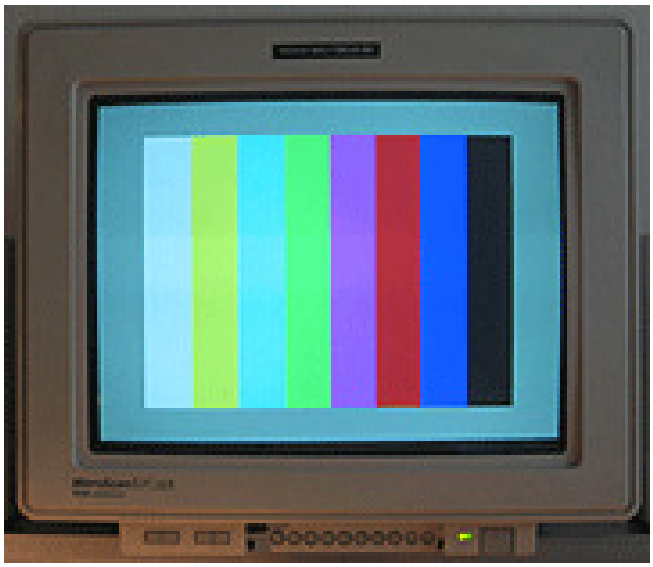
Jonathan N. Tombs

Fernando Muñoz Chavero

Miguel A. Aguirre

Hipólito Guzmán

Javier Nápoles



Segunda práctica de VHDL.
Control de un monitor VGA

La tarjeta S3 dispone de un conector VGA cableado a la FPGA. Utilizando dicho conector es posible conectar directamente un monitor compatible con el estándar VGA.

El objetivo de la práctica consiste en:

1. *Diseñar un driver VGA que permita representar imágenes con un formato de 60Hz, 640x480 y 8 colores. Este driver será muy útil para la mayor parte de los proyectos de la asignatura.*
2. *Ver un caso práctico de la utilización de GENERICS.*

Esta práctica es la segunda de un grupo de tres dedicada a entrenar al alumno con el entorno de trabajo Xilinx ISE 8.2 y la tarjeta Digilent S3.

1. Objetivo propuesto

En la presente práctica conectaremos directamente la FPGA a un monitor compatible VGA (figura 1). Para ello, mediante una serie de contadores síncronos, generaremos todas las señales necesarias para controlar el monitor en modo 60Hz, 640x480 y 8 colores.

Para conseguir nuestro objetivo necesitaremos diseñar:

- Un contador con reset síncrono y número de bits parametrizable mediante *GENERICS*.
- Un comparador síncrono cuyos umbrales sean programables mediante *GENERICS*.
- Un bloque que, a partir de las coordenadas de pantalla, genere los tres bits de color necesarios.

