**Objetivo**

El alumno diseñará un emulador de display de 7 segmentos empleando una tarjeta de desarrollo y un monitor.   
El alumno implementará los números del 0 al 9 y los desplegará en un monitor.

**Introducción**

La VGA, que significa "Video Graphics Array", es un estándar de señal de video que ha desempeñado un papel fundamental en la visualización de imágenes en computadoras y monitores durante décadas. Aunque ha sido ampliamente superado por tecnologías más modernas como el HDMI y el DisplayPort, la VGA sigue siendo relevante en algunos contextos, especialmente en equipos y sistemas más antiguos.

Una de las características distintivas de la señal de video VGA es que consta de cinco señales activas. Estas señales activas son esenciales para transmitir información de video desde una computadora o dispositivo a un monitor o pantalla. Cada una de estas señales activas desencadena la generación de una imagen en el monitor, y juntas, permiten la representación de una imagen completa en la pantalla.

Las cinco señales activas en una señal de video VGA incluyen la señal de sincronización horizontal (HSYNC) y la señal de sincronización vertical (VSYNC), que se encargan de coordinar la posición y el refresco de la imagen en la pantalla. Además, se encuentran las señales de color: rojo (R), verde (G) y azul (B), que determinan los colores y la intensidad de la imagen que se mostrará. Al combinar estas cinco señales activas de manera adecuada, se logra la representación de imágenes nítidas y a todo color en un monitor VGA.

**Desarrollo**

Diagrama

Descripción generada automáticamente

Figura 1. Asignación de cada segmento con sus respectivas coordenadas

Letra, su color y código correspondiente con el que se construirán los números:

**A azul**

If ((row > 200 and row <210) and

(column>110 and column<140)) then – A azul

Red <= (others => ‘0’);

Green <= (others => ‘0’);

Blue <= (others => ‘1’);

**B verde**

Elsif ((row > 210 and row <240) and

(column>140 and column<150)) then – B verde

Red <= (others => ‘0’);

Green <= (others => ‘1’);

Blue <= (others => ‘0’);

**C rojo**

Elsif ((row > 250 and row <280) and

(column>140 and column<150)) then – C rojo

Red <= (others => ‘1’);

Green <= (others => ‘0’);

Blue <= (others => ‘0’);

**D blanco**

Elsif ((row > 280 and row <290) and

(column>110 and column<140)) then – D blanco

Red <= (others => ‘1’);

Green <= (others => ‘1’);

Blue <= (others => ‘1’);

**E cian**

Elsif ((row > 250 and row <280) and

(column>100 and column<110)) then – E cian

Red <= (others => ‘0’);

Green <= (others => ‘1’);

Blue <= (others => ‘1’);

**F amarillo**

Elsif ((row > 210 and row <240) and

(column>100 and column<110)) then – F amarillo

Red <= (others => ‘1’);

Green <= (others => ‘1’);

Blue <= (others => ‘0’);

**G violeta**

Elsif ((row > 240 and row <250) and

(column>110 and column<140)) then – G violeta

Red <= (others => ‘1’);

Green <= (others => ‘0’);

Blue <= (others => ‘1’);

ACTIVIDAD COMPLEMENTARIA:

En esta práctica se mostró como codificar para que se emulen los números 1, 2 y 9 en la pantalla VGA. El alumno implementará además los números 0, 3, 4, 5, 6, 7 y 8.

Para completar esta actividad tuvimos que analizar la forma en que se imprimían cada segmento de la simulación del display y el color que se usaba.

**Código del programa**

Library ieee;

Use ieee.std\_logic\_1164.all;

Use ieee.numeric\_std.all;

Entity p12 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); -- al monitor

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;

Dipsw: in std\_logic\_vector(3 downto 0); -- numeros para

A,B,C,D,E,F,G: out std\_logic ); -- decodificador

End entity p12;

Architecture behavioral of p12 is

Constant h\_period : integer := h\_pulse+h\_bp+h\_pixels+h\_fp;

Constant v\_period : integer := v\_pulse+v\_bp+v\_pixels+v\_fp;

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’);

Constant b0:std\_logic\_vector(3 downto 0):=(others => ‘0’);

Signal h\_count : integer range 0 to h\_period-1 := 0;

Signal v\_count : integer range 0 to v\_period-1 := 0;

Signal conectornum:std\_logic\_vector(6 downto 0); -- coneccion del

Signal reloj\_pixel : std\_logic;

Signal column : integer := 0;

Signal row : integer :=0;

Signal display\_ena : std\_logic ;

Begin

Relojpixel: process (clk50MHz) is

Begin

If rising\_edge(clk50MHz) then

Reloj\_pixel <= not reloj\_pixel;

End if;

End process relojpixel; -- 25mh

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 process coords\_pixel;

Generador\_imagen: process(display\_ena, row, column)

Begin

If(display\_ena = ‘1’) then

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 > 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 > 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’);

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’);

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 > 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>140 and column<150)) then – C rojo

Red <= (others => ‘1’);

Green <= (others => ‘0’);

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 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’);

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>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 seis=>

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 > 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 > 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 siete=>

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’);

Else – fondo

Red <= (others => ‘0’);

Green <= (others => ‘0’);

Blue <= (others => ‘0’);

End if;

When ocho=>

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 > 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 > 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 nueve=>

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 > 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 others=>

Red <= (OTHERS => ‘1’);

Green <= (OTHERS => ‘1’);

Blue <= (OTHERS => ‘0’);

End case;

Else

Red<= (others => ‘0’);

Green <= (others => ‘0’);

Blue<= (others => ‘0’);

End if;

End process generador\_imagen;

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”,

“1110011” when “1001”,

“0000000” when others;

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;

Para el funcionamiento se realizaron las modificaciones siguientes:

Primero al código proporcionado anteriormente se acomodó de forma correcta para hacerlo funcionar para esto se agregó en la sección qué imprime los datos con ayuda de un while e if para identificar cada caso.

Como no estaba completo el código proporcionado en la parte de impresión se agregaron los correspondientes números 3, 4, 5, 6, 7, 8 en el display. Para esto se ubicó el punto de inicio y fin de cada línea del display en el vga

Para mostrar en el display se usó la disposición y color mostrado en el desarrollo de esta práctica.

La asignación de pines fue la siguiente:

Aplicación, Tabla

Descripción generada automáticamente

**Video del funcionamiento del programa:**

<https://youtu.be/ch1m-GuQrXM?si=YNIrCT-JDNF2rIcQ>

**Bibliografía**

Technologies, T. (2020, 5 junio). *DE-10 Lite User Manual*. Copyright © 2003-2017. <https://www.terasic.com.tw/cgi-bin/page/archive_download.pl?Language=China&No=1021&FID=a13a2782811152b477e60203d34b1baa>

Universidad Complutense de Madrid. (2014). *Introducción a la programación en VHDL.* <https://eprints.ucm.es/id/eprint/26200/1/intro_VHDL.pdf>

Instituto Tecnológico de Querétaro. (2016). *Guía básica del VHDL.* <http://www.itq.edu.mx/carreras/IngElectronica/archivos_contenido/Apuntes%20de%20materias/Apuntes_VHDL_2016.pdf>