Protocolo VGA basado en VHDL para tarjeta Basys 3

Gabriel de la Iglesia, Laura Granados
Estudiantes (Universidad Latina de Panamá, Vía Ricardo J. Alfaro, Panamá, Panamá 87-877)

Resumen – Este proyecto es un demo desarrollado con Vivado e implementando lenguaje VHDL aplicado a la tarjeta Basys 3 haciendo uso del hardware para la implementación practica de un controlador VGA con generador de patrones definidos por el usuario.

I. INTRODUCCIÓN

A. VIDEO GRPAHICS ARRAY (VGA)

Se trata de un sistema gráfico que presentó la empresa estadounidense International Business Machines (IBM) a fines de la década de 1980 y que se convirtió, por su popularidad, en una especie de estándar para las computadoras. [1]

Ilustración 1. Conector VGA

La implementación de este arreglo posibilita una resolución de 320 x 200 (empleando 256 colores) o de 640 x 480 (usando 16 colores). Cabe resaltar, por otra parte, que el conector VGA, presente en gran parte de los monitores de PC y de las placas de video, dispone de 15 pines.

Un frame de VGA posee normalmente 480 líneas y cada una de ellas 680 pixeles, para conseguir pintar un frame, los circuitos deflectores se encargan de desviar el haz de electrones desde la izquierda a la derecha y de arriba abajo a través de toda la pantalla. Los circuitos de deflexión necesitan dos señales de sincronización para conseguir controlar el inicio y la parada de estos.

B. FIELD PROGRAMMABLE GATE ARRAY (FPGA)

Las FPGAs son dispositivos digitales, cuyo nombre se traduce a Matriz de Puertas de Cambo Programables, y estas se constituyen internamente de cables, puertas lógicas, biestables, y puertos de entrada y salida.

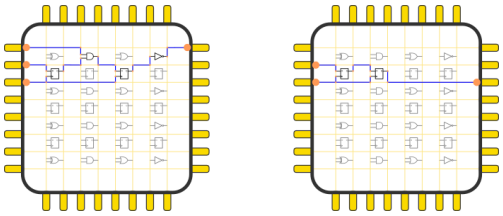

Ilustración 2. Diagrama de FPGA

El usuario programa en un lenguaje de descripción de Hardware (VHDL o Verilog) y este describe las conexiones físicas que debe realizar el dispositivo para ejecutar la tarea descrita. Estas instrucciones son cargadas al dispositivo mediante un archivo Bitstream, un archivo que comunica una serie de bits al dispositivo para realizar las interconexiones. [2]

II. OBJETIVOS

En el desarrollo del proyecto, se plantearon algunos objetivos:

1. Desarrollar un controlador VGA de fácil implementación para la Basys 3.
2. Demostrar mediante ejemplos prácticos, la flexibilidad del uso de VGA para transmisión de información visual.
3. Plantear de manera práctica, los conceptos de la programación de VHDL para FPGA.

III. MATERIALES Y MÉTODOS

A. Materiales

Para el desarrollo de este proyecto, se hizo uso de:

- Digilent Basys 3
- Cable VGA
- Monitor/Proyector

B. Métodos

Para hacer posible el diseño y desarrollo del modelo, se tuvo que establecer un proceso de desarrollo, el cual consistía en:

1. Revisión de material bibliográfico: Se realizó una búsqueda de fuentes de información relacionadas al tema a desarrollar. Estas fuentes debían contener información técnica sobre el tema a desarrollar, ejemplos de proyectos similares o especificaciones sobre los materiales a utilizar
2. Definición de requisitos para el desarrollo del proyecto: Al finalizar la revisión bibliográfica, se definieron las variables intrínsecas al proyecto, los elementos fundamentales, los adicionales no esenciales y el concepto general de funcionamiento.
3. Desarrollo de fundamentos del proyecto: Se inició el proceso de desarrollo de prototipos, teniendo como objetivo realizar un prototipo que cumpliera con la función principal, para luego desarrollar los detalles a partir de ahí.

IV. Resultados

A. *Divisor de Reloj*

En los estándares para el desarrollo de controladores VGA, se debe manejar un reloj a frecuencia especifica según la resolución a trabajar. En nuestro caso se maneja una resolución de 640*480 pixeles, lo que nos indica que debemos trabajar con una frecuencia de reloj de 25MHz.

Sl No.	Resolution @ 60Hz	Clock frequency (MHz)
1	640 x 480(VGA)	25.175
2	800 x 600(SVGA)	40
3	1024 x 768(XGA)	65
4	1366 x 768(WXGA)	85.86

Ilustración 3. Frecuencia de reloj según resolución

Para el desarrollo de la señal de reloj a 25MHz es necesario crear un divisor de reloj [3]. Esto ya que la tarjeta Basys 3 contiene un reloj que funciona a una frecuencia de 100MHz.
En nuestro caso, se hizo uso del IP Core de Clock Wizard que se encuentra en el Vivado HLx [4]

```
component clk_wiz_0
port (
CLK_IN1: in std_logic;-- RELOJ DE ENTRADA= 100MHZ
CLK_OUT1: out std_logic);--RELOJ DE SALIDA = 25MHZ
end component;
```

B. *Sincronización Horizontal*

Para el desarrollo de un controlador VGA funcional, es necesario contar con señales de sincronización horizontal, así como un contador de los pixeles en pantalla.

