Escuela Politécnica Superior de Linares

Grado en Ingeniería Telemática

UNIVERSIDAD DE JAÉN
Escuela Politécnica Superior de Linares

Trabajo Fin de Grado

CONTROLADORA PARA PANTALLA TFT LCD 800x600 BASADA EN FPGA PARA OSCILOSCOPIOS LECROY 93XX

Alumno: **David Godoy Chiclana**

Tutor: Prof. D. Gregorio Godoy Vilches
Depto.: Ingeniería Electrónica y Automática

Septiembre, 2017

ÍNDICE GENERAL

ÍNDICE DE LA MEMORIA

ÍNDICE DEL ANEXO

ÍNDICE DE PLANOS

ÍNDICE DE LOS PRESUPUESTOS

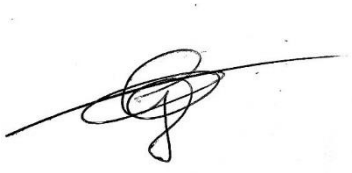
Índice de Tablas

Memoria

Controladora para pantalla TFT-LCD 800x600 basada en FPGA para osciloscopios LeCroy 93XX

Alumno: David Godoy Chiclana

Tutor: Prof. D. Gregorio Godoy Vilches



Departamento: Ingeniería Electrónica y Automática

Septiembre, 2017

ÍNDICE DE LA MEMORIA

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

ÍNDICE DE ECUACIONES

1 OBJETO

El presente documento especifica cómo realizar la interconexión entre un osciloscopio LeCroy de la serie 93XX que, a pesar de estar obsoleto, posee grandes prestaciones, y una pantalla TFT-LCD de 800x600 píxeles mediante una controladora basada en FPGA.

Se representará lo más fielmente posible lo que es mostrado por la pantalla de tubo de rayos catódicos del osciloscopio LeCroy de la serie 93XX, así como aprovechar las características que incorpora la pantalla TFT-LCD. Con este fin, se detallará las soluciones contempladas que puedan llevar a la resolución del proyecto y se justificará la solución final adoptada.

Se elaborará una memoria justificativa del estudio realizado y del diseño adoptado, así como unos planos para la placa controladora que se diseñe, unos anexos que contengan el software del proyecto y un presupuesto con el precio de los distintos elementos que forman parte de la solución final del proyecto.

Es también objeto del proyecto crear un prototipo funcional y llevar a cabo las pruebas necesarias para verificar el correcto funcionamiento de la solución final adoptada.

2 ALCANCE

El alcance de este proyecto incluye el diseño tanto del hardware como del software necesario para conectar un osciloscopio LeCroy de la serie 93XX a una pantalla TFT-LCD de 800x600 píxeles.

A nivel de hardware, se detallarán todos los cambios que son necesarios realizar en el osciloscopio LeCroy de la serie 93XX, para hacerlo compatible con el sistema de desarrollo basado en FPGA, que permitan su interconexión con la pantalla TFT-LCD de 800x600 píxeles, así como la descripción y el diseño de la placa controladora basada, justificando su uso y explicando su funcionamiento.

A nivel del software, se diseñará y describirá el software que debe ejecutar la FPGA y que la hará compatible el osciloscopio con la pantalla LCD-TFT de 800x600 píxeles.

Es también parte del alcance del proyecto, la realización de un prototipo con la solución final adoptada, así como las pruebas para corroborar su correcto funcionamiento.

Por último, el proyecto se realizará en tres fases en las que se pretende:

En la primera, ser capaz de representar la información tal cual es enviada por la tarjeta de control del osciloscopio LeCroy de la serie 93XX, tal y como sale de esta, a la pantalla TFT-LCD de 800x600 píxeles.

En la segunda, ser capaz de representar la información que proviene del osciloscopio, de forma que se pueda visualizar correctamente en la pantalla TFT-LCD de 800x600 píxeles.

En la tercera fase, ser capaz de representar la información que proviene del osciloscopio LeCroy de la serie 93XX, de forma que se pueda visualizar correctamente y, además, que se pueda modificar el color que se representa en pantalla mediante los mandos de la botonera del osciloscopio.

3 ANTECEDENTES

3.1 Sistema de imagen de los osciloscopios LeCroy de la serie 93XX

El sistema de representación de imágenes de los osciloscopios LeCroy de la serie 93XX es un tubo de rayos catódicos como el de la Figura 3.1. Antes de pasar a explicar las particularidades de esta gama de osciloscopios, se va a explicar el funcionamiento genérico de un tubo de rayos catódicos.

Figura 3.1: Tubo de rayos catódicos del osciloscopio

3.2 Tubo de rayos catódicos

Un tubo de rayos catódicos consiste en un tubo de vacío, en el que un cañón dispara un haz de electrones, que son dirigidos mediante un campo eléctrico hacia una pantalla de vidrio recubierta con fósforo y plomo (ver Figura 3.2).

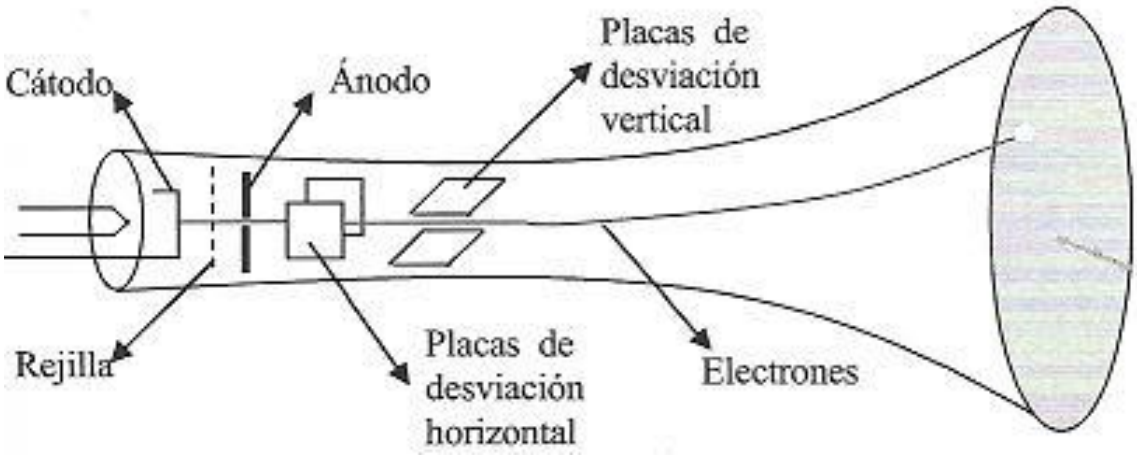


Figura 3.2: Esquema de un tubo de rayos catódicos

Fuente: http://intercentres.edu.gva.es/iesleonardodavinci/Fisica/Campo-electrico/Electrico12.htm

3.2.1 Funcionamiento

El haz de electrones se genera mediante emisión termoiónica en el cátodo, gracias al campo eléctrico que se genera entre este y el ánodo. Este campo eléctrico también es el responsable de acelerar los electrones. Este haz de electrones se pasa por una rejilla para conseguir un haz fino. Posteriormente mediante otros dos campos eléctricos, generados por las placas de desviación horizontal y las placas de desviación vertical, se consigue dirigir el haz hasta un punto específico en la pantalla, este punto donde se conduce el haz para que incida, se denomina píxel.

Cuando el haz de electrones incide en la pantalla, estos producen un destello proporcional al número e intensidad de los electrones.

En el caso de los tubos de rayos catódicos a color, se utilizan tres cañones de electrones que disparan tres haces independientes pero que son dirigidos al mismo píxel para producir color, como puede verse en la Figura 3.3.

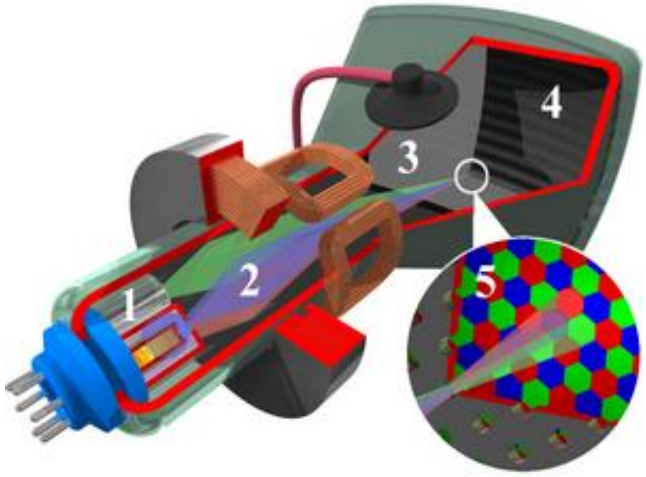


Figura 3.3: Haz de rayos para un tubo de rayos catódicos a color

Fuente: http://fisica-teleco.blogspot.com.es/2008/02/tubo-de-rayos-catdicos.html

El fósforo se utiliza para generar el color de la pantalla, en el caso de las pantallas a color, se utiliza fósforo de tres colores, rojo, verde y azul, que, al combinarse, generan toda la gama de colores. El plomo es utilizado para bloquear los rayos X que se generan, y así, proteger al usuario.

3.2.2 Barrido

En los tubos de rayos catódicos, el haz de electrones debe ser movido por toda la pantalla, llegando a todos los píxeles y lo suficientemente rápido para crear la sensación de imagen en movimiento. Esto se consigue gracias a las placas de desviación horizontal y a las placas de desviación vertical, que generan campos eléctricos que mueven el haz de electrones al punto correspondiente.

Este movimiento que cubre toda la pantalla se denomina barrido.

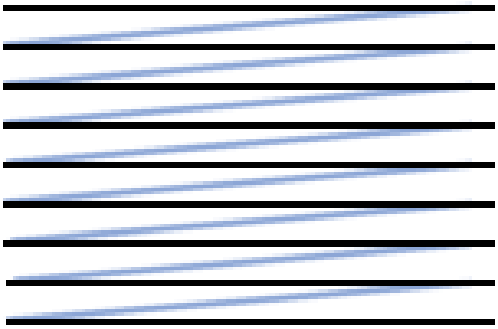


Figura 3.4: Barrido en un tubo de rayos catódicos

Fuente: https://www.ecured.cu/Tubo_de_rayos_cat%C3%B3dicos

El barrido se realiza como se muestra en la Figura 3.4, es decir:

- Se barren todos los píxeles de una fila de la pantalla.
- Se salta de fila y se repite el proceso hasta barrer todos los píxeles.
- Por último, se reconduce el haz a la posición inicial.

3.3 VGA

Video Graphics Array o VGA, es un estándar de visualización gráfica analógico lanzado por IBM en 1987, para conectar ordenadores a monitores analógicos, en concreto, de tubo de rayos catódicos. Los componentes responsables de generar, procesar y almacenar señales de video son el controlador gráfico y controlador del monitor, como puede verse en la Figura 3.5.

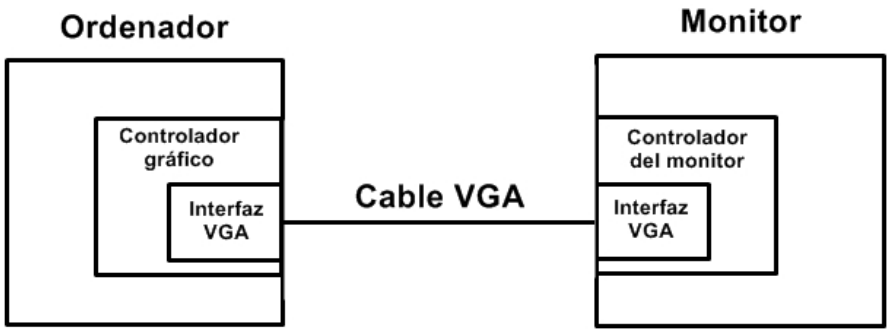


Figura 3.5: Elementos VGA

VGA se caracteriza por una resolución de 640x480 píxeles a 60Hz, utilizar tres colores para cada píxel y un conector de 15 pines. De esos 15 pines, cinco son para señales activas, en concreto, dos pines son para sincronización y tres para color.

3.3.1 Pines

El conector que utiliza VGA para la transmisión de datos es de 15 pines y se denomina DB15. En concreto, para un conector hembra, la posición y ordenación de los pines es la mostrada en la Figura 3.6.

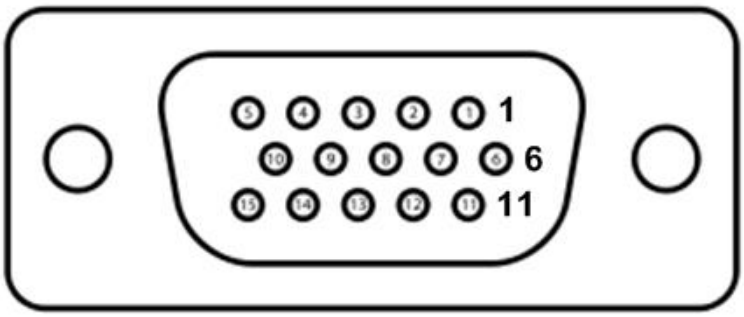


Figura 3.6: Puerto VGA

Fuente:
https://upload.wikimedia.org/wikipedia/commons/thumb/3/30/DE15_Connector_Pinout.svg/300px-DE15_Connector_Pinout.svg.png

Según la Video Electronics Standards Association (VESA), a través del documento llamado Display Data Channel (DDC), se define cada pin y su uso. En el caso de un DB15 y según la DDC1 y utilizando la configuración más simple obtendríamos la Tabla 3.1:

Pin	Nombre	Dirección	Configuración	Impedancia/Nivel
1	Rojo	Al monitor	Conexión analógica	75 Ω, 0V-0.7V p-p
2	Verde	Al monitor	Conexión analógica	75 Ω, 0V-0.7V p-p ó 0.3V-1V p-p
3	Azul	Al monitor	Conexión analógica	75 Ω, 0V-0.7V p-p
4	ID2	Del monitor	N/C	
5	GND	Al monitor	Masa	
6	RGND	Al monitor	Masa del rojo	
7	VGND	Al monitor	Masa del verde	
8	AGND	Al monitor	Masa del azul	
9	Sin uso o +5V	Al monitor	N/C	
10	SGND	Al monitor	Masa de sincronización	
11	ID0	Del monitor	N/C	
12	Data	Del monitor	N/C	
13	HSYNC	Al monitor	Conexión digital	
14	VSYNC	Al monitor	Conexión digital	
15	ID3	Del monitor	N/C	

Tabla 3.1: Señales de puerto VGA

3.3.2 Color

En VGA, la representación de imágenes se realiza gracias a una tarjeta gráfica que convierte señales digitales (con las que trabaja la tarjeta) en analógicas y que son mandadas al monitor correspondiente.

La tarjeta, para representar el color, utiliza bits y dispone de tres pines para transmitírselo al monitor. Si utilizásemos N bits por color, se pueden obtener 2^{N} niveles analógicos. Si tienes tres señales, obtendrás 2^{3N} colores. Esta configuración se denomina RGB (Red, Green, Blue) porque utiliza tres señales para representar el color de un píxel.

Por ejemplo, si se usase un bit por color, se obtendrían 8 colores, como puede verse en la Tabla 3.2:

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

Tabla 3.2: Mezcla de colores VGA con 1 bit

Por último, las señales analógicas que se generan, tienen una tensión entre 0 y 0.7 voltios y van sobre cables con dos resistencias en paralelo de 75 Ω (37.5 Ω). Como se mostró en la Tabla 3.2, la tensión de la señal verde puede ser 0.3 voltios mayor.

3.3.3 Sincronización

Una pantalla típica de VGA tiene 480 líneas con 640 píxeles por línea. Para indicar qué píxel se va a pintar, se utilizan señales de sincronización. En VGA, el barrido se realiza de izquierda a derecha y de arriba abajo. Las señales encargadas de este fin son la señal de sincronización horizontal (HSYNC) y vertical (VSYNC). Ver la Figura 3.7.

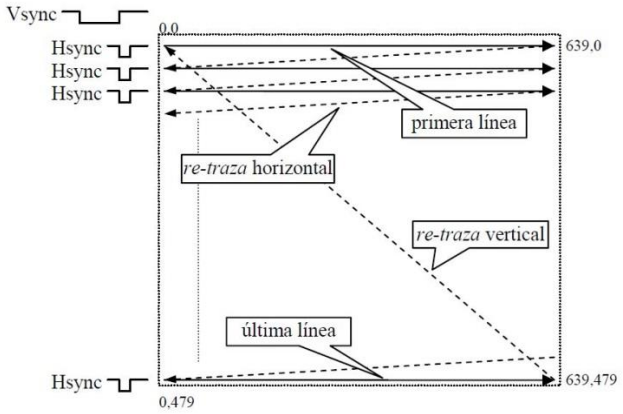


Figura 3.7: Barrido en VGA

Fuente: http://www2.elo.utfsm.cl/~lsb/elo211/labs/2006/elo212-lab08-0106.pdf

Ambas señales, HSYNC y VSYNC, son generadas por el controlador gráfico utilizando la señal de reloj. Este reloj, para la configuración típica, tiene una frecuencia de 25.175MHz.

Los intervalos de tiempo en los que se dividen ambos sincronismos son los siguientes:

- HSYNC_T: Duración total del pulso.
- HSYNC_W: Señal de sincronismo.
- HSYNC_E: 'Front porch'
- HSYNC_S: 'Back porch'
- X_ON: Tiempo en el que se está pintando un píxel.
- X_OFF: Tiempo en el que no se está pintando.

3.3.3.1 *Sincronización horizontal*

En VGA, la sincronización horizontal se realiza barriendo una fila de píxeles de la pantalla. Esta se compone de pulsos, uno por línea. Habiendo un pulso de reloj por píxel. Los tiempos correspondientes a los periodos de esta sincronización, se muestran la Tabla 3.3.

Horizontal	# de Píxeles	Tiempo en µs
HSYNC_T	800	31.77
HSYNC_W	96	3.77
HSYNC_E	16	0.94
HSYNC_S	48	1.89
X_ON	640	25.17
X_OFF	160	6.6

Tabla 3.3: Temporización horizontal en VGA

En la Figura 3.8, se muestran esos intervalos de tiempo:

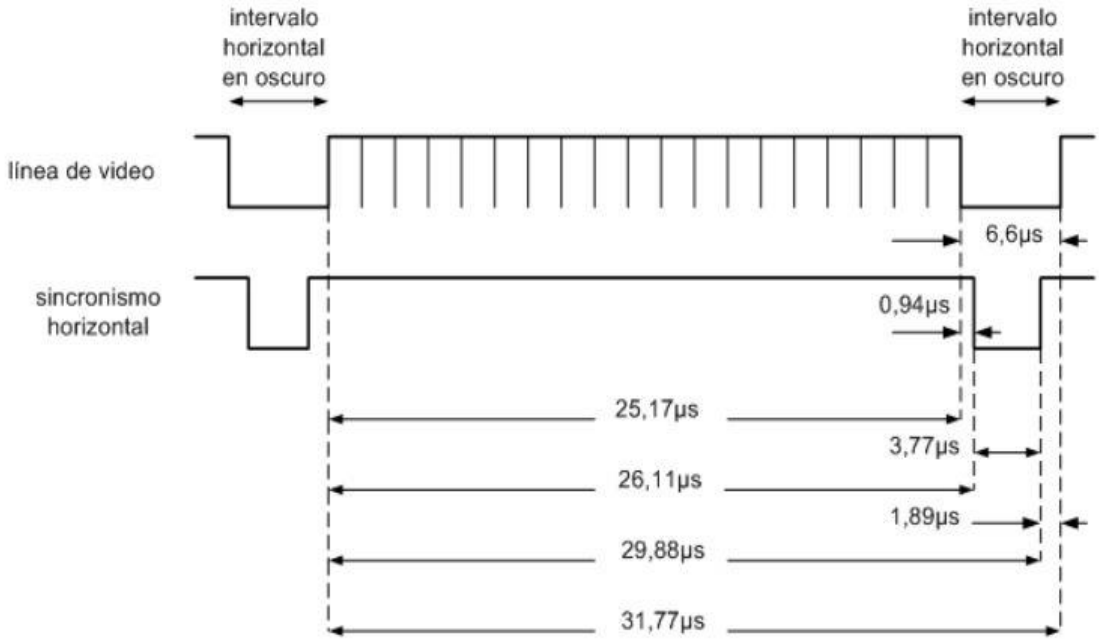


Figura 3.8: Sincronización Horizontal en VGA

Fuente: file:///C:/Users/David/Downloads/Practica_3_DSEDA_CTRL_VGA.pdf

3.3.3.2 *Sincronización vertical*

En VGA, los pulsos de la sincronización vertical marcan cuando se ha de comenzar a representar una nueva pantalla. Los tiempos correspondientes a los periodos de esta sincronización, se muestran la Tabla 3.4.

Vertical	# de líneas verticales	Tiempo en ms
VSYNC_T	525	16.784
VSYNC_W	10	0.064
VSYNC_E	2	0.45
VSYNC_S	33	1.02
Y-ON	480	15.25
Y-OFF	45	1.534

Tabla 3.4: Temporización vertical en VGA

En la Figura 3.9, se muestran esos intervalos de tiempo:

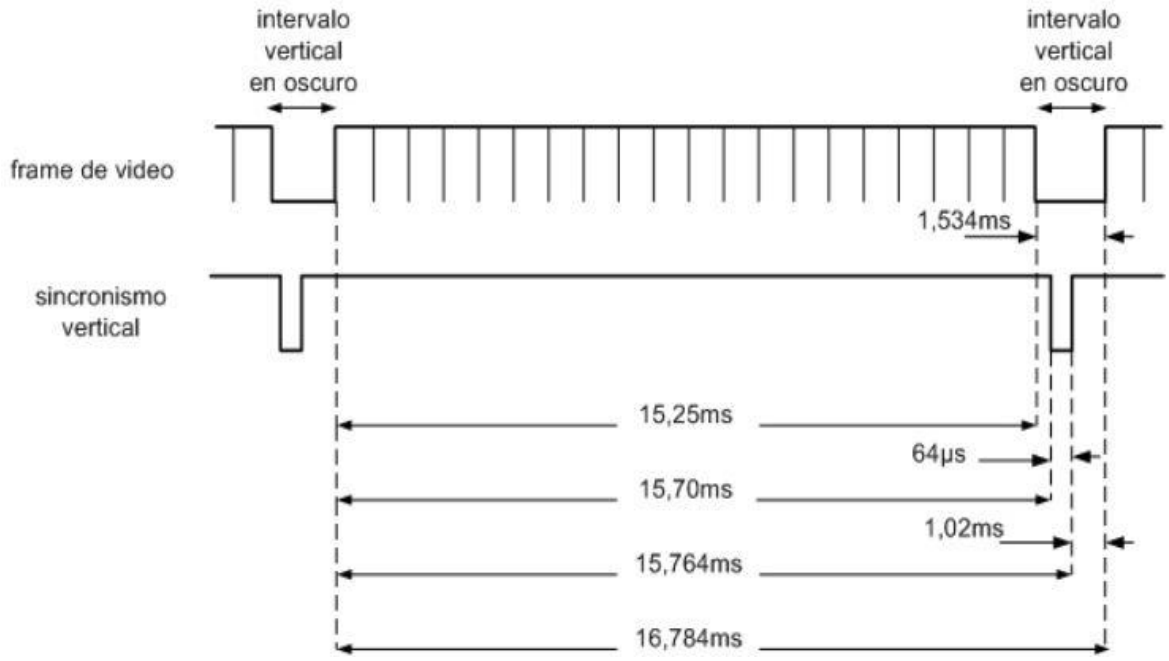


Figura 3.9: Sincronización Vertical VGA

Fuente: file:///C:/Users/David/Downloads/Practica_3_DSEDA_CTRL_VGA.pdf

3.4 TFT-LCD

La pantalla por la que se sustituye el tubo de rayos catódicos del osciloscopio LeCroy de la serie 93XX, es una pantalla de tipo Thin Film Transistor-Liquid Crystal Display o (TFT-LCD) de 800x600 píxeles.

Una pantalla LCD consta de los elementos mostrados en la Figura 3.10:

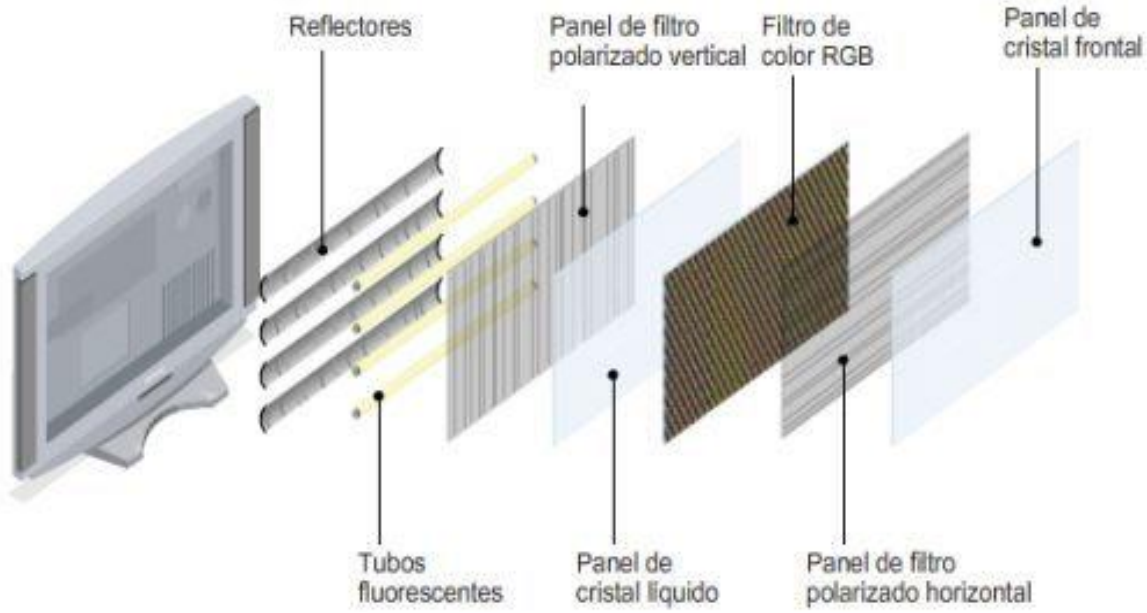


Figura 3.10: Componente de una pantalla TFT-LCD

Fuente: https://www.xataka.com/alta-definicion/como-funciona-un-televisor-lcd

- Reflectores y tubos fluorescentes: Las pantallas LCD requieren una fuente de luz externa para funcionar, ya que no producen propia luz. En versiones más modernas esto se sustituye por light-emitting diodes (LEDs).
- Pantalla de filtro de polarizado vertical: Polariza verticalmente la luz que entra al panel de cristal líquido.
- Panel de cristal líquido: Está compuesto por moléculas helicoidales de cristal líquido, al aplicar un campo eléctrico, podemos orientar estas moléculas, que a su vez son las responsables de polarizar la luz que pasa a través de ellas.
- Filtro de color RGB: Es la responsable de dar color.
- Panel de filtro polarizado horizontal. Filtra toda la luz que no haya sido polarizada horizontalmente.

3.5 Características de la pantalla de los osciloscopios LeCroy 93XX

Los osciloscopios LeCroy de la serie 93XX incorporan como pantalla un tubo de rayos catódicos. En este apartado, se van a explicar las características básicas del sistema de imagen de esta serie de osciloscopios y de las señales que controlan la representación de las imágenes en la pantalla.

Características:

- Pantalla monocromática naranja de nueve pulgadas.
- Pantalla con tratamiento anti reflejante.
- Resolución de 810x696 píxeles a una frecuencia de 50Hz o 60Hz.
- Frecuencia de píxel de 48MHz.
- Barrido vertical.
- Deflector electromagnético.
- Tiempo de subida y bajada del control de intensidad mayor de 12ns.
- Entrada de intensidad analógica.
- Entrada de sincronización con niveles TTL.
- Tamaño nominal horizontal: 165 mm para un X-ON = 15.39 ms.
- Tamaño de ajuste horizontal: > +/- 5mm.
- Ajuste de offset horizontal: +/- 5mm.
- Tamaño nominal vertical: 120 mm para un Y-ON = 14.5 µs.
- Tamaño de ajuste vertical: > +/- 5mm.
- Ajuste de offset vertical: +/- 5mm.
- No linealidad diferencial de X y Y: 10 %.
- Cada línea de píxeles se barre verticalmente, de abajo arriba.
- Las líneas se barren horizontalmente, de izquierda a derecha.
- Las señales de sincronismo dependen de la señal de reloj.

3.5.1 Sincronismo vertical

La función del sincronismo vertical, que presenta la serie de osciloscopios LeCroy de la serie 93XX, es indicar cuando barrer una línea de píxeles. Este sincronismo presenta las siguientes características:

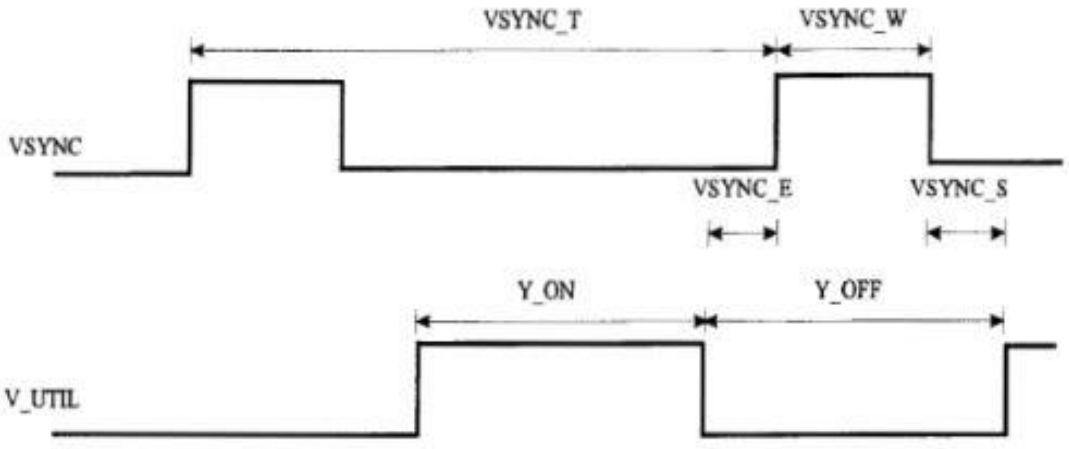


Figura 3.11: Sincronización vertical en el osciloscopio

Fuente: LeCroy 9314AX Service Manual

Los tiempos correspondientes a los intervalos mostrados en la Figura 3.11 para el sincronismo vertical, aparecen en la Tabla 3.5.

Vertical	# de Píxeles	Tiempo en µs
VSYNC_T	912	19.000
VSYNC_W	136	2.833
VSYNC_E	0	0.000
VSYNC_S	80	1.666
Y-ON	696	14.500
Y-OFF	216	4.500

Tabla 3.5: Temporización vertical del osciloscopio

3.5.2 *Sincronismo horizontal*

La función del sincronismo horizontal, que presenta la serie de osciloscopios LeCroy de la serie 93XX, es indicar el comienzo del barrido de una pantalla. Este sincronismo presenta las siguientes características:

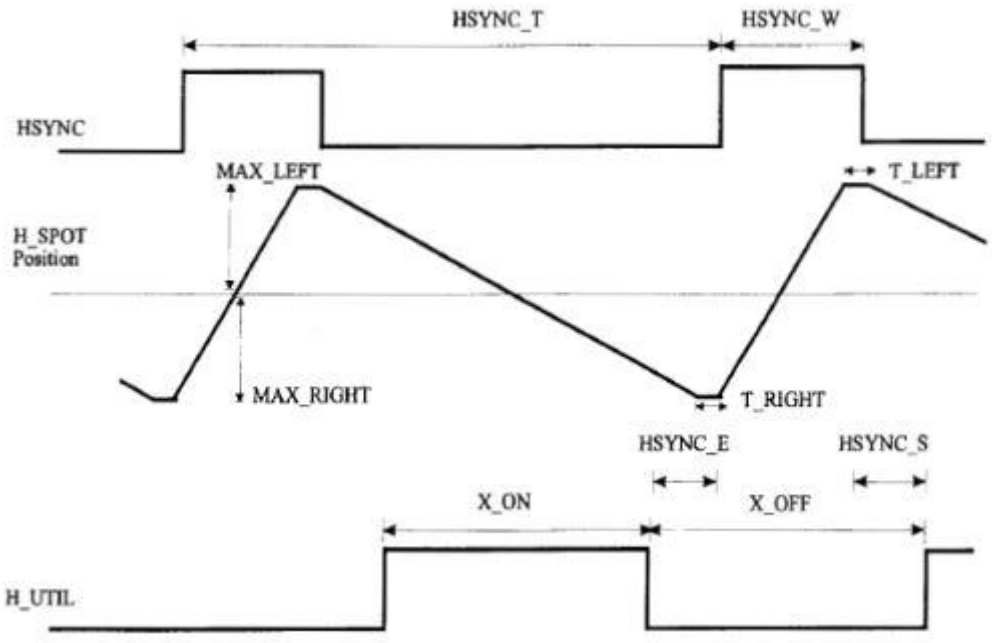


Figura 3.12: Sincronismo horizontal en el osciloscopio

Fuente: LeCroy 9314AX Service Manual

Los tiempos correspondientes a los intervalos mostrados en la Figura 3.12 para el sincronismo horizontal, aparecen en la Tabla 3.6.

Horizontal	# de líneas verticales	Tiempo en ms
HSYNC_T	842	15.998
HSYNC_W	22	0.418
HSYNC_E	4	0.076
HSYNC_S	6	0.114
X-ON	810	15.390
X-OFF	32	0.608

Tabla 3.6: Temporización horizontal del osciloscopio

3.6 FPGA

Field Programmable Gate Array o FPGA es circuito integrado programable que permite, utilizando un lenguaje de descripción de hardware especializado, simular circuitos lógicos, siendo estos, además, reprogramables.

Una FPGA se compone de un array de bloques lógicos programables, que pueden contener todo tipo de puertas lógicas (AND, OR, NOT…) y una jerarquía de interconexiones reconfigurables, que permiten conectar los bloques lógicos de formas distintas, pudiendo crear así cualquier dispositivo digital. Estos bloques pueden incluir también memorias, con lo que se aumenta la complejidad que se puede lograr. Todo esto se puede ver en la Figura 3.13.

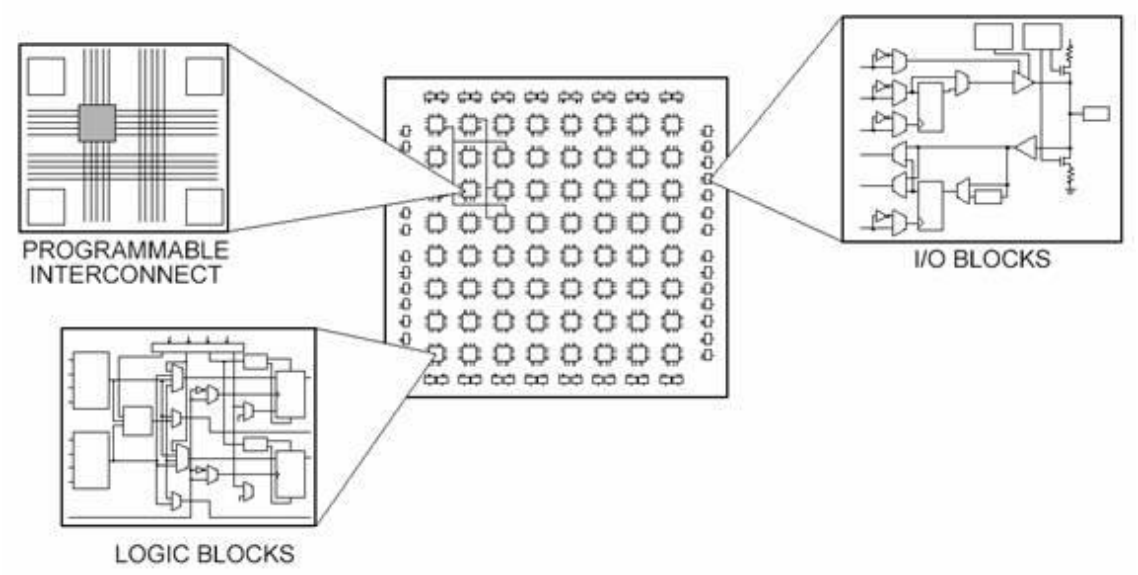


Figura 3.13: Esquema de una FPGA

Fuente:
http://sine.ni.com/np/app/main/p/ap/imc/lang/es/pg/1/sn/n17:imc,n19:FPGA_target/fmid/2170/

Para realizar la programación, se utiliza un Hardware Description Language o HDL, que es un lenguaje informático usado para describir la estructura y el comportamiento de un circuito lógico.

3.7 VHDL

VHSIC Hardware Description Language o VHDL es un lenguaje de descripción de hardware. Su código describe el comportamiento o estructura de un circuito electrónico, a partir del cual se puede obtener un circuito físico gracias a un compilador. VHDL permite tanto la simulación como la síntesis de circuitos. Fue el primer lenguaje de descripción de hardware estandarizado por el IEEE.

