**Objetivo**

El alumno aplicara los conocimientos y las habilidades obtenidas en el manejo de la señalización VGA, para definir una unidad de control de una cámara digital. Aprenderá además la señalización requerida en el almacenamiento de imágenes digitales en una tarjeta de desarrollo (FPGA-INTEL).

**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.

La cámara digital OV7670 captura imágenes 640x480 pixeles. Opera a 3.3V, aunque cuenta con un regulador que permite polarización de hasta 5V. El formato de salida de video, por defecto es el YUV (A:2:2), aunque puede generar RGB 4:2:2 y RGB 565/555/444. El protocolo de comunicación con la cámara es el SCCB, compatible con el protocolo de comunicación I2C (Integer Integrated Ciucuit). La cámara incluye un módulo para el control del color, de la saturación, del límite, de gama, y de realzado de bordes, entre otros. Estos deben ser configurados escribiendo los valores adecuados en los registros correspondientes.

La cámara opera por default en formato YUV 4:2:2 de 640x480. De la señal entrega por la cámara, solamente se recupera la componente de luminancia y esta componente alimenta a un monitor con entrada VGA, y dependiendo de la capacidad del FPGA es la cantidad de bits que define la componente de luminancia.

La imagen que entrega la cámara es almacenada en una memoria de doble puerto (escritura y lectura) dentro del FPGA.

**Desarrollo**

El código que utilizamos para esta práctica es el mismo que nos fue proporcionado, no se hicieron cambios.

**Código del programa**

library IEEE;

USE IEEE.STD\_LOGIC\_1164.ALL;

USE IEEE.STD\_LOGIC\_ARITH.ALL;

USE IEEE.STD\_LOGIC\_UNSIGNED.ALL;

ENTITY P13 IS

Generic(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; --for this example is 50MHz

red: out std\_logic\_vector (3 downto 0);

green: out std\_logic\_vector (3 downto 0);

blue: out std\_logic\_vector (3 downto 0);

n\_sync: out std\_logic;

n\_blank: out std\_logic;

h\_sync: out std\_logic;

v\_sync: out std\_logic;

sio\_c: out std\_logic:='0';

sio\_d: out std\_logic:='0';

pwdn: out std\_logic:='0';

resetcamera: out std\_logic:='1';

xclk: out std\_logic;

pclk: in std\_logic;

vsync: in std\_logic;

href: in std\_logic;

sample:in std\_logic\_vector (7 downto 0) );

END ENTITY P13;

ARCHITECTURE BEHAVIORAL OF P13 IS

--CONTADORES

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

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

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

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

--vram

type matrix is array (0 to 307199) of std\_logic\_vector (3 downto 0);

signal memory: matrix;

signal memorysize : integer:=h\_pixels\*v\_pixels;

signal pixel: std\_logic\_vector( 3 downto 0);

--pixel\_capture - vram

signal data: std\_logic\_vector( 3 downto 0);

signal wrclk: std\_logic;

--pixel\_capture: internal signals

type states is (e0,e1,e2,e3,e4,e5,e6,e7);

signal state: states:=e0;

signal wraddr: integer range 0 to h\_pixels\*v\_pixels:=0;

signal clken: std\_logic:='0';

--Reloj de pixel

signal reloj\_pixel: STD\_LOGIC:='0';

--vga\_controller - vram

signal rdaddress: integer:=0;

--vga\_controller - hw\_image\_generator

signal display\_ena: std\_logic:='1'; --display enable ('1' = display time, '0' = blanking time)

signal column: integer; --horizontal pixel coordinate

signal row: integer; --vertical pixel coordinate

BEGIN

div25MHz: process (clk50MHz) is

begin

if rising\_edge(clk50MHz) then

reloj\_pixel <= not reloj\_pixel;

end if;

end process div25MHz;

-- Reloj a la camara

XCLK<=reloj\_pixel;

-- Controlador del monitor

Contadores : process (reloj\_pixel)