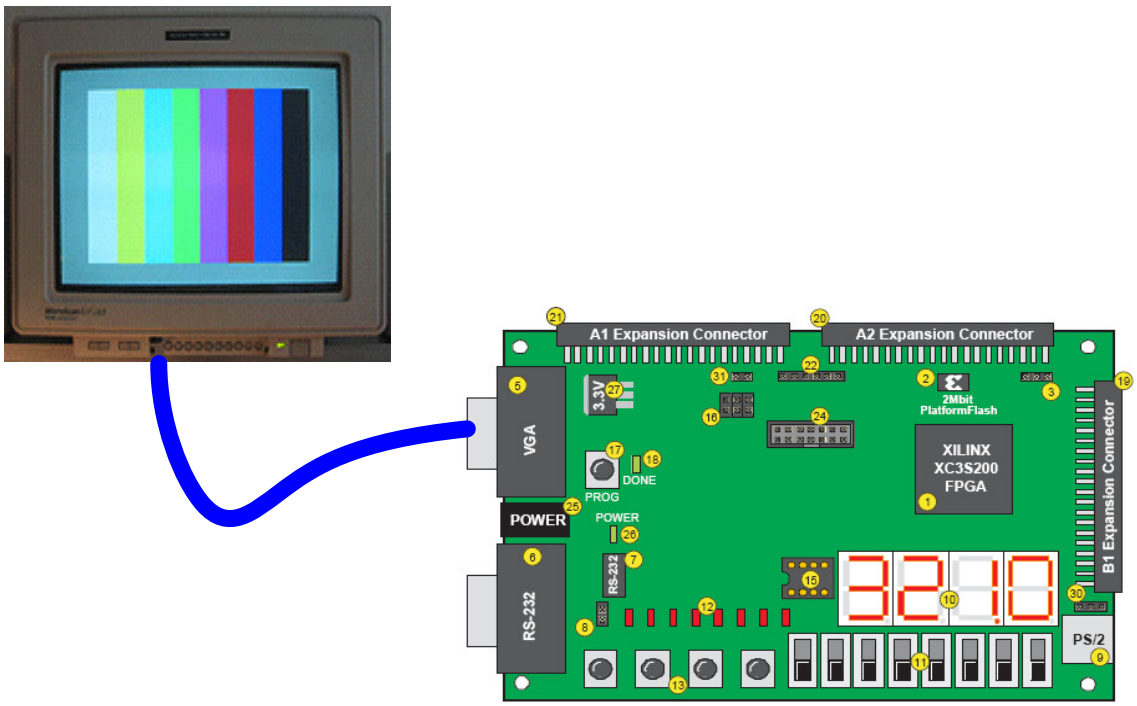


Figura 1. Objetivo de la práctica

2. Conexión VGA en la Tarjeta S3 de Digilent

La tarjeta S3 dispone de un conector DB15 cableado a puertos de la FPGA. Dicho conector permite la conexión con la pantalla del PC utilizando un cable estándar VGA. La conexión entre la FPGA y el conector se muestra en la figura 2.

Se utilizarán cinco puertos: HS (sincronismo horizontal), VS (sincronismo vertical), R (rojo), G (verde) y B (azul).