3.7.1 Estructura del código

Vamos a explicar brevemente las partes más importantes de un código VHDL.

```
----------Serializador----------------------------------
-- Define un registro de desplazamiento de 7 bits ------------
--------------------------------------------------------------
LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY serializador IS
   PORT (clk280   : IN STD_LOGIC;
         din      : IN std_logic_vector(6 downto 0);
         dout     : OUT STD_LOGIC);
END serializador;

ARCHITECTURE serializador OF serializador IS
   SIGNAL internal: STD_LOGIC_VECTOR(6 DOWNTO 0);
BEGIN
   PROCESS(clk280)
      VARIABLE count: INTEGER RANGE 0 TO 7 := 0;
   BEGIN
      IF rising_edge(clk280) THEN
         count := count + 1;
         IF (count = 6) THEN
            internal <= din;           -- Lee siguiente palabra
         ELSIF (count = 7) THEN
            count := 0;                -- Inicializa para enviar MSB primero
         END IF;
         dout <= internal(6-count);    -- MSB primero
      END IF;
   END PROCESS;
END serializador;
```

Figura 3.14: Ejemplo de código en VHDL

Un código VHDL contiene las siguientes partes, como puede verse en la Figura 3.14:

- LIBRARY/PACKAGE: Contiene una lista de todas las librerías y paquetes necesarios para el diseño, siendo los más comunes ieee, std, y work.
- ENTITY: Especifica principalmente los puertos de entrada y salida. Opcionalmente se pueden declarar constantes.
- ARCHITECTURE: Contiene el código VHDL que describe como es el circuito y a partir de este, se obtiene el hardware que lo realiza.

3.8 LVDS

Low-Voltage Differential Signaling o LVDS es un estándar que especifica las características técnicas de un protocolo de comunicaciones serie diferencial. Al ser solo una especificación física, muchos estándares de comunicación y aplicaciones lo pueden usar.

Fue creado en 1994 y se ha hecho muy popular en aparatos como los televisores LCD, ya que su aplicación típica es la transmisión de datos a gran velocidad, como en el caso del video de alta calidad.

LVDS es un sistema de señalización diferencial, lo que significa que transmite la información a través de dos líneas complementarias por las que se transmite dos señales de polaridad opuesta. Esto presenta la ventaja de hacer al sistema resistente al ruido en modo común, ya que se mide la diferencia de tensión entre las señales diferenciales que viajan entre ambas líneas. El ruido acoplado es visto como modulaciones de modo común por los receptores y es eliminado.

Los receptores responden solo al voltaje diferencial, por lo que, para recuperar la información enviada, se calcula la diferencia entre ambas señales y se le asigna un valor lógico correspondiente.

Otra ventaja de LVDS es que esta tecnología no requiere de un nivel específico de alimentación, es decir, puede trabajar con fuentes de alimentación de distintas tensiones, como 5V, 3.3V, 2V… Esta característica es especialmente deseable cuando no se conoce la tensión con la que se va a trabajar.

El valor de tensión típico en LVDS es de 1.25V, con una amplitud de ±0.2V, por lo que los niveles lógicos se encuentran entre 1.05V y 1.45V. Al ser esta diferencia de tensión muy pequeña, se reduce también el ruido electromagnético generado. También hace que LVDS consuma una cantidad de corriente constante.

Teóricamente, LVDS podría llegar a trabajar en el rango de los Gbps, aunque las velocidades típicas oscilan entre 500Mbps y 1000 Mbps, siendo su principal limitación la dependencia del medio de transmisión.

LVDS trabaja tanto con transmisión serie como paralela. Si es paralela, cada par lleva varias señales en una, incluyendo la de reloj para sincronizar los datos. Si es serie, las señales se serializan a suficiente velocidad como para que no se pierda información.

3.8.1 Recomendaciones para LVDS

Entre las recomendaciones que hace la especificación de LVDS para su correcto funcionamiento se encuentran:

1. Que el medio de transmisión termine en una resistencia del valor de la impedancia característica del medio, de forma que se prevengan reflexiones en la línea e interferencias electromagnéticas. El valor típico está entre 100Ω y 120Ω por cada par.
2. Se recomienda colocar una resistencia de 100Ω entre los drivers que generan los canales diferenciales LVDS y los receptores de las señales.
3. Hacer uso de componentes de montaje superficial para evitar efectos parásitos.
4. Hacer uso de pistas con la misma impedancia diferencial (mismo grosor y longitud) en el medio de transmisión.
5. Colocar las pistas para los canales diferenciales lo más juntas posibles con una separación máxima de 10mm. Esto ayuda a eliminar reflexiones y asegura que el ruido se acople en modo común. Esto se debe a que, cuanto más juntas se encuentren las líneas de los canales diferenciales, mayor es la cancelación del campo magnético radiado.
6. Hacer las pistas de los canales diferenciales de la misma longitud para evitar que las diferencias de fase estropeen la cancelación de los campos magnéticos.
7. Evitar giros de 90º, ya que causan discontinuidades de la impedancia. Se recomiendan arcos de 45º.

A modo ilustrativo, en la Figura 3.15, se muestra como un fabricante recomienda el diseño de las pistas LVDS para su integrado.

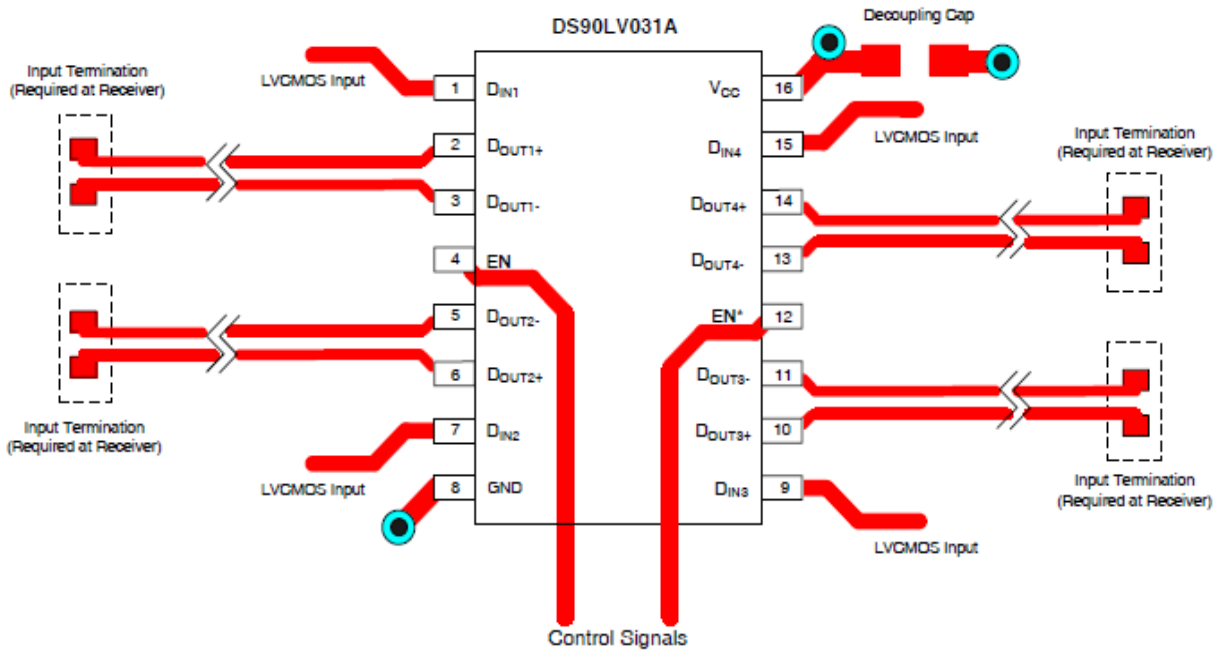


Figura 3.15: Recomendaciones para pistas LVDS

4 NORMAS Y REFERENCIAS

4.1 Bibliografía

1. **Pedroni, Volnei A.** *Circuit Desing and Simulation with VHDL.* London : MIT Press, 2010.

2. **Analog Devices.** AD8002. *Datasheet.* [En línea] 2015. http://www.analog.com/media/en/technical-documentation/data-sheets/AD8002.pdf.

3. **Texas Instruments.** Ds90c031. *Datasheet.* [En línea] 2011. http://www.ti.com/lit/ds/snls095a/snls095a.pdf.

4. **Terasic.** *DE0-CV User Manual.* [En línea] 2015. https://www2.pcs.usp.br/~labdig/material/DE0_CV_User_Manual.pdf.

5. **Altera Corporation.** *Introduction to the Quartus II Software.* [En línea] 2010. https://www.altera.com/content/dam/altera-www/global/en_US/pdfs/literature/manual/intro_to_quartus2.pdf.

6. **LeCroy.** *LeCroy Service Manual 9314A/C AM/CM/AL/CL.* [En línea] 1994. https://www.google.es/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&ved=0ahUKEwjE-oyQzYHWAhUDJVAKHTHkAT0QFggqMAA&url=http%3A%2F%2Fwww.f4ctz.fr%2F%3Fpage_id%3D321%26aid%3D358%26sa%3D1&usg=AFQjCNEpWkbUPrVdp__Vnp8kw_5Z5ykdSg.

7. **SiliconBlue.** *Application Note AN014: iCE65TM mobileFPGATM as an LVDS, FPD-Link Display Driver.* [En línea] 2010. http://www.prevailing-technology.com/publications/iCE65_LVDS_Display.pdf.

8. **AU Optronics.** B101EW05. *Datasheet.* [En línea] 2011. http://boundarydevices.com/wp-content/uploads/2014/12/B101EW05-V1-Ver0.1-20110125_201111101610.pdf.

9. **Chimei INNOLUX.** AT080TN52 V.3. *Datasheet.* [En línea] 2010. http://www.datasheetspdf.com/PDF/AT080TN52-V3/788928/1.

10. **Texas Instruments.** LM7905. *Datasheet.* [En línea] 2017. http://www.ti.com/lit/ds/symlink/lm79.pdf.

11. **Analog Devices.** AD9057. *Datasheet.* [En línea] 2017. http://www.analog.com/media/en/technical-documentation/data-sheets/AD9057.pdf.

12. **Monolithic Power Systems.** MP1584. *Datasheet.* [En línea] 2011. http://pdf1.alldatasheet.es/datasheet-pdf/view/551592/MPS/MP1584.html.

13. **TRACO POWER.** TSR-1 Series, 1 Amp. *Datasheet.* [En línea] 2017. https://www.tracopower.com/products/tsr1.pdf.

4.2 Programas de cálculo

Durante la realización de este proyecto, se han utilizado los siguientes programas informáticos:

Para el diseño del software con el que se controla la FPGA, haciendo uso de un lenguaje de descripción de hardware, se ha utilizado el programa Quartus II 13.0sp1 de Altera.

Para el diseño de las placas de circuito impreso, se ha utilizado el programa Eagle 7.6.0 de Autodesk.

Para la simulación de circuitos y componentes, se ha utilizado el programa OrCAD 9.2.

Para la redacción del proyecto, se ha utilizado Microsoft Word Professional Plus 2016.

4.3 Plan de gestión de calidad aplicado durante la redacción

La redacción y estructura del proyecto sigue la Norma UNE 157001:14, donde se describen los criterios generales para la elaboración formal de los documentos que constituyen un proyecto técnico.

La estructura en forma de capítulos y apartados, se enumeran de acuerdo con la Norma UNE 50132:94.

Todos los planos utilizan un mismo criterio de acotación y rotulación. Así como los tamaños y dimensiones de los formatos de papel a utilizar lo serán de la Serie A indicados en las normas UNE-EN ISO 5457:2000 y en UNE-EN ISO 5457:2000/A1:2010.

Para la redacción de referencias bibliográficas y citas, se ha empleado la norma UNE-ISO 690:2013.

5 DEFINICIONES Y ABREVIATURAS

5.1 Definiciones

AC: Alternating Current o Corriente Alterna, es aquel tipo de corriente eléctrica en la que la magnitud y el sentido varían con el tiempo.

AS: Active Serial, permite configurar el sistema de desarrollo basado en FPGA, de forma que al apagar este, el programa en ejecución no se pierde y, además, se carga cuando se enciende el sistema de desarrollo.

DAC: Digital to Analog Converter o Conversor Digital/Analógico, es un conversor capaz de transformar señales digitales en analógicas.

DC: Direct Current o Corriente Continua, es aquel tipo de corriente eléctrica en la que la magnitud y el sentido de la corriente no varía con el tiempo.

DDC: Display Data Channel, es un protocolo para la comunicación entre tarjetas de video y pantallas. Permite ajustar los parámetros de la pantalla.

FPGA: Field Programmable Gate Array, es un dispositivo reprogramable que contiene bloques lógicos que, mediante un lenguaje de descripción de hardware, permite emular cualquier dispositivo digital.

GPIO: General Purpose Input/Output, es un tipo de pin genérico en un chip, en el que cada pin no tiene un uso predeterminado.

HDL: Hardware Description Language, es un lenguaje de programación que permite definir circuitos electrónicos, así como su análisis y su simulación.

HPD: Hot Plug Detect, permite conocer cuando una pantalla es conectada a un equipo si esta no se encontraba conectada durante el arranque del mismo.

HSYNC: Horizontal Sync o Sincronismo Horizontal, es un pulso de sincronismo que se utiliza para indicar el principio y el fin del barrido de una línea de píxeles en una pantalla.

JTAG: Joint Test Action Group, es el nombre utilizado para referirse a la norma del IEEE 1149.1. Se utiliza para testear placas de circuito impreso. Permite cargar el programa directamente en el sistema de desarrollo.

LCD: Liquid Crystal Display, es un tipo de pantalla plana donde los píxeles se colocan delante de una fuente de luz.

LED: Light-Emitting Display, es un tipo de diodo el cual, al entrar en conducción, emiten luz.

PLL: Phase-Locked Loop, es un dispositivo que, a partir de una señal de reloj de una frecuencia dada, permite generar otra señal de reloj de una frecuencia diferente.

RAM: Random-Access Memory, es un tipo de memoria que permite leer o escribir en cualquier posición de memoria con el mismo tiempo de espera.

RGB: Permite, mezclando los tres colores primarios, generar cualquier otro color. Cuantos más bits por color, más combinaciones de colores posibles.

SDRAM: Synchronous Dynamic Random-Access Memory. Pertenece a la familia de las memorias aleatorias de acceso dinámico, teniendo la característica de que está sincronizada con el bus del sistema al que pertenece.

SMD: Surface Mount Technology, describe a aquellos componentes de los que se montan en las placas de circuito impreso, que van instalados sobre la placa sin llegar a atravesarla.

SOP: Small Outline Package, es un estándar para los encapsulados de montaje superficial, que define, entre otras cosas, la distancia entre pines.

SPI: Serial Peripheral Interface, es un estándar de comunicaciones para la transmisión de datos entre circuitos integrados.

TMDS: Transition Minimized Differential Signaling, es una tecnología para la transmisión de datos en serie a alta velocidad.

TFT: Thin Film Transistor, es un tipo especial de transistor de efecto de campo.

TTL: Transistor-Transistor Logic, es un estándar el diseño de lógica digital.

VESA: Video Electronics Standards Association, es una asociación internacional de fabricantes de electrónica

VGA: Video Graphics Array, es un estándar para la transmisión y generación de vídeo que emula el comportamiento de las señales utilizadas en los tubos de rayos catódicos.

VHSIC: Very High Speed Integrated Circuit, hace referencia a un tipo de circuito de lógica digital de alta velocidad.

VSYNC: Vertical Sync o Sincronismo Vertical, es un pulso de sincronismo que se utiliza para indicar el principio y el fin de un barrido vertical de píxeles en una pantalla.

5.2 Abreviaturas

AC	→	Alternating Current o Corriente Alterna
AS	→	Active Serial
DAC	→	Digital to Analog Converter o Conversor Digitar/Analógico
DC	→	Direct Current
DDC	→	Display Data Channel
FPGA	→	Field Programmable Gate Array
GPIO	→	General Purpose Input/Output
HDL	→	Hardware Description Language
HPD	→	Hot Plug Detect
HSYNC	→	Horizontal Sync o Sincronismo Horizontal
JTAG	→	Joint Test Action Group
LCD	→	Liquid Cristal Display
LED	→	Light-Emitting Diodes
PLL	→	Phase-Locked Loop
RAM	→	Random-Access Memory
RGB	→	Red, Green, Blue
SDRAM	→	Synchronous Dynamic Random-Access Memory
SMD	→	Surface Mount Technology
SOP	→	Small Outline Package
SPI	→	Serial Peripheral Interface
TMDS	→	Transition Minimized Differential Signaling
TFT	→	Thin Film Transistor
TTL	→	Transistor-Transistor Logic
VESA	→	Video Electronics Standards Association
VGA	→	Video Graphics Array
VHDL	→	VHSIC Hardware Description Language
VHSIC	→	Very High Speed Integrated Circuit
VSYNC	→	Vertical Sync o Sincronismo Vertical

6 REQUISITOS DE DISEÑO

6.1 LeCroy 93XX

Los osciloscopios LeCroy de la serie 93XX, presentan una serie de características relevantes para el desarrollo de este proyecto y que se van a explicar:

- Pantalla con resolución de 810x696 píxeles.
- Reloj del sistema de imagen de 48MHz.
- El barrido que recorre una línea de píxeles se realiza de abajo a arriba.
- El barrio que recorre las líneas se realiza de izquierda a derecha.

6.1.1 MDS410

Como se puede observar en la Figura 6.1, las señales de sincronismo HSYNC y VSYNC se generan en el integrado MDS410.

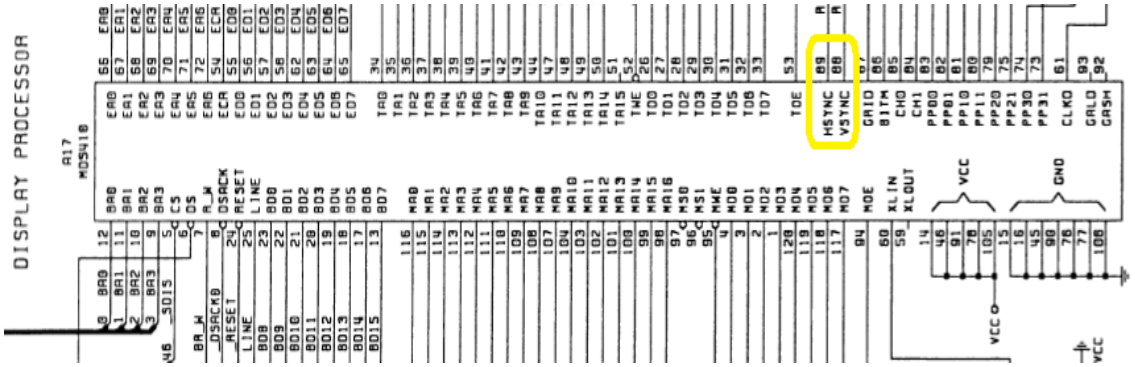


Figura 6.1: Señales de sincronismo en MDS410

Fuente: LeCroy 9314AX Service Manual

En el mismo integrado (ver la Figura 6.2), se puede obtener la señal de reloj CLK0.

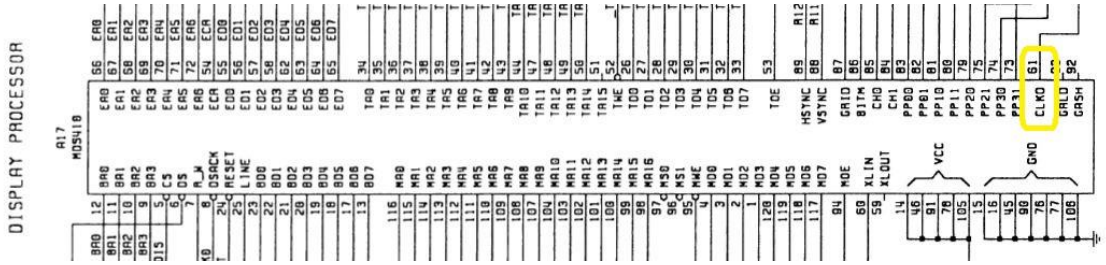


Figura 6.2: Señal de reloj en MDS410

Fuente: LeCroy 9314AX Service Manual

6.1.2 Video DAC SMBT475

Como se ve en la Figura 6.3, las señales de color RGB, se generan en el integrado VIDEO DAC, donde la señal para el color azul está a masa y las señales del rojo y el verde se unen para formar la intensidad (INTENS) que establece el nivel de iluminación del píxel.

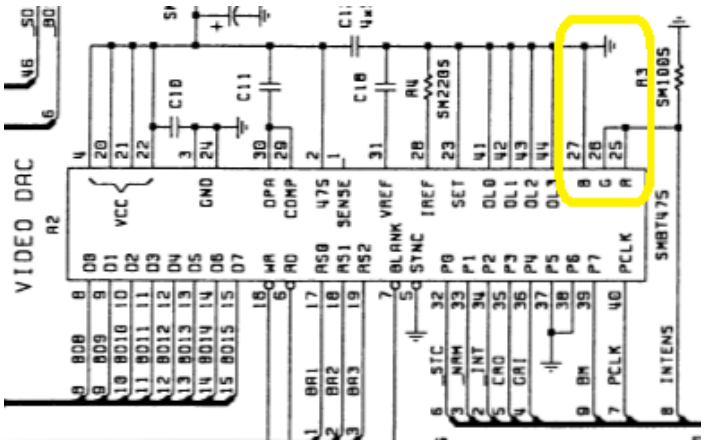


Figura 6.3: Señales RGB en DAC SMBT475

Fuente: LeCroy 9314AX Service Manual

6.1.3 Señales de sincronismo

Las señales de sincronismo, presentan las características que se expusieron en los apartados 3.5.1 y 3.5.2. Como información complementaria obtenida de su estudio, es importante señalar que, como se muestra en la Figura 6.4, el flanco negativo de la señal de sincronismo horizontal (arriba), coincide con el flanco positivo de uno de los pulsos de la señal de sincronismo vertical (abajo).

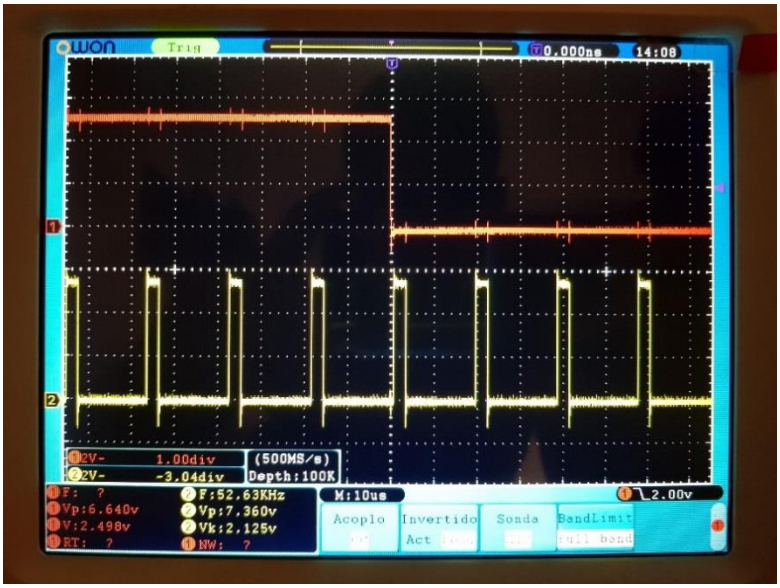


Figura 6.4: Señales de sincronización horizontal y vertical

6.2 Pantalla TFT-LCD de 800x600 píxeles

La pantalla TFT-LCD de 800x600 elegida es la Chimei INNOLUX de 8.0 pulgadas, 24 bits y 40 pines, TFT-LCD AT080TN52 V.5 SVGA RGB de 800x600, que presenta las siguientes características:

- Pantalla con resolución de 800x600 píxeles.
- Puerto para cable plano de 40 pines TTL.
- Conector Backlight.

6.2.1 Placa controladora de video PCB800099

La pantalla TFT-LCD, mencionada en el apartado 6.2, necesita de una placa controladora de video. Esta presenta las siguientes características relevantes para el proyecto:

- Puerto para cable plano de 40 pines TTL.
- Puerto Backlight.
- Puerto VGA.

6.3 Pantalla TFT-LCD de 1280x800 píxeles

La pantalla TFT-LCD de 1280x800 elegida es la AUO B101EW05 de 10.1 pulgadas, TFT-LCD de WXGA de 1280x800 píxeles (16:10) con Backlight, que presenta las siguientes características:

- Pantalla con resolución de 1280x800 píxeles.
- Conector LVDS.
- Conector Backlight.

6.4 DE0-CV

DE0-CV es el sistema de desarrollo elegido para este proyecto (ver la Figura 6.5 y la Figura 6.6), construido alrededor de una FPGA Cyclone V de Altera. De todas las características que presenta este sistema de desarrollo, se expondrán aquellas que son relevantes para el proyecto:

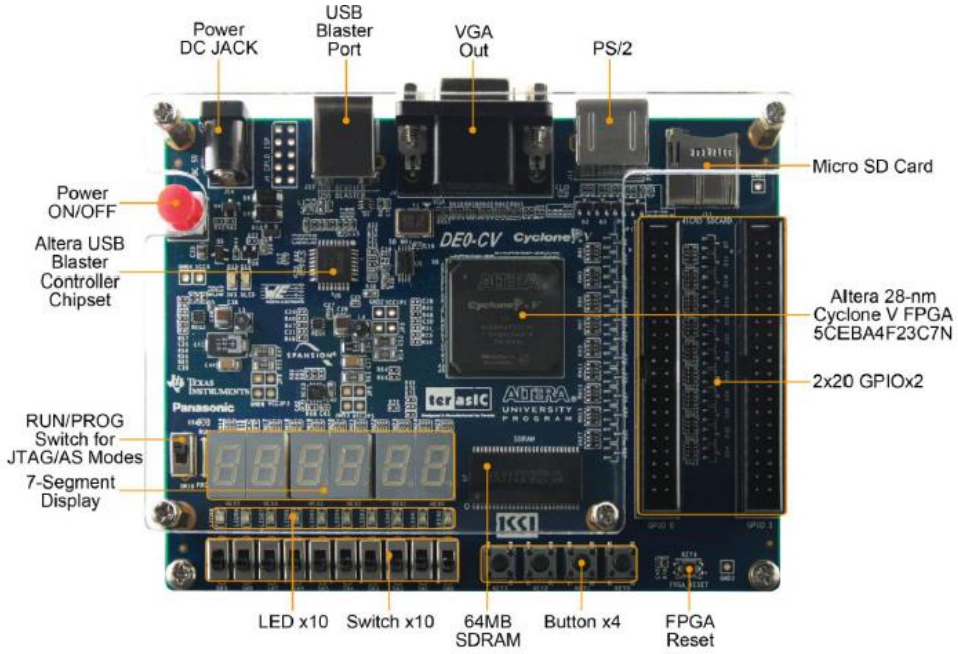


Figura 6.5: Cara superior del sistema de desarrollo

Fuente: DE0-DV User Manual

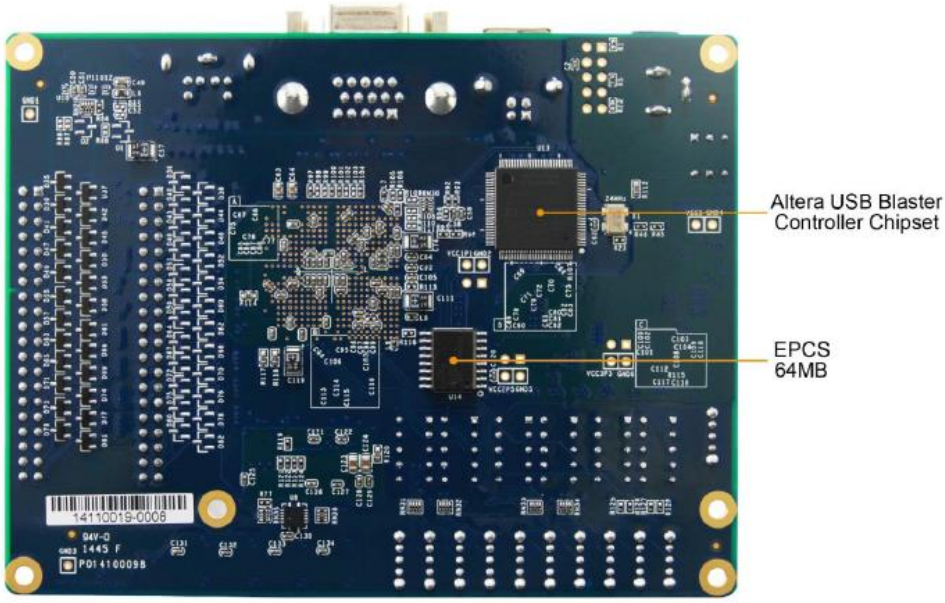


Figura 6.6: Cara inferior del sistema de desarrollo

Fuente: DE0-DV User Manual

6.4.1 Hardware

El hardware que incorpora el sistema de desarrollo de la DE0-CV, se muestra en la Figura 6.7:

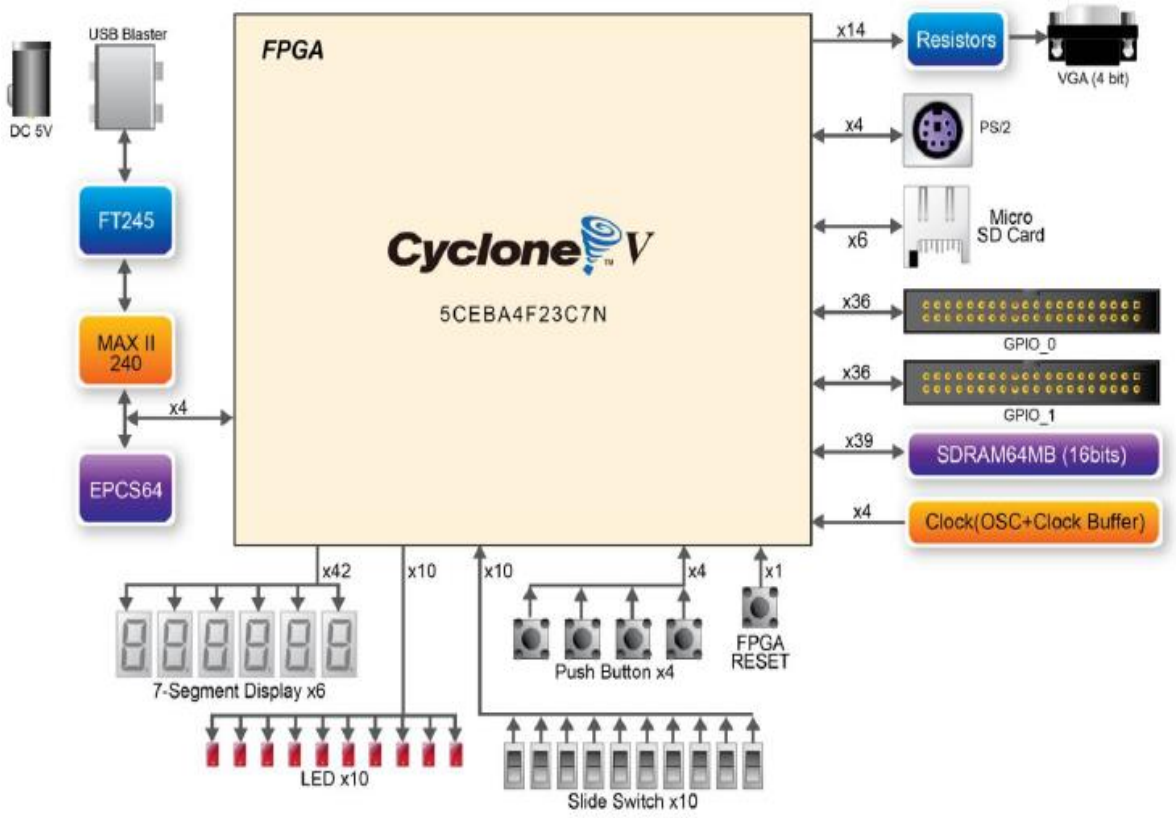


Figura 6.7: Esquema de componentes del sistema de desarrollo

Fuente: DE0-DV User Manual

6.4.1.1 Dispositivos FPGA

- Cyclone V 5CEBA4F23C7N.
- 49K elementos lógicos programables.
- 3080 Kbit de memoria interna.
- Cuatro Phase-Locked Loops (PLLs) fraccionales.

6.4.1.2 Configuración y Debug

- Dispositivo de configuración serie de tipo EPCS64.
- Puerto USB tipo B.
- Soporta configuración en modo Joint Test Action Group (JTAG) y Active Serial (AS).

6.4.1.3 *Dispositivo de memoria*

- Synchronous Dynamic Random-Access Memory (SDRAM) de 64MB con un bus de datos de x16 bits.

6.4.1.4 *Conectores*

El sistema de desarrollo dispone de dos conectores GPIO 2x20 pines.

Cada conector dispone de 36 pines de datos conectados directamente a la FPGA Cyclone V. También dispone de un pin de para alimentación de +5V, otro para alimentación de +3,3V y sus respectivos pines de masa.

En la Figura 6.8, se muestran la numeración de los pines, su posición y el nombre que recibe cada pin.

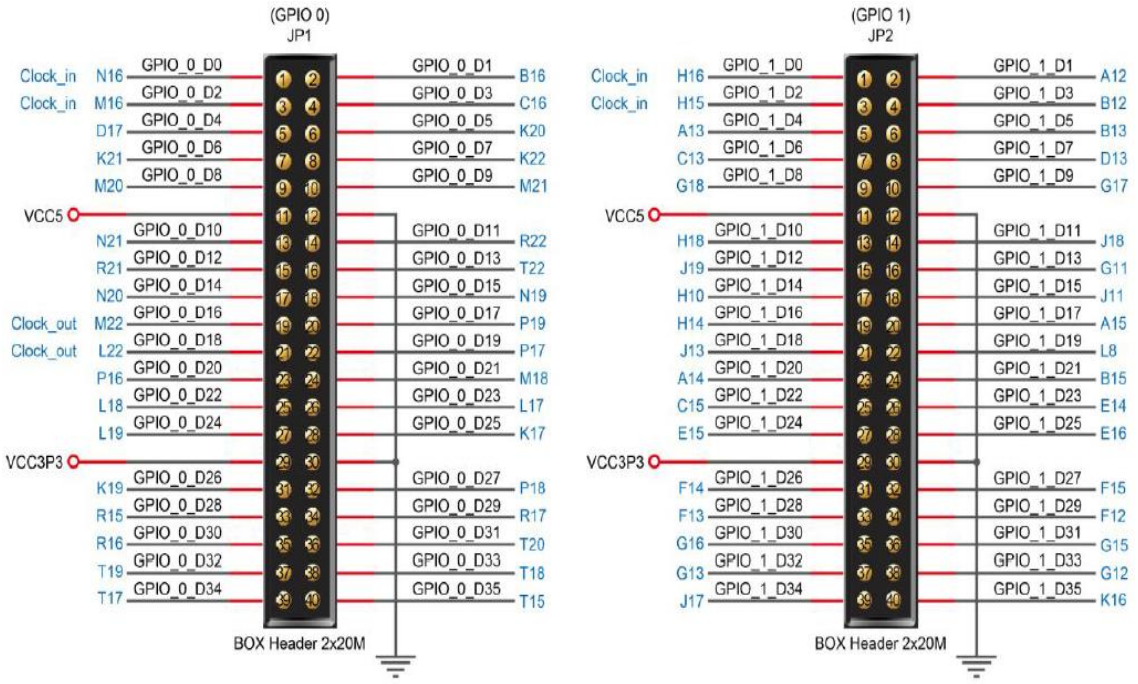


Figura 6.8: Conectores GPIO

Fuente: DE0-DV User Manual

En la Tabla 6.1 se muestra, para cada pin, que señal le corresponde, su número y la conexión que realiza.

Señal	Nº de pin FPGA	Conexión	Señal	Nº de pin FPGA	Conexión
GPIO_0_D0	PIN_N16	0[0]	GPIO_1_D0	PIN_H16	1[0]
GPIO_0_D1	PIN_B16	0[1]	GPIO_1_D1	PIN_A12	1[1]
GPIO_0_D2	PIN_M16	0[2]	GPIO_1_D2	PIN_H15	1[2]
GPIO_0_D3	PIN_C16	0[3]	GPIO_1_D3	PIN_B12	1[3]
GPIO_0_D4	PIN_D17	0[4]	GPIO_1_D4	PIN_A13	1[4]
GPIO_0_D5	PIN_K20	0[5]	GPIO_1_D5	PIN_B13	1[5]
GPIO_0_D6	PIN_K21	0[6]	GPIO_1_D6	PIN_C13	1[6]
GPIO_0_D7	PIN_K22	0[7]	GPIO_1_D7	PIN_D13	1[7]
GPIO_0_D8	PIN_M20	0[8]	GPIO_1_D8	PIN_G18	1[8]
GPIO_0_D9	PIN_M21	0[9]	GPIO_1_D9	PIN_G17	1[9]
GPIO_0_D10	PIN_N21	0[10]	GPIO_1_D10	PIN_H18	1[10]
GPIO_0_D11	PIN_R22	0[11]	GPIO_1_D11	PIN_J18	1[11]
GPIO_0_D12	PIN_R21	0[12]	GPIO_1_D12	PIN_J19	1[12]
GPIO_0_D13	PIN_T22	0[13]	GPIO_1_D13	PIN_G11	1[13]
GPIO_0_D14	PIN_N20	0[14]	GPIO_1_D14	PIN_H10	1[14]
GPIO_0_D15	PIN_N19	0[15]	GPIO_1_D15	PIN_J11	1[15]
GPIO_0_D16	PIN_M22	0[16]	GPIO_1_D16	PIN_H14	1[16]
GPIO_0_D17	PIN_P19	0[17]	GPIO_1_D17	PIN_A15	1[17]
GPIO_0_D18	PIN_L22	0[18]	GPIO_1_D18	PIN_J13	1[18]
GPIO_0_D19	PIN_P17	0[19]	GPIO_1_D19	PIN_L8	1[19]
GPIO_0_D20	PIN_P16	0[20]	GPIO_1_D20	PIN_A14	1[20]
GPIO_0_D21	PIN_M18	0[21]	GPIO_1_D21	PIN_B15	1[21]
GPIO_0_D22	PIN_L18	0[22]	GPIO_1_D22	PIN_C15	1[22]
GPIO_0_D23	PIN_L17	0[23]	GPIO_1_D23	PIN_E14	1[23]
GPIO_0_D24	PIN_L19	0[24]	GPIO_1_D24	PIN_E15	1[24]
GPIO_0_D25	PIN_K17	0[25]	GPIO_1_D25	PIN_E16	1[25]
GPIO_0_D26	PIN_K19	0[26]	GPIO_1_D26	PIN_F14	1[26]
GPIO_0_D27	PIN_P18	0[27]	GPIO_1_D27	PIN_F15	1[27]
GPIO_0_D28	PIN_R15	0[28]	GPIO_1_D28	PIN_F13	1[28]
GPIO_0_D29	PIN_R17	0[29]	GPIO_1_D29	PIN_F12	1[29]
GPIO_0_D30	PIN_R16	0[30]	GPIO_1_D30	PIN_G16	1[30]

GPIO_0_D31	PIN_T20	0[31]	GPIO_1_D31	PIN_G15	1[31]
GPIO_0_D32	PIN_19	0[32]	GPIO_1_D32	PIN_G13	1[32]
GPIO_0_D33	PIN_18	0[33]	GPIO_1_D33	PIN_G12	1[33]
GPIO_0_D34	PIN_17	0[34]	GPIO_1_D34	PIN_J17	1[34]
GPIO_0_D35	PIN_T15	0[35]	GPIO_1_D35	PIN_K16	1[35]

Tabla 6.1: Señales del puerto GPIO

Fuente: DE0-DV User Manual

A todos los pines de datos se les aplica la protección que se muestra en la Figura 6.9, que consiste en conectar estos pines a dos diodos zener y una resistencia para protegerlos de niveles de tensión mayores que los que soporta la FPGA.