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;

rdaddress <= column + (row \* h\_pixels);

senial\_hsync : process (h\_count)

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 (v\_count)

begin

--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\_col: process(h\_count)

begin

if (h\_count < h\_pixels) then

column <= h\_count;

end if;

end process coords\_pixel\_col;

coords\_pixel\_row: process(v\_count)

begin

if (v\_count < v\_pixels) then

row <= v\_count;

end if;

end process coords\_pixel\_row;

captura\_pixel:process (pclk)

variable scanrow : integer range 0 to v\_pixels:=0;

begin

if rising\_edge(pclk) then

case state is

when e0=>

if vsync='1' then

state<=e1;

end if;

when e1=>

if vsync='0' then

state<=e2;

end if;

when e2=>

if href='1' then

state<=e3;

end if;

when e3=>

data<=sample(3 downto 0);

state<=e4;

when e4=>

clken<='1';

state<=e5;

when e5=>

data<=sample(3 downto 0);

clken<='0';

wrAddr<=wrAddr+1;

state<=e6;

when e6=>

clken<='1';

if href='1' then

state<=e5;

else

state<=e7;

end if;

when e7=>

clken<='0';

wrAddr<=wrAddr+1;

if scanrow<v\_pixels-1 then

scanrow:=scanrow+1;

state<=e2;

else

scanrow:=0;

wrAddr<=0;

state<=e0;

end if;

end case;

end if;

end process captura\_pixel;

generador\_imagen: process(display\_ena, row, column,pixel)

begin

if(display\_ena = '1') then

if ((row > 0 and row <480) and

(column>0 and column<640)) THEN

red <= (others => '0');

green<=(others => '0');

blue<=(others => '0');

elsif ((row > 0 and row <480) and

(column>0 and column<640)) then

red <= (others => '0');

green<=(others => '0');

blue<=(others => '0');

elsif ((row > 0 and row <480) and

(column>0 and column<640)) then

red <= (others => '0');

green<=(others => '0');

blue<=(others => '0');

elsif (row<v\_pixels and column<h\_pixels) then

red<= pixel;

green<= pixel;

blue<= pixel;

else

red <= (others => '1');

green<= (others => '1');

blue <= (others => '1');

end if;

else

red<= (others => '1');

green <= (others => '1');

blue<= (others => '1');

end if;

end process generador\_imagen;

wrvram:process (clken) -- sección de escritura

begin

if rising\_edge(clken)then

memory(wraddr)<=data;

end if;

end process wrvram;

rdvram: process (reloj\_pixel) --sección de lectura

begin

if falling\_edge(reloj\_pixel) then

pixel<=memory(rdaddress);

end if;

end process rdvram;

END BEHAVIORAL;

Al inicio del código se declaran todas las bibliotecas que vamos a utilizar en el código. Enseguida se crea la entidad para la práctica, donde declaramos las constantes y variables de entrada y salida que utilizaremos.

Dentro de la arquitectura de la práctica primero declaramos todas las señales que se utilizarán para posteriormente comenzar a declarar todos los procesos (módulos) que se necesitan. Primero tenemos el proceso para crear un divisor de frecuencia a 25MHz. El segundo proceso es el de los contadores que cuentan los pixeles horizontales y verticales. El tercer y cuarto proceso son las sincronizaciones horizontal y vertical, respectivamente, que nos permiten llevar un control sobre la posición en la que nos entramos en el monitor, es por eso que son una sincronización. El quinto y sexto proceso son los que nos permiten llevar un control sobre las filas y las columnas de pixeles en la pantalla. El séptimo proceso es el que captura el pixel actual que esté leyendo la tarjeta. El octavo proceso es el generador de imagen, que es el que permite dibujar el pixel actual en pantalla. Finalmente, los noveno y décimo proceso son unas memoria RAM, una para escritura y otra para lectura, respectivamente, que sirven para almacenar en memoria los datos que envíe la cámara (la foto en sí).

La asignación de pines fue la siguiente:

**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>