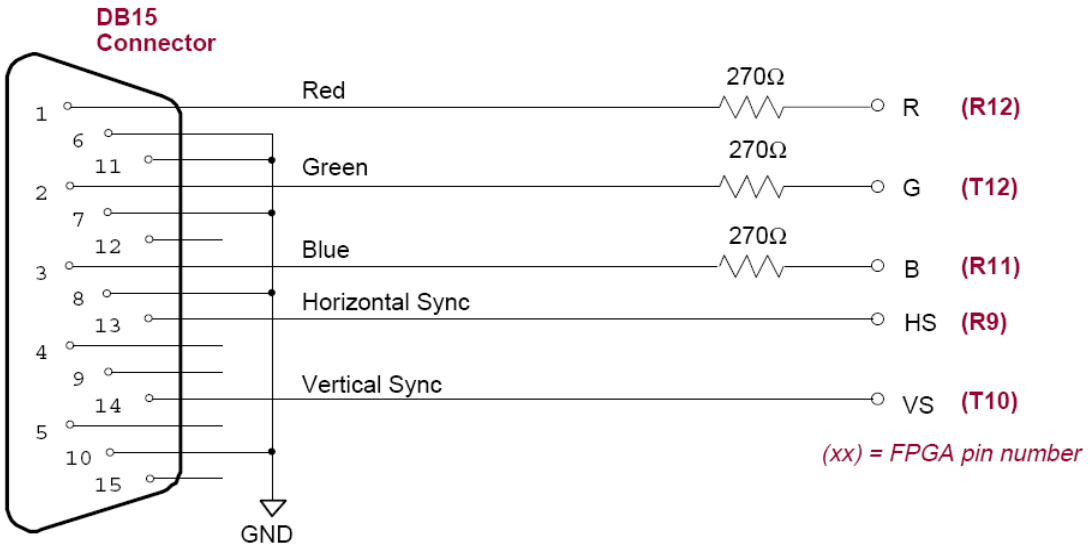


Figura 2. Conexión entre la FPGA y el conector VGA

Teóricamente, Las señales de color (R, G y B) son analógicas y determinan la intensidad de color de cada píxel. Sin embargo, en la placa están conectadas a un puerto digital (entre 0 y 3.3V) a través de una resistencia de 270Ω. Al estar el cable VGA terminado a 75Ω, un '1' en la FPGA fijará una señal de 0.7V (máxima intensidad de color) y un '0' será 0V. Por tanto, tendremos sólo dos niveles por color, pudiendo representar hasta 8 colores como se muestran en la tabla 1.

R	G	B	Color
0	0	0	Negro
0	0	1	Azul
0	1	0	Verde
0	1	1	Cian
1	0	0	Rojo
1	0	1	Magenta
1	1	0	Amarillo
1	1	1	Blanco

Tabla 1. Representación de 8 colores

3. Monitor VGA en modo 640x480

En este apartado se explicará cómo deben generarse las señales HS, VS, R, G y B para representar una imagen en un monitor en modo 60Hz, 640x480 y 8 colores. Es decir, explicaremos brevemente el protocolo VGA.

Para explicar el protocolo VGA es necesario conocer el funcionamiento básico de un Tubo de Rayos Catódicos (CRT)[1]. El funcionamiento básico de un CRT se explica en la figura 3.

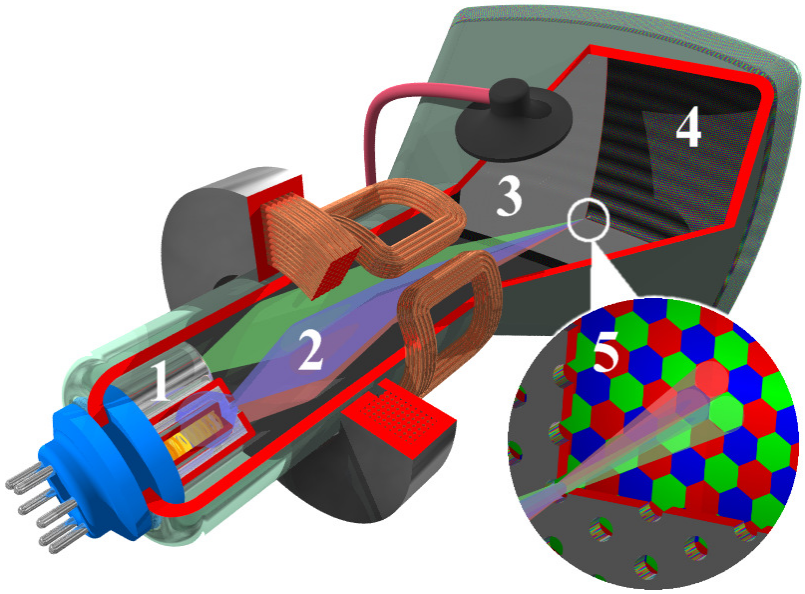


Figura 3. Funcionamiento del CRT

Tres cañones de electrones (1) producen tres haces de electrones, uno por cada color básico (rojo, azul y verde). Estos haces de electrones se focalizan (2), utilizando una bobina, para conseguir que converjan en un punto de la pantalla. Posteriormente, mediante un campo magnético generado por un par de bobinas (horizontal y vertical) se direccionan al punto deseado de la pantalla. En la pantalla de visualización (5), los rayos son separados (3) por una máscara e inciden en una capa fosforescente con zonas receptivas para cada color (4).

Por otro lado, la intensidad luminosa dependerá de la potencia a la que se emite el haz de electrones, controlable mediante las señales analógicas R, G y B.