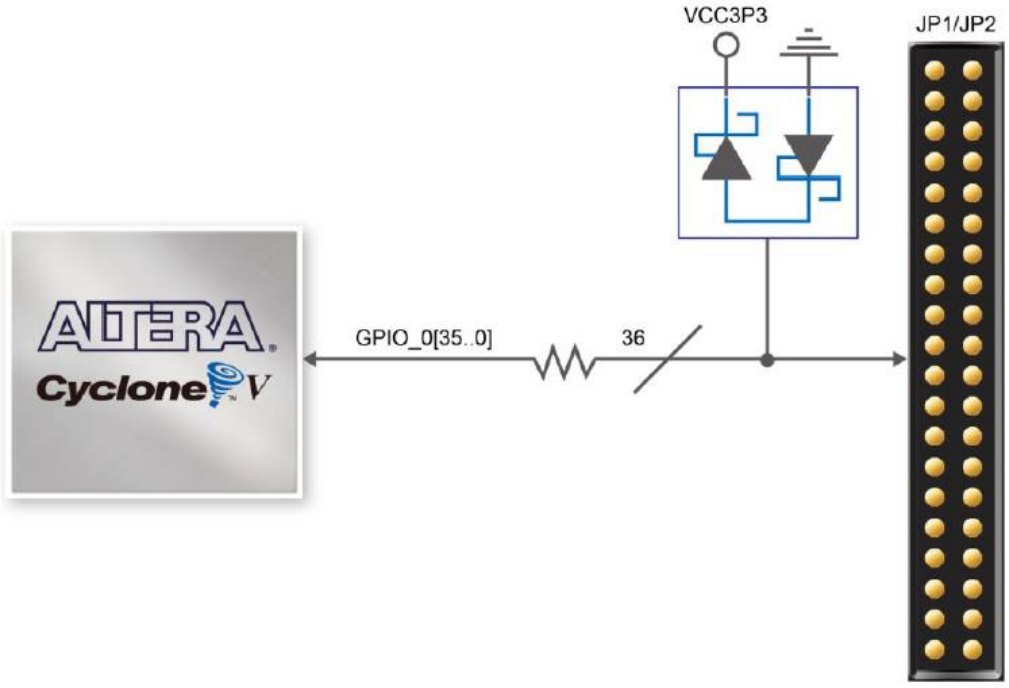


Figura 6.9: Protección para la Cyclone V

Fuente: DE0-DV User Manual

6.4.1.5 *Puerto VGA*

El sistema de desarrollo DE0-CV incluye un puerto D-sub de 15 pines como salida VGA. Las señales de sincronismo horizontal y vertical las proporciona directamente la Cyclone V, mientras que las señales analógicas de datos (rojo, verde y azul), se proporcionan por un Digital to Analog Converter o Conversor Digital/Analógico (DAC) de red de resistencias de 4 bits a partir de unas señales digitales que proporciona la Cyclone V.

La Figura 6.10, muestra el esquema de conexionado entre la Cyclone V y el conector D-sub de 15 pines, que es el que permite soportar el estándar VGA (640x480 píxeles a 25MHz).

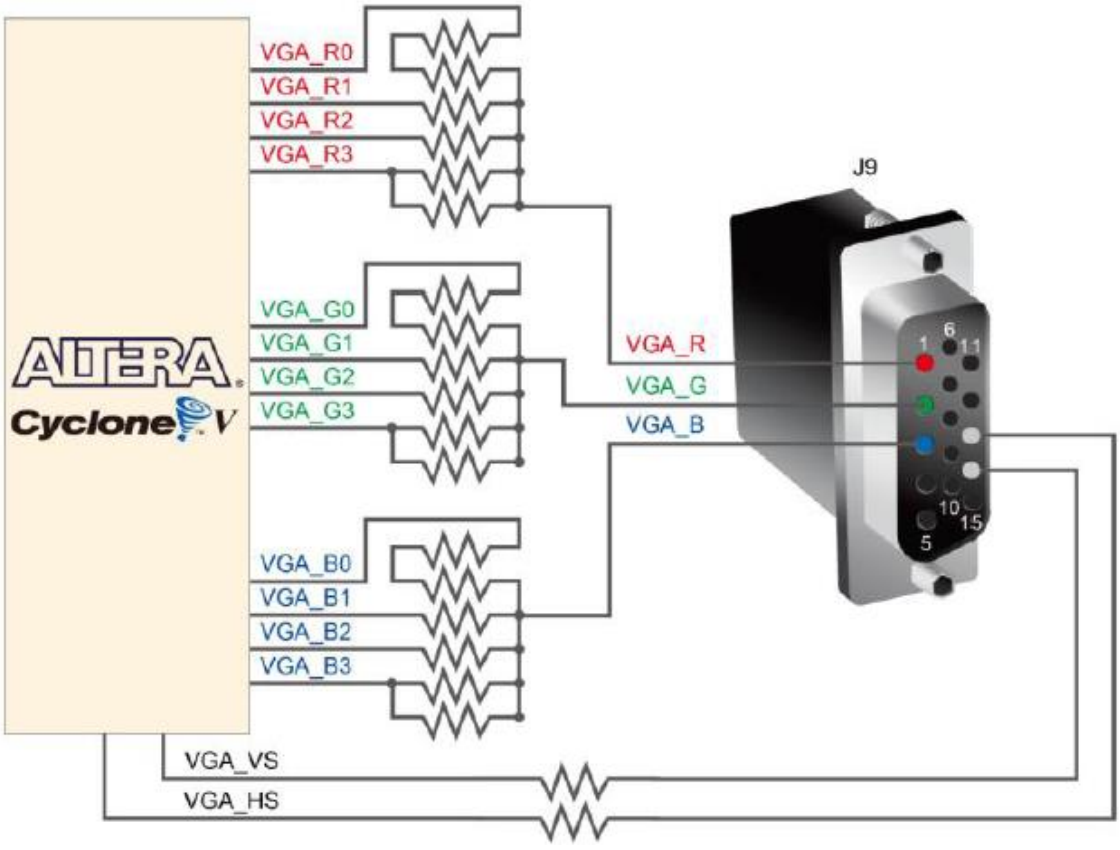


Figura 6.10: DAC VGA

Fuente: DE0-DV User Manual

En la Tabla 6.2, se muestra para cada una de las señales VGA, el pin de la FPGA al que se encuentran conectados y la conexión que realizan:

Señal	Nº de pin FPGA	Conexión
VGA_R0	PIN_A9	Red[0]
VGA_R1	PIN_B10	Red[1]
VGA_R2	PIN_C9	Red[2]
VGA_R3	PIN_A5	Red[3]
VGA_G0	PIN_L7	Green[0]
VGA_G1	PIN_K7	Green [1]
VGA_G2	PIN_J7	Green [2]
VGA_G3	PIN_J8	Green [3]
VGA_B0	PIN_B6	Blue[0]
VGA_B1	PIN_B7	Blue [1]
VGA_B2	PIN_A8	Blue [2]
VGA_B3	PIN_A7	Blue[3]
VGA_HS	PIN_H8	HSYNC
VGA_VS	PIN_G8	VSYNC

Tabla 6.2: Conexiones del DAC VGA

Fuente: DE0-DV User Manual

6.4.2 Software y conexionado

Para trabajar con la FPGA y su sistema de desarrollo, se requiere cargar el programa que se desee que ejecute la FPGA, para ello, hay que conectar el sistema de desarrollo al equipo en el que se está diseñando el software mediante un puerto USB de tipo B.

6.5 Requisitos impuestos por el proyecto

- Sustituir la pantalla de tubo de rayos catódicos por una pantalla TFT-LCD, consiguiendo una imagen adecuada.
- Como la pantalla TFT-LCD es a color, se pretende idear algún mecanismo para sacar provecho de esta característica.
- Comprobar que el osciloscopio genera señales a color.

7 ANÁLISIS DE SOLUCIONES

El objetivo principal de este proyecto es sustituir la pantalla de tubo de rayos catódicos del osciloscopio LeCroy de la serie 93XX por una pantalla TFT-LCD.

Lo primero que se ha hecho al comenzar este proyecto, es buscar soluciones que resuelvan el objeto del proyecto. Al no encontrar ninguna, se ideó una serie etapas para resolver todos los puntos del proyecto.

7.1 Llevar las señales del osciloscopio a la pantalla TFT-LCD

Una vez que ha comprobado que no hay una solución específica al problema que propone el proyecto, el primer paso es buscar una forma de obtener las señales del osciloscopio LeCroy de la serie 93XX y representarlas en la pantalla TFT-LCD. Esto es importante, porque determina qué elementos serán necesarios de partida. Para ello se proponen varias alternativas:

7.1.1 Opción 1: Conexión directa

Esta opción consiste en intentar llevar las señales directamente del osciloscopio LeCroy de la serie 93XX a la pantalla TFT-LCD. Para ello, habría que hacer uso directo de las señales RGB que se encuentran en el integrado VIDEO DAC SMBT475 de la placa de control del osciloscopio. Esto requeriría desoldarlas y llevar las señales de video RGB a un conector VGA, tras conectarlas a una resistencia de 100Ω. El principal inconveniente de esta opción, es que existe la posibilidad de inutilizar el osciloscopio al desoldarlas.

Otra solución aún más sencilla, pero que sigue permitiendo hacer una conexión directa es, ya que el sistema tiene cortocircuitada la señal B y unidas las señales R y G Figura 7.1, podría cogerse la señal de intensidad tal cual, llevarla a los conectores de R y G y dejar a masa B. El inconveniente es que con esto se conseguiría que lo que apareciese por pantalla fuese siempre del mismo color (tal y como ocurre con la pantalla de tubo de rayos catódicos del osciloscopio).

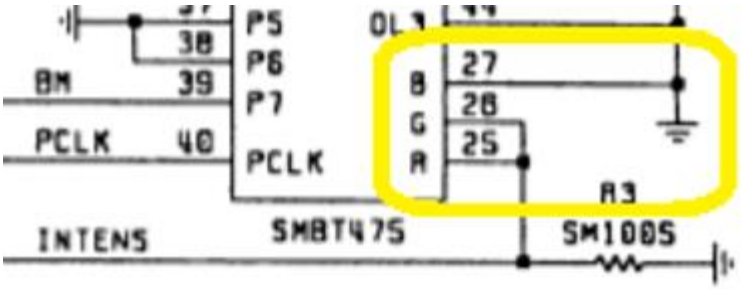


Figura 7.1: Generación de las señales RGB en le osciloscopio

Fuente: LeCroy 9314AX Service Manual

Respecto a las señales de sincronismo, en ambos casos, habría que invertir las señales HSYNC y VSYNC, puesto que el video activo se produce cuando las señales de sincronismo están a nivel bajo y en VGA es cuando las señales de sincronismo están a nivel alto.

Otro inconveniente añadido es que la forma en la que trabajan los sincronismos es distinta. En el caso del osciloscopio LeCroy de la serie 93XX, el barrido se realiza de abajo arriba y de izquierda a derecha, mientras que, en VGA, es de izquierda a derecha y arriba abajo. Esto puede verse en la Figura 7.2.

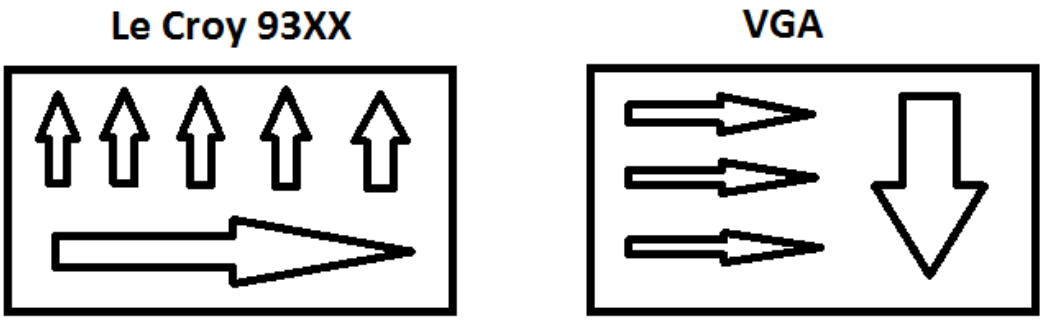


Figura 7.2: Comparativa entre el sincronismo del osciloscopio y de VGA

Como puede verse en la Figura 7.2, es posible hacer coincidir el sincronismo del osciloscopio LeCroy de la serie 93XX con el sincronismo de VGA, pero se obtendría la pantalla rotada 90º hacia la derecha, de forma que la resolución que se obtendría al representarla en la pantalla TFT-LCD sería de 696x810 píxeles. Por lo que se obtendría una pantalla como la que se muestra en la Figura 7.3:

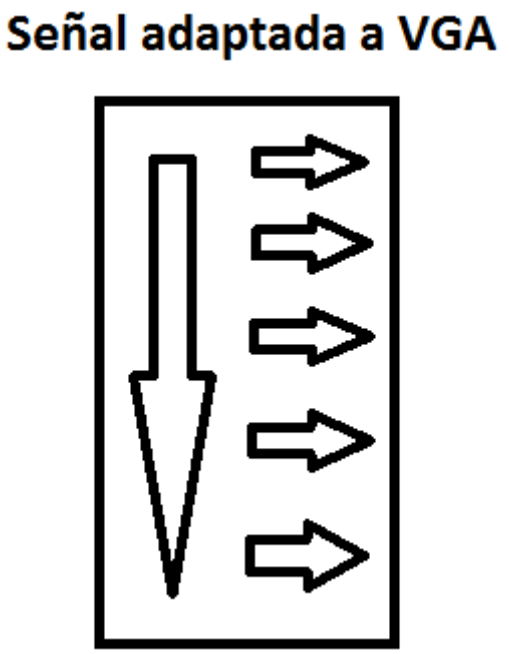


Figura 7.3: Pantalla del osciloscopio con sincronismo VGA

Como puede verse en la Figura 7.3, que la pantalla aparezca rotada no es muy elegante, ya que requeriría de una pantalla de resolución muy grande, por ejemplo, de 1280x960 píxeles, para que cupiesen los 696x810, desperdiciando gran parte de la pantalla además de aparecer esta rotada 90º.

7.1.2 Opción 2: Almacenar la pantalla en memoria RAM

La segunda opción, consiste en conectar la señal de intensidad a una entrada digital, de forma que se asignaría el valor '0' cuando el valor de la intensidad estuviese por debajo de algún valor prefijado, para no mostrar nada en pantalla y '1' cuando estuviese por encima (conversor A/D de 1 bit), de forma que apareciese por pantalla lo que muestra el osciloscopio. Por tanto, en RAM se almacenaría un '0' o un '1' del píxel en cuestión.

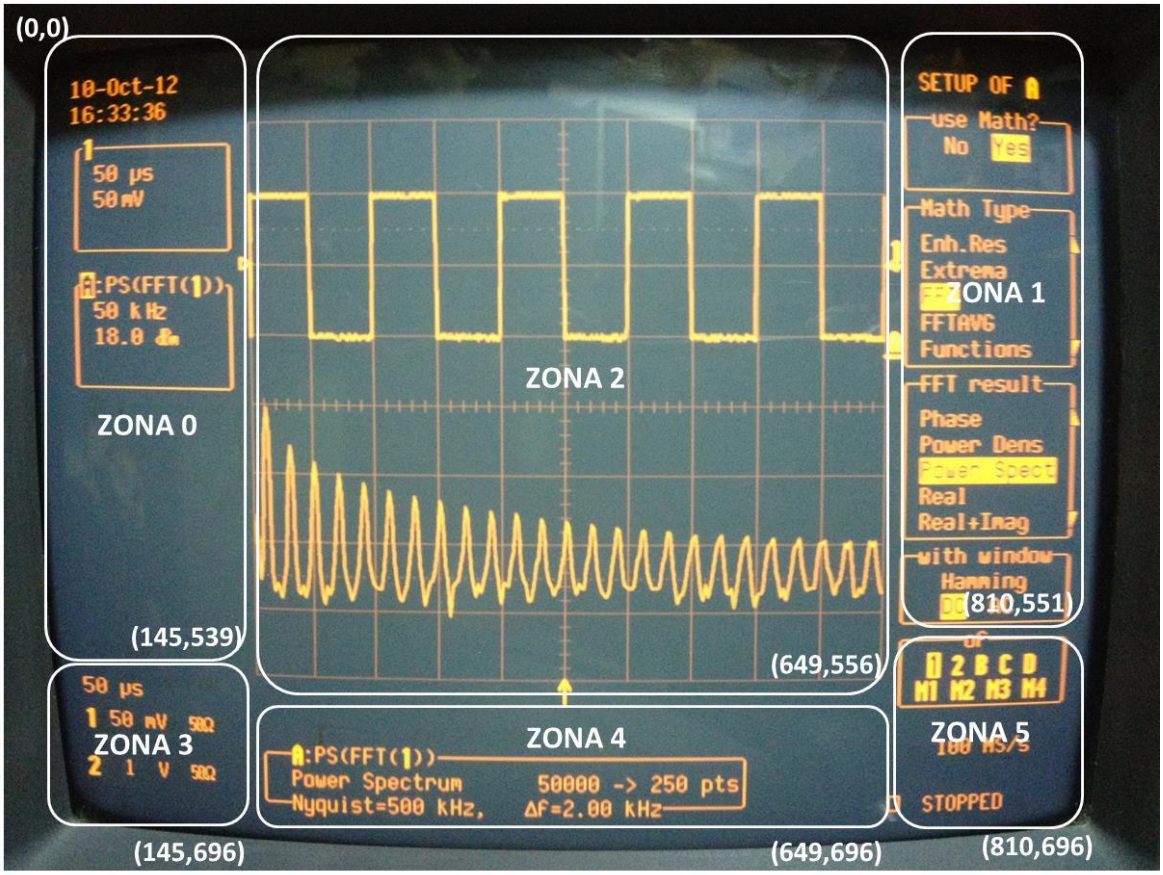


Figura 7.4: Pantalla del osciloscopio dividida por zonas

Para dar color a la pantalla, esta podría dividirse en zonas, como se ve en la Figura 7.4, asignando un color a cada zona. El problema es que cuando se accede a algunos menús de estos osciloscopios como: Adquisition, System, Test & Times, Waveform o Memory Used, estas pantallas tienen una distribución diferente a la mostrada en la Figura 7.4. Esto podría resolverse obteniendo la señal de los pulsadores de la botonera para procesarla y cambiar la distribución de los colores. El inconveniente de todo esto es que aumenta la complejidad del software y del hardware a diseñar.

En la Figura 7.5, se puede ver una pantalla del osciloscopio (System) con una distribución diferente.

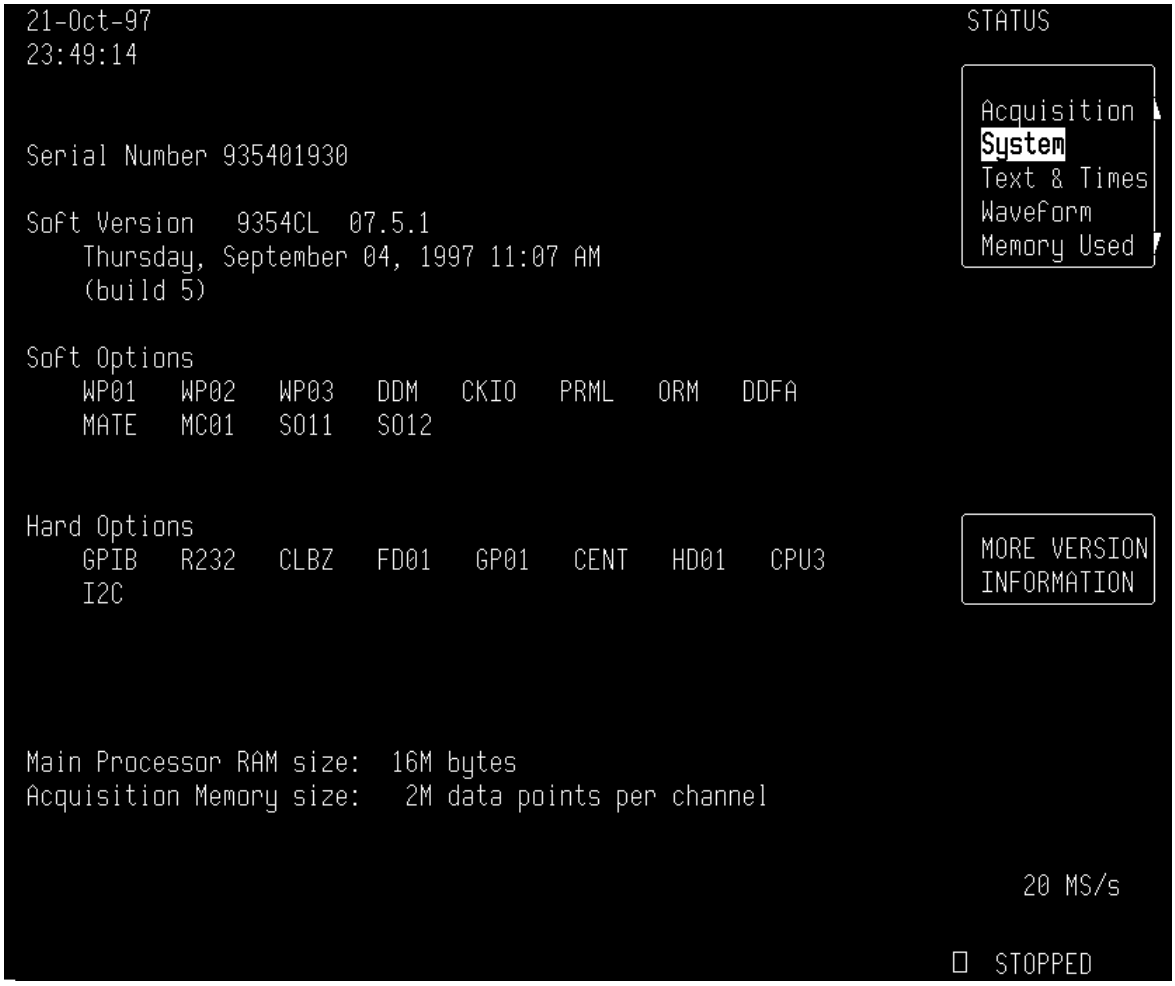


Figura 7.5: Pantalla del osciloscopio sin señales

7.1.3 Opción 3: Almacenar las pantallas en memoria y añadir un A/D

Esta opción es muy similar a la del apartado 7.1.2, solo que, en vez de prefijar un color para cada zona, utilizamos un conversor analógico-digital para convertir el valor de la intensidad del primer píxel de cada zona en una combinación de valores RGB. Esto es factible, ya que hay dos mandos en la botonera del osciloscopio que, aunque independientes, modifican la señal de intensidad, así, hay uno que controla la intensidad del 'grid' (la rejilla sobre la que se pintan las señales) y otro que se encarga de la intensidad del resto de elementos que aparecen por pantalla, como las señales y todo el texto.

Así, al modificar con los controles la intensidad, estaríamos cambiando el color de las zonas en función de cual sea la asignación de colores que se haga en cada zona, en función del píxel convertido.

Por último, señalar que tanto en las opciones 2 y 3 (ver los apartados 7.1.2 y 7.1.3), hay que salvar el contenido de la información las señales procedentes del osciloscopio en memoria, pero solo es necesario almacenar un bit por pixel, ya que solo se pretende saber si hay que representarlo o no.

7.1.4 Representación de señales

Tras estudiar las señales del osciloscopio, y debido al tamaño de su pantalla, que es de 810x696 píxeles, se presenta el problema de cómo ajustar la información de una pantalla de mayor resolución en otra de menor resolución, puesto que la pantalla TFT-LCD especificada en el título del TFG era de una resolución 800x600 píxeles.

La solución ideada fue la de eliminar líneas de video vía software. En concreto, respecto de las líneas verticales, eliminar las 5 primeras y las 5 últimas líneas, al ser las primeras y las últimas líneas, no contienen información relevante, de esta manera se consigue fácilmente pasar de 810 a 800 líneas.

El problema se encuentra en las líneas horizontales, donde habría que eliminar 96 de las 696. Se puede optar por eliminar las 48 primeras y las 48 últimas líneas, pero seguro que de esta manera se elimina información relevante. Otra solución es eliminar 1 de cada 8 líneas, de esta manera se reparte la pérdida de información por toda la pantalla, consiguiéndose un total de 609 líneas, tras esto, y a modo de ejemplo, solo quedaría eliminar las 5 primeras líneas y las 4 últimas o viceversa. La solución ideal sería ver de dónde se pueden eliminar más líneas por presentar información menos relevante.

Finalmente, se optó por adquirir una pantalla con resolución suficiente para poder acomodar toda la pantalla del osciloscopio sin pérdida de información. Así que se adquirió una pantalla TFT-LCD de 1280x800 píxeles, que, al ser de mayor resolución, permite representar toda la información de la pantalla del osciloscopio corregida (girada 90º hacia la derecha), no tal cual es generada.

7.2 Justificación del sistema de desarrollo con FPGA elegido

El proyecto requiere almacenar en memoria RAM la información de la pantalla del osciloscopio LeCroy de la serie 93XX, lo que suponen 810x696 = 563760 bits, ya que solo se pretende saber si hay que representar un píxel o no. Es por ello, que se requiere una memoria, de como mínimo, ese tamaño. Además, la memoria debe ser capaz de trabajar con una frecuencia de reloj de como mínimo 48MHz, que es la que utiliza el osciloscopio (ver el apartado 3.5).

Respecto a la memoria de las FPGAs, no hay problemas ni de lectura ni escritura, pues permiten frecuencias de reloj que alcanzan los 200MHz o 300MHz, sin embargo, las memorias SDRAM que incorporan estos sistemas de desarrollo basados en FPGA no soportan estas velocidades de acceso. Además, las memorias que habitualmente incorporan, son de un solo puerto.

El sistema de desarrollo A-C8V4, que había sido adquirido para el desarrollo de este proyecto, no disponía de una memoria RAM externa a la FPGA lo suficientemente rápida, además, la cantidad de memoria que incorporaba su FPGA era de un tamaño insuficiente, por lo que fue descartado.

El proceso de trabajo con la memoria es sencillo, para cada píxel proveniente del osciloscopio, se escribe en memoria si está encendido o no, y, a la vez, esta información se lee de la memoria y se envía a la pantalla TFT-LCD.

Respecto de la lectura de memoria, hay dos alternativas en función del tipo de memoria:

Si la memoria es de puerto único, solamente se puede realizar una operación a la vez, es decir, o se escribe en memoria o se lee de esta, lo que provoca que en un periodo de reloj se lea y en otro se escriba. Esto no es un gran problema a la hora de escribir en memoria, pues solo se escribirían la mitad de las pantallas que se generan, el problema está a la hora de leer de memoria, pues en un periodo se envía la imagen y en el siguiente nada.

Tras realizar pruebas con esta opción, se ha visto que funciona, pero que produce un parpadeo molesto por la alternancia de enviar información y no.

Otra opción, es utilizar una memoria de doble puerto, lo que permite realizar las dos operaciones simultáneamente y, si es necesario, hasta con dos relojes diferentes. Esta es la opción elegida, ya que permite leer y escribir simultáneamente, lo que evita la pérdida de información y además no produce parpadeos molestos. Además, la opción de emplear relojes diferentes ha sido inevitable al usar transmisión LVDS. En definitiva, la RAM que se emplea para video es de doble puerto.

En cualquier caso, la memoria interna que incorpora la FPGA se diseña vía software, mediante algún lenguaje de descripción de hardware, es decir, se define el tamaño del bus de datos, el del bus de direcciones, el tipo de acceso (síncrono o no) y el número de puertos. Así, los factores limitantes para el sistema de desarrollo siguen siendo la cantidad de memoria y la velocidad a la que esta puede trabajar.

Teniendo en cuenta todo lo anterior, se optó por adquirir el sistema de desarrollo DE0-CV. Este cuenta con una memoria de 3080 Kbit de memoria interna. Más que suficiente para los requerimientos del proyecto y lo suficientemente rápido para soportar los 48MHz del osciloscopio.

7.2.1 Programación

Para programar el sistema de desarrollo, se propone hacer uso del compilador Quartus II de Altera, ya que el sistema de desarrollo con FPGA es del mismo fabricante. Para ello, también se propone realizar el código haciendo uso del lenguaje de descripción de hardware VHDL que emplea el compilador y es conocido, pues se estudia en la Escuela Politécnica Superior de Linares.

Para cargar el programa generado gracias a Quartus II, se propone hacer uso de la configuración JTAG a la hora de realizar las distintas pruebas con el software, ya que carga el programa directamente en la Cyclone V, y al apagar el sistema de desarrollo, se borra. La configuración JTAG se muestra en la Figura 7.6:

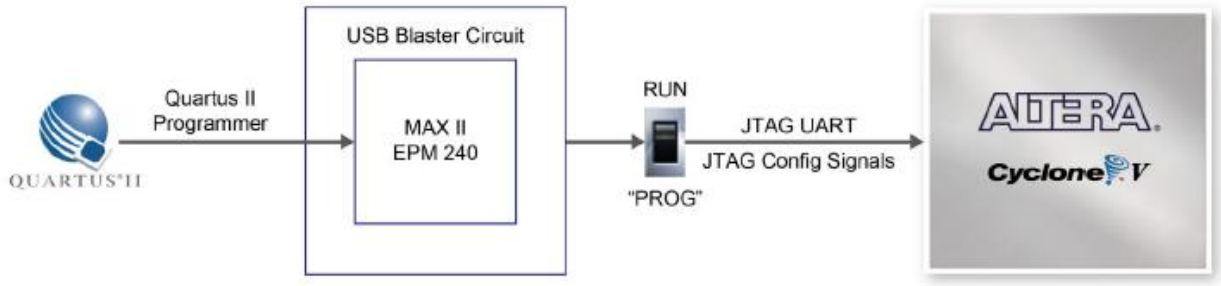


Figura 7.6: Conexión JTAG

Fuente: DE0-DV User ManualUna vez que se haya creado la versión definitiva del código, se propone hacer uso de la configuración AS, que descarga el programa en la memoria EPCS64. Esto permite que cuando se apague el sistema de desarrollo no se borre el programa, posteriormente, este se carga en la FPGA desde la memoria automáticamente al encender el sistema de desarrollo. Esta configuración se muestra en la Figura 7.7:

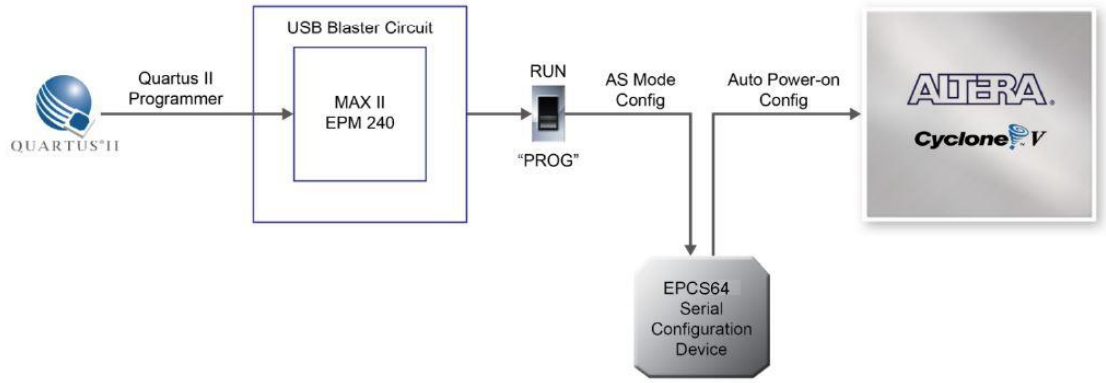


Figura 7.7: Conexión AS

Fuente: DE0-DV User Manual

7.2.2 Funcionalidades extra

Entre las funcionalidades extra que incorpora el sistema de desarrollo y que son interesantes para el proyecto, es que este incorpora un puerto VGA, que permite diseñar una placa controladora para el proyecto más sencilla, al no tener que incorporar el puerto VGA en esta. Esta característica que incorpora el sistema de desarrollo, hace que se opte por trabajar con VGA.

Otro añadido interesante son los dos puertos GPIO que incorpora y que permiten manejar gran cantidad de señales.

7.3 Diseño de una placa controladora

Puesto que no es posible representar de manera adecuada, el video procedente del osciloscopio haciendo uso de sus señales, tal y como salen de este, es necesario someterlas a algún tipo de transformación para adecuarlas, para ello se necesita una placa controladora que conecte el osciloscopio LeCroy de la serie 93XX con el sistema de desarrollo FPGA y con la pantalla TFT-LCD.

7.3.1 Diseño de una placa controladora mediante circuito impreso

Puesto que este proyecto lo componen una gran cantidad de elementos, como son: el osciloscopio, la pantalla TFT-LCD, la FPGA con su sistema de desarrollo y la controladora de video (en el caso de utilizar el puerto VGA del sistema de desarrollo), es interesante tener algún elemento que agrupe todo el desarrollo de hardware adicional que se requiere, como pueden los conectores, amplificadores o atenuadores de tensión para adaptar las señales, etc…

Con este fin, se propone para este proyecto la fabricación de una placa controladora de circuito impreso en la que se incluyan todos los elementos hardware que se diseñen con el fin de ahorrar espacio y simplificar el montaje.

Una vez diseñada la placa, para el trazado de las pistas se ha probado software de Multisim, Orcad e Eagle. Para este proyecto, se ha optado por la versión de Eagle 7.6.0, pues es, para la placa diseñada, la que ofrecía un enrutado automático más completo y, a la hora de componer los esquemas, muestra información más completa y detallada de los componentes utilizados.

7.3.2 Tecnologías para conectarse con la pantalla TFT-LCD

Lo primero que se debe hacer antes de empezar a diseñar la placa controladora propuesta para este proyecto que ayuda en la tarea de conectar y adaptar las señales procedentes del osciloscopio, es seleccionar aquella o aquellas tecnologías que soporte la pantalla TFT-LCD y que sean interesantes para el proyecto.

Se debe tener en cuenta que la pantalla TFT-LCD de la que disponemos solo es compatible con LVDS, por lo que siempre se va a necesitar trabajar con LVDS.

Hay que recordar que LVDS es un estándar para la transmisión de datos, así que, sobre él, se podrían utilizar otros estándares de video como VGA, DVI, HDMI, S-

video, etc., es decir, se podrían generar el video con alguno de estos protocolos y transformarlo para transmitirlo sobre LVDS.

Para las pantallas investigadas, se conoce que LVDS solo soporta VGA y no el resto de estándares de video.

Otra opción es, con un conector del protocolo elegido, llevar las señales generadas de ese protocolo a una placa controladora de video intermedia que se encargue de hacer las conversiones necesarias a LVDS.

Convertir las señales directamente a LVDS es, en cualquier caso, la opción más interesante, porque no requiere de la placa controladora de video antes mencionada.

Es por esto que, de las tecnologías elegidas, podría ser interesante tener un puerto de su estándar para poder enviar las señales a la placa controladora de video y a la vez, tener la posibilidad de poder enviarlas directamente a la pantalla TFT-LCD a través de LVDS cuando sea posible.

7.3.3 Conexiones requeridas

Teniendo en cuenta las tecnologías con la que se tiene pensado trabajar y los elementos que van a ser conectados a la placa controladora diseñada, se van a enumerar los puertos que van a ser necesarios para este proyecto:

7.3.3.1 Puerto para conectar el osciloscopio LeCroy de la serie 93XX con la placa controladora diseñada

Para conectar el osciloscopio con la placa controladora diseñada, se propone buscar algún conector desde el cual se puedan obtener las señales requeridas para el proyecto. En concreto, se propone hacer uso del conector que se encuentra en la placa de control del osciloscopio y al que va conectada la placa de video del osciloscopio. Este conector presenta las señales mostradas en la Figura 7.8.