La señal de sincronización horizontal (HSync) es una señal dada al monitor que le indica que deje de dibujar la línea horizontal actual y comience a dibujar la siguiente línea. La cantidad de tiempo que esto tarda en ocurrir se mide en Hz (Hertz), que es una medida de ciclos por segundo. [5]
En nuestro proyecto, esta señal es una salida declarada.

```
Hsync : out  STD_LOGIC;
```

En el caso del contador horizontal, utilizamos la señal Hcounter, la cual manejara números enteros.
Para le control de las líneas horizontales en nuestro protocolo, se utilizo el siguiente código:

```
if(hcounter = 799) then
    hcounter <= 0;
```

El código en la parte superior, fue elaborado para cumplir la función de reiniciar el contador horizontal luego de exceder el numero indicado.
En el caso del protocolo de control, se estructuro de la siguiente manera:

```
    -- PULSO DE SINCRONIZACION HORIZONTAL
    --INICIA NUEVA FILA
if (hcounter >= 656 and hcounter < 752) then
    Hsync <= '0';
else
    Hsync <= '1';
    end if;
```

C. *Sincronización Vertical*

Adicionalmente, fue necesaria la implementación de una sincronización vertical, el cual es una opción que obliga a que la frecuencia de fotogramas de un juego de computadora coincida con la frecuencia de actualización del monitor de pantalla. Aunque efectivamente establece un límite en los cuadros por segundo. [6]
Al igual que HSync, VSync es declarada como una salida en nuestro proyecto.

```
Vsync : out  STD_LOGIC;
```

Para este caso, también es necesario el uso de un contador vertical, y se controlo el reinicio del mismo con el código a continuación:

```
if vcounter = 524 then
    vcounter <= 0;
```

Y por último, la sincronización de las señales se planteo de la siguiente manera:

```
    -- PULSO DE SINCRONIZACION VERTICAL
    --INICIA NUEVA COLUMNA
if (vcounter >= 490 and vcounter < 492) then
    Vsync <= '0';
else
    Vsync <= '1';
    end if;
```

D. *Posibles salidas de colores*

Se tiene la salida de los canales R (rojo), G (verde) y B (azul), cada uno cuenta con 4 bits para su escala. Se declararon salidas de tipo vector para el manejo de ellas.

```
vgaR : out  STD_LOGIC_VECTOR (3 downto 0) := "0000";
vgaG : out  STD_LOGIC_VECTOR (3 downto 0) := "0000";
vgaB : out  STD_LOGIC_VECTOR (3 downto 0) := "0000"
```

Se producen las siguientes opciones de colores:

Ilustración 4. Escala de Colores del Controlador

E. Patrones de Salida

Habiendo desarrollado las diferentes partes del controlador, se desarrollaron 4 patrones de salida para el sistema.

1. Franjas Horizontales
2. Franjas Verticales
3. Cruza blanca con esquinas negras
4. Patrón No identificado

El caso del patron con franjas horizontales, se programo de la siguiente manera:

```
            else if (hcounter < 640 and vcounter < 480) and (sw<= "01") then
                if (vcounter < 80) then
--FRANJA EN PANTALLA
                        vgaR <= "1111";
                        vgaG <= "0000";
                        vgaB <= "0000";
                elsif ( vcounter < 160) and (vcounter > 79) then
--FRANJA EN PANTALLA
                        vgaR <= "0000";
                        vgaG <= "1111";
                        vgaB <= "0000";
                elsif (vcounter < 240)and (vcounter > 159) then
--FRANJA EN PANTALLA
                        vgaR <= "0000";
                        vgaG <= "0000";
                        vgaB <= "1111";
                elsif (vcounter < 320)and (vcounter > 239) then
--FRANJA  EN PANTALLA
                        vgaR <= "0000";
                        vgaG <= "1111";
                        vgaB <= "0000";
                elsif (vcounter < 400)and (vcounter > 319) then
--FRANJA  EN PANTALLA
                        vgaR <= "1111";
                        vgaG <= "0000";
                        vgaB <= "0000";
                else
--FRANJA EN PANTALLA
                        vgaR <= "0000";
                        vgaG <= "0000";
                        vgaB <= "1111";
                    end if;
```

Lo que producía una salida de la siguiente manera:

Ilustración 5. Salida simulada del patrón 1

Al desarrollar el patron de franjas verticales, tuvimos que codificarlo de la siguiente manera:

```
            if (hcounter < 640 and vcounter < 480) and (sw<= "00") then
                if (hcounter < 213) then
--FRANJA  EN PANTALLA
                        vgaR <= "0000";
                        vgaG <= "0000";
                        vgaB <= "1111";
                    elsif (hcounter < 426) and (hcounter > 212) then
--FRANJA EN PANTALLA
                        vgaR <= "1111";
                        vgaG <= "0000";
                        vgaB <= "0000";
                    else
-- FRANJA EN PANRALLA
                         vgaR <= "0000";
                         vgaG <= "1111";
                         vgaB <= "0000";
                    end if;
```

Produciendo una salida como se muestra a continuación:

Ilustración 6. Salida simulada del patrón 2

Para el desarrollo de la salida de la cruz con esquinas negras, se utilizo la posición de los pixeles para ubicar los cuadrantes blancos para formar la cruz, de la siguiente manera:

```
            else if (hcounter < 640 and vcounter < 480) and (sw<= "10") then
                    if (hcounter < 213) and (vcounter < 160) then
--CUADRO BLANCO EN PANTALLA
                        vgaR <= "1111";
                        vgaG <= "1111";
                        vgaB <= "1111";
                    elsif (hcounter < 213) and (vcounter > 320) then
--RECUADRO BLANCO EN PANTALLA
                        vgaR <= "1111";
                        vgaG <= "1111";
                        vgaB <= "1111";
                    elsif (hcounter > 426) and (vcounter < 160) then
-- RECUADRO BLANCO EN PANTALLA
                         vgaR <= "1111";
                         vgaG <= "1111";
                         vgaB <= "1111";
                    elsif (hcounter > 426) and (vcounter > 320) then
-- RECUADRO BLANCO EN PANTALLA
                         vgaR <= "1111";
                         vgaG <= "1111";
                         vgaB <= "1111";
                        else
--ESPACIO RESTANTE EN NEGRO
                        vgaR<=  "0000";
                        vgaG<=  "0000";
                        vgaB<=  "0000";
                    end if;
```

Al momento de mostrarse la salida, se dio de la siguiente manera:

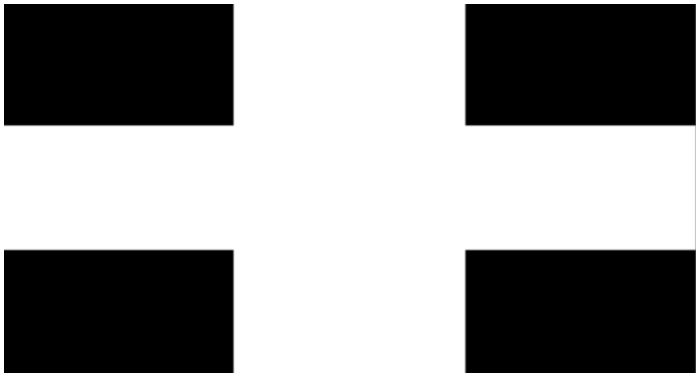

Ilustración 7. Salida simulada del patrón 3

Y por ultimo, se desarrollo el patrón no identificado, el cual consistía en programar un patrón que generara pixeles blancos en pantalla, con un fondo rojo.

```
                if (hcounter > 160) and (hcounter<480) and (vcounter >
                160) and (vcounter <320) then
--PIXELES  EN PANTALLA
                        vgaR <= "1111";
                        vgaG <= "1111";
                        vgaB <= "1111";
                    else
-- RESTO
                        vgaR <= "1111";
                        vgaG <= "0000";
                        vgaB <= "0000";
                    end if;
```

La salida generaba un patrón un poco defectuoso, pero era posible apreciar el área de diferente color.

V. Conclusiones

Luego de haber realizado el protocolo controlador de VGA fue posible notar la importancia de la implementación de las señales de sincronización, así como la inclusión de señales para realizar operaciones antes de controlar las salidas.

A pesar de que el protocolo diseñado contara con un rango limitado de patrones de salida disponibles, fue posible diseñar un controlador que manejara de manera precisa los colores a la salida, permitiendo una mayor precisión al momento de transmitir información visual.

VI. Referencias

[1] Definicion.de, «VGA,» [En línea]. Available: https://definicion.de/vga/.

[2] Planet Chatbot, «Qué es una FPGA y por qué jugarán un papel clave en el futuro,» [En línea]. Available: https://planetachatbot.com/qué-es-una-fpga-y-por-qué-jugarán-un-papel-clave-en-el-futuro-e76667dbce3e.

[3] Surf VHDL, «How to implement a Clock Divider,» [En línea]. Available: https://surf-vhdl.com/how-to-implement-clock-divider-vhdl/.

[4] Xilinx, «Clocking Wizard v6.0 LogiCORE IP Product Guide,» Xilinx, 2020.

[5] ComputerHope, «HSync,» 2017. [En línea]. Available: https://www.computerhope.com/jargon/h/hsync.htm.

[6] ComputerHope, «Vsync,» 2017. [En línea]. Available: https://www.computerhope.com/jargon/v/vsync.htm.