Se representarán imágenes en pantalla haciendo que el haz de electrones la recorra en una sucesión de líneas (de izquierda a derecha) comenzando por la esquina superior izquierda (figura 4).

[1] El tubo de rayos catódicos está quedando obsoleto, sin embargo los nuevos monitores LCD mantienen la misma temporización en las señales de entrada VGA.

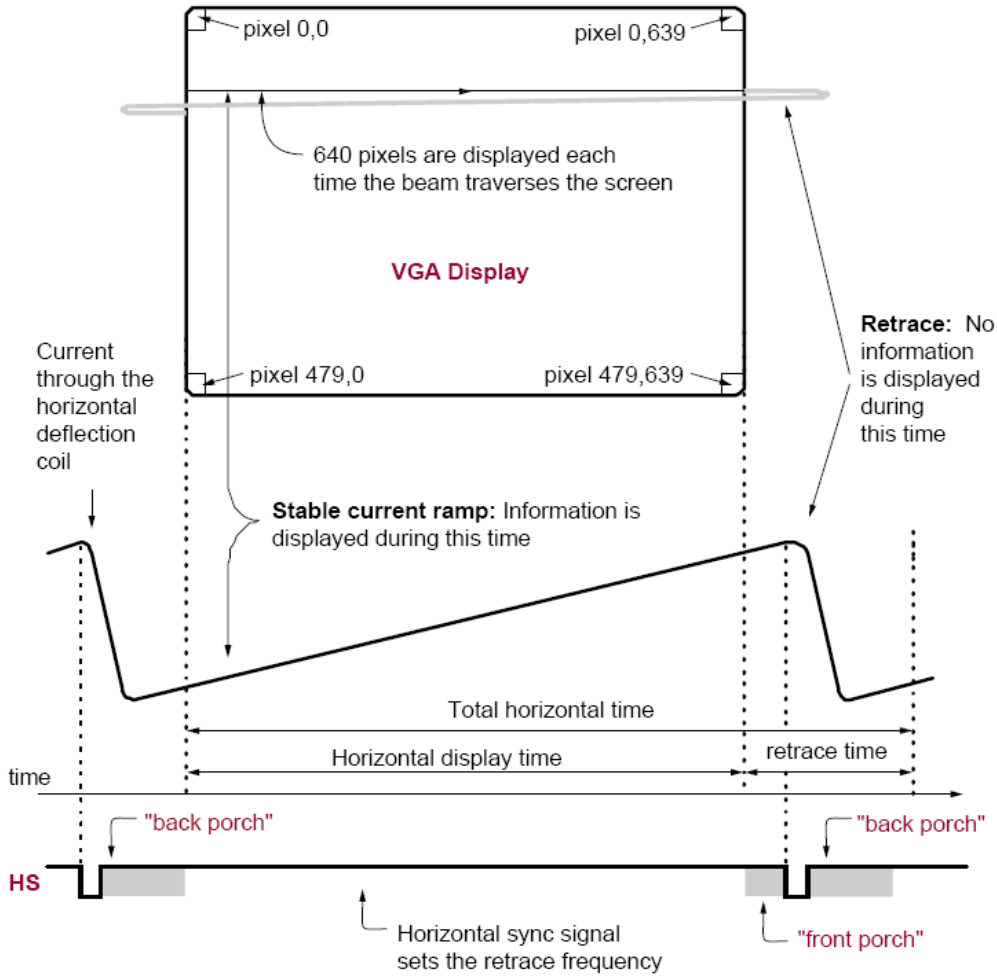


Figura 4. Representación de una imagen en un CRT

En la figura 4, la onda en forma de diente de sierra representa la corriente que pasa por la bobina que crea el campo magnético para la deflexión horizontal. Se puede comprobar, que se representa la imagen (Horizontal display time) cuando el haz de electrones va de izquierda a derecha y está apuntando correctamente a la pantalla, existiendo un tiempo (retrace time) para que el cañón vuelva al comienzo de la siguiente línea. El pulso de sincronismo vertical (HS) marca el instante en que el cañón debe comenzar una nueva línea, pudiendo distinguir un tiempo anterior ("front porch") en que el cañón está apuntando fuera de la pantalla y un tiempo posterior ("pack porch") necesario para que el haz se posicione en el inicio de la siguiente línea. Es importante que durante todo el "retrace time" los cañones de electrones estén apagados.