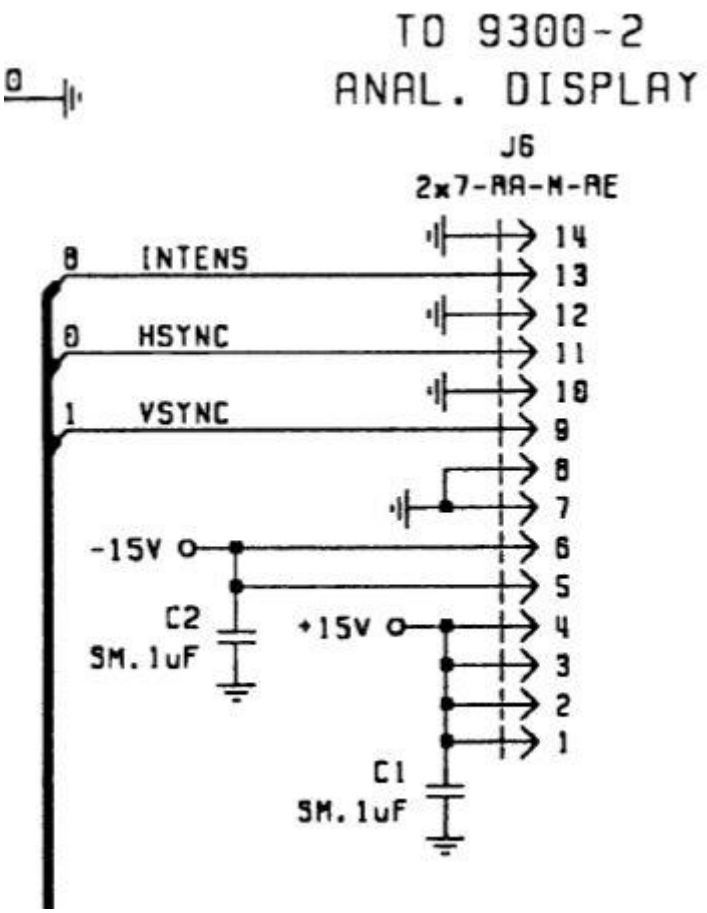


Figura 7.8: Puerto con todas las señales requeridas del osciloscopio

Fuente: LeCroy 9314AX Service Manual

Todas las señales requeridas del osciloscopio, como puede verse en la Figura 7.8, son: la de intensidad (pin 13) y las de sincronismo (pines 9 y 11). Estas pueden ser tomadas conjuntamente de uno de los puertos del osciloscopio (ver Figura 7.9). Además, este conector proporciona alimentación de ±15V, lo que servirá para alimentar componentes en la placa controladora diseñada.

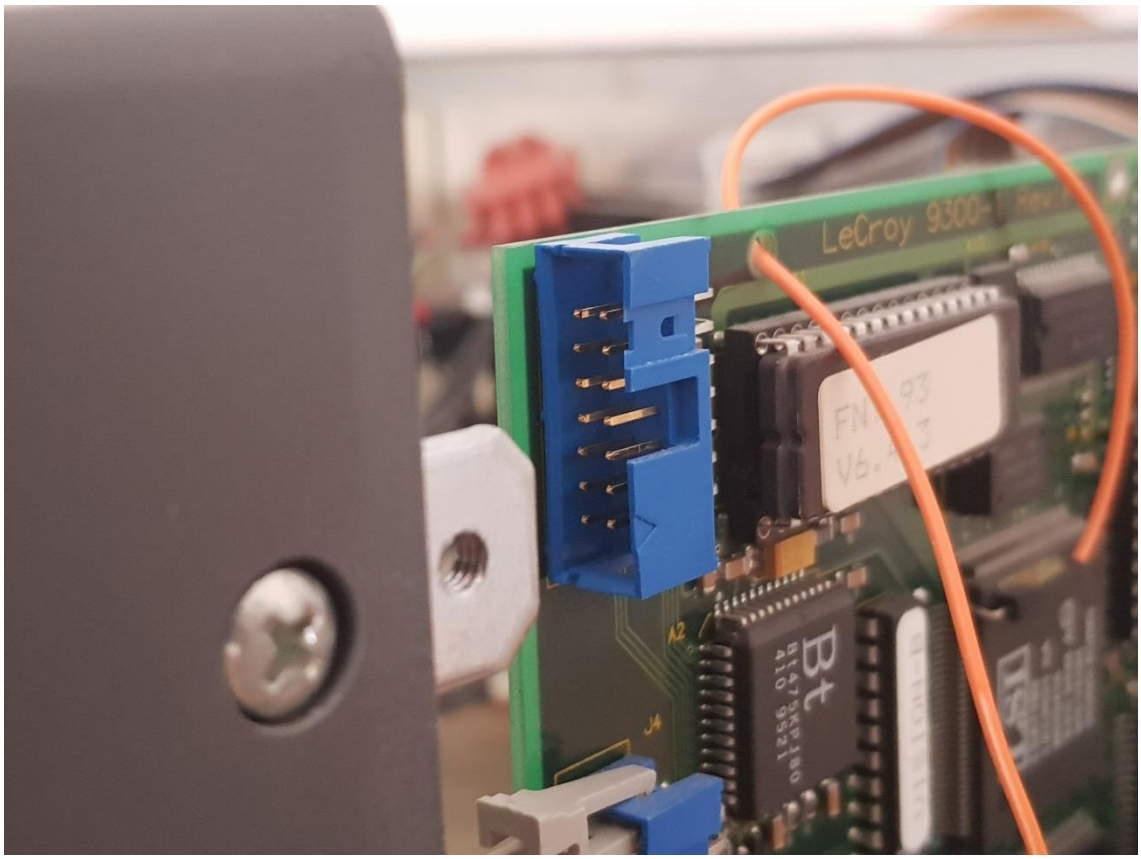


Figura 7.9: Puerto del que obtener las señales del osciloscopio

El puerto que requerirá la placa será idéntico al de la Figura 7.9, con 14 pines distribuidos en 2x7. La separación entre pines es la estándar de décima de pulgada. Destacar que el pin 1, de todos los pines mostrados en la Figura 7.9, corresponde con el pin que se encuentra en la parte inferior izquierda.

La última señal que se necesita del osciloscopio es la de reloj. Esta señal permite marcar cada píxel y junto a las señales de sincronismo, su posición en la pantalla.

Para obtener esta señal del osciloscopio debemos conectar un cable desde donde se encuentra la señal en la placa de control del osciloscopio a la placa controladora diseñada donde será utilizada. En concreto, esta señal se obtiene del integrado DISPLAY PROCESSOR:

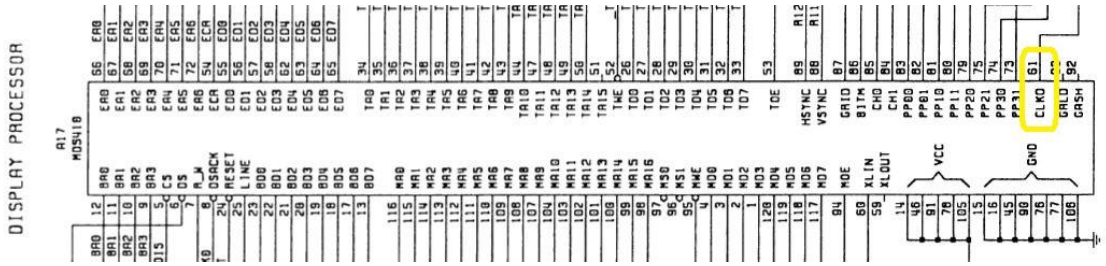


Figura 7.10: Señal de reloj en el osciloscopio

Fuente: LeCroy 9314AX Service Manual

Como se puede ver en la Figura 7.10, la señal de reloj (CLK0) corresponde al pin 61 del circuito integrado situado en la placa de control del osciloscopio denominado DISPLAY PROCESSOR. Para llevar esa señal a la placa controladora diseñada, se propone una conexión en el osciloscopio como se muestra en la Figura 7.11:

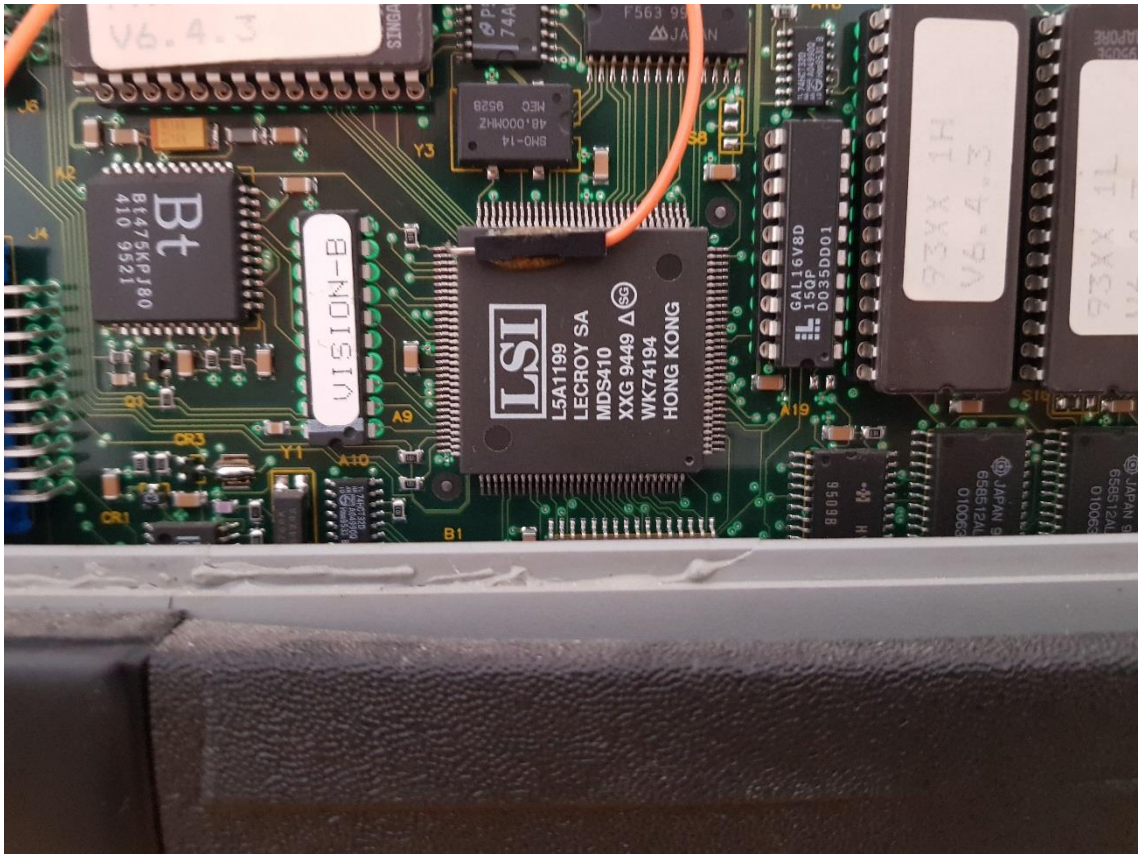


Figura 7.11: Conexión para obtener la señal de reloj del osciloscopio

Al conectar un cable directamente al pin del que sale la señal, como se muestra en la Figura 7.11, solo habría que llevar ese cable a algún conector en la placa controladora diseñada para poder trabajar con la señal de reloj de 48MHz que genera el osciloscopio.

7.3.3.2 Puerto para conectar el sistema de desarrollo basado en FPGA con la placa controladora diseñada

Para llevar las señales del osciloscopio al sistema de desarrollo basado en FPGA, así como llevar aquellas señales generadas en la FPGA hacia la pantalla TFT-LCD, se propone utilizar un puerto GPIO, igual que el que incorpora el sistema de desarrollo.

Las ventajas de este puerto son el gran número de pines que tiene para la transmisión de múltiples señales y la resistencia mecánica que presenta.

Figura 7.12: Puertos GPIO del sistema de desarrollo

Como se observa en la Figura 7.12, este puerto tiene una distribución de pines de 2x20. La separación entre pines es la estándar de décima de pulgada. Además, presenta la ventaja de que es de posición única (ver la hendidura en mitad del puerto).

La posición del pin 1 se corresponde, en la Figura 7.12, con el que se encuentra en la fila superior derecha.

7.3.3.3 Puerto para enviar las señales VGA que sean generadas

Para transmitir las señales VGA que se generen, se requiere de un puerto VGA, la ventaja es que, con el sistema de desarrollo elegido, no es necesario incluir uno en la placa, puesto que ya lo incluye el sistema de desarrollo, por lo que se puede

tener una placa controladora diseñada más pequeña y de menor complejidad (ver la Figura 6.5).

7.3.3.4 Puerto para enviar las señales LVDS que sean generadas hacia la pantalla TFT-LCD

Para poder transmitir las señales LVDS que se generen, se requiere de un puerto LVDS.

Las pantallas TFT-LCD de las que se dispone para el proyecto tienen un conector de 2x15 pines con una separación no estándar de 2mm. Las señales que corresponden a cada pin siguen la distribución estándar que se muestra en la Tabla 7.1:

Pin	Símbolo	Descripción
1	VSEL	Alimentación de la pantalla, 3.3V típicos
2	VSEL	Alimentación de la pantalla, 3.3V típicos
3	VSEL	Alimentación de la pantalla, 3.3V típicos
4	GND	Masa
5	GND	Masa
6	NC	Sin conexión
7	TXO0-	Señal impar LVDS 0-
8	TXO0+	Señal impar LVDS 0+
9	TXO1-	Señal impar LVDS 1-
10	TXO1+	Señal impar LVDS 1+
11	TXO2-	Señal impar LVDS 2-
12	TXO2+	Señal impar LVDS 2+
13	GND	Masa
14	GND	Masa
15	TXOC-	Señal impar LVDS de reloj-
16	TXOC+	Señal impar LVDS de reloj+
17	TXO3-	Señal impar LVDS 3-
18	TXO3+	Señal impar LVDS 3+
19	TXE0-	Señal par LVDS 0-
20	TXE0+	Señal par LVDS 0+
21	TXE1-	Señal par LVDS 1-
22	TXE1	Señal par LVDS 1+

23	TXE`2-	Señal par LVDS 2-
24	TXE2+	Señal par LVDS 2+
25	GND	Masa
26	GND	Masa
27	TXEC-	Señal par LVDS de reloj-
28	TXEC+	Señal par LVDS de reloj+
29	TXE3-	Señal par LVDS 3-
30	TXE3+	Señal par LVDS 3+

Tabla 7.1: Señales en un puerto LVDS

Como ya se mencionó en el apartado 3.8.1, LVDS también necesita que se coloque una resistencia de 100Ω al final de cada línea diferencial (donde se encuentra el puerto LVDS) para evitar reflexiones.

Adicionalmente al puerto LVDS, hay pantallas TFT-LCD que no se alimentan a través del conector LVDS, sino que requieren de otro conector a través del cual se alimentan, este conector se denomina Backlight.

Este puerto es estándar y presenta 1x6 pines con una separación de 2mm. Las señales que corresponden a cada pin se muestran en la Tabla 7.2:

Pin	Señal
1	+12V
2	+12V
3	+5V
4	Sin uso
5	GND
6	GND

Tabla 7.2: Señales en un puerto Backlight

7.3.4 Adaptación de señales

Las señales relevantes para el proyecto que genera el osciloscopio, tienen los siguientes niveles de tensión:

- La señal de intensidad, que es una señal analógica entre 0V y 1V.
- La señal de sincronismo horizontal y vertical, que es una señal digital entre 0V y 5V.

7.3.4.1 Adaptación para la FPGA

La FPGA, entiende las señales de tensión con niveles TTL entre 0V y 3.3V, por tanto, aquellas señales que necesite leer la FPGA, tendrán que ser adaptadas primero. Este es el caso de la señal de intensidad y de la señal de sincronismo vertical.

Para adaptar las señales de sincronismo horizontal y vertical, se propone un divisor de tensión para atenuar de 5V a 3.3V.

Para adaptar la señal de intensidad, se proponer hacer uso de un amplificador operacional con configuración estándar de ganancia positiva, para amplificar de 1V a 3.3V

7.3.4.2 Adaptación a VGA

Las señales VGA que se generan a través de la FPGA y su sistema de desarrollo, no requieren de ninguna adaptación y son enviadas directamente a la placa controladora de video.

7.3.4.3 Adaptación a LVDS

Como ya se explicó en el apartado 3.8, LVDS utiliza una transmisión diferencial de la información. Cada pareja de cables diferenciales forma un canal. Para el caso concreto de este proyecto, se requiere transmitir video VGA sobre LVDS, por lo que se debe hacer uso de aquella configuración que requiera LVDS para este fin.

En LVDS, el número de interfaces de salida que se utilizan depende del número de bits de las componentes RGB de entrada, cuando se usan 6 bits, se obtiene un interfaz de salida de 18 bits y se necesitan 3 + 1 canales, mientras que, si para cada componente se usan 8 bits, el interfaz de salida es de 24 bits y se necesitan 4 + 1 canales, siendo en ambos casos, el canal individual el de reloj. En este proyecto, se ha implementado el primer ejemplo, en el que cada una de las componentes RGB de VGA está formada por 6 bits.

Respecto a la señal de reloj, se parte de una VGA estándar a 40MHz, por lo que se necesita un reloj de 280MHz (7 x 40MHz). Para el caso de 6 bits por componente de entrada, se tienen 18 bits, más las señales de sincronismo (horizontal y vertical) y video activo, lo que es un total de 21 bits/píxel. Como de cada canal se envían 7 bits, se necesitan 21/7 = 3 canales. Para ello, se utilizará un PLL doble, que

obtenga, a partir de los 50MHz que proporciona el reloj del sistema de desarrollo de la FPGA, tanto los 40MHz de VGA como los 280MHz de LVDS.

Es por esto, que se ha optado por la configuración de la Figura 7.13, que es la más sencilla, para facilitar el desarrollo del proyecto.

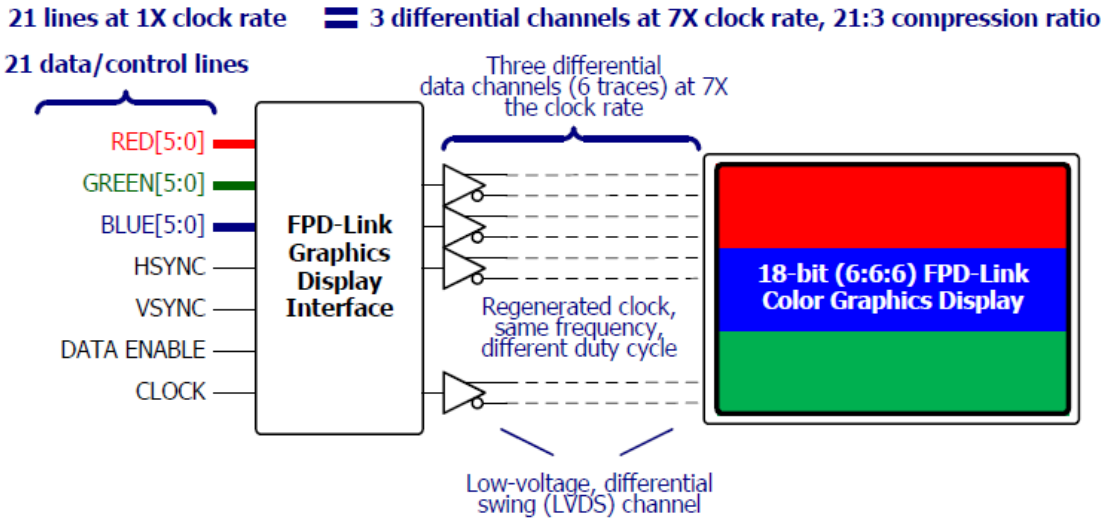


Figura 7.13: Esquema LVDS

Fuente: http://www.prevailing-technology.com/publications/iCE65_LVDS_Display.pdf

Como se muestra en la Figura 7.13 y se ha explicado anteriormente, se requiere hacer uso de 6 bits para cada componente RGB del color, además, se necesita un FPD-Link Graphics Display Interface. Puesto que para este proyecto no se dispone de los medios adecuados para fabricar placas de circuito impreso con componentes de tipo Surface Mount Technology (SMD) complejos, se propone realizar, vía software, la conversión de VGA a LVDS.

Ya que la salida de la FPGA tiene niveles TTL, se necesitará usar algún otro dispositivo, para transformar los niveles de esas señales en niveles LVDS, con sus canales diferenciales.

En concreto, se busca un driver como el de la Figura 7.14, que convierte los niveles TTL a LVDS y además genera los canales diferenciales.

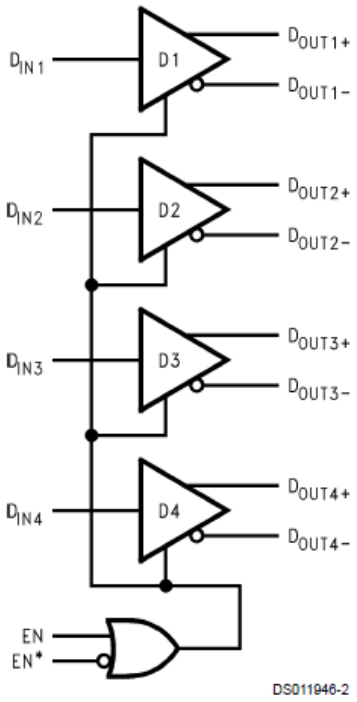


Figura 7.14: Driver LVDS

Fuente: http://www.ti.com/lit/ds/snls095a/snls095a.pdf

El driver anterior es el encargado de generar los niveles de los canales LVDS, por tanto, se requiere generar los canales TTL con los que trabaja el driver. Para este proyecto, en el que se usa VGA con 6 bits para cada componente RGB, al driver, hay que pasarle cuatro canales TTL.

En la Figura 7.15, se muestran los canales mencionados anteriormente, y cómo han de ser multiplexados (que en dicha figura son: PAIR0, PAIR1, PAIR2), teniendo en cuenta que el cuarto canal, el de reloj (CLOCK), lleva la frecuencia original de estos sin multiplexar pero con un ciclo de trabajo diferente.

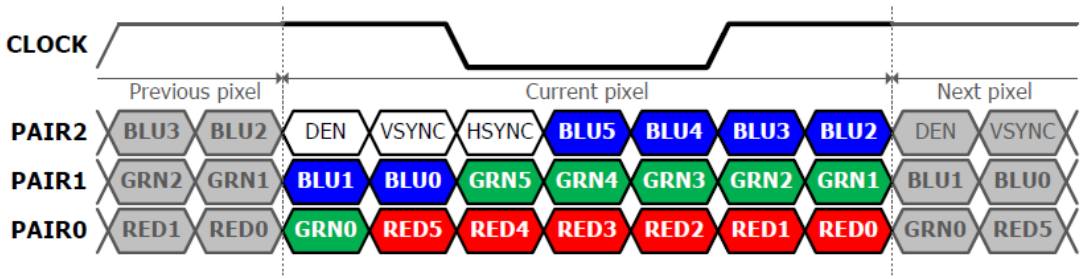


Figura 7.15: VGA de 6 bits sobre LVDS

Fuente: http://www.prevailing-technology.com/publications/iCE65_LVDS_Display.pdf

Como se puede ver en la Figura 7.13 y en la Figura 7.15, por uno de los canales se envía la señal de reloj con la que se trabaja, mientras que el resto de canales se multiplexan a siete veces dicha frecuencia de reloj.

7.3.5 Alimentaciones

Las tensiones disponibles son aquellas que pueden proporcionar la FPGA (+3.3V) y el osciloscopio LeCroy de la serie 93XX (±15V). Por tanto, en función de los componentes que se instalen en la placa controladora diseñada, se podría requerir de otros niveles de tensión continua, generados a partir de las anteriores.

7.4 Placa controladora de video

Puesto que se ha adquirido una pantalla TFT-LCD de 1280x800 píxeles, se requiere una nueva placa controladora de video que pueda trabajar con esta, ya que, estas placas, solo trabajan en un margen fijado de resoluciones y frecuencias.

Esta placa controladora de video, tiene por referencia VS-V59AV, y presenta las siguientes características relevantes para el proyecto:

- Puerto de entrada VGA.
- Puerto de salida LVDS.
- Puerto de salida Backlight.

7.5 Diseño del software

Una vez que se ha propuesto y explicado qué elementos hardware de interconexión son necesarios entre los distintos elementos del proyecto, es el momento de pasar a analizar qué software se requiere desarrollar para el proyecto.

7.5.1 Tecnologías contempladas

Para el software que se ha diseñado, se han tenido en cuenta distintos puntos que se expondrán a continuación:

VGA es un estándar que incluye tanto la forma de generar y procesar las señales como la forma de transmitirlas y los requerimientos que esto tiene. Además, VGA es un estándar de video basado en el funcionamiento analógico de las señales que se transmitían a los tubos de rayos catódicos, por lo tanto, es el estándar de video ideal al que convertir las señales procedentes del osciloscopio.

Hay otros estándares digitales como HDMI o DVI, que a priori, podrían requerir mayor esfuerzo, puesto que habría que transformar las señales analógicas del osciloscopio en digitales.

LVDS es sólo un estándar para la transmisión de datos, ampliamente utilizado en la transmisión de video por las altas velocidades a las que trabaja, es por esto que, aunque se haga uso de LVDS para transmitir los datos a la pantalla TFT-LCD, se seguirá requiriendo dar algún formato estándar a las señales de video procedentes del osciloscopio.

Por todo lo anterior, se generan dos tipos de formato de video, uno para la transmisión de video por un canal de su tecnología (VGA) y otro para enviar ese vídeo directamente a través de LVDS.

7.5.2 Simulación

Un elemento vital para desarrollar el software, y que este cumpla adecuadamente con los requerimientos del proyecto, es la simulación, ya que permite, emulando las señales de entrada, ver cómo se comportan las señales que es necesario generar.

Es por esto que, al igual que se requiere diseñar un software que transforme las señales según los requerimientos, también es necesario diseñar otro software para simular el comportamiento del diseño software antes de probarlo sobre la placa. El software que se propone utilizar si se ha utilizado Quartus II, es ModelSim-Altera

Este es el programa propuesto ya que es gratuito y durante la realización del proyecto, nunca ha presentado ninguna limitación.

8 RESULTADOS FINALES

Una vez que se ha explicado qué elementos se han tenido en cuenta para la realización del proyecto, se va a exponer qué se ha hecho exactamente para dar solución al objeto del proyecto.

8.1 Montaje realizado

Se han realizado dos tipos de montajes (ver esquema de interconexión en las Figura 8.1 y la Figura 8.2), cuando se transmite el video por VGA se necesita la controladora adquirida y cuando se transmite mediante LVDS hace uso de la controladora diseñada.

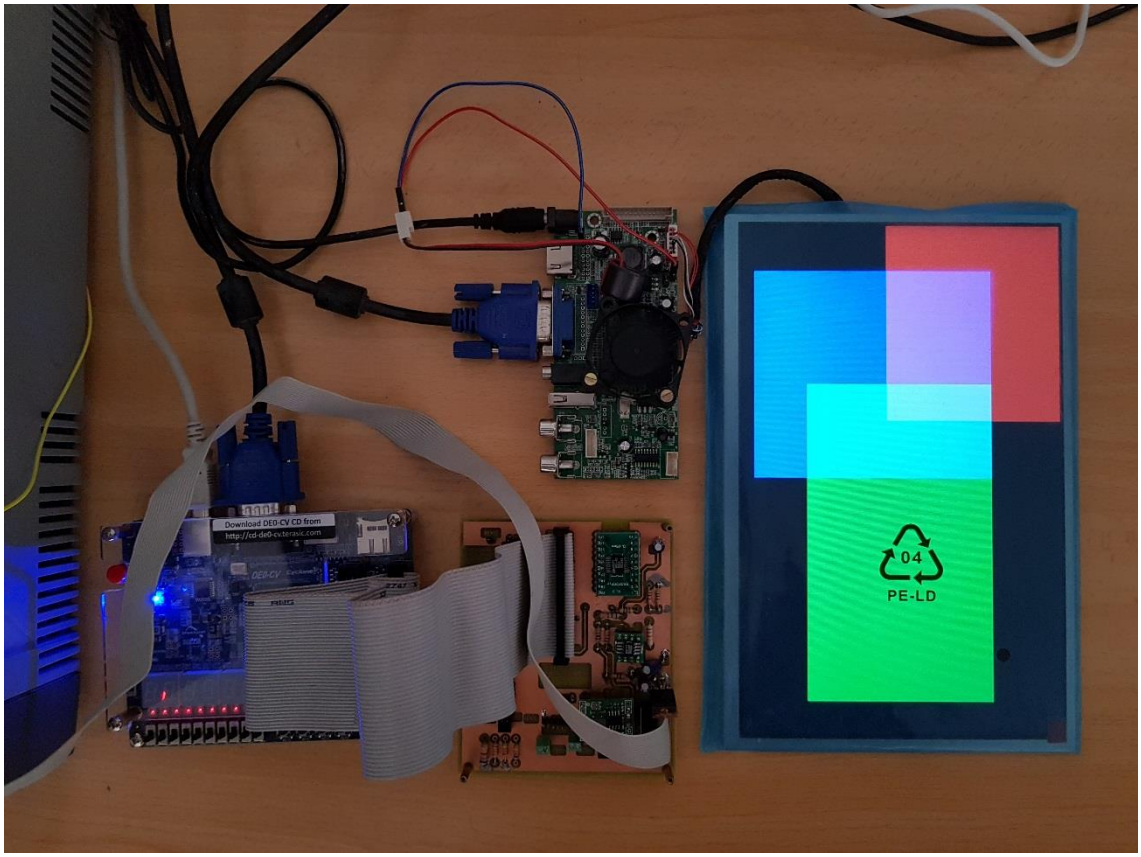


Figura 8.1: Componentes del montaje con VGA

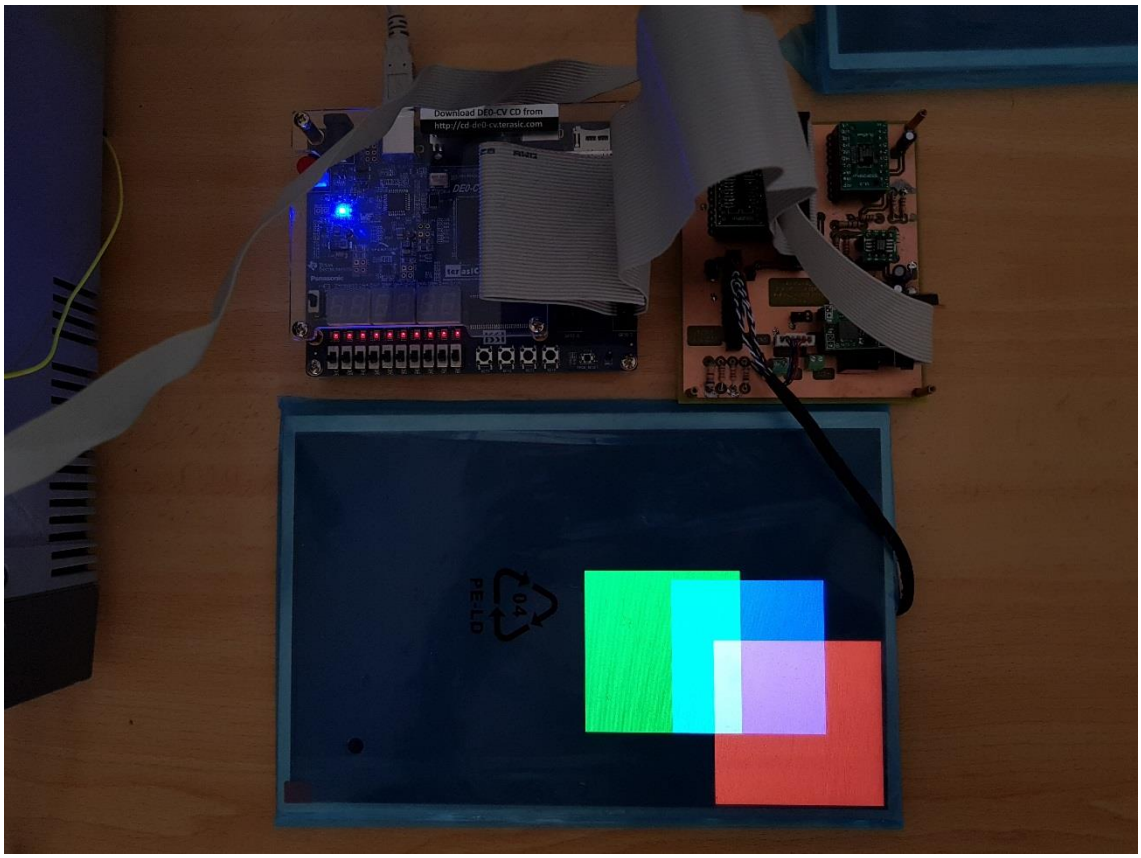


Figura 8.2: Componentes del montaje con LVDS

8.2 Placa controladora diseñada

La placa controladora diseñada es la encargada de recibir las señales del osciloscopio, adaptarlas si es necesario y ponerlas a disposición de otros componentes del sistema.

Es por esto que se va a explicar individualmente cada elemento de hardware que forma parte de esta placa.

8.2.1 *Puerto de 14 pines para las señales procedentes del osciloscopio*

En este puerto de 2x7 pines (ver la Figura 8.3), se reciben en la placa controladora diseñada las señales y alimentaciones procedentes del osciloscopio.

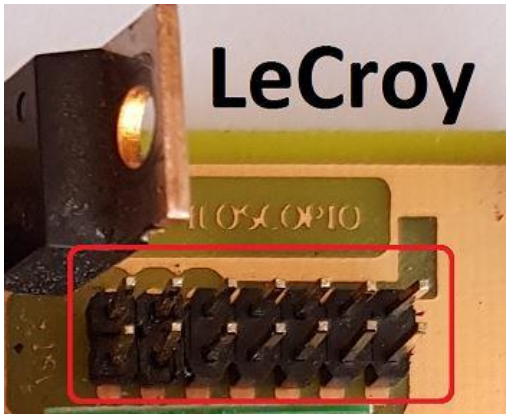


Figura 8.3: Puerto para las señales del osciloscopio

En la Tabla 8.1, se recoge qué señal lleva asociada cada pin:

Pin	Señal
1	+15V
2	+15V
3	+15V
4	+15V
5	-15V
6	-15V
7	GND
8	GND
9	VSYNC
10	GND
11	HSYNC
12	GND
13	Intensidad
14	GND

Tabla 8.1: Señales provenientes del osciloscopio

Es a través de este puerto por el que se proporcionan ±15V para alimentar la placa controladora diseñada.

8.2.2 *Puerto para la señal de reloj del osciloscopio*

Como ya se propuso en el apartado 7.3.3.1, se ha añadido un pin para recibir en la placa controladora diseñada la señal de reloj de 48MHz que genera el osciloscopio. Ver la Figura 8.4.

Figura 8.4: Puerto para las señal de reloj del osciloscopio

8.2.3 *Puerto GPIO de 40 pines*

El siguiente puerto de 2x20 pines (ver la Figura 8.5), se utiliza tanto para enviar señales desde la placa controladora diseñada, como para enviar y recibir señales desde la FPGA.

Figura 8.5: Puerto GPIO

En la Tabla 8.2, se recoge qué señal lleva asociada cada pin y si esta es enviada o recibida por la FPGA:

Pin	FPGA	Tipo	Señal	Procedencia
3	GPIO2	OUT	Reloj a 48MHz	FPGA
4	GPIO3	IN	D7	AD9057
5	GPIO4	IN	Reloj a 48MHz	LeCroy
6	GPIO5	IN	D6	AD9057
7	GPIO6	OUT	RO1+	LVDS
8	GPIO7	IN	D5	AD9057
9	GPIO8	OUT	RO0+	LVDS
10	GPIO9	IN	D4	AD9057
12	GND			
13	GPIO10	OUT	RO2+	LVDS
14	GPIO11	IN	D3	AD9057
15	GPIO12	OUT	CLK+	LVDS
16	GPIO13	IN	D2	AD9057
18	GPIO15	IN	D1	AD9057
20	GPIO17	IN	D0	AD9057
29	3.3V			
30	GND			
38	GPIO33	IN	Intensidad	AD8002
39	GPIO34	IN	HSYNC	LeCroy
40	GPIO35	IN	VSYNC	LeCroy

Tabla 8.2: Señales en el puerto GPIO de la placa controladora diseñada

Aclarar que la terminología IN OUT obedece a cómo son las señales para la FPGA: IN para aquellas señales que lee la FPGA y que son procesadas por esta, OUT para las señales que genera la FPGA.

La distribución de pines anterior, se ha hecho bajo el criterio de simplificar el enrutado de la placa controladora diseñada.

8.2.4 *Puerto Backlight TFT*

Este puerto de 1x6 pines (ver la Figura 8.6), es usado por aquellas pantallas TFT-LCD que no se alimentan a través del puerto LVDS:

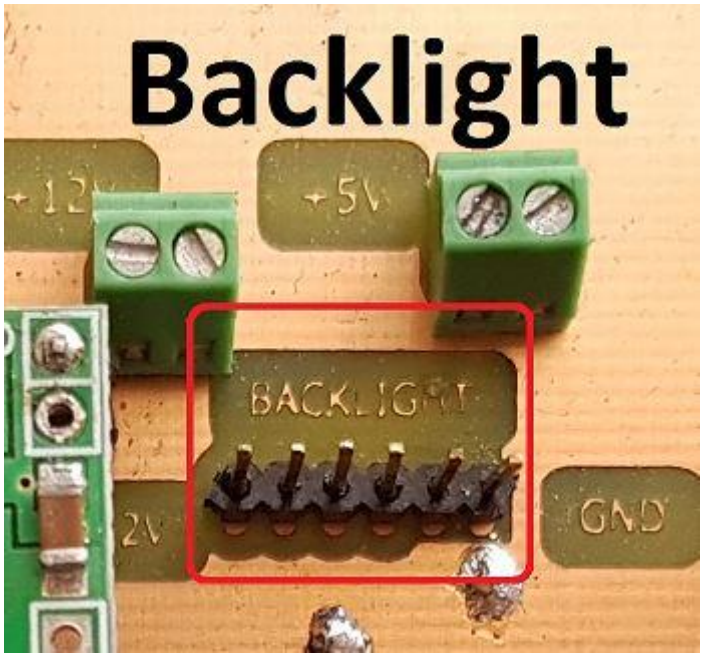


Figura 8.6: Puerto Backlight

En la Tabla 8.3, se recoge qué señal lleva asociada cada pin:

Pin	Señal
1	+12V
2	+12V
3	+5V
4	Sin uso
5	GND
6	GND

Tabla 8.3: Señales en el puerto Backlight

8.2.5 Puerto LVDS

Este puerto de 2x15 pines (ver la Figura 8.7), se utiliza para enviar el video a la pantalla TFT-LCD.

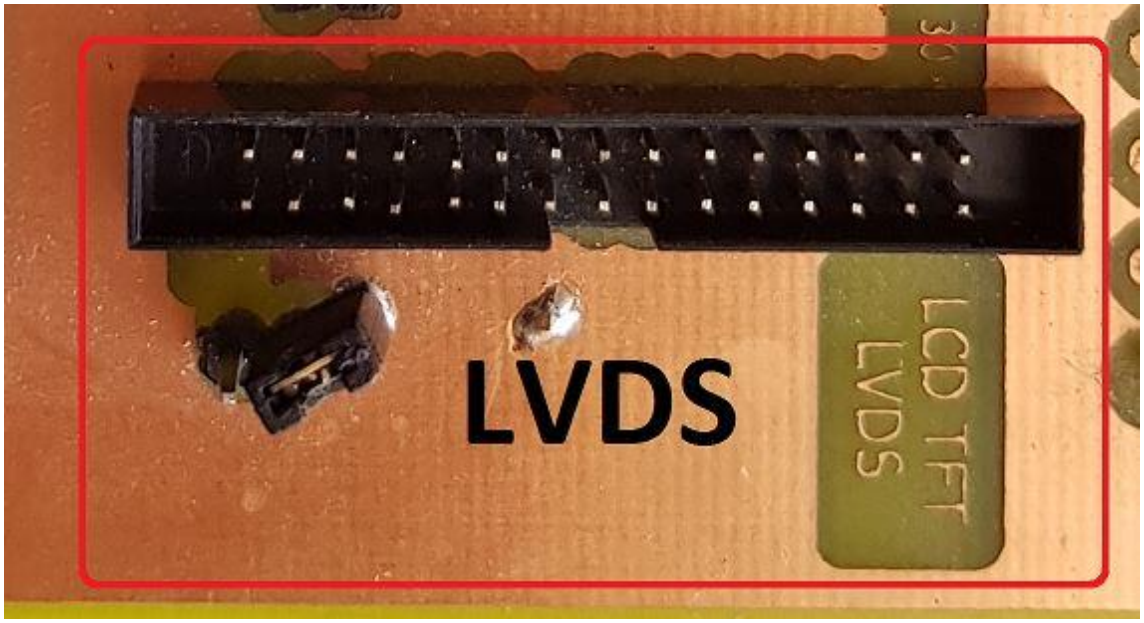


Figura 8.7: Puerto LVDS

En la Tabla 7.1 se recoge qué señal lleva asociada cada pin:

Este esquema de conexiones se corresponde con que cada componente de la señal VGA que se genera es de 6 bits.

Figura 8.8: Resistencias para LVDS

Las resistencias que se muestran en la Figura 8.8, como se ha explicado en el apartado 3.8.1, sirven para impedir reflexiones en las señales enviadas.

8.2.6 Regletas de conexión de dos contactos

Las regletas que se muestran en la Figura 8.9 se utilizan para alimentar otros componentes del proyecto, en concreto, la regleta que proporciona +12V, se utiliza para alimentar la placa controladora de video adquirida y la otra regleta, que proporciona +5V, se utiliza para alimentar el sistema de desarrollo basado en FPGA.

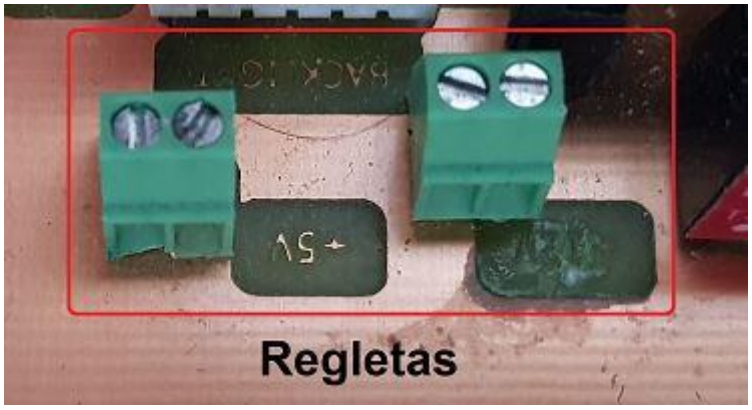


Figura 8.9: Regletas para alimentación

8.2.7 Tracopower TRS 1-2450

Este conversor DC/DC según se utiliza para proporcionar +5V y 1A a su salida. Los +15V que necesita el conversor Tracopower TRS 1-2450 (ver la Figura 8.10) se los proporciona el osciloscopio a través del puerto de 14 pines, como se explica en el apartado 8.2.1.

Figura 8.10: Tracopower TRS 1-2450

El esquema del conversor se muestra en la Figura 8.11:

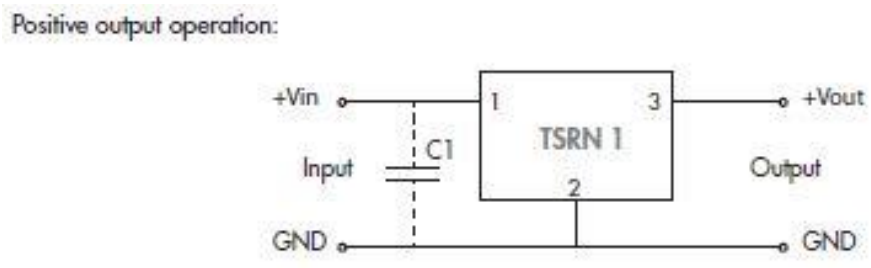


Figura 8.11: Esquema Tracopower TRS 1-2450

Fuente: https://assets.tracopower.com/20170830075536/TSRN1/documents/tsrn1-datasheet.pdf

El regulador es de tipo 'step-down'. Proporciona una tensión fija de +5V a su salida, a partir de +15V en su entrada, todo esto, sin que requiera de disipador.

8.2.8 Regulador de tensión LM7905

Este regulador se utiliza para proporcionar en $V_0 = -5V$ y 1.5A cuando se conecte V_i a -15V (ver la Figura 8.12). Los -15V que necesita el regulador de tensión LM7905 se los proporciona el osciloscopio a través del puerto de 14 pines. En su salida de -5V, se necesita únicamente en la entrada de alimentación negativa de los amplificadores operacionales.

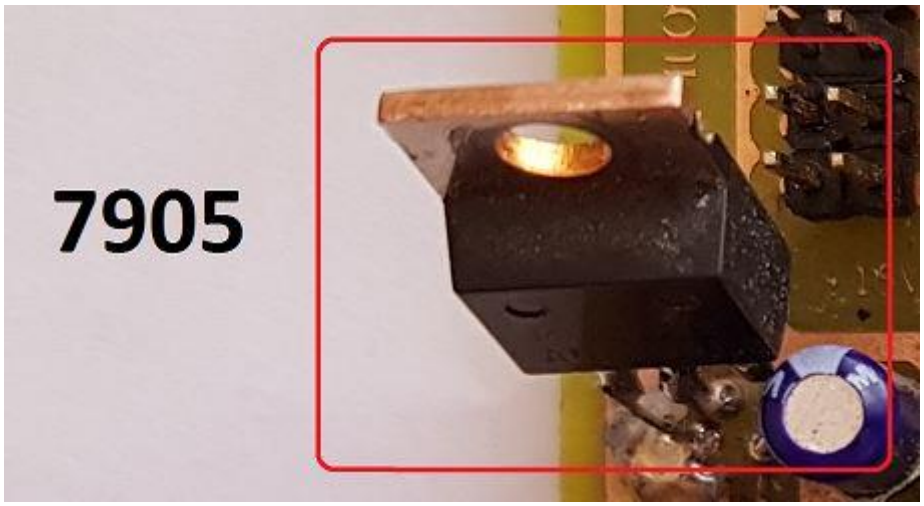


Figura 8.12: LM7905

El esquema propuesto para este integrado se puede en la Figura 8.13:

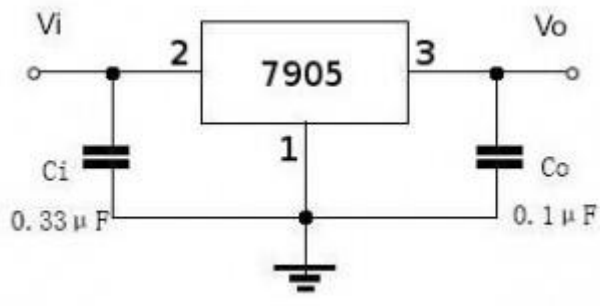


Figura 8.13: Esquema LM7905

Fuente: http://electronica-teoriaypractica.com/reguladores-de-tension-7905-7912-7915-y-7924/

8.2.9 Regulador de tensión MP1584

Este regulador es utilizado para proporcionar en $V_0 = +12V$ y 3A cuando se conecte V_i a +15V. Los +15V que necesita el regulador de tensión MP1584 se los proporciona el osciloscopio a través del puerto de 14 pines. Su salida es variable según el potenciómetro que aparece en su parte inferior (estrella), como puede verse en la Figura 8.14.

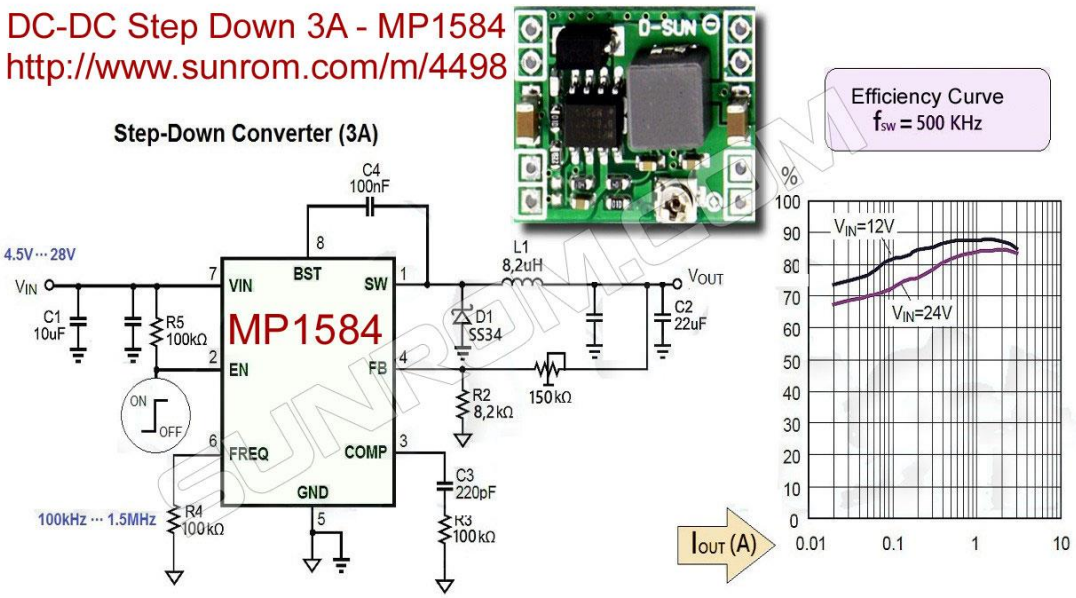


Figura 8.14: MP1584 con esquema

Fuente: http://jghttrading.com/p/dc-dc-step-down-3a-mp1584.htm

El regulador es de tipo 'step-down', proporcionando una tensión inferior a la de entrada. Es el regulador elegido ya que proporciona la tensión que se requiere y además no requiere de elementos de disipación de potencia.

8.2.10 Operacional AD8002

Este integrado contiene dos amplificadores operacionales en la misma pastilla y requiere de una alimentación simétrica de ±5V. Uno de los operacionales se utiliza para amplificar la señal de intensidad de 1V a 3.3V para la FPGA y el otro para adaptar y amplificar la señal de intensidad de entre 0V y 1V a entre 2V y 3V para el conversor analógico digital AD9057.

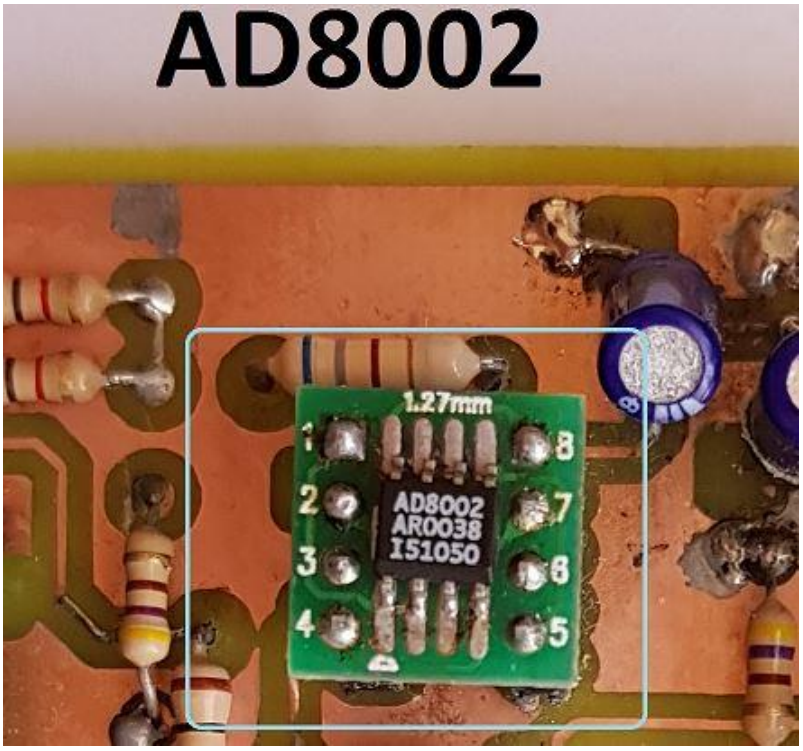


Figura 8.15: AD8002

Como puede verse en la Figura 8.15, al ser el encapsulado SOP, requiere de una placa enrutada y preestañada que facilite su instalación en la placa controladora diseñada. Su esquema de conexionado se muestra en la Figura 8.16:

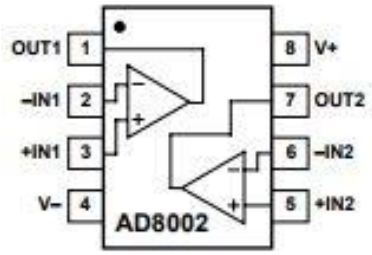


Figura 8.16: Esquema AD8002

Fuente: http://www.analog.com/media/en/technical-documentation/data-sheets/AD8002.pdf

8.2.11 Conversor AD9057

Este integrado (ver la Figura 8.17), es el encargado de convertir la señal analógica de intensidad, procedente del osciloscopio, en niveles digitales a partir de los cuales se pretende asignar colores en la pantalla TFT-LCD.

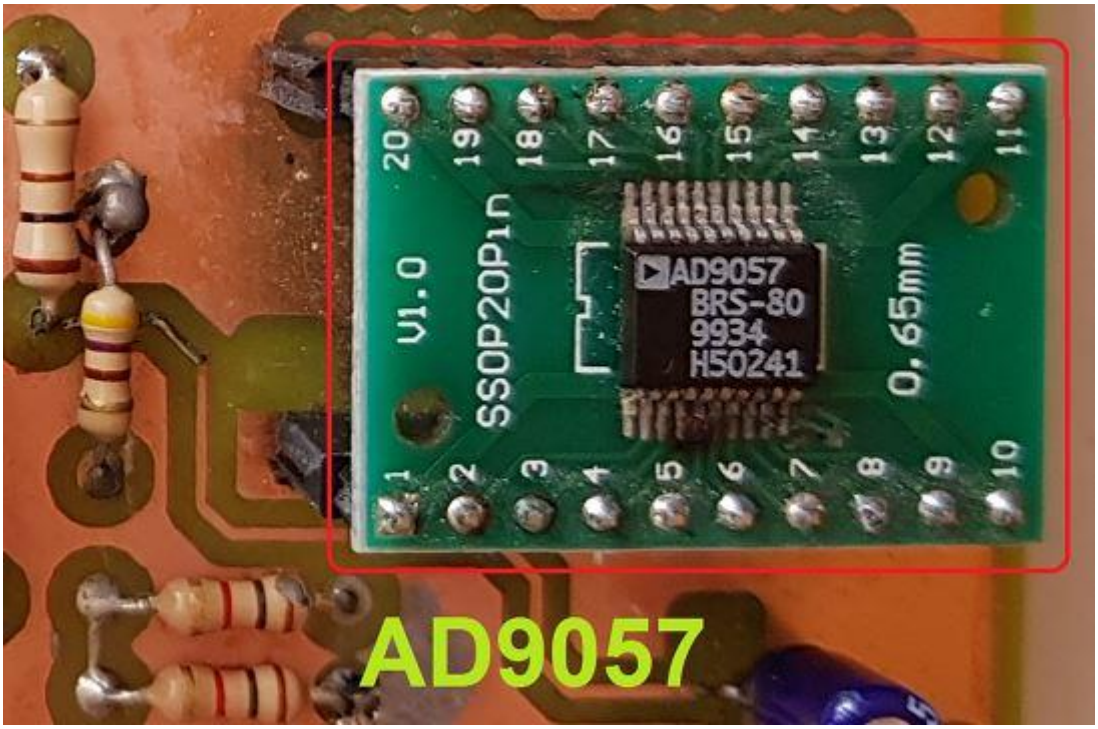


Figura 8.17: AD9057

El esquema general del conversor AD9057 puede verse en la Figura 8.18:

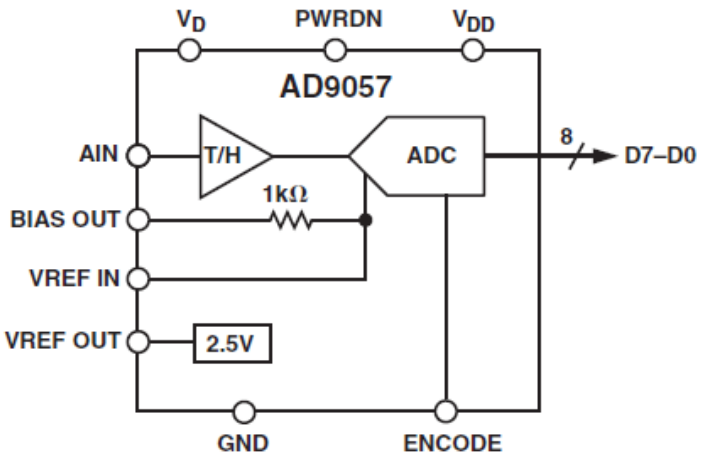


Figura 8.18: Esquema AD9057

Fuente: http://www.analog.com/en/products/analog-to-digital-converters/ad9057.html#product-overview

El esquema de montaje que propone el fabricante se muestra en la Figura 8.19:

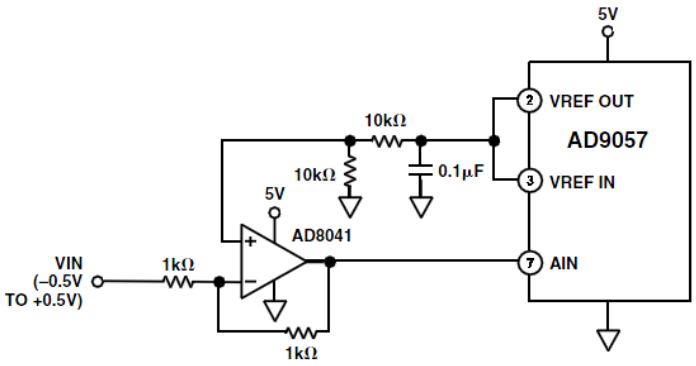


Figure 3. DC-Coupled AD9057 (Inverted VIN)

Figura 8.19: Esquema de conexión para el AD9057

Fuente: http://www.analog.com/en/products/analog-to-digital-converters/ad9057.html#product-overview

Para ello se deben conectar los terminales de la siguiente manera:

- PWRDN: debe ser conectado a masa.
- BIAS OUT: debe ser conectado a la entrada analógica AIN, pero solo si la entrada analógica es sinusoidal.
- Se deben conectar VREF IN y VREF OUT.
- Se debe conectar V_D a 5V y V_{DD} a 3.3V.
- ENCODE se debe conectar a la señal de reloj, en este caso, a 48MHz.

Gracias a esta configuración, resulta que el valor de la entrada analógica del conversor V_{AIN}, en función de la entrada V_{IN} y de la tensión en el terminal no inversor V^+ vale:

$$V_{AIN} = \left(1 + \frac{R_2}{R_1}\right) V^+ - \frac{R_2}{R_1} V_{IN}$$

(8.1)

Si en el terminal de referencia V_{REF} colocamos un resistivo formado por R_3 y R_4 conectado a masa, de modo que:

$$V^+ = \frac{R_4}{R_3 + R_4} V_{REF}$$

(8.2)

Al sustituir (8.2) en (8.1) obtenemos:

$$V_{AIN} = \left(1 + \frac{R_2}{R_1}\right)\frac{R_4}{R_3 + R_4}V_{REF} - \frac{R_2}{R_1}V_{IN}$$

(8.3)

Puesto que $\mathrm{V_{REF}} = \mathrm{V_{REF\,IN}} = \mathrm{V_{REF\,OUT}} = 2.5\mathrm{V}$, que $\mathrm{R_1} = \mathrm{R_2} = 1\mathrm{K\Omega}$ y que $\mathrm{R_3} = \mathrm{R_4} = 10\mathrm{K\Omega}$. Al sustituir estos valores en (8.3) se obtiene:

$$\mathrm{V_{AIN}} = \left(1 + \frac{\mathrm{R_2}}{\mathrm{R_1}}\right)\frac{\mathrm{R_4}}{\mathrm{R_3 + R_4}}\mathrm{V_{REF}} - \frac{\mathrm{R_2}}{\mathrm{R_1}}\mathrm{V_{IN}} = \mathrm{V_{REF}} - \mathrm{V_{IN}} = 2.5 - \mathrm{V_{IN}}$$

(8.4)

Para que funcione bien, la señal de estada $\mathrm{V_{IN}}$ debe variar entre -0.5V y 0.5V, por lo que la señal de salida del amplificador y entrada al conversor variará entre 2V y 3V. Esto se debe a que la relación entre la entrada y la salida del conversor AD9057 es Figura 8.20:

Table I. Digital Coding (VREF = 2.5 V)

Analog Input	Voltage Level	Digital Output
3.0 V	Positive Full Scale	1111 1111
2.502 V	Midscale +1/2 LSB	1000 0000
2.498 V	Midscale –1/2 LSB	0111 1111
2.0 V	Negative Full Scale	0000 0000

Figura 8.20: Relación entrada/salida en el AD9057

Fuente: http://www.analog.com/en/products/analog-to-digital-converters/ad9057.html#product-overview

Es importante observar que el conversor tiene una salida de referencia de 2.5V disponible en la unión de los terminales $\mathrm{V_{REF\,IN}} = \mathrm{V_{REF\,OUT}}$.

El diseño anterior tiene varios problemas (ver ecuación (8.4)). Por un lado, la entrada del conversor $\mathrm{V_{AIN}}$ se encuentra desfasada 180º respecto de la entrada de intensidad del osciloscopio $\mathrm{V_{IN}}$. Por otro lado, la señal $\mathrm{V_{IN}}$ debe estar comprendida entre -0.5V y 0.5V.

Por esta razón, se propone el diseño de la Figura *8.21*, que solventa los dos inconvenientes anteriores. Se parte de un amplificador diferencial típico con la siguiente estructura:

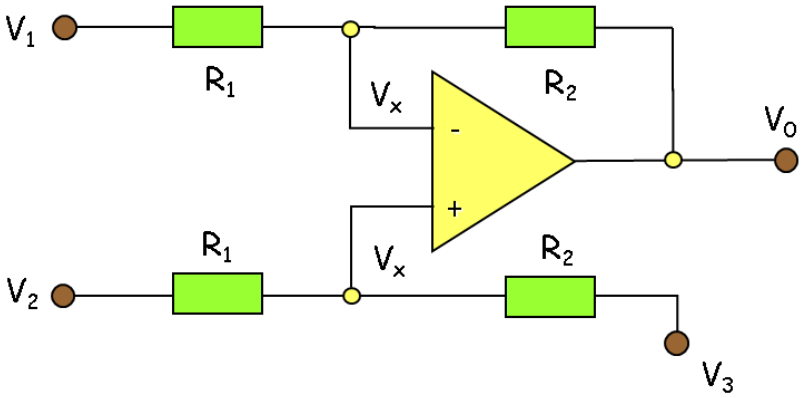


Figura 8.21: Esquema general de un amplificador diferencial

La salida V_0, en función de las entradas V_1, V_2 y V_3 vale:

$$V_0 = \frac{R_2}{R_1}(V_2 - V_1) + V_3 \tag{8.5}$$

Haciendo que $V_2 = V_{IN}$ y $V_1 = 0V$ en (8.5), se obtiene:

$$V_0 = \frac{R_2}{R_1}V_{IN} + V_3 \tag{8.6}$$

Donde R_2/R_1 es la ganancia y V_3 el offset.

Como la señal a amplificar, V_{IN} (INTENSIDAD), varía entre 0V y 1V.

A partir de (8.6), para $V_{IN} = 0V$, se tiene que $V_0 = 2V$, por lo tanto $V_0 = V_3 \rightarrow V_3 = 2V$. Para $V_{IN} = 1V$, se tiene que $V_0 = 3V$, por lo tanto $3 = \frac{R_2}{R_1} + 2$ es decir, que la relación es $\frac{R_2}{R_1} = 1$. En este caso, se toma $R_1 = R_2 = 1K\Omega$.

Lo cierto es que el amplificador necesario, va a necesitar la alimentación simétrica de ±5V, que se genera a partir de las alimentaciones en el mismo conector que las señales de sincronismo e intensidad del osciloscopio.

Se necesita también otro diseño basado en un amplificador que obtenga, a partir de la señal de intensidad (V_{IN}), una salida para conectarse a una entrada digital de la FPGA, por lo que es necesario amplificar dicha señal de entrada hasta los 3.3V.

Bastaría con un amplificador de ganancia positiva de valor $G = 3.3 = 1 + \frac{R_2}{R_1}$, por lo tanto $\frac{R_2}{R_1} = 2.3$. Por último, tras realizar simulaciones con el amplificador operacional elegido, el AD8002, se ha comprobado que funciona mejor con resistencias relativamente pequeñas, así que los valores asignados son: $R_1 = 470\Omega$ y $R_2 = 1K\Omega$.

8.2.12 Jumper para seleccionar el reloj usado por el conversor AD9057

A este jumper llegan los relojes de 48MHz que genera el osciloscopio y el que se genera con el sistema de desarrollo de la FPGA. Ver la Figura 8.22.

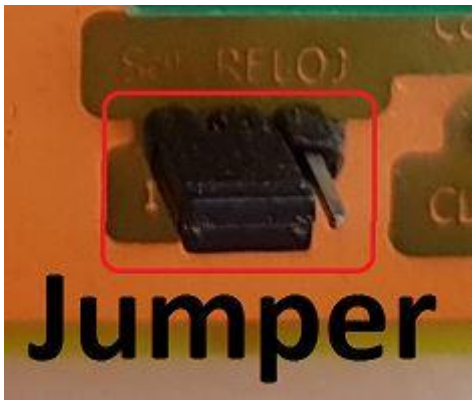


Figura 8.22: Jumper para alternar entre las señales de reloj

Para este proyecto se hace uso del que se genera con el sistema de desarrollo de la FPGA porque da un mejor resultado a la hora de mostrar el video en la pantalla TFT-LCD.

8.2.13 2 Jumpers para LVDS

Aunque el puerto LVDS es estándar, hay pantallas TFT-LCD que requieren que algunos pines tengan alimentaciones diferentes a la del estándar, en concreto:

Uno de los jumpers se utiliza para seleccionar que en el pin 3 se tenga 3.3V o se quede sin conectar. Ver la Figura 8.23.

Figura 8.23: Jumper para poner el pin 3 a 3.3V

El otro jumper se utiliza para que en el pin 4 se tenga masa o se quede sin conectar. Ver la Figura 8.24.

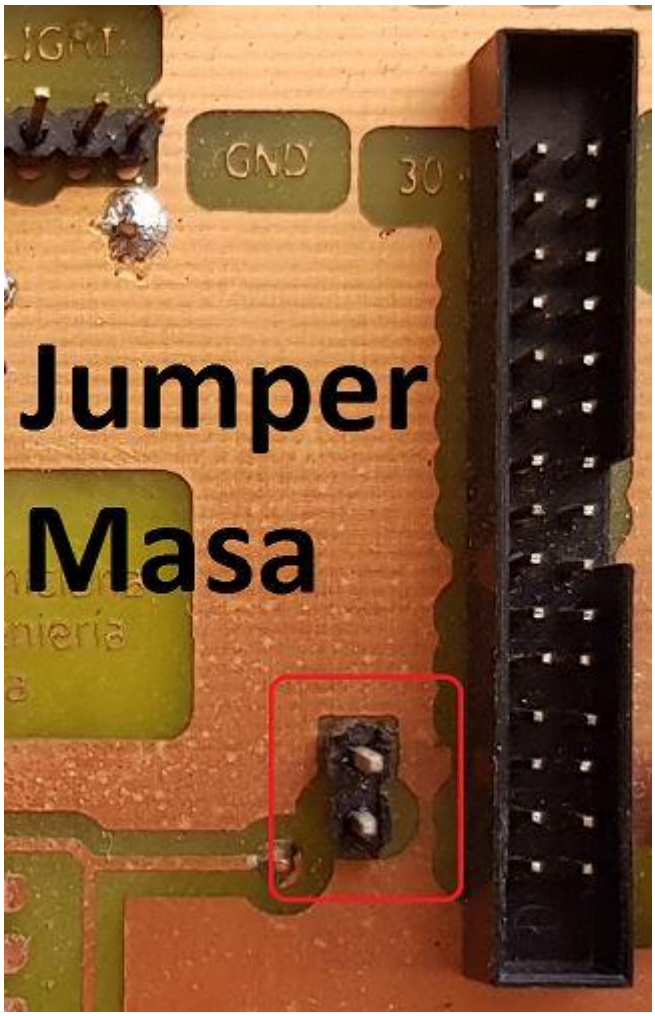


Figura 8.24: Jumper para poner el pin 4 a masa

8.2.14 Driver de línea Ds90c031

Este integrado sirve para convertir los niveles TTL a LVDS y generar además los canales diferenciales

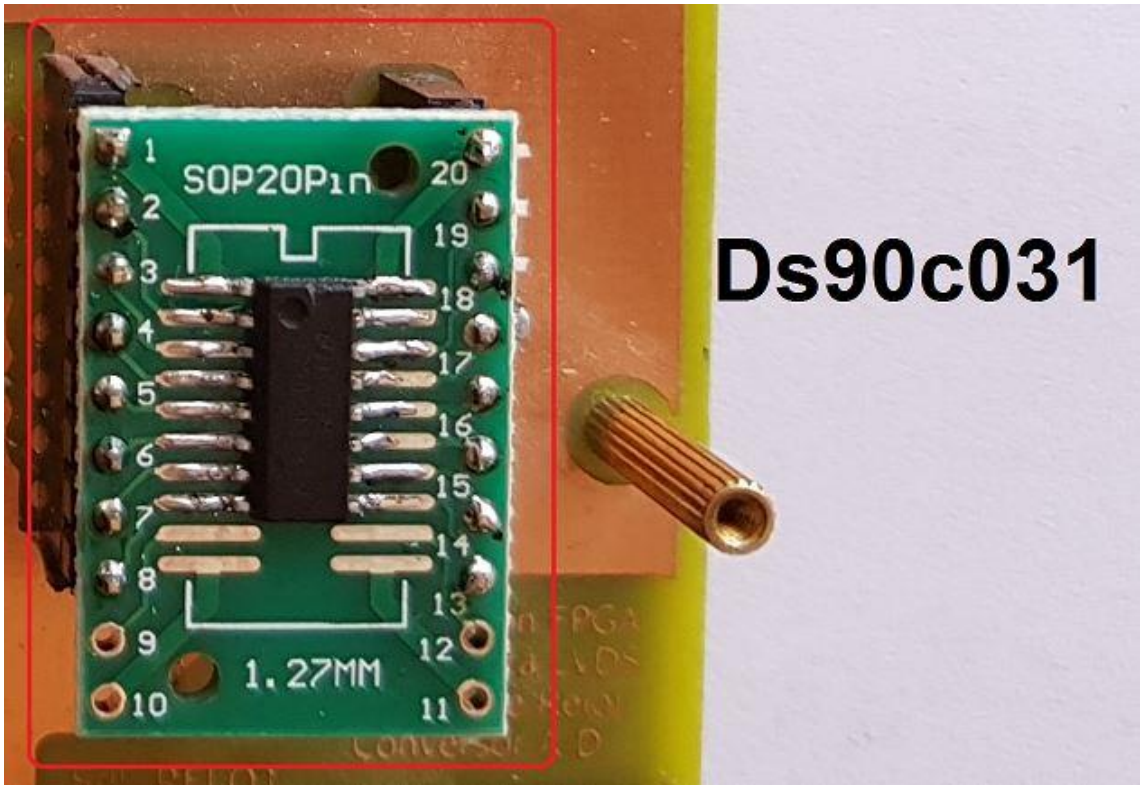


Figura 8.25: Ds90c031

Como puede verse en la Figura 8.25, al ser un integrado con encapsulado Small Outline Package (SOP), se requiere una placa enrutada y preestañada que facilite su instalación en la placa controladora diseñada. Su esquema de conexionado se muestra en la Figura 8.26:

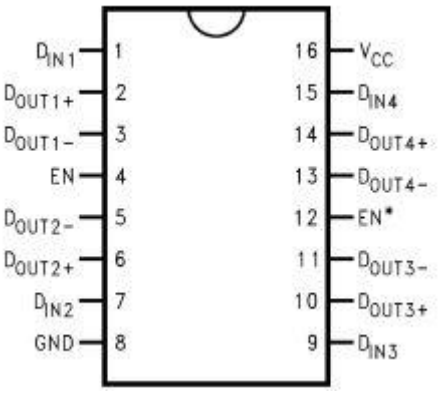


Figura 8.26: Esquema Ds90c031

Fuente: http://www.ti.com/lit/ds/snls095a/snls095a.pdf

Las señales de tipo D_{IN} las genera la FPGA. Las señales de tipo D_{OUT} son los canales LVDS que van al puerto con el mismo nombre. Las señales EN y EN* sirven para habilitar el integrado. Por último el integrado requiere una alimentación entre -0.3V hasta +6V, para este proyecto es de $V_{CC} = 3.3V$.

8.2.15 Resistencias

A continuación, se detallará el uso de las resistencias que se han instalado en la placa controladora diseñada:

En el apartado 8.2.5, se detalla el uso de 4 resistencias de 100Ω.

En el apartado 8.2.11, se expusieron una serie de cálculos, el resultado de estos, es la necesidad de colocar 4 resistencias 1KΩ para adaptar la señal de intensidad que se dirige al conversor AD9057, además, puesto que se requiere una referencia de tensión de 2V para los cálculos, se ha diseñado un divisor de tensión con dos resistencias, una de 470Ω y otra de 100Ω.

En el mismo apartado (8.2.11), se propone otro diseño para adaptar esa misma señal de intensidad a la entrada digital de la FPGA, tras realizar los cálculos, se obtienen unas resistencias de 1KΩ y 470Ω, pero, tras realizar algunas pruebas, se ha visto que reduciendo la ganancia se obtenía un mejor resultado, por lo que la resistencia de 1KΩ, se ha acabado sustituyendo por otra de 680Ω.

Por último, con dos resistencia de 33KΩ y otras dos de 18KΩ, se ha diseñado dos divisor de tensión, para reducir la tensión de las señales de sincronismo horizontal y vertical procedentes del osciloscopio, que como se expone en el apartado 7.3.4.1, son de 5V y, al dirigirse a la entrada digital de la FPGA, se requiere un nivel de tensión de 3.3V.

A modo de aclaración, en la placa controladora diseñada, uno de los dos juegos de resistencias al anterior descrito, para atenuar la señal de sincronismo horizontal, se ha puesteado, pues durante las pruebas que se han realizado, ha resultado que proporciona mejores resultados que la señal atenueda, como puede verse en la Figura 8.28.

8.2.16 Condensadores

Hay 3 condensadores de 1µF 50V, que han sido colocados en base a las recomendaciones de los distintos fabricantes.

Los condensadores son de 50 voltios porque tienen que ser de más del doble de la máxima tensión utilizada.

8.2.17 Esquema de conexionado

Todos los elementos mencionados anteriormente han sido interconectados, como puede verse en el Plano 1, el Plano 2 y el Plano 3.

El resultado de todas estas conexiones es una placa controladora con unas dimensiones de 108x90mm y que se presenta enrutada a dos caras. A continuación, se muestra el resultado final, que, una vez que se ha fabricado la placa y se le han soldado los componentes, como se puede ver en la Figura 8.27 y la Figura 8.28.