La frecuencia a la que se proporcionan los pulsos de sincronismos vertical y horizontal determina la velocidad y el número de veces que el haz de electrones recorre la pantalla. De esta forma seleccionamos la resolución de visualización de la

imagen en el monitor. El estándar VGA admite diferentes resoluciones, para las que tendríamos que utilizar diferentes frecuencias en HS y VS[2].

En esta práctica vamos diseñar un controlador VGA para la resolución 640x480 píxeles y 60 Hz de refresco de pantalla. Consideraremos una frecuencia de píxel de 25MHz, para lo que dividiremos por dos la frecuencia de reloj disponible de 50MHz utilizando un biestable[3].

Symbol	Parameter	Vertical Sync			Horizontal Sync	
		Time	Clocks	Lines	Time	Clocks
T_S	Sync pulse time	16.7 ms	416,800	521	32 µs	800
T_{DISP}	Display time	15.36 ms	384,000	480	25.6 µs	640
T_{PW}	Pulse width	64 µs	1,600	2	3.84 µs	96
T_{FP}	Front porch	320 µs	8,000	10	640 ns	16
T_{BP}	Back porch	928 µs	23,200	29	1.92 µs	48

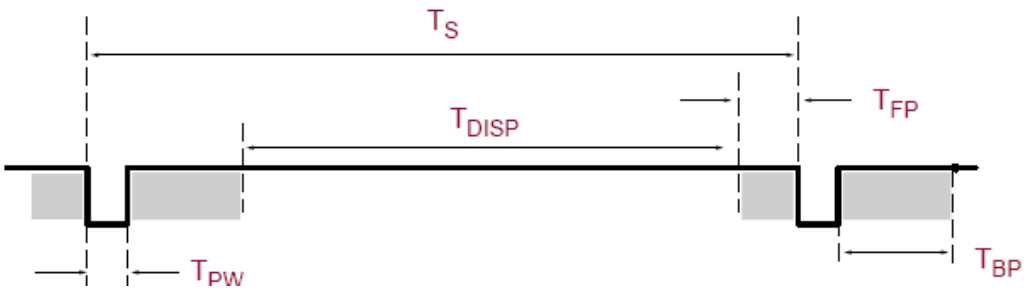


Figura 5. Temporización necesaria para la resolución 640x480 píxel.

Todos los datos necesarios para generar correctamente las señales VGA se proporcionan en la figura 5. En la tabla de la figura 5, el número de periodos (Clocks) se refiere siempre a la frecuencia de píxel (25MHz) y no al reloj general del sistema (50MHz).

Por tanto, el problema a resolver es cómo generar los pulsos de VS y HS con duración y frecuencia adecuados para que sean entendidos por un monitor compatible con el estándar VGA. En el siguiente apartado estudiaremos una solución sencilla y eficiente a este problema.

[2] En http://www.epanorama.net/documents/pc/vga_timing.html se proporciona un estudio que permite seleccionar la frecuencia de HS y VS en función de la resolución de pantalla que queremos conseguir.

[3] También es posible configura la frecuencia de reloj a la que funcionarán los circuitos en la FPGA utilizando el Digital Clock Manager de la Spartan 3.