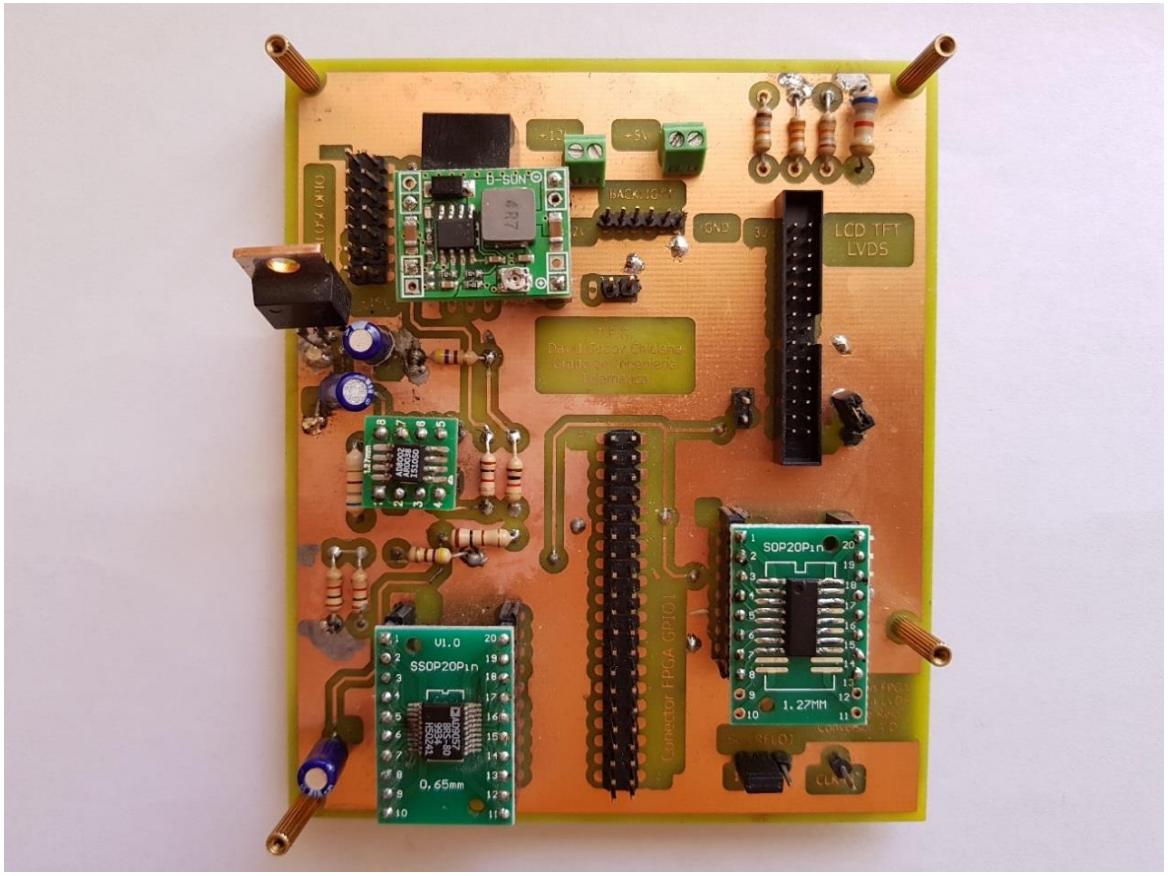


Figura 8.27: Cara superior de la placa controladora diseñada

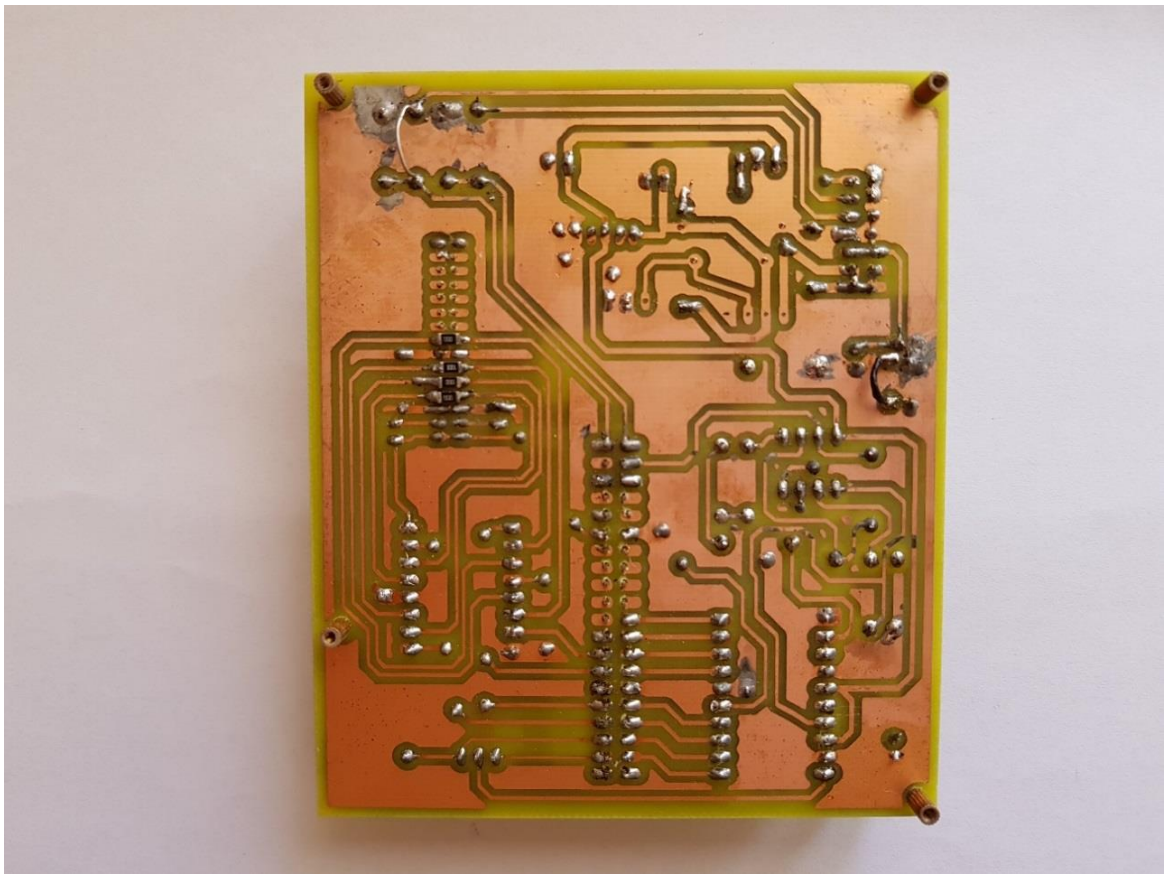


Figura 8.28: Cara inferior de la placa controladora diseñada

8.3 Software

En este apartado, se explicará el código que ha sido diseñado utilizando el programa Quartus II y el lenguaje de descripción de hardware VHDL.

En el Anexo 1, se encuentra todo el código del proyecto en formato de texto, pero aquí se van a exponer los distintos ficheros que forman el software del proyecto, detallando el funcionamiento de las partes relevantes de cada uno.

8.3.1 Consideraciones iniciales

Con el objetivo de dar color a la pantalla TFT-LCD de 1280x800 píxeles e intentando simplificar el código del programa, se ha optado por trabajar con RGB haciendo uso de un único bit por color. Sin embargo, en el apartado 7.3.4.3, se expuso que la configuración más simple para transmitir VGA sobre LVDS requiere hacer uso de 6 bits por cada componente del color.

Teniendo esto en cuenta, se trabajará con 6 bits como si fuese 1 bit, es decir, cuando se quiera enviar ‘1’, se enviaría ‘111111’.

Por último, y tras hacer pruebas a distintas frecuencias, se ha acabado optando por trabajar con un reloj de 40MHz para la transmisión con LVDS, ya que frecuencias superiores ofrecen peores resultados (debido seguramente a las limitaciones de las pistas en la placa controladora diseñada, véase el apartado 3.8.1). Por tanto, los datos son multiplexados a 280MHz (7 × 40MHz).

8.3.2 Herencia

VHDL es un lenguaje fuertemente jerarquizado y dependiente de los tipos de datos. Por tanto, no solo se explicará cómo funciona cada entidad que compone el software, sino también la jerarquía adoptada.

Los ficheros de los que consta este proyecto son:

- vga696810.vhd: Entidad superior de la jerarquía, desde la que se instancian las demás entidades.
- accede_VIDEO_RAM.vhd: Almacena y lee las pantallas del osciloscopio.
- ADC9057.vhd: Máquina de estados para la conversión A/D.
- video_ram_dual_port.vhd: Define una RAM de doble puerto, y el tipo de acceso permitido en la RAM interna de la FPGA para almacenar la pantalla.
- zona_dual_port.vhd: Define una RAM de doble puerto, y el tipo de acceso permitido en la RAM interna de la FPGA para almacenar los valores convertidos con el AD9057.
- Interface_FDPLink_18bits.vhd: Genera los canales de datos multiplexados que son enviados al driver que adapta los niveles TTL a LVDS (Ds90c031).
- serializador.vhd: Registro de desplazamiento para transformador de paralelo a serie.

En la Figura 8.29 se muestra la jerarquía de los distintos ficheros que intervienen en este proyecto.

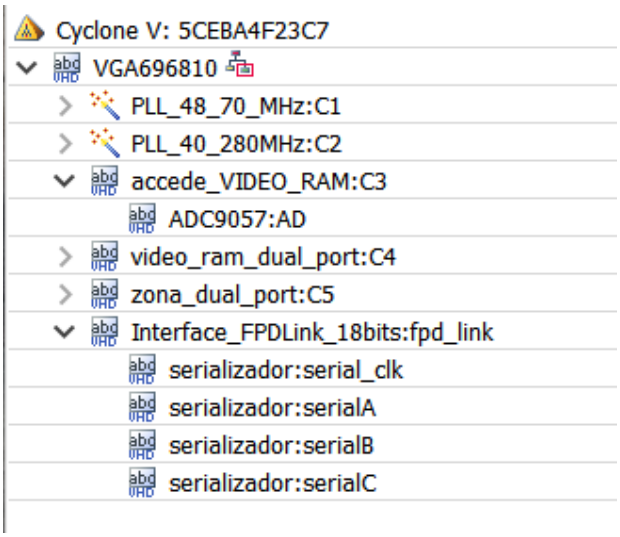


Figura 8.29: Jerarquía adoptada en el proyecto

8.3.3 vga696810.vhd

Esta es la entidad superior de la jerarquía y es la encargada de instanciar al resto de entidades.

8.3.4 accede_VIDEO_RAM.vhd

Esta entidad, es la encargada de guardar en RAM la pantalla de 696x810 píxeles a 48MHz procedente del osciloscopio LeCroy de la serie 93XX. También es la encargada de leer de RAM y generar toda la información necesaria, para representar adecuadamente en la pantalla TFT-LCD de 1280x800 píxeles la pantalla procedente del osciloscopio. Está compuesta por tres procesos, y de esta depende también la entidad ADC9057.vhd:

En el primer proceso, como se muestra en la Figura 8.30, es el encargado de detectar cuando comienza una nueva pantalla. Desactiva la señal una vez ha finalizado el video activo de la actual.

```
-- Detecta cada FRAME del osciloscopio mediante
-- el flanco negativo del sincronismo horizontal
DETECTA_FRAME:PROCESS(VGA_H,HPOS48,VPOS48)
BEGIN
   IF (HPOS48=821) AND (VPOS48=0) THEN    -- Condición asíncrona para inicializar FRAME
      FRAME <= FALSE;
   ELSIF falling_edge(VGA_H) THEN         -- Las señales del osciloscopio estan invertidas
                                          -- respecto a una VGA estandar, por eso se
                                          -- detecta el flanco negativo
      FRAME <= TRUE;                      -- Detectado FRAME
   END IF;
END PROCESS DETECTA_FRAME;
```

Figura 8.30: DETECTA_FRAME

En el segundo proceso como puede verse en la Figura 8.31, la Figura 8.32 y la Figura 8.33, cuando se lee un nuevo píxel, se detecta a qué zona de la pantalla pertenece y, haciendo uso de ZONA_W, se le indica al conversor AD9057, que comprobará si para esa zona se ha convertido algún otro píxel y, en caso negativo, será el encargado de obtener un color que se almacena en la RAM del conversor. La detección de la zona del píxel, se consigue mediante los contadores de fila y columna HPOS_48 y VPOS_48, habilitados para tal fin.

El almacenamiento en RAM de video se realiza en posiciones de memoria de tamaño byte, es decir, se almacenan 8 píxeles en cada posición de memoria, ocupando posiciones consecutivas a partir de la posición 0.

```
SALVAR_VIDEO_RAM: PROCESS(CLK48)

    VARIABLE dato: STD_LOGIC_VECTOR((ANCHO_DATO-1) DOWNTO 0) := (OTHERS => '0');  -- dato a escribir en RAM
    VARIABLE conta_dir: NATURAL RANGE 0 TO (2**LONG_DIR)-1 := 0;                  -- direccion RAM
    VARIABLE conta_b: STD_LOGIC_VECTOR(3-1 DOWNTO 0) := (OTHERS => '0');          -- contador de bit (8 bit/dato)
    VARIABLE HPO,VPO  : STD_LOGIC_VECTOR(10 DOWNTO 0) := (OTHERS => '0');         -- contador de columna y fila

BEGIN
IF rising_edge(CLK48) THEN
   IF FRAME THEN -- Una vez detectado nuevo FRAME
      -- Condicion de VIDEO ACTIVO
      IF (VPOS48 >= P48 + S48 AND VPOS48 < P48 + S48 + R48) AND (HPOS48 >= E48 AND HPOS48 < E48 + D48) THEN
         -- Con video activo, con cada ciclo de pixel clock se lee la senal INTENSIDAD
         -- del osciloscopio y se va almacenando en la senal dato hasta que se completan
         -- los 8 bits, mientras tanto no se accede a la RAM de VIDEO

         HPO := HPOS48 - E48;
         VPO := VPOS48 - (P48 + S48);

         IF INTENSIDAD_DIG = '1' THEN -- Solo asigna zona cuando existe pixel
         -- Se identifica la zona de pantalla
            IF                       (HPO <= ZONA0_H) AND (VPO >= R48-ZONA0_V AND VPO < R48) THEN    -- ZONA 0
               ZONA_W <= 0;
            ELSIF (HPO > ZONA2_H) AND (HPO <= ZONA1_H) AND (VPO >= R48-ZONA1_V AND VPO < R48) THEN    -- ZONA 1
               ZONA_W <= 1;
            ELSIF (HPO > ZONA0_H) AND (HPO <= ZONA2_H) AND (VPO >= R48-ZONA2_V AND VPO < R48) THEN    -- ZONA 2
               ZONA_W <= 2;
            ELSIF                    (HPO <= ZONA3_H) AND (VPO <= R48-ZONA0_V) THEN                  -- ZONA 3
               ZONA_W <= 3;
```

Figura 8.31: SALVAR_VIDEO_RAM 1

```
            ELSIF (HPO > ZONA3_H) AND (HPO <= ZONA4_H) AND (VPO <= R48-ZONA2_V) THEN                  -- ZONA 4
               ZONA_W <= 4;
            ELSIF (HPO > ZONA4_H) AND (HPO <= ZONA5_H) AND (VPO <= R48-ZONA5_V) THEN                  -- ZONA 5
               ZONA_W <= 5;
            ELSE
               ZONA_W <= 6;                                                                  -- Zona inexistente
            END IF;
         END IF;

         dato(conv_integer(conta_b)) := INTENSIDAD_DIG;  -- Almacena bit de INTENSIDAD en dato

         IF conta_b < 7 THEN
            -- Hasta que no se completan los 8 bits de INTENSIDAD...
            VIDEO_DATA_48 <= (OTHERS => 'Z');   -- Datos: alta Z
            VIDEO_WE_48 <= '1';                 -- Lectura
         ELSE
            -- Guarda en RAM la INTENSIDAD.
            VIDEO_ADD_48 <= conta_dir;     -- Direccion a escribir
            VIDEO_DATA_48 <= dato;         -- Dato de 8 bits que se guarda
            VIDEO_WE_48 <= '0';            -- Escritura

            -- Una vez que se han completado 8 bits de INTENSIDAD se guardan en RAM
            IF conta_dir < TAM_MEM-1 THEN     -- TAM_MEM = 696x810/8 = 70470. Numero ...
                                              -- de posiciones a almacenar por pantalla.
               conta_dir := conta_dir + 1;    -- Siguiente direccion
            ELSE
               conta_dir := 0;                -- Inicializa contador de direcciones RAM
            END IF;
```

Figura 8.32: SALVAR_VIDEO_RAM 2

```
        END IF;
        conta_b := conta_b + 1;          -- incrementa contador de bits (0..7)
     ELSE
        -- Condicion de VIDEO NO ACTIVO
        VIDEO_DATA_48 <= (OTHERS => 'Z');   -- Datos: Alta Z
        VIDEO_WE_48 <= '1';                 -- Lectura
        ZONA_W <= 7;                        -- Zona inexistente
     END IF;

     -- Se ponen en marcha los contadores de fila y columna
     IF (VPOS48 < 048) THEN           -- 048 = 912 filas
        VPOS48 <= VPOS48 + 1;         -- contador de filas
     ELSE
        VPOS48 <= (OTHERS => '0');
        IF (HPOS48 < A48)THEN         -- A48 = 842 columnas
           HPOS48 <= HPOS48 + 1;      -- contador de columnas
        ELSE
           HPOS48 <= (OTHERS => '0');
        END IF;
     END IF;
  ELSE
  -- Inicializar contadores de fila y columna y la direccion RAM a escribir
     HPOS48 <= (OTHERS => '0'); -- De columna
     VPOS48 <= (OTHERS => '0'); -- De fila
  END IF;
END IF;

conta_bit_48 <= conta_b;

END PROCESS SALVAR_VIDEO_RAM;
```

Figura 8.33: SALVAR_VIDEO_RAM 3

En el tercer proceso, como puede verse en la Figura 8.34, la Figura 8.35, la Figura 8.36, la Figura 8.37 y la Figura 8.38, se generan las señales de sincronismo para VGA. Además, cuando se va a representar un píxel, primero se comprueba si este está activo (bit_SRAM = '1'), en caso afirmativo, se detecta a qué zona pertenece, y una vez conocida, mediante la variable ZONA se obtiene el color. La detección, en este proceso, se realiza a través de los contadores de fila y columna HPOS_TFT y VPOS_TFT, utilizados también para generar las señales de sincronismo.

En las 4 primeras líneas del código de la Figura 8.36, es donde se produce la rotación de 90º de la pantalla, siendo este es único tratamiento que se le aplica a la imagen previamente almacenada en RAM de video.

```
LEER_VIDEO_RAM: PROCESS(CLK_TFT)

-- Constantes para centrar la pantalla
  CONSTANT HC : INTEGER := (D60 - D48) / 2;
  CONSTANT VC : INTEGER := (R60 - R48) / 2;

  VARIABLE pixel,resto: STD_LOGIC_VECTOR((LONG_DIR-1+1+3) DOWNTO 0);              -- Numero de pixel (0..810x696-1)
  VARIABLE conta_dir : STD_LOGIC_VECTOR((LONG_DIR-1) DOWNTO 0) := (OTHERS => '0'); -- direccion RAM
  VARIABLE conta_bit : STD_LOGIC_VECTOR((3-1) DOWNTO 0) := (OTHERS => '0');        -- contador de bit (8 bit/dato)
  VARIABLE bit_SRAM  : STD_LOGIC := '0';
  VARIABLE BIT_RGB   : STD_LOGIC_VECTOR((3-1) DOWNTO 0);
  VARIABLE COLOR     : INTEGER RANGE 0 TO 7 := 7;                          -- BLANCO por defecto
  VARIABLE HS,VS     : STD_LOGIC_VECTOR(10 DOWNTO 0) := (OTHERS => '0');   -- Contador de columna y fila
  VARIABLE HPO,VPO   : STD_LOGIC_VECTOR(10 DOWNTO 0) := (OTHERS => '0');   -- Contador de columna y fila desfasado
  VARIABLE ZONA      : INTEGER RANGE 0 TO 7 := 0;                          -- IZQUIERDA por defecto
```

Figura 8.34: LEER_VIDEO_RAM 1

```
BEGIN

IF rising_edge(CLK_TFT) THEN
    -- Genera señal de SINCRONISMO HORIZONTAL
   IF (HS > E60 + D60 + C60 AND HS < E60 + D60 + C60 + B60) THEN
      HSYNC <= '0';
   ELSE
      HSYNC <= '1';
   END IF;
   -- Genera señal de SINCRONISMO VERTICAL
   IF(VS > R60 + Q60 AND VS < R60 + Q60 + P60) THEN
      VSYNC <= '0';
   ELSE
      VSYNC <= '1';
   END IF;
   -- Condicion de VIDEO ACTIVO
   IF (HS >= E60 AND HS < E60 + D60) AND (VS < R60) THEN
      BLANK <= '1';                    -- Video ACTIVO
      COLOR := NEGRO;
      VIDEO_WE_TFT <= '1';             -- Lectura SRAM
      IF (HS >= E60 + HC AND HS < E60 + D60 - HC) AND (VS >= VC) AND (VS < R60 - VC) THEN
         ------------------------------------------------------------------------
         -- Obtiene la posicion de memoria RAM (conta_dir) donde se encuentra almacenado el pixel
         -- que toca enviar a pantalla y el bit (conta_bit) que corresponde dentro de esa posición
         HPO := HS - (E60 + HC);
         VPO := VS - VC;
```

Figura 8.35: LEER_VIDEO_RAM 2

```
         pixel := (HPO + 1) * SLV_R48 - (VPO + 1);     -- pixel := (HPO + 1) * R60 - (VPO + 1);
         conta_dir := pixel((LONG_DIR-1+3) DOWNTO 3); -- dirección VIDEO_RAM := pixel/8
         resto := pixel - (conta_dir & "000");        -- conta_bit <= pixel - (conta_dir * 8)
         conta_bit := resto(2 DOWNTO 0);              -- nº de bit a enviar a pantalla
      ------------------------------------------------------------------------

         -- Direccion RAM de VIDEO a leer
         VIDEO_ADD_TFT <= conv_integer(conta_dir);            -- Dirección RAM de VIDEO
         VIDEO_WE_TFT <= '1';                                 -- Lectura RAM de VIDEO
         bit_SRAM := VIDEO_DATA_TFT(conv_integer(conta_bit)); -- Lee pixel de RAM de VIDEO

         -- PANTALLA DIVIDIDA EN LAS 5 ZONAS A LAS QUE HACE REFERENCIA EL FICHERO Docu Lecroy
         -- Cuando el bit esta activo se pinta de color segun el valor convertido de INTENSIDAD
         IF bit_SRAM = '1' THEN
            -- Se identifica la zona de pantalla
            IF                       (HPO <= ZONA0_H) AND (VPO <= ZONA0_V) THEN                      -- ZONA 0
               ZONA := 0;
            ELSIF (HPO > ZONA2_H) AND (HPO <= ZONA1_H) AND (VPO <= ZONA1_V) THEN                     -- ZONA 1
               ZONA := 1;
            ELSIF (HPO > ZONA0_H) AND (HPO <= ZONA2_H) AND (VPO <= ZONA2_V) THEN                     -- ZONA 2
               ZONA := 2;
            ELSIF                    (HPO <= ZONA3_H) AND (VPO > ZONA0_V) AND (VPO <= ZONA3_V) THEN  -- ZONA 3
               ZONA := 3;
            ELSIF (HPO > ZONA0_H) AND (HPO <= ZONA4_H) AND (VPO > ZONA2_V) AND (VPO <= ZONA4_V) THEN  -- ZONA 4
               ZONA := 4;
            ELSIF (HPO > ZONA4_H) AND (HPO <= ZONA5_H) AND (VPO > ZONA1_V) AND (VPO <= ZONA5_V) THEN  -- ZONA 5
               ZONA := 5;
```

Figura 8.36: LEER_VIDEO_RAM 3

```
            ELSE
               ZONA := 6;                          -- Zona inexistente
            END IF;

            AD_RAM_ADDR_TFT <= ZONA;                                  -- Dirección
            COLOR := conv_integer(AD_RAM_DOUT_TFT(7 downto 5));       -- Valor convertido
            COLOR := AMARILLO;
         ELSE
            COLOR := NEGRO;
            ZONA := 7;                                                -- Zona inexistente
         END IF;
      END IF;
   ELSE
   -- Video NO ACTIVO: Sin acceso a memoria RAM de VIDEO
      BLANK <= '0';                          -- Video NO ACTIVO
      COLOR := NEGRO;
      ZONA := 7;                             -- Zona inexistente
      VIDEO_WE_TFT <= '1';                   -- Lectura RAM de VIDEO
   END IF;

   -- Genera los contadores de columna HS y fila VS
   IF (HS < A60) THEN
      HS := HS + 1;     -- Contador de columnas
   ELSE
      HS := (OTHERS => '0');
      IF (VS < O60) THEN
         VS := VS + 1;  -- Contador de filas
```

Figura 8.37: LEER_VIDEO_RAM 4

```
      ELSE
         VS := (OTHERS => '0');
      END IF;
   END IF;
END IF;

-- Se descompone el COLOR del pixel en sus componentes RGB
BIT_RGB := conv_std_logic_vector(COLOR,3);

-- A continuación, descompone color de pixel en sus componentes RGB
VR <= BIT_RGB(2); -- Rojo (MSB según la tabla que se adjunta)
VG <= BIT_RGB(1); -- Verde
VB <= BIT_RGB(0); -- Azul (LSB)

-- Contadores de fila y columna para leer de RAM de VIDEO y enviar a pantalla LCD-TFT
HPOS_TFT <= HPO;
VPOS_TFT <= VPO;

AD_RAM_WE_TFT <= '1';    -- Lectura del valor convertido

END PROCESS LEER_VIDEO_RAM;
```

Figura 8.38: LEER_VIDEO_RAM 5

A modo de posible aclaración, al igual que sucede para leer y escribir en RAM de video, donde se emplean relojes de 48MHz y 40MHz, a la hora de guardar y leer los valores convertidos de los píxeles de una zona, se emplean las mismas frecuencias de reloj, 48MHz para escribir en RAM y 40MHz para leer de RAM.

8.3.5 ADC9057.vhd

Esta entidad, realiza la conversión analógica/digital, gracias al conversor AD9057 y es usada en accede_VIDEO_RAM.vhd. Para ello, se digitaliza el primer píxel de cada zona. Cuando la conversión ha finalizado, escribe el resultado en una zona específica de la RAM que no se corresponde con la zona de memoria RAM reservada para almacenar las pantallas, es decir, la RAM de VIDEO. Puesto que la conversión se realiza por zonas, y en una misma zona todos los píxeles son del mismo color, marca esa zona como convertida, de forma que no se convierten más píxeles de dicha zona.

El proceso anteriormente explicado, se ha resuelto con una máquina de estados. Como toda máquina de estados de Moore en VHDL, se puede modelar mediante dos procesos: uno secuencial, que es idéntico en todas las máquinas de estado, donde se define el siguiente estado con cada flanco activo de reloj y otro combinacional, donde para cada estado, se definen las salidas y las transiciones.

```
ARCHITECTURE my_rtl OF ADC9057 IS

-- Declaración de estados de la maquina de MOORE
TYPE ESTADO IS (COMPRUEBA, CICLO1, CICLO2, FIN_CONV);
-- Declaración señal de tipo estado
SIGNAL ACTUAL, SIGUIENTE : ESTADO := COMPRUEBA;
```

Figura 8.39: Definición de estados

Cada vez que es detectada la nueva pantalla, se debe inicializar marca_ZONA, utilizada para ir marcando el primer píxel detectado de una nueva zona, que es el único que se digitaliza de dicha zona. Como se ha mencionado, esto se realiza cuando la nueva pantalla es detectada y, por tanto, HPOS_48=0 y VPOS_48=0. Además, en este instante el conversor se habilita (EOC=TRUE), quedando listo para realizar la primera conversión. Esta se produce por primera vez cuando se lee el primer bit a ‘1’ de la RAM de video.

El proceso secuencial se corresponde a las primeras líneas de código de la Figura 8.40.

```
BEGIN

-- Proceso secuencial: asignación de estado
   SEQ: PROCESS (CLK48)
      BEGIN
         IF rising_edge (CLK48) THEN
            ACTUAL <= SIGUIENTE;
         END IF;
      END PROCESS SEQ;

-- Proceso combinacional: salida y estado siguiente
   COMB: PROCESS (ACTUAL, ZONA, DATA, marca_ZONA, EOC, VPOS, HPOS, FRAME)
   VARIABLE var_ZONA: INTEGER RANGE 0 TO 7;
   BEGIN
      IF FRAME AND (HPOS = 0) AND (VPOS = 0) THEN
         -- Al comienzo de cada FRAME en que toca salvar a SRAM ...
         -- se inicializa el marcador de cada zona marca_ZONA y se activa
         -- la posibilidad de usar en conversor
         marca_ZONA <= (OTHERS => '0');      -- Inicializa marcador de zona
         AD_RAM_WE_48 <= '1';                -- Se lee
         EOC <= TRUE;                        -- Fin de conversión activo: necesario para la siguiente conversión
```

Figura 8.40: Procesos de la máquina de estados 1

```
      ELSE
         -- Ahora se realiza la conversión con el ADC9057, esta es ...
         -- la parte puramente combinacional de la maquina de MOORE,
         -- donde hay que definir los distintos estados
         CASE ACTUAL IS
            WHEN COMPRUEBA =>
               -- Permanece en este estado hasta que se encuentre un pixel
               -- de una zona en la que todavia no se haya convertido su INTENSIDAD
               -- y no se este en el proceso de conversión de un pixel anterior
               AD_RAM_WE_48 <= '1';          -- Lectura
               var_ZONA := ZONA;             -- Se guarda ZONA en este instante
               AD_RAM_ADDR_48 <= ZONA;       -- Dirección de escritura = el valor de ZONA
               IF (marca_ZONA(var_ZONA) = '0') AND EOC THEN
                  SIGUIENTE <= CICLO1;
               END IF;
            WHEN CICLO1 =>
               marca_ZONA(var_ZONA) <= '1';  -- Se marca la zona
               EOC <= FALSE;                 -- Se inhabilita el conversor
               SIGUIENTE <= CICLO2;
            WHEN CICLO2 =>
               SIGUIENTE <= FIN_CONV;
            WHEN OTHERS =>
               EOC <= TRUE;                  -- Fin de conversion
               AD_RAM_WE_48 <= '0';          -- Escritura
               AD_RAM_DIN_48 <= DATA;        -- Escribe valor convertido en RAM de FPGA
               SIGUIENTE <= COMPRUEBA;       -- Vuelve al comienzo, para la siguiente conversión
         END CASE;
      END IF;
   END PROCESS COMB;
END my_rtl;
```

Figura 8.41: Procesos de la máquina de estados 2

El proceso combinacional, como puede verse en la Figura 8.40 y la Figura 8.41, está formado por los siguientes estados, definidos en la Figura 8.39:

- Estado COMPRUEBA

  Es el estado inicial. Se mantiene en este estado hasta que se detecte un píxel de una zona que no haya sido convertida. Cuando se da el caso, pasa al siguiente ciclo de la máquina de estado.

  Para este estado, la salida es la dirección donde se encuentra el píxel y la transición es CICLO1.

- Estado CICLO1

  En este ciclo se inhabilita el conversor (EOC=FALSE), para indicar que el conversor AD9057 está convirtiendo un valor y, además, se marca la zona.

  Para este estado, la transición es CICLO2.

- Estado CICLO2

  Este estado solo sirve para esperar un ciclo adicional.

  Para este estado, la transición es FIN_CONV.

- Estado FIN_CONV

  Es el último ciclo, en él, se prepara la máquina de estado para una nueva conversión haciendo EOC=TRUE y se escribe el valor resultante de la conversión DATA en la RAM del conversor en la dirección ZONA.

  Para este estado, la transición es COMPRUEBA.

8.3.6 video_ram_dual_port.vhd

Esta entidad, es la encargada de definir el acceso a la memoria RAM de video.

8.3.7 zona_dual_port.vhd

Esta entidad, es la encargada de definir el acceso a la memoria RAM del conversor AD9057.

8.3.8 Interface_FDPLink_18bits.vhd

Esta entidad, como puede verse en la Figura 8.42, es la encargada de generar los canales de datos que son enviados al driver convertidor de niveles TTL a LVDS (Ds90c031), y que en última instancia forman las señales que se envían por el puerto LVDS.

Es capaz, a partir de las componentes R, G y B de cada píxel, el indicador de video activo, los sincronismos horizontal y vertical, y un reloj de 280MHz (7 × 40Mhz), generar los tres canales de datos y el canal de reloj que se envían al driver convertidor de niveles TTL a LVDS (Ds90c031), haciendo uso de la entidad serializador.vhd.

```
ARCHITECTURE FPDLink_interface_18bits OF Interface_FPDLink_18bits IS
    SIGNAL dataA, dataB, dataC, data_clk: STD_LOGIC_VECTOR(6 DOWNTO 0);
    COMPONENT serializador IS
       PORT( clk280: IN STD_LOGIC;
             din: IN STD_LOGIC_VECTOR(6 DOWNTO 0);
             dout: OUT STD_LOGIC);
    END COMPONENT;
 BEGIN
 --------------Mapeador--------------------------------------
    dataA <= G(0) & R(5 DOWNTO 0);
    dataB <= B(1 DOWNTO 0) & G(5 DOWNTO 1);
    dataC <= video_activo & Vsync & Hsync & B(5 DOWNTO 2);
    data_clk <= "1100011";
 --------------Serializador----------------------------------
    serialA : serializador PORT MAP (clk280, dataA, RO0);
    serialB : serializador PORT MAP (clk280, dataB, RO1);
    serialC : serializador PORT MAP (clk280, dataC, RO2);
    serial_clk : serializador PORT MAP (clk280, data_clk, CLK);
END FPDLink_interface_18bits;
```

Figura 8.42: FPDLink

8.3.9 Serializador.vhd

Esta entidad, como puede verse en la Figura 8.43, que depende jerárquicamente de la entidad Interface_FPDLink_18bits.vhd, realiza la conversión de paralelo a serie. Para ello, requiere un reloj de 280MHz. Constituye básicamente un registro de desplazamiento de 7 bits. Envía siempre primero el bit más significativo de la palabra de 7 bits que convierte.

```
ARCHITECTURE serializador OF serializador IS
    SIGNAL internal: STD_LOGIC_VECTOR(6 DOWNTO 0);
BEGIN
    PROCESS(clk280)
       VARIABLE count: INTEGER RANGE 0 TO 7 := 0;
    BEGIN
       IF rising_edge(clk280) THEN
          count := count + 1;
          IF (count = 6) THEN
             internal <= din;           -- Lee siguiente palabra
          ELSIF (count = 7) THEN
             count := 0;                -- Inicializa para enviar MSB primero
          END IF;
          dout <= internal(6-count);    -- MSB primero
       END IF;
    END PROCESS;
 END serializador;
```

Figura 8.43: Serializador

8.4 Conclusiones y Líneas Futuras

8.4.1 Conclusiones

El objetivo principal del proyecto que se planteó es el de conseguir representar fielmente la imagen que se envía al tubo de rayos catódicos del osciloscopio LeCroy de la serie 93XX en una pantalla TFT-LCD. Con ese fin, se planteó una metodología con la que probar distintas alternativas para satisfacer el objeto del proyecto, ordenadas en función de la complejidad que se pensaba inicialmente que dichas alternativas presentaban.

Según se fue desarrollando el proyecto, se vio que la primera alternativa ponía en riesgo la integridad del osciloscopio así que no fue llevada a cabo. La segunda alternativa se llevó a cabo y funciona satisfactoriamente cumpliendo todos los objetivos, mientras que la tercera funciona, puesto que la imagen se representa adecuadamente en la pantalla TFT-LCD, pero el color que proporciona el conversor no es el esperado. En ambos casos, si se ha conseguido transmitir la información a la pantalla TFT-LCD utilizando tanto una conexión directa entre la pantalla y la placa controladora diseñada mediante LVDS como mediante la placa controladora de video intermedia adquirida, haciendo uso del conector VGA del sistema de desarrollo.

Tras haber realizado todo el trabajo anterior, se han redactado una serie de documentos, como son la memoria, los planos y el anexo, donde se recoge toda la información necesaria para replicar el proyecto tanto en el apartado hardware como software.

Por último, se han elaborado unos presupuestos, a modo orientativo, del precio que tendría replicar el proyecto.

8.4.2 Líneas futuras

Para futuros desarrollos o ampliaciones, se propone:

- Representar adecuadamente el color haciendo uso del conversor, ya que no ha llegado a funcionar satisfactoriamente.
- Buscar otro sistema de desarrollo basado en FPGA más barato que cumpla con los requerimientos del proyecto.
- Diseñar la placa controladora haciendo uso de herramientas que permitan el diseño de pistas específicas para LVDS, y así, conseguir mayores velocidades.
- Adquirir un cableado mejor o hacer uso de un conector más fiable.
- Conseguir unos medios de fabricación con capacidad para fabricar una placa de circuito impreso multicapa: que es la que se necesitaría para contener todos los elementos necesarios, como son: la FPGA, los medios para trabajar con ella y los elementos instalados en la placa controladora diseñada.

9 ORDEN DE PRIORIDAD ENTRE DOCUMENTOS

El orden de prioridad a tener en cuenta, si se encontraran discrepancias entre los diferentes documentos básicos que componen este proyecto, es el siguiente:

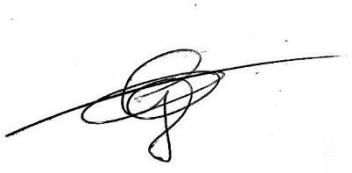
- Memoria
- Planos
- Anexos
- Presupuestos

Anexos

Controladora para pantalla TFT-LCD 800x600 basada en FPGA para osciloscopios LeCroy 93XX

Alumno: David Godoy Chiclana

Tutor: Prof. D. Gregorio Godoy Vilches



Departamento: Ingeniería Electrónica y Automática

Septiembre, 2017

ÍNDICE DEL ANEXO

1 ANEXO 1

1.1 vga696810.vhd

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
use ieee.std_logic_unsigned.all;

ENTITY VGA696810 IS
GENERIC (ANCHO_DIR      : INTEGER := 17;      -- Ancho del bus de direcciones
                        ANCHO_DATO      : INTEGER := 8;       -- Ancho del bus de
datos

        -- CONSTANTES PARA LA PANTALLA DE 696x810 a 48MHz
        A48 : INTEGER := 841;       -- Ciclo de Sincronización Horizontal (B + C + D +
E): 15.998 ms
  B48 : INTEGER := 22;       -- Pulso de Sincronización Horizontal: 0.418 ms

        C48 : INTEGER := 4;         -- Front Porch Horizontal: 0.076 ms

        D48 : INTEGER := 810;       -- Video Activo Horizontal: 15.39 ms
        E48 : INTEGER := 6;         -- Back Porch Horizontal: 0.114 ms

        O48 : INTEGER := 911;       -- Ciclo de Sincronización Vertical (P + Q + R + S):
19.0 us
        P48 : INTEGER := 136;       -- Pulso de sincronización Vertical: 2.83 us

        Q48 : INTEGER := 0;         -- Front Porch Vertical: 0.0 us

        R48 : INTEGER := 696;       -- Video Activo Vertical: 14.5 us

        S48 : INTEGER := 80;        -- Back Porch Vertical: 1.6 us



        --  CONSTANTES PARA LA PANTALLA DE 1500X860 a 40MHz
        A60 : INTEGER := 1499;      -- Ciclo de Sincronización Horizontal (B + C + D +
E): 31.77 us
```

```
        B60 : INTEGER := 100;       -- Pulso de Sincronización Horizontal: 3.77 us

        C60 : INTEGER := 60;                -- Front Porch Horizontal: 1.89 us

        D60 : INTEGER := 1280;      -- Video Activo Horizontal: 25.17 us

        E60 : INTEGER := 60;                -- Back Porch Horizontal: 0.94 us

        O60 : INTEGER := 859;       -- Ciclo de Sincronización Vertical (P + Q + R + S):
16.6 ms
        P60 : INTEGER := 20;                -- Pulso de sincronización Vertical: 64 us

        Q60 : INTEGER := 20;                -- Front Porch Vertical: 1.02 ms

        R60 : INTEGER := 800;       -- Video Activo Vertical: 15.25 ms

        S60 : INTEGER := 20);       -- Back Porch Vertical: 0.35 ms


PORT(
        -- Relojes de 50MHz a partir de los cuales se generan el resto de relojes
        CLOCK_50,CLOCK2_50: IN STD_LOGIC;

        --      Sirve para detectar el comienzo del video activo horizontal
        FRAME                                   : INOUT BOOLEAN := TRUE;

        -- Señales procedentes del osciloscopio
        VIDEO_RAM_CLK_48        : INOUT STD_LOGIC;
        HS_48,VS_48             : INOUT STD_LOGIC;
        INTENSIDAD_DIG          : INOUT STD_LOGIC;

        -- Señales de sincronismo a 40MHz para escribir en la pantalla TFT
        VGA_HS,VGA_VS           : INOUT STD_LOGIC;
        VGA_R,VGA_G,VGA_B : INOUT STD_LOGIC_VECTOR (3 DOWNTO 0);

        -- Señales para leer la señal del osciloscopio y escribir en RAM de VIDEO
```

```
        HPOS48, VPOS48              : INOUT STD_LOGIC_VECTOR (10 DOWNTO 0)
:= (OTHERS => '0');   -- Contadores de columna y fila
        conta_bit_48            : INOUT STD_LOGIC_VECTOR(3-1 DOWNTO 0) :=
(OTHERS => '0');      -- Contador de bit (16 bit/dato)
        VIDEO_RAM_ADDR_48       : INOUT natural RANGE 0 TO 2**ANCHO_DIR-1;
        VIDEO_RAM_DIN_48        : INOUT STD_LOGIC_VECTOR ((ANCHO_DATO-
1) DOWNTO 0);
        VIDEO_RAM_WE_48         : INOUT STD_LOGIC := '1';

-- Para almacenar el valor convertido del A/D
        HPOS_TFT, VPOS_TFT: INOUT STD_LOGIC_VECTOR (10 downto 0) :=
(OTHERS => '0');      -- Contadores de fila y columna
  conta_bit_TFT                 : INOUT STD_LOGIC_VECTOR (3-1 DOWNTO 0) :=
(OTHERS => '0');      -- Contador de bit para la TFT
        VIDEO_RAM_CLK_TFT       : INOUT STD_LOGIC;
        VIDEO_RAM_ADDR_TFT: INOUT NATURAL RANGE 0 TO 2**(ANCHO_DIR)-
1;
        VIDEO_RAM_DOUT_TFT: INOUT STD_LOGIC_VECTOR((ANCHO_DATO-1)
DOWNTO 0);
        VIDEO_RAM_WE_TFT        : INOUT STD_LOGIC;

-- Señales para acceder a la RAM del AD en escritura
        AD_RAM_WE_48            : INOUT std_logic := '1';
        AD_RAM_ADDR_48          : INOUT natural RANGE 0 TO 7;
        AD_RAM_DIN_48           : INOUT std_logic_vector(7 DOWNTO 0);

-- Señales para leer la INTENSIDAD de la RAM de la FPGA
        AD_RAM_WE_TFT           : INOUT std_logic := '1';
        AD_RAM_ADDR_TFT         : INOUT natural RANGE 0 TO 7;
        AD_RAM_DOUT_TFT         : INOUT std_logic_vector(7 DOWNTO 0);

-- Señales del conversor AD9057
        GPIO_1_D                : INOUT STD_LOGIC_VECTOR (35 DOWNTO 0);
        -- Conector placa
        DATA                            : INOUT STD_LOGIC_VECTOR (7
DOWNTO 0);          -- Señal DATA del A/D
```

```
  marca_ZONA                    : INOUT STD_LOGIC_VECTOR (7 DOWNTO
0):=(OTHERS => '0');
        ZONA_W                  : INOUT INTEGER RANGE 0 TO 7 := 0;
                -- Zona de pantalla para el A/D
        EOC                     : INOUT BOOLEAN := TRUE);

END VGA696810;

ARCHITECTURE MAIN OF VGA696810 IS

-- Componentes LVDS
SIGNAL RO0: STD_LOGIC;
        -- Canal 0 LVDS
SIGNAL RO1: STD_LOGIC;
        -- Canal 1 LVDS
SIGNAL RO2: STD_LOGIC;
        -- Canal 2 LVDS
SIGNAL CLK: STD_LOGIC;
        -- Canal reloj LVDS
SIGNAL R, G, B: STD_LOGIC_VECTOR(5 DOWNTO 0); -- Componentes R,G,B de 6
bits
SIGNAL clk40: STD_LOGIC;                                        -- Reloj
de 40MHz
SIGNAL clk280: STD_LOGIC;
        -- Reloj de 280MHz

SIGNAL
VIDEO_RAM_CLK_FPGA,VIDEO_RAM_CLK_OSCILOSCOPIO:STD_LOGIC;  --
Sirven para seleccionar el reloj
SIGNAL VR,VG,VB: STD_LOGIC;                                     --
Componentes del color de 1 bit
SIGNAL video_activo : STD_LOGIC;                                --
Señal de video activo (dena)

-- Señales para acceder a la RAM del conversor, para guardar/leer el valor convertido
del A/D
```

```
SIGNAL AD_RAM_DIN_TFT: STD_LOGIC_VECTOR((ANCHO_DATO-1) DOWNTO
0);
SIGNAL AD_RAM_DOUT_48: STD_LOGIC_VECTOR((ANCHO_DATO-1) DOWNTO
0);
SIGNAL VIDEO_RAM_DOUT_48: STD_LOGIC_VECTOR((ANCHO_DATO-1)
DOWNTO 0);
SIGNAL VIDEO_RAM_CLK: STD_LOGIC;
SIGNAL VIDEO_RAM_DIN_TFT: STD_LOGIC_VECTOR((ANCHO_DATO-1)
DOWNTO 0);
SIGNAL CLK_TFT,VIDEO_RAM_CLK_70: STD_LOGIC;

-- Se define el PLL que genera pixel clock de 48MHz y 70MHz
COMPONENT PLL_48_70_MHz IS
        PORT (
                refclk   : IN  std_logic := '0';   -- refclk.clk
                rst      : IN  std_logic := '0';    -- reset.reset
                outclk_0 : OUT std_logic;                       -- outclk0.clk
                outclk_1 : OUT std_logic                        -- outclk1.clk
        );
END COMPONENT PLL_48_70_MHz;

-- Se define el PLL que genera pixel clock de 40MHz y 280MHz
COMPONENT PLL_40_280MHz IS
        PORT (
                refclk   : IN  std_logic := '0';   -- refclk.clk
                rst      : IN  std_logic := '0';    -- reset.reset
                outclk_0 : OUT std_logic;                       -- outclk0.clk
                outclk_1 : OUt std_logic                        -- outclk1.clk
        );
END COMPONENT PLL_40_280MHz;

-- Se definen los componentes de las señales para transmitir LVDS
COMPONENT Interface_FPDLink_18bits IS
        PORT(
                clk280 : IN STD_LOGIC;
                Hsync, Vsync, video_activo: IN STD_LOGIC;
                R, G, B          : IN STD_LOGIC_VECTOR(5 DOWNTO 0);
```

```
                RO0             : OUT STD_LOGIC;
                RO1             : OUT STD_LOGIC;
                RO2             : OUT STD_LOGIC;
                CLK             : OUT STD_LOGIC);
END COMPONENT;

-- Se define accede_VIDEO_RAM
COMPONENT accede_VIDEO_RAM IS
GENERIC(
        LONG_DIR                : INTEGER;    -- Tamaño del bus de direcciones
        ANCHO_DATO              : INTEGER;    -- Tamaño del bus de datos

 --  CONSTANTES PARA ESCRIBIR EN LA PANTALLA DE 696X810 48MHz Y LA DE
1500x860 a 40MHz
  A48,B48,C48,D48,E48,O48,P48,Q48,R48,S48,
        A60,B60,C60,D60,E60,O60,P60,Q60,R60,S60 : INTEGER);

PORT(
        -- Escribir de RAM de VIDEO
        CLK48                   : IN STD_LOGIC;                         -- Pixel
clock (señal del osciloscopio)
        VGA_H,VGA_V             : IN STD_LOGIC;                         --
Sincronismo vertical del osciloscopio
        INTENSIDAD_DIG    : IN STD_LOGIC;                       -- Intensidad
DIGITAL del osciloscopio
        FRAME                           : INOUT BOOLEAN := FALSE;       --
Detectado FRAME

  HPOS48, VPOS48  : INOUT STD_LOGIC_VECTOR (10 DOWNTO 0) := (OTHERS
=> '0'); -- Contadores de columna y fila
        conta_bit_48   : INOUT STD_LOGIC_VECTOR(3-1 DOWNTO 0) := (OTHERS
=> '0'); -- Contador de bit (16 bit/dato)

        -- Señales para escribir en la RAM de la FPGA (Lee de la pantalla del
osciloscopio y lo salvara a RAM)
        VIDEO_ADD_48       : OUT NATURAL RANGE 0 TO (2**LONG_DIR)-1;
                -- Direcciones RAM
```

```
        VIDEO_DATA_48     : OUT STD_LOGIC_VECTOR ((ANCHO_DATO-1)
DOWNTO 0);  -- Datos in RAM
        VIDEO_WE_48               : OUT STD_LOGIC := '1';
                                        -- R/W, Output Enable y Chip Enable de
RAM

        -- Señales para acceder a la RAM DE VIDEO en lectura
        CLK_TFT                   : IN STD_LOGIC;                   -- Pixel
clock (señal para la TFT)
        HSYNC,VSYNC               : OUT STD_LOGIC;                  --
Sincronismo horizontal, vertical, video activo para generar
        VR,VG,VB                  : OUT STD_LOGIC;                  --
Componentes RGB generadas para la LCD-TFT
        HPOS_TFT,VPOS_TFT: INOUT STD_LOGIC_VECTOR (10 DOWNTO 0) :=
(OTHERS => '0');      -- Contadores de fila y columna
  BLANK                           : OUT STD_LOGIC;                  --
Video activo

        -- Señales para leer de la memoria VIDEO_RAM
        VIDEO_ADD_TFT     : OUT NATURAL RANGE 0 TO (2**LONG_DIR)-1;
        -- Direcciones VIDEO_RAM
        VIDEO_DATA_TFT    : IN STD_LOGIC_VECTOR (ANCHO_DATO-1 DOWNTO
0);       -- Datos VIDEO_RAM
        VIDEO_WE_TFT      : OUT STD_LOGIC;
                                        -- R/W, output enable, chip enable

        -- Señales para guardar la INTENSIDAD en RAM de la FPGA
        AD_RAM_WE_48      : OUT std_logic := '1';
        AD_RAM_ADDR_48  : OUT natural RANGE 0 TO 7;
        AD_RAM_DIN_48     : OUT std_logic_vector(7 DOWNTO 0);

        -- Señales para leer la INTENSIDAD de la RAM de la FPGA
        AD_RAM_WE_TFT   : OUT std_logic := '1';
        AD_RAM_ADDR_TFT: OUT natural RANGE 0 TO 7;
        AD_RAM_DOUT_TFT: IN std_logic_vector(7 DOWNTO 0);

        -- Señales del AD9057
```

```
        DATA                    : IN STD_LOGIC_VECTOR (7 DOWNTO 0);
                                        -- Valor convertido del A/D
        marca_ZONA      : INOUT std_logic_vector(7 DOWNTO 0) := (OTHERS =>
'0'); -- Marca la zona de pantalla
        ZONA_W                  : INOUT INTEGER RANGE 0 TO 7 := 0;
                                        -- Zona de pantalla para el A/D
        EOC                     : INOUT BOOLEAN := TRUE);
                                                -- Indica el fin de conversión

END COMPONENT accede_VIDEO_RAM;

-- Memoria DUAL PORT para guardar resultado del A/D
COMPONENT zona_dual_port IS

        GENERIC
        (
                ADDR_WIDTH : natural := 3;
                DATA_WIDTH : natural := 8
        );

        PORT
        (
                clk_a           : IN std_logic;
                clk_b           : IN std_logic;
                addr_a: IN natural RANGE 0 TO 2**ADDR_WIDTH - 1;
                addr_b: IN natural RANGE 0 TO 2**ADDR_WIDTH - 1;
                data_a: IN std_logic_vector ((DATA_WIDTH-1) DOWNTO 0);
                data_b: IN std_logic_vector ((DATA_WIDTH-1) DOWNTO 0);
                we_a            : IN std_logic := '1';
                we_b            : IN std_logic := '1';
                q_a             : OUT std_logic_vector ((DATA_WIDTH-1) DOWNTO 0);
                q_b             : OUT std_logic_vector ((DATA_WIDTH-1) DOWNTO 0)
        );

END COMPONENT;

-- Memoria de RAM de VIDEO
```

```
COMPONENT video_ram_dual_port IZ
	GENERIC
	(
		ADDR_WIDTH : natural;
		DATA_WIDTH : natural
	);
	PORT
	(
		clk_a		: IN std_logic;
		clk_b		: IN std_logic;
		addr_a	: IN natural RANGE 0 TO 2**ADDR_WIDTH - 1;
		addr_b	: IN natural RANGE 0 TO 2**ADDR_WIDTH - 1;
		data_a	: IN std_logic_vector (DATA_WIDTH-1 DOWNTO 0);
		data_b	: IN std_logic_vector (DATA_WIDTH-1 DOWNTO 0);
		we_a		: IN std_logic := '1';
		we_b		: IN std_logic := '1';
		q_a		: OUT std_logic_vector (DATA_WIDTH-1 DOWNTO 0);
		q_b		: OUT std_logic_vector (DATA_WIDTH-1 DOWNTO 0)
	);

END COMPONENT;

-- Arquitectura
BEGIN
	-- Obtiene pixel clock de 48MHz y 70.55MHz a partir de un reloj de 50MHz (usa uno de los 4 PLL)
	C1: PLL_48_70_MHz PORT MAP (CLOCK_50,'0',VIDEO_RAM_CLK_FPGA,VIDEO_RAM_CLK_70); -- Obtiene pixel clock de 48 y 40MHz

	-- Obtiene pixel clock de 40MHz y 280MHz a partir de un reloj de 50MHz (usa uno de los 4 PLL)
	C2: PLL_40_280MHz PORT MAP (CLOCK2_50,'0',clk40,clk280); -- Obtiene pixel clock de 48MHz y 280MHz

	-- FPDLink Interface
	fpd_link: Interface_FPDLink_18bits PORT MAP(
```

```
        clk280, VGA_HS, VGA_VS, video_activo, R, G, B,
        RO0, RO1, RO2, CLK);

    -- Salva a VIDEO RAM la pantalla (operacion de ESCRITURA en RAM)
    C3: accede_VIDEO_RAM GENERIC
MAP(ANCHO_DIR,ANCHO_DATO,A48,B48,C48,D48,E48,O48,P48,Q48,

    R48,S48,A60,B60,C60,D60,E60,O60,P60,Q60,R60,S60)
                                PORT MAP
(VIDEO_RAM_CLK_48,HS_48,VS_48,INTENSIDAD_DIG,FRAME,HPOS48,VPOS48,c
onta_bit_48,

    VIDEO_RAM_ADDR_48,VIDEO_RAM_DIN_48,VIDEO_RAM_WE_48,VIDEO_
RAM_CLK_TFT,VGA_HS,VGA_VS,

    VR,VG,VB,HPOS_TFT,VPOS_TFT,video_activo,VIDEO_RAM_ADDR_TFT,VID
EO_RAM_DOUT_TFT,VIDEO_RAM_WE_TFT,

    AD_RAM_WE_48,AD_RAM_ADDR_48,AD_RAM_DIN_48,AD_RAM_WE_TFT,
AD_RAM_ADDR_TFT,AD_RAM_DOUT_TFT,

    DATA,marca_ZONA,ZONA_W,EOC);

    -- Memoria RAM de FPGA para acceder a la MEMORIA DE VIDEO;
    C4: video_ram_dual_port GENERIC MAP (ANCHO_DIR,ANCHO_DATO)
                                PORT MAP
(VIDEO_RAM_CLK_48,VIDEO_RAM_CLK_TFT,VIDEO_RAM_ADDR_48,VIDEO_RA
M_ADDR_TFT,VIDEO_RAM_DIN_48,

    VIDEO_RAM_DIN_TFT,VIDEO_RAM_WE_48,VIDEO_RAM_WE_TFT,VIDEO_
RAM_DOUT_48,VIDEO_RAM_DOUT_TFT);

    -- Memoria RAM de FPGA para guardar la conversion del A/D;
    C5: zona_dual_port GENERIC MAP (ANCHO_DIR,ANCHO_DATO)
                                PORT MAP
(VIDEO_RAM_CLK_48,VIDEO_RAM_CLK_TFT,AD_RAM_ADDR_48,AD_RAM_ADD
R_TFT,AD_RAM_DIN_48,
```

```
AD_RAM_DIN_TFT,AD_RAM_WE_48,AD_RAM_WE_TFT,AD_RAM_DOUT_4
8,AD_RAM_DOUT_TFT);

DATA <= GPIO_1_D(3) & GPIO_1_D(5) & GPIO_1_D(7) & GPIO_1_D(9)
& GPIO_1_D(11) &    GPIO_1_D(13) & GPIO_1_D(15) & GPIO_1_D(17);

INTENSIDAD_DIG <= GPIO_1_D(33);              -- Señal de intensidad del
osciloscopio
HS_48 <= GPIO_1_D(34);                       -- Sincronimo
horizontal del osciloscopio
VS_48 <= GPIO_1_D(35);                       -- Sincronismo
vertical del osciloscopio

GPIO_1_D(2) <= VIDEO_RAM_CLK_FPGA;                          -- Reloj
48 MHz procedente de la FPGA que se envia a la placa
VIDEO_RAM_CLK_OSCILOSCOPIO <= GPIO_1_D(4);   -- Reloj 48 MHz
procedente del OSCILOSCOPIO que se envía a la FPFA

                -- desde el GPIO_1

-- Permite seleccionar vía software si se va autilizar el reloj del osciloscopio o el
que se genera
--VIDEO_RAM_CLK_48 <= VIDEO_RAM_CLK_OSCILOSCOPIO;        --
Descomentar si usamos el reloj del OSCILOSCOPIO
VIDEO_RAM_CLK_48 <= VIDEO_RAM_CLK_FPGA;                 --
Descomentar si se usa el reloj de la FPGA


VIDEO_RAM_CLK_TFT <= clk40;          -- Reloj de 40MHz

GPIO_1_D(12) <= CLK;                 -- CLK+
GPIO_1_D(10) <= RO2;                 -- RO2+
GPIO_1_D(8) <= RO0;                  -- RO0+
GPIO_1_D(6) <= RO1;                  -- RO1+

R <= (OTHERS => VR);                 -- Asigna componente R 6 bits
```

```
        G <= (OTHERS => VG);            -- Asigna componente G 6 bits
        B <= (OTHERS => VB);            -- Asigna componente B 6 bits

        VGA_R <= (OTHERS => VR);        -- Asigna componente R 4 bits
        VGA_G <= (OTHERS => VG);        -- Asigna componente G 4 bits
        VGA_B <= (OTHERS => VB);        -- Asigna componente B 4 bits

END MAIN;
```

1.2 accede_VIDEO_RAM.vhd

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
use ieee.std_logic_arith.all;

ENTITY accede_VIDEO_RAM IS

GENERIC(
        LONG_DIR                : INTEGER;    -- Tamaño del bus de direcciones
        ANCHO_DATO              : INTEGER;    -- Tamaño del bus de datos

        --  CONSTANTES PARA ESCRIBIR EN LA PANTALLA DE 696X810 A 48MHz
Y LA DE 1500X860 A 40MHz
  A48,B48,C48,D48,E48,O48,P48,Q48,R48,S48,
        A60,B60,C60,D60,E60,O60,P60,Q60,R60,S60 : INTEGER);



PORT(
        -- Escribir en RAM de VIDEO
        CLK48                   : IN STD_LOGIC;                         -- Pixel
clock (señal del osciloscopio)
        VGA_H,VGA_V             : IN STD_LOGIC;                         --
Sincronismo vertical del osciloscopio
        INTENSIDAD_DIG    : IN STD_LOGIC;                       -- Intensidad
DIGITAL del osciloscopio
        FRAME                           : INOUT BOOLEAN := FALSE;       --
Detectado FRAME

        -- Junto con el pixel clock de 48MHz, solo se necesita el sincronismo horizontal,
        -- para acceder a la memoria para salvar la pantalla de 696x810 en RAM.
        -- El resto de señales de esta entidad son las que se usan para leer la RAM.

  HPOS48, VPOS48 : INOUT STD_LOGIC_VECTOR (10 DOWNTO 0) := (OTHERS
=> '0'); -- Contadores de columna y fila
        conta_bit_48  : INOUT STD_LOGIC_VECTOR(3-1 DOWNTO 0) := (OTHERS
=> '0'); -- Contador de bit (16 bit/dato)
```

```
        -- Señales para escribir en la RAM de la FPGA (Lee de la pantalla del
osciloscopio y lo salvara a RAM)
        VIDEO_ADD_48       : OUT NATURAL RANGE 0 TO (2**LONG_DIR)-1;
                -- Direcciones RAM
        VIDEO_DATA_48      : OUT STD_LOGIC_VECTOR ((ANCHO_DATO-1)
DOWNTO 0);  -- Datos in RAM
        VIDEO_WE_48                : OUT STD_LOGIC := '1';
                                        -- R/W, Output Enable y Chip Enable de
RAM

        -- Señales para acceder a la RAM DE VIDEO en lectura
        CLK_TFT                    : IN STD_LOGIC;                          -- Pixel
clock (señal para la TFT)
        HSYNC,VSYNC                : OUT STD_LOGIC;                         --
Sincronismo horizontal, vertical, video activo para generar
        VR,VG,VB                   : OUT STD_LOGIC;                         --
Componentes RGB generadas para la LCD-TFT
        HPOS_TFT,VPOS_TFT: INOUT STD_LOGIC_VECTOR (10 DOWNTO 0) :=
(OTHERS => '0');       -- Contadores de fila y columna
  BLANK                            : OUT STD_LOGIC;                         --
Video activo

        -- Señales para leer de la memoria VIDEO_RAM
        VIDEO_ADD_TFT      : OUT NATURAL RANGE 0 TO (2**LONG_DIR)-1;
        -- Direcciones VIDEO_RAM
        VIDEO_DATA_TFT     : IN STD_LOGIC_VECTOR (ANCHO_DATO-1 DOWNTO
0);       -- Datos VIDEO_RAM
        VIDEO_WE_TFT       : OUT STD_LOGIC;
                                        -- R/W, output enable, chip enable

        -- Señales para guardar la INTENSIDAD en RAM de la FPGA
        AD_RAM_WE_48       : OUT std_logic := '1';
        AD_RAM_ADDR_48  : OUT natural RANGE 0 TO 7;
        AD_RAM_DIN_48      : OUT std_logic_vector(7 DOWNTO 0);

        -- Señales para leer la INTENSIDAD de la RAM de la FPGA
```

```
        AD_RAM_WE_TFT   : OUT std_logic := '1';
        AD_RAM_ADDR_TFT: OUT natural RANGE 0 TO 7;
        AD_RAM_DOUT_TFT: IN std_logic_vector(7 DOWNTO 0);

        -- Señales del AD9057
        DATA                    : IN STD_LOGIC_VECTOR (7 DOWNTO 0);
                                        -- Valor convertido del A/D
        marca_ZONA      : INOUT std_logic_vector(7 DOWNTO 0) := (OTHERS =>
'0');   -- Marca la zona de pantalla
        ZONA_W                  : INOUT INTEGER RANGE 0 TO 7 := 0;
                                        -- Zona de pantalla para el A/D
        EOC                     : INOUT BOOLEAN := TRUE);
                                        -- Indica el fin de conversión

END accede_VIDEO_RAM;

ARCHITECTURE MAIN OF accede_VIDEO_RAM IS

        -- Colores RGB. Valor decimal correspondiente a tres bits
        CONSTANT NEGRO          : INTEGER := 0;
        CONSTANT AZUL           : INTEGER := 1;
        CONSTANT VERDE  : INTEGER := 2;
        CONSTANT CYAN           : INTEGER := 3;
        CONSTANT ROJO           : INTEGER := 4;
        CONSTANT MAGENTA        : INTEGER := 5;
        CONSTANT AMARILLO : INTEGER := 6;
        CONSTANT BLANCO         : INTEGER := 7;

        CONSTANT SLV_R48        : std_logic_vector :=
conv_std_logic_vector(R48,10);      -- Convierte a STD_LOGIC
        CONSTANT TAM_MEM : INTEGER := D48 * R48 / 8;                    --
Tamano de memoria de 70470 bytes para almacenar una pantalla

        SIGNAL dato : STD_LOGIC_VECTOR((ANCHO_DATO-1) DOWNTO 0);
                -- Dato a escribir en RAM
        SIGNAL ZONA_R: INTEGER RANGE 0 TO 7 := 0;
                        -- Zona de pantalla
```

```
        SIGNAL CONTA: STD_LOGIC_VECTOR(24 DOWNTO 0):=(OTHERS =>'0');
        -- Permite la conversión

        -- Limites de cada zona de pantalla
        CONSTANT  ZONA0_H : INTEGER := 145;
        -- IZQUIERDA
        CONSTANT  ZONA0_V : INTEGER := R48-ZONA0_H-12;
        CONSTANT  ZONA1_H : INTEGER := D48;
        -- DERECHA
        CONSTANT  ZONA1_V : INTEGER := R48-ZONA0_H;
        CONSTANT  ZONA2_H : INTEGER := D48-ZONA0_H;             -- CENTRAL
        CONSTANT  ZONA2_V : INTEGER := R48-ZONA0_H+5;
        CONSTANT  ZONA3_H : INTEGER := ZONA0_H;                 --
INF_IZQUIERDA
        CONSTANT  ZONA3_V : INTEGER := R48;
        CONSTANT  ZONA4_H : INTEGER := D48-ZONA0_H-5;  -- INF_CENTRAL
        CONSTANT  ZONA4_V : INTEGER := R48;
        CONSTANT  ZONA5_H : INTEGER := D48;
        -- INF_DERECHA
        CONSTANT  ZONA5_V : INTEGER := R48;

-- Conversor A/D
-- Para almacenar el resultado de la conversión, la memoria necesaria tambien
-- tiene que ser de doble puerto ya que se tiene que escribir y leer al mismo tiempo.
COMPONENT ADC9057 IS
        PORT(
                CLK48                   : IN STD_LOGIC;
                        -- Reloj 48MHz
                FRAME                           : IN BOOLEAN;
                                        -- Detectado frame
                VPOS, HPOS          : IN STD_LOGIC_VECTOR (10 DOWNTO 0);
        -- Contador de fila y columna
                EOC                     : INOUT BOOLEAN;
                        -- Fin de conversion

                marca_ZONA          : INOUT STD_LOGIC_VECTOR (7 DOWNTO 0); -
- Byte con las zonas marcadas
```

```
                ZONA                    : IN INTEGER RANGE 0 TO 7 := 0;
        -- Zona de pantalla

                -- Señales para acceder al resultdo de la conversion del A/D
                AD_RAM_WE_48    : OUT STD_LOGIC := '1';
        -- R/W para acceder a RAM
                AD_RAM_ADDR_48  : OUT NATURAL RANGE 0 TO 7;
        -- Dirección para acceder a RAM
                AD_RAM_DIN_48   : OUT STD_LOGIC_VECTOR (7 DOWNTO 0);
        -- Datos para escribir en RAM

                DATA                    : IN STD_LOGIC_VECTOR (7 DOWNTO
0));    -- Valor convertido por el AD9057
END COMPONENT ADC9057;

BEGIN

-- Instanciacion del componente ADC9057
AD: ADC9057 PORT MAP
(CLK48,FRAME,VPOS48,HPOS48,EOC,marca_ZONA,ZONA_W,
                AD_RAM_WE_48,AD_RAM_ADDR_48,AD_RAM_DIN_48,DATA);

-- Detecta cada FRAME del osciloscopio mediante
-- el flanco negativo del sincronismo horizontal
DETECTA_FRAME:PROCESS(VGA_H,HPOS48,VPOS48)
BEGIN
        IF (HPOS48=821) AND (VPOS48=0) THEN             -- Condición
asíncrona para inicializar FRAME
                FRAME <= FALSE;
        ELSIF falling_edge(VGA_H) THEN                  -- Las señales del
osciloscopio estan invertidas

                -- respecto a una VGA estandar, por eso se

                -- detecta el flanco negativo
                FRAME <= TRUE;                                          --
Detectado FRAME
```

```
        END IF;
END PROCESS DETECTA_FRAME;

SALVAR_VIDEO_RAM: PROCESS(CLK48)

        VARIABLE dato: STD_LOGIC_VECTOR((ANCHO_DATO-1) DOWNTO 0) :=
(OTHERS => '0');       -- dato a escribir en RAM
        VARIABLE conta_dir: NATURAL RANGE 0 TO (2**LONG_DIR)-1 := 0;
                                -- direccion RAM
        VARIABLE conta_b: STD_LOGIC_VECTOR(3-1 DOWNTO 0) := (OTHERS =>
'0');                           -- contador de bit (8 bit/dato)
        VARIABLE HPO,VPO          : STD_LOGIC_VECTOR(10 DOWNTO 0) :=
(OTHERS => '0');                -- contador de columna y fila

BEGIN
IF rising_edge(CLK48) THEN
        IF FRAME THEN -- Una vez detectado nuevo FRAME
                -- Condicion de VIDEO ACTIVO
                IF (VPOS48 >= P48 + S48 AND VPOS48 < P48 + S48 + R48) AND
(HPOS48 >= E48 AND HPOS48 < E48 + D48) THEN
                        -- Con video activo, con cada ciclo de pixel clock se lee la senal
INTENSIDAD
                  -- del osciloscopio y se va almacenando en la senal dato hasta que se
completan
                        -- los 8 bits, mientras tanto no se accede a la RAM de VIDEO

                        HPO := HPOS48 - E48;
                        VPO := VPOS48 - (P48 + S48);

                        IF INTENSIDAD_DIG = '1' THEN -- Solo asigna zona cuando
existe pixel
                        -- Se identifica la zona de pantalla
                                IF                                              (HPO
<= ZONA0_H) AND (VPO >= R48-ZONA0_V AND VPO < R48) THEN    -- ZONA 0
                                        ZONA_W <= 0;
                                ELSIF (HPO > ZONA2_H) AND (HPO <= ZONA1_H)
AND (VPO >= R48-ZONA1_V AND VPO < R48) THEN     -- ZONA 1
```

```
                                ZONA_W <= 1;
                            ELSIF (HPO > ZONA0_H) AND (HPO <= ZONA2_H)
AND (VPO >= R48-ZONA2_V AND VPO < R48) THEN     -- ZONA 2
                                ZONA_W <= 2;
                            ELSIF                                   (HPO <=
ZONA3_H) AND (VPO <= R48-ZONA0_V) THEN
        -- ZONA 3
                                ZONA_W <= 3;
                            ELSIF (HPO > ZONA3_H) AND (HPO <= ZONA4_H)
AND (VPO <= R48-ZONA2_V) THEN                                   --
ZONA 4
                                ZONA_W <= 4;
                            ELSIF (HPO > ZONA4_H) AND (HPO <= ZONA5_H)
AND (VPO <= R48-ZONA5_V) THEN                                   --
ZONA 5
                                ZONA_W <= 5;
                            ELSE
                                ZONA_W <= 6;

                                -- Zona inexistente
                            END IF;
                    END IF;

                    dato(conv_integer(conta_b)) := INTENSIDAD_DIG;          --
Almacena bit de INTENSIDAD en dato

                    IF conta_b < 7 THEN
                        -- Hasta que no se completan los 8 bits de
INTENSIDAD...
                        VIDEO_DATA_48 <= (OTHERS => 'Z');      -- Datos: alta
Z
                        VIDEO_WE_48 <= '1';
        -- Lectura
                    ELSE
                        -- Guarda en RAM la INTENSIDAD.
                        VIDEO_ADD_48 <= conta_dir;        -- Direccion a escribir
```

```
                                VIDEO_DATA_48 <= dato;                  -- Dato de 8
bits que se guarda
                                VIDEO_WE_48 <= '0';                             --
Escritura

                                -- Una vez que se han completado 8 bits de INTENSIDAD
se guardan en RAM
                                IF conta_dir < TAM_MEM-1 THEN    -- TAM_MEM =
696x810/8 = 70470. Numero ...

                        -- de posiciones a almacenar por pantalla.
                                        conta_dir := conta_dir + 1;    -- Siguiente direccion
                                ELSE
                                        conta_dir := 0;
        -- Inicializa contador de direcciones RAM
                                END IF;

                        END IF;
                        conta_b := conta_b + 1;                         -- incrementa
contador de bits (0..7)
                ELSE
                        -- Condicion de VIDEO NO ACTIVO
                        VIDEO_DATA_48 <= (OTHERS => 'Z');      -- Datos: Alta Z
                        VIDEO_WE_48 <= '1';
        -- Lectura
                        ZONA_W <= 7;
        -- Zona inexistente
                END IF;

                -- Se ponen en marcha los contadores de fila y columna
                IF (VPOS48 < O48) THEN                          -- O48 = 912 filas

                        VPOS48 <= VPOS48 + 1;                   -- contador de filas
                ELSE
                        VPOS48 <= (OTHERS => '0');
                        IF (HPOS48 < A48)THEN                   -- A48 = 842
columnas
```

```
                                HPOS48 <= HPOS48 + 1;              -- contador de
columnas
                        ELSE
                                HPOS48 <= (OTHERS => '0');
                        END IF;
                END IF;
        ELSE
        -- Inicializar contadores de fila y columna y la direccion RAM a escribir
                HPOS48 <= (OTHERS => '0'); -- De columna
                VPOS48 <= (OTHERS => '0'); -- De fila
        END IF;
END IF;

conta_bit_48 <= conta_b;

END PROCESS SALVAR_VIDEO_RAM;

-- Genera las señales de sincronismo para la pantalla TFT.
-- Ademas, lee de RAM y la envia a la pantalla TFT de
-- 810x696 @ 62.5Hz con un pixel clock de 60MHz

LEER_VIDEO_RAM: PROCESS(CLK_TFT)

-- Constantes para centrar la pantalla
  CONSTANT HC : INTEGER := (D60 - D48) / 2;
  CONSTANT VC : INTEGER := (R60 - R48) / 2;

  VARIABLE pixel,resto: STD_LOGIC_VECTOR((LONG_DIR-1+1+3) DOWNTO 0);
                                        -- Numero de pixel (0..810x696-1)
  VARIABLE conta_dir : STD_LOGIC_VECTOR((LONG_DIR-1) DOWNTO 0) :=
(OTHERS => '0');       -- direccion RAM
  VARIABLE conta_bit          : STD_LOGIC_VECTOR((3-1) DOWNTO 0) := (OTHERS
=> '0');                    -- contador de bit (8 bit/dato)
  VARIABLE bit_SRAM  : STD_LOGIC := '0';
  VARIABLE BIT_RGB          : STD_LOGIC_VECTOR((3-1) DOWNTO 0);
  VARIABLE COLOR            : INTEGER RANGE 0 TO 7 := 7;
                                                -- BLANCO por defecto
```

```
 VARIABLE HS,VS              : STD_LOGIC_VECTOR(10 DOWNTO 0) := (OTHERS =>
'0');              -- Contador de columna y fila
 VARIABLE HPO,VPO        : STD_LOGIC_VECTOR(10 DOWNTO 0) := (OTHERS =>
'0');              -- Contador de columna y fila desfasado
 VARIABLE ZONA              : INTEGER RANGE 0 TO 7 := 0;
                                                -- IZQUIERDA por defecto

BEGIN

IF rising_edge(CLK_TFT) THEN
        -- Genera señal de SINCRONISMO HORIZONTAL
        IF (HS > E60 + D60 + C60 AND HS < E60 + D60 + C60 + B60) THEN
                HSYNC <= '0';
        ELSE
                HSYNC <= '1';
        END IF;
        -- Genera señal de SINCRONISMO VERTICAL
        IF(VS > R60 + Q60 AND VS < R60 + Q60 + P60) THEN
                VSYNC <= '0';
        ELSE
                VSYNC <= '1';
        END IF;
        -- Condicion de VIDEO ACTIVO
        IF (HS >= E60 AND HS < E60 + D60) AND (VS < R60) THEN
                BLANK <= '1';                                                   --
Video ACTIVO
                COLOR := NEGRO;
                VIDEO_WE_TFT <= '1';                                    -- Lectura
SRAM
                IF (HS >= E60 + HC AND HS < E60 + D60 - HC) AND (VS >= VC) AND
(VS < R60 - VC) THEN
                        ------------------------------------------------------------------
                        -- Obtiene la posicion de memoria RAM (conta_dir) donde se
encuentra almacenado el pixel
                        -- que toca enviar a pantalla y el bit (conta_bit) que corresponde
dentro de esa posición
                        HPO := HS - (E60 + HC);
```

```
                    VPO := VS - VC;

                    pixel := (HPO + 1) * SLV_R48 - (VPO + 1);  -- pixel := (HPO + 1) *
R60 - (VPO + 1);
                    conta_dir := pixel((LONG_DIR-1+3) DOWNTO 3); -- dirección
VIDEO_RAM := pixel/8
                    resto := pixel - (conta_dir & "000");              -- conta_bit <=
pixel - (conta_dir * 8)
                    conta_bit := resto(2 DOWNTO 0);
        -- nº de bit a enviar a pantalla
                ----------------------------------------------------------------------

                    -- Direccion RAM de VIDEO a leer
                    VIDEO_ADD_TFT <= conv_integer(conta_dir);
        -- Dirección RAM de VIDEO
                    VIDEO_WE_TFT <= '1';
                            -- Lectura RAM de VIDEO
                    bit_SRAM := VIDEO_DATA_TFT(conv_integer(conta_bit));  --
Lee pixel de RAM de VIDEO

                    -- PANTALLA DIVIDIDA EN LAS 5 ZONAS A LAS QUE HACE
REFERENCIA EL FICHERO Docu Lecroy
                    -- Cuando el bit esta activo se pinta de color segun el valor
convertido de INTENSIDAD
                    IF bit_SRAM = '1' THEN
                        -- Se identifica la zona de pantalla
                        IF          (HPO <= ZONA0_H) AND (VPO <=
ZONA0_V) THEN                                       -- ZONA 0
                            ZONA := 0;
                        ELSIF (HPO > ZONA2_H) AND (HPO <= ZONA1_H)
AND (VPO <= ZONA1_V) THEN                                   --
ZONA 1
                            ZONA := 1;
                        ELSIF (HPO > ZONA0_H) AND (HPO <= ZONA2_H)
AND (VPO <= ZONA2_V) THEN                                   --
ZONA 2
                            ZONA := 2;
```

```
                        ELSIF                                   (HPO <=
ZONA3_H) AND (VPO > ZONA0_V) AND (VPO <= ZONA3_V) THEN    -- ZONA 3
                                ZONA := 3;
                        ELSIF (HPO > ZONA0_H) AND (HPO <= ZONA4_H)
AND (VPO > ZONA2_V) AND (VPO <= ZONA4_V) THEN -- ZONA 4
                                ZONA := 4;
                        ELSIF (HPO > ZONA4_H) AND (HPO <= ZONA5_H)
AND (VPO > ZONA1_V) AND (VPO <= ZONA5_V) THEN -- ZONA 5
                                ZONA := 5;
                        ELSE
                                ZONA := 6;
                -- Zona inexistente
                        END IF;

                        AD_RAM_ADDR_TFT <= ZONA;
                                -- Dirección
                        COLOR := conv_integer(AD_RAM_DOUT_TFT(7 downto
5));    -- Valor convertido
                        COLOR := AMARILLO;
                ELSE
                        COLOR := NEGRO;
                        ZONA := 7;
                                                -- Zona inexistente
                END IF;
        END IF;
    ELSE
    -- Video NO ACTIVO: Sin acceso a memoria RAM de VIDEO
        BLANK <= '0';
    -- Video NO ACTIVO
        COLOR := NEGRO;
        ZONA := 7;
    -- Zona inexistente
        VIDEO_WE_TFT <= '1';
    -- Lectura RAM de VIDEO
    END IF;

    -- Genera los contadores de columna HS y fila VS
```

```
        IF (HS < A60) THEN
                HS := HS + 1;           -- Contador de columnas
        ELSE
                HS := (OTHERS => '0');
                IF (VS < O60) THEN
                        VS := VS + 1; -- Contador de filas
                ELSE
                        VS := (OTHERS => '0');
                END IF;
        END IF;
END IF;

-- Se descompone el COLOR del pixel en sus componentes RGB
BIT_RGB := conv_std_logic_vector(COLOR,3);

-- A continuación, descompone color de pixel en sus componentes RGB
VR <= BIT_RGB(2); -- Rojo (MSB según la tabla que se adjunta)
VG <= BIT_RGB(1); -- Verde
VB <= BIT_RGB(0); -- Azul (LSB)

-- Contadores de fila y columna para leer de RAM de VIDEO y enviar a pantalla LCD-
TFT
HPOS_TFT <= HPO;
VPOS_TFT <= VPO;

AD_RAM_WE_TFT <= '1';    -- Lectura del valor convertido

END PROCESS LEER_VIDEO_RAM;

END MAIN;
```