4. Realización de la práctica

En la figura 6 se representa el diagrama de bloques del diseño a realizar en la presente práctica. Podemos distinguir:

1. CONTADOR: Contador con reset síncrono y señal de habilitación. El número de bits del contador será parametrizable utilizando *GENERICS*.
2. COMPARADOR: Comprador síncrono. Los umbrales de comparación serán parametrizables mediante *GENERICS*.
3. FREC_PIXEL: Generador de la frecuencia de píxel. Dividirá la frecuencia de reloj por dos para obtener una señal de 25MHz. Es un bloque tan sencillo que puede ser diseñado directamente en el nivel de jerarquía superior.
4. GENCOLOR: Este bloque asegura que el cañón sólo esté activo el tiempo en que está apuntando a la pantalla. El resto del tiempo, que denominamos en el apartado anterior *retrace time* el color presentantazo debe ser negro (R='0', G='0' y B='0'). Es decir, el cañón debe estar apagado el tiempo de retorno. Se recomienda que este bloque se diseñe directamente en el nivel de jerarquía superior.
5. DIBUJA: Genera los colores (R, G y B) deseados en función de las coordenadas de pantalla. Dependiendo cómo se diseñe este bloque se representará una imagen diferente en pantalla.

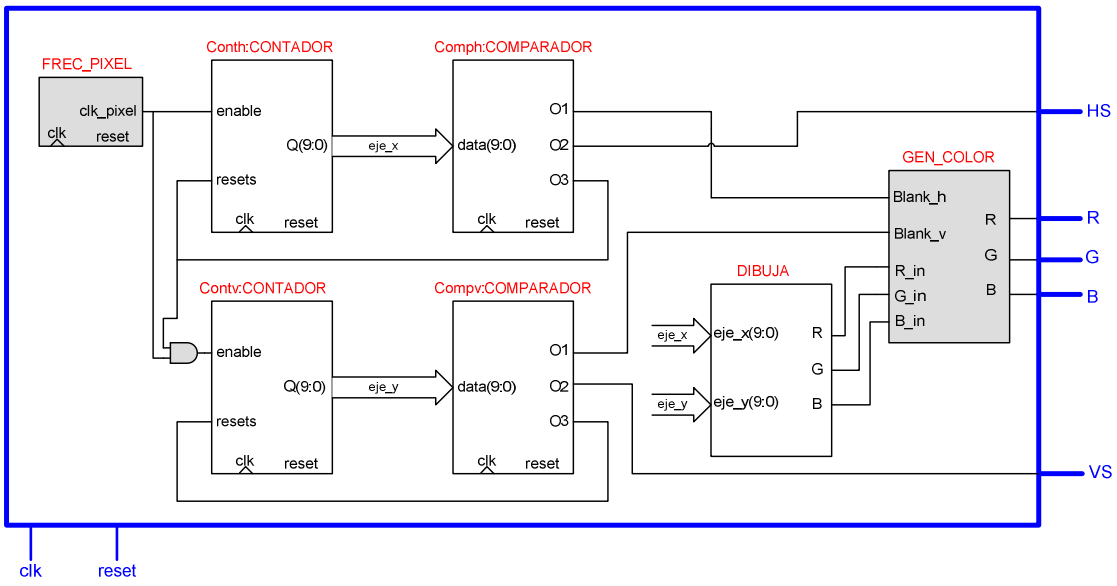


Figura 6. Diagrama de bloques del controlador VGA

Contador

CONTADOR	
Descripción	Contador síncrono de Nbit con habilitación y reset síncrono

Entidad	`entity contador is`
	`  Generic (Nbit: INTEGER := 8);`
	`  Port ( clk : in STD_LOGIC;`
	`  reset : in STD_LOGIC;`
	`  enable : in STD_LOGIC;`
	`  resets : in STD_LOGIC;`
	`  Q : out STD_LOGIC_VECTOR (Nbit-1 downto 0));`
	`end contador;`

Descripción de los puertos	
clk	Reloj
reset	Reset asíncrono activo a nivel alto
enable	Señal de habilitación activa a nivel alto. Si enable='1' el contador avanzará uno en la cuenta en el flanco positivo de reloj.
resets	Reset síncrono activo a nivel alto. Si resets='1' el contador pasará a cero en el siguiente flanco positivo de reloj.
Q	Salida de Nbit igual al valor de la cuenta.

Comparador

COMPARADOR	
Descripción	Comparado síncrono. Las salidas O1, O2 y O3 cambiarán en los flancos positivos de reloj[4].

Entidad	`entity comparador is`
	`  Generic (Nbit: integer :=8;`
	`  End_Of_Screen: integer :=10;`
	`  Start_Of_Pulse: integer :=20;`
	`  End_Of_Pulse: integer := 30;`
	`  End_Of_Line: integer := 40);`
	`  Port ( clk : in STD_LOGIC;`
	`  reset : in STD_LOGIC;`
	`  data : in STD_LOGIC_VECTOR (Nbit-1 downto 0);`
	`  O1 : out STD_LOGIC;`
	`  O2 : out STD_LOGIC;`
	`  O3 : out STD_LOGIC);`
	`end comparador;`

Descripción de los puertos	
clk	Reloj
reset	Reset asíncrono activo a nivel alto
data	Dato de entrada de Nbit

[4] Se realiza el comparador síncrono para filtrar los posibles *gitches*. Es crítico que los pulsos de sincronismo (HS y VS) estén limpios de glitches, pudiendo incluso dañar el monitor CRT.

O1	Cuando se produzca un flanco positivo de reloj: O1='1' si data>End_Of_Screen O1='0' en caso contrario
O2	Cuando se produzca un flanco positivo de reloj: O2='0' si Start_Of_Pulse<data<End_Of_Pulse O2='1' en caso contrario
O3	Cuando se produzca un flanco positivo de reloj: O3='1' si data=End_Of_Line O3='0' en caso contrario

Dibuja

DIBUJA	
Descripción	Bloque combinacional que generará, a partir de las coordenadas de pantalla (eje_x y eje_y), las salidas de color necesarias (R,G y B) para producir una imagen. Se deja la funcionalidad a la imaginación del alumno[5].

Entidad	`entity dibuja is`
	`  Port ( eje_x : in STD_LOGIC_VECTOR (9 downto 0);`
	`  eje_y : in STD_LOGIC_VECTOR (9 downto 0);`
	`  R : out STD_LOGIC;`
	`  G : out STD_LOGIC;`
	`  B : out STD_LOGIC);`
	`end dibuja;`

Descripción de los puertos	
eje_x	Coordenada horizontal del píxel. Sólo estarán dentro de la pantalla los píxeles con coordenadas comprendidas entre 0 y 639.
eje_y	Coordenada vertical del píxel. Sólo estarán dentro de la pantalla los píxeles con coordenadas comprendidas entre 0 y 479.
R	Salida de color rojo para el píxel
G	Salida de color verde para el píxel
B	Salida de color azul para el píxel

[5] Se puede comenzar, por deferencia al profesor Jon Tombs, con el siguiente código:

```
process(eje_x, eje_y)
begin
        G<='1'; B<='1'; R<='1';
        if ((eje_x>279 and eje_x<359) or (eje_y>209 and eje_y<269)) then
                R<='1';G<='0';B<='0';
        end if;
end process;
```

Gen_color

<table>
<tr><th colspan="2">DIBUJA</th></tr>
<tr><td>Descripción</td><td>Bloque combinacional que asegura que el cañón de electrones esté apagado cuando apunta fuera de la pantalla o está volviendo al comienzo de otra línea.
Se puede diseñar:
a. Como un bloque independiente, con la entidad especificada a continuación.
b. Como un process (con una sentencia if) en el nivel de jerarquía superior[6].</td></tr>
<tr><td>Entidad (opción a)</td><td><pre>
entity gen_color is
    Port (blank_h : in STD_LOGIC;
           blank_v : in STD_LOGIC;
           R_in : in STD_LOGIC;
           G_in : in STD_LOGIC;
           B_in : in STD_LOGIC;
           R : out STD_LOGIC;
           G : out STD_LOGIC;
           B : out STD_LOGIC);
end gen_color;
</pre></td></tr>
<tr><td>Process (opción b)</td><td><pre>
gen_color:process(Blank_H, Blank_V, R_in, G_in, B_in)
begin
      if (Blank_H='1' or Blank_V='1') then
            R<='0'; G<='0'; B<='0';
      else
            R<=R_in; G<=G_in; B<=B_in;
      end if;
end process;
</pre></td></tr>
<tr><td colspan="2">Descripción de los puertos</td></tr>
<tr><td>Blank_h</td><td>Indicador de que está fuera de pantalla del comparador horizontal</td></tr>
<tr><td>Blank_v</td><td>Indicador de que está fuera de pantalla del comparador vertical</td></tr>
<tr><td>R_in, G_in, B_in</td><td>Entradas de color procedentes del bloque DIBUJA</td></tr>
<tr><td>R, G, B</td><td>Salidas de color del controlador VGA</td></tr>
</table>

[6] Opción recomentada

Frec_pixel

FREC_PIXEL	
Descripción	Divide por dos la frecuencia de reloj, produciendo en su salida una señal de 25MHz: Se puede diseñar: c. Como un bloque independiente, con la entidad especificada. d. Como un *process* síncrono en el nivel de jerarquía superior[7].

Entidad (opción a)	`entity frec_pixel is`
	`  Port ( clk : in STD_LOGIC;`
	`  reset : in STD_LOGIC;`
	`  clk_pixel : out STD_LOGIC);`
	`end frec_pixel;`

Proceso (opción b)	`div_frec:process(clk,reset)`
	`begin`
	`  if (reset='1') then`
	`  clk_pixel<='0';`
	`  elsif (clk='1' and clk'event) then`
	`  clk_pixel<= not clk_pixel;`
	`  end if;`
	`end process;`

Descripción de los puertos	
clk	Reloj de entrada
reset	Reset asíncrono activo a nivel alto
Clk_pixel	Señal generada de 25MHz

[7] Opción recomendada

Nivel de jerarquía superior

<table>
<tr><th colspan="2">VGA_DRIVER</th></tr>
<tr><td>Descripción</td><td>Nivel de jearquía superior mostrado en la figura 6.

El valor de los generics para las diferentes instancias vale[8]:
a. Conth: Nbit=10.
b. Contv: Nbit=10.
c. Comph: Nbit=10, End_Of_Screen=639, Start_Of_Pulse=655, End_Of_Pulse=751, End_Of_Line=799.
d. Compv: Nbit=10, End_Of_Screen=479, Start_Of_Pulse=489, End_Of_Pulse=491, End_Of_Line=520.</td></tr>
<tr><td>Entidad</td><td><pre>entity VGA_driver is
    Port (clk : in STD_LOGIC;
           reset : in STD_LOGIC;
           VS : out STD_LOGIC;
           HS : out STD_LOGIC;
           R : out STD_LOGIC;
           G : out STD_LOGIC;
           B : out STD_LOGIC);
end VGA_driver;</pre></td></tr>
<tr><td colspan="2">Descripción de los puertos</td></tr>
<tr><td>clk</td><td>Reloj de entrada</td></tr>
<tr><td>reset</td><td>Reset asíncrono activo a nivel alto</td></tr>
<tr><td>VS</td><td>Sincronismo vertical</td></tr>
<tr><td>HS</td><td>Sincronismo horizontal</td></tr>
<tr><td>R</td><td>Rojo</td></tr>
<tr><td>G</td><td>Verde</td></tr>
<tr><td>B</td><td>Azul</td></tr>
</table>

[8] El valor asignado a los generics se ha calculado a partir de los tiempos proporcionados en la figura 5