1.3 ADC9057.vhd

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
use ieee.numeric_std.all;

ENTITY ADC9057 IS
	PORT(
		CLK48				: IN STD_LOGIC;
				-- Reloj 48MHz
		FRAME					: IN BOOLEAN := FALSE;
				-- Detectado frame
		VPOS, HPOS		: IN STD_LOGIC_VECTOR (10 DOWNTO 0);
	-- Contador de fila y columna
		EOC				: INOUT BOOLEAN := TRUE;
			-- Fin de conversion

		marca_ZONA		: INOUT STD_LOGIC_VECTOR (7 DOWNTO 0);
	-- Byte con las zonas marcadas
		ZONA				: IN INTEGER RANGE 0 TO 7;
		-- Zona de pantalla

		-- Señales para acceder al resultdo de la conversion del A/D
		AD_RAM_WE_48	: OUT STD_LOGIC := '1';
	-- R/W para acceder a RAM
		AD_RAM_ADDR_48 : OUT NATURAL RANGE 0 TO 7;
	-- Dirección para acceder a RAM
		AD_RAM_DIN_48	: OUT STD_LOGIC_VECTOR (7 DOWNTO 0);
	-- Datos para escribir en RAM

		DATA				: IN STD_LOGIC_VECTOR (7 DOWNTO
0));	-- Valor convertido por el AD9057
END ADC9057;

ARCHITECTURE my_rtl OF ADC9057 IS

-- Declaración de estados de la maquina de MOORE
```

```
TYPE ESTADO IS (COMPRUEBA, CICLO1, CICLO2, FIN_CONV);
-- Declaración señal de tipo estado
SIGNAL ACTUAL, SIGUIENTE : ESTADO := COMPRUEBA;
BEGIN

-- Proceso secuencial: asignación de estado
        SEQ: PROCESS (CLK48)
                BEGIN
                        IF rising_edge (CLK48) THEN
                                ACTUAL <= SIGUIENTE;
      END IF;
    END PROCESS SEQ;

-- Proceso combinacional: salida y estado siguiente
        COMB: PROCESS (ACTUAL, ZONA, DATA, marca_ZONA, EOC, VPOS,
HPOS, FRAME)
        VARIABLE var_ZONA: INTEGER RANGE 0 TO 7;
        BEGIN
                IF FRAME AND (HPOS = 0) AND (VPOS = 0) THEN
                        -- Al comienzo de cada FRAME en que toca salvar a SRAM ...
                        -- se inicializa el marcador de cada zona marca_ZONA y se
activa
                        -- la posibilidad de usar en conversor
                        marca_ZONA <= (OTHERS => '0');          -- Inicializa marcador
de zona
                        AD_RAM_WE_48 <= '1';
        -- Se lee
                        EOC <= TRUE;
        -- Fin de conversión activo: necesario para la siguiente conversión

                ELSE
                        -- Ahora se realiza la conversión con el ADC9057, esta es ...
                        -- la parte puramente combinacional de la maquina de MOORE,
                        -- donde hay que definir los distintos estados
                        CASE ACTUAL IS
                                WHEN COMPRUEBA =>
```

```
                                    -- Permanece en este estado hasta que se
encuentre un pixel
                                    -- de una zona en la que todavia no se haya
convertido su INTENSIDAD
                                    -- y no se este en el proceso de conversión de un
pixel anterior
                                    AD_RAM_WE_48 <= '1';                    --
Lectura
                                    var_ZONA := ZONA;           -- Se guarda ZONA
en este instante
                                    AD_RAM_ADDR_48 <= ZONA;                 --
Dirección de escritura = el valor de ZONA
                                    IF (marca_ZONA(var_ZONA) = '0') AND EOC
THEN
                                        SIGUIENTE <= CICLO1;
                                    END IF;
                                WHEN CICLO1 =>
                                    marca_ZONA(var_ZONA) <= '1';    -- Se marca la
zona
                                    EOC <= FALSE;
        -- Se inhabilita el conversor
                                    SIGUIENTE <= CICLO2;
                                WHEN CICLO2 =>
                                    SIGUIENTE <= FIN_CONV;
                                WHEN OTHERS =>
                                    EOC <= TRUE;
        -- Fin de conversion
                                    AD_RAM_WE_48 <= '0';                    --
Escritura
                                    AD_RAM_DIN_48 <= DATA;                  --
Escribe valor convertido en RAM de FPGA
                                    SIGUIENTE <= COMPRUEBA;
        -- Vuelve al comienzo, para la siguiente conversión
                        END CASE;
                END IF;
        END PROCESS COMB;
END my_rtl;
```

1.4 Interface_FPDLink_18bits.vhd

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY Interface_FPDLink_18bits IS
	PORT(
		clk280: IN STD_LOGIC;
	-- Reloj de 280MHz
		Hsync, Vsync, video_activo: IN STD_LOGIC;		--
Sincronismos y video activo
		R, G, B: IN STD_LOGIC_VECTOR(5 DOWNTO 0);		--
Componentes R,G,B de 6 bits
		RO0	: OUT STD_LOGIC;
	-- Canal 0 LVDS
		RO1	: OUT STD_LOGIC;
	-- Canal 1 LVDS
		RO2	: OUT STD_LOGIC;
	-- Canal 2 LVDS
		CLK	: OUT STD_LOGIC);
	-- Canal reloj LVDS
END Interface_FPDLink_18bits;


ARCHITECTURE FPDLink_interface_18bits OF Interface_FPDLink_18bits IS
	SIGNAL dataA, dataB, dataC, data_clk: STD_LOGIC_VECTOR(6 DOWNTO 0);
	COMPONENT serializador IS
		PORT(clk280: IN STD_LOGIC;
				din: IN STD_LOGIC_VECTOR(6 DOWNTO 0);
				dout: OUT STD_LOGIC);
	END COMPONENT;
BEGIN
----------Mapper----------------------------------
	dataA <= G(0) & R(5 DOWNTO 0);
	dataB <= B(1 DOWNTO 0) & G(5 DOWNTO 1);
	dataC <= video_activo & Vsync & Hsync & B(5 DOWNTO 2);
	data_clk <= "1100011";
----------Serializer-----------------------------
```

```
        serialA : serializador PORT MAP (clk280, dataA, RO0);
        serialB : serializador PORT MAP (clk280, dataB, RO1);
        serialC : serializador PORT MAP (clk280, dataC, RO2);
        serial_clk : serializador PORT MAP (clk280, data_clk, CLK);
END FPDLink_interface_18bits;
```

1.5 Serializador.vhd

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY serializador IS
	PORT (clk280 : IN STD_LOGIC;
			din		: IN std_logic_vector(6 downto 0);
			dout		: OUT STD_LOGIC);
END serializador;

ARCHITECTURE serializador OF serializador IS
	SIGNAL internal: STD_LOGIC_VECTOR(6 DOWNTO 0);
BEGIN
	PROCESS(clk280)
		VARIABLE count: INTEGER RANGE 0 TO 7 := 0;
	BEGIN
		IF rising_edge(clk280) THEN
			count := count + 1;
			IF (count = 6) THEN
				internal <= din;				-- Lee
siguiente palabra
			ELSIF (count = 7) THEN
				count := 0;					--
Inicializa para enviar MSB primero
			END IF;
			dout <= internal(6-count);		-- MSB primero
		END IF;
	END PROCESS;
END serializador;
```

1.6 video_ram_dual_port.vhd

```
library ieee;
use ieee.std_logic_1164.all;

ENTITY video_ram_dual_port IS

	GENERIC
	(
		ADDR_WIDTH : natural := 17;
		DATA_WIDTH : natural := 8
	);

	PORT
	(
		clk_a		: IN std_logic;
		clk_b		: IN std_logic;
		addr_a	: IN natural RANGE 0 TO 2**ADDR_WIDTH - 1;
		addr_b	: IN natural RANGE 0 TO 2**ADDR_WIDTH - 1;
		data_a	: IN std_logic_vector ((DATA_WIDTH-1) DOWNTO 0);
		data_b	: IN std_logic_vector ((DATA_WIDTH-1) DOWNTO 0);
		we_a		: IN std_logic := '1';
		we_b		: IN std_logic := '1';
		q_a		: OUT std_logic_vector ((DATA_WIDTH-1) DOWNTO 0);
		q_b		: OUT std_logic_vector ((DATA_WIDTH-1) DOWNTO 0)
	);

END video_ram_dual_port;

ARCHITECTURE rtl OF video_ram_dual_port IS

	-- Build a 2-D array type for the RAM
	SUBTYPE word_t IS std_logic_vector ((DATA_WIDTH-1) DOWNTO 0);
	TYPE memory_t IS ARRAY(2**ADDR_WIDTH - 1 DOWNTO 0) OF word_t;

	-- Declare the RAM
	SHARED VARIABLE ram : memory_t;  -- Para que se pueda acceder por mas
...
```

```
        -- de un puerto es necesario que el ...

        -- su tipo sea: shared variable
        ATTRIBUTE ramstyle : string;

BEGIN

        -- Port A
        PROCESS(clk_a)
        BEGIN
        IF(rising_edge(clk_a)) THEN
                IF(we_a = '0') THEN
                        ram(addr_a) := data_a;
                END IF;
                q_a <= ram(addr_a);
        END IF;
        END PROCESS;

        -- Port B
        PROCESS(clk_b)
        BEGIN
        IF(rising_edge(clk_b)) THEN
                IF(we_b = '0') THEN
                        ram(addr_b) := data_b;
                END IF;
                q_b <= ram(addr_b);
        END IF;
        END PROCESS;

END rtl;
```

1.7 zona_dual_port.vhd

```
library ieee;
use ieee.std_logic_1164.all;

ENTITY zona_dual_port IS

	GENERIC
	(
		ADDR_WIDTH : natural := 3;
		DATA_WIDTH : natural := 8
	);

	PORT
	(
		clk_a		: IN std_logic;
		clk_b		: IN std_logic;
		addr_a	: IN natural RANGE 0 TO 2**ADDR_WIDTH - 1;
		addr_b	: IN natural RANGE 0 TO 2**ADDR_WIDTH - 1;
		data_a	: IN std_logic_vector ((DATA_WIDTH-1) DOWNTO 0);
		data_b	: IN std_logic_vector ((DATA_WIDTH-1) DOWNTO 0);
		we_a		: IN std_logic := '1';
		we_b		: IN std_logic := '1';
		q_a		: OUT std_logic_vector ((DATA_WIDTH-1) DOWNTO 0);
		q_b		: OUT std_logic_vector ((DATA_WIDTH-1) DOWNTO 0)
	);

END zona_dual_port;

ARCHITECTURE rtl OF zona_dual_port IS

	-- Build a 2-D array type for the RAM
	SUBTYPE word_t IS std_logic_vector ((DATA_WIDTH-1) DOWNTO 0);
	TYPE memory_t IS ARRAY(2**ADDR_WIDTH - 1 DOWNTO 0) OF word_t;

	-- Declare the RAM
	SHARED VARIABLE ram : memory_t;  -- Para que se pueda acceder por mas
...
```

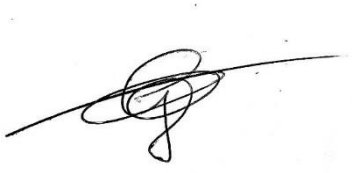
```
        -- de un puerto es necesario que el ...

        -- su tipo sea: shared variable
        ATTRIBUTE ramstyle : string;

BEGIN

        -- Port A
        PROCESS(clk_a)
        BEGIN
        IF(rising_edge(clk_a)) THEN
                IF(we_a = '0') THEN
                        ram(addr_a) := data_a;
                END IF;
                q_a <= ram(addr_a);
        END IF;
        END PROCESS;

        -- Port B
        PROCESS(clk_b)
        BEGIN
        IF(rising_edge(clk_b)) THEN
                IF(we_b = '0') THEN
                        ram(addr_b) := data_b;
                END IF;
                q_b <= ram(addr_b);
        END IF;
        END PROCESS;

END rtl;
```

Planos

Controladora para pantalla TFT-LCD 800x600 basada en FPGA para osciloscopios LeCroy 93XX

Alumno: David Godoy Chiclana

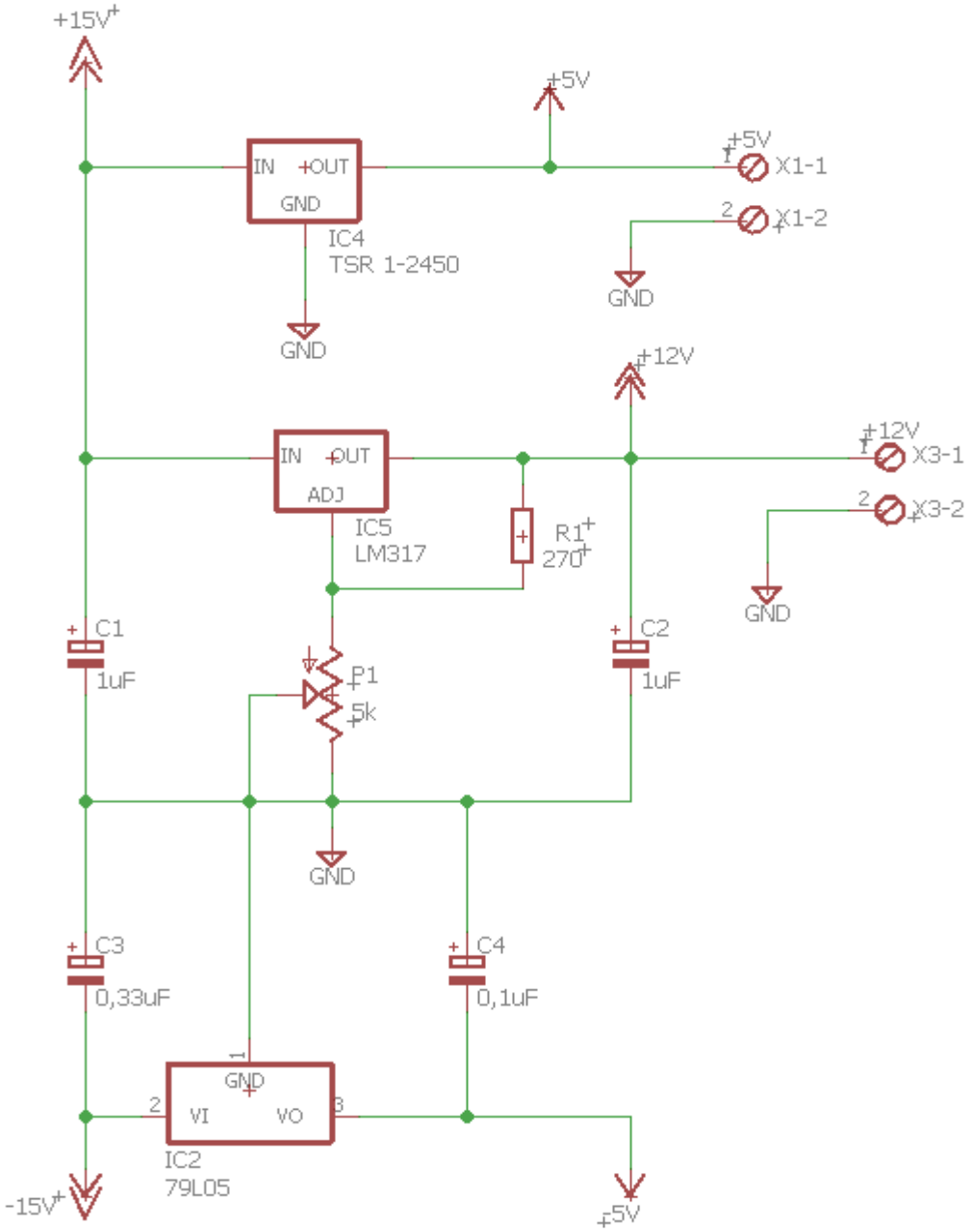
Tutor: Prof. D. Gregorio Godoy Vilches



Departamento: Ingeniería Electrónica y Automática

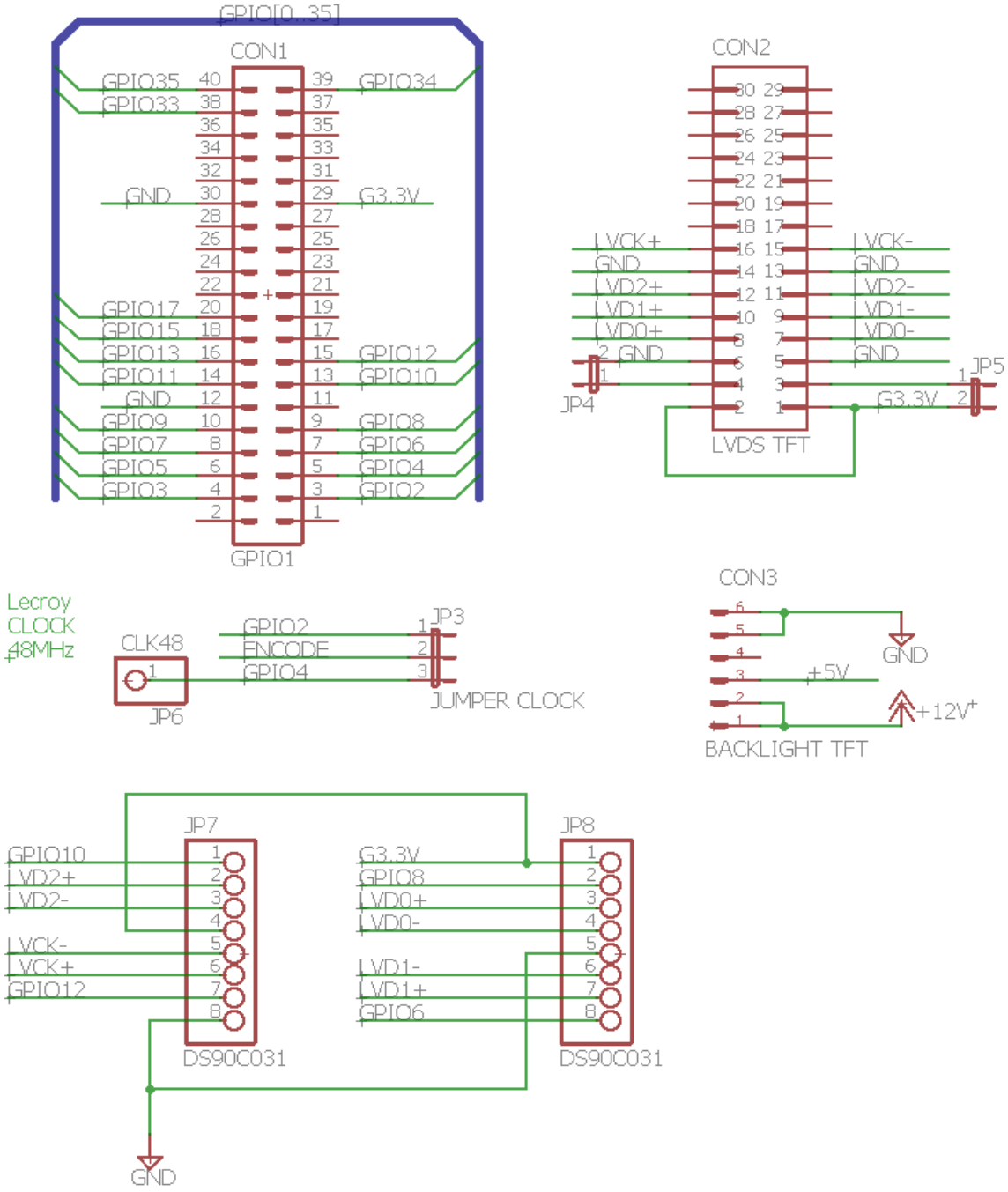
Septiembre, 2017

ÍNDICE DE PLANOS



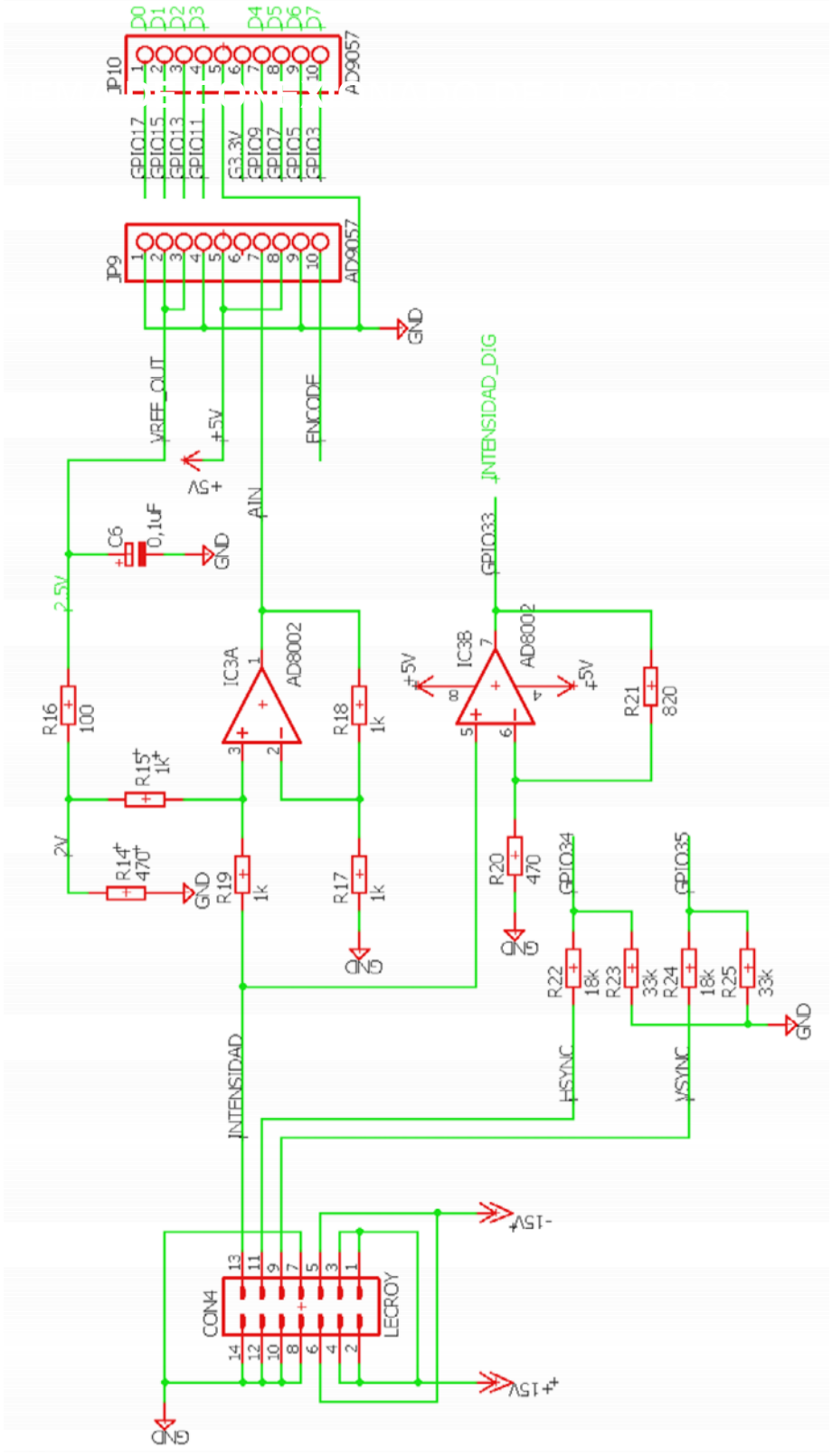
	FECHA	NOMBRE	FIRMA	ESCUELA POLITÉCNICA SUPERIOR DE LINARES
DIBUJADO	15/06/17	David		
COMPROBADO		Godoy		
		Chiclana		

ESCALA:	Controladora para pantalla TFT LCD de 800x600 basada en FPGA para osciloscopios LeCroy 93XX Esquema de conexionado de la PCB 1	Nº PLANO 1/5 SUSTITUYE A: SUSTITUIDO POR:



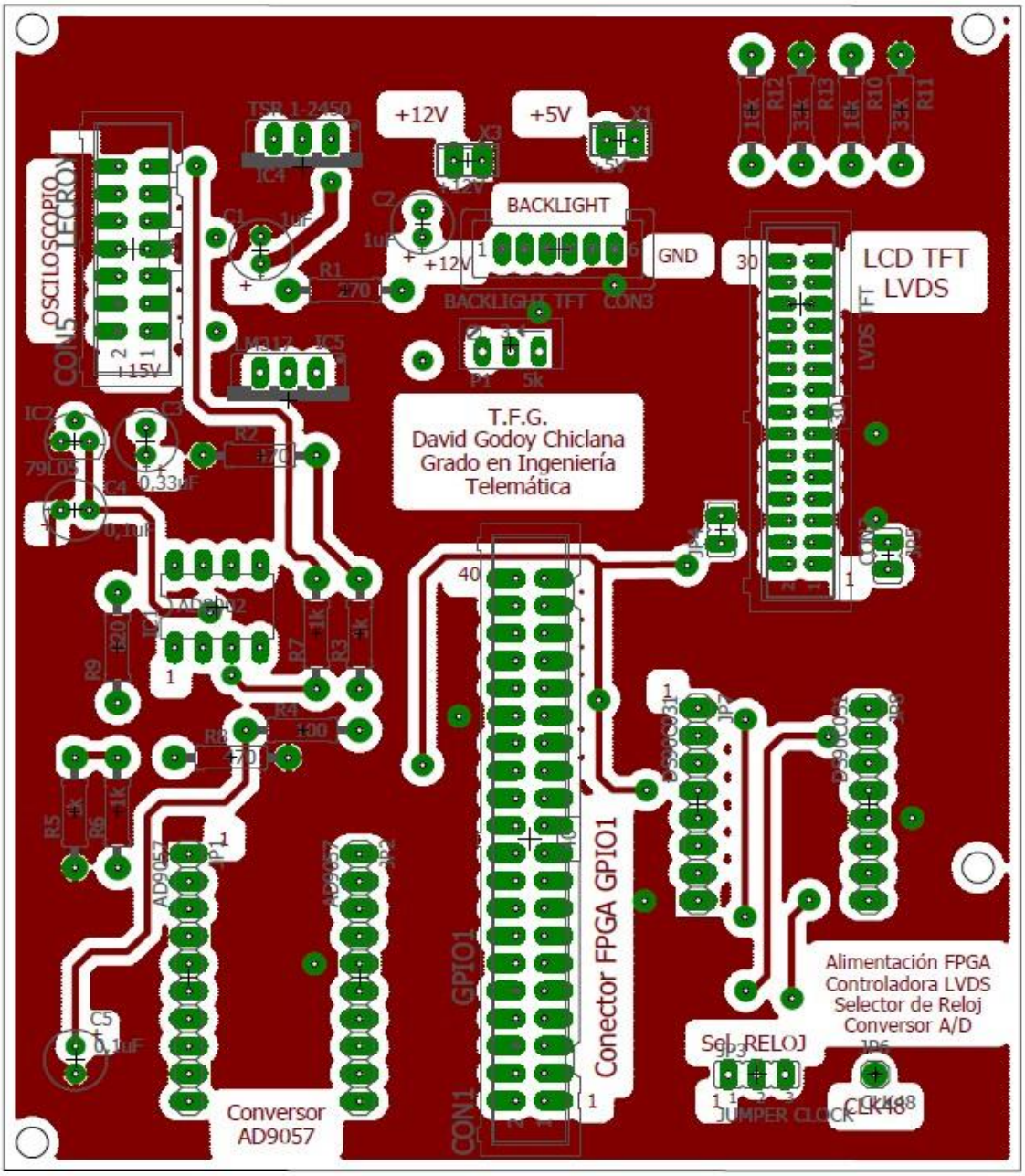
	FECHA	NOMBRE	FIRMA	ESCUELA POLITÉCNICA SUPERIOR DE LINARES
DIBUJADO	15/06/17	David		
COMPROBADO		Godoy		
		Chiclana		

ESCALA:	Controladora para pantalla TFT LCD de 800x600 basada en FPGA para osciloscopios LeCroy 93XX Esquema de conexionado de la PCB 2	Nº PLANO 2/5 SUSTITUYE A: SUSTITUIDO POR:



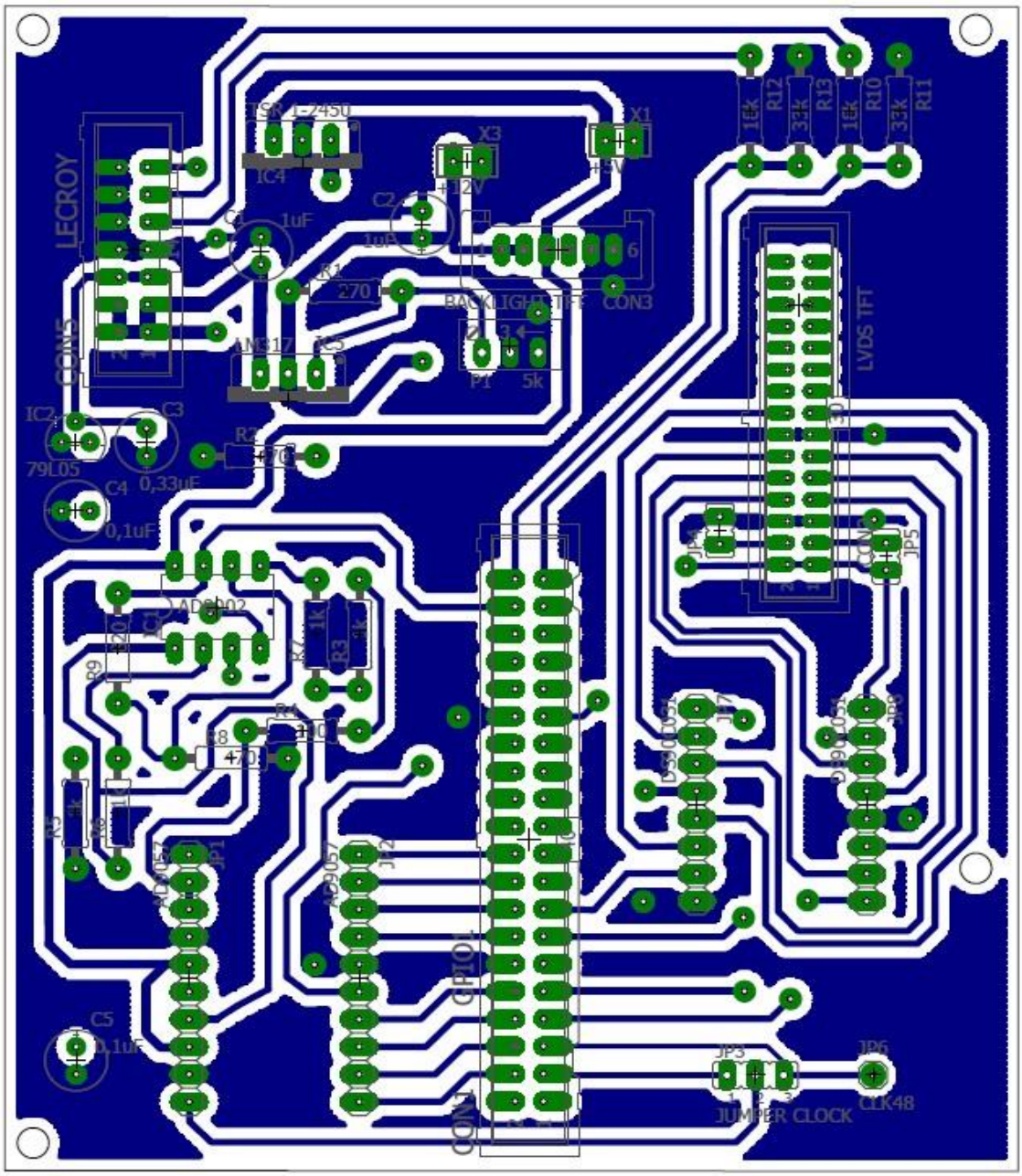
	FECHA	NOMBRE	FIRMA	ESCUELA POLITÉCNICA SUPERIOR DE LINARES
DIBUJADO	15/06/17	David		
COMPROBADO		Godoy		
		Chiclana		

ESCALA:	Controladora para pantalla TFT LCD de 800x600 basada en FPGA para osciloscopios LeCroy 93XX Esquema de conexionado de la PCB 3	Nº PLANO 3/5 SUSTITUYE A: SUSTITUIDO POR:



	FECHA	NOMBRE	FIRMA	ESCUELA POLITÉCNICA SUPERIOR DE LINARES
DIBUJADO	15/06/17	David		
COMPROBADO		Godoy		
		Chiclana		

ESCALA:	Controladora para pantalla TFT LCD de 800x600 basada en FPGA para osciloscopios LeCroy 93XX Cara superior de la PCB	Nº PLANO 4/5
		SUSTITUYE A:
		SUSTITUIDO POR:



	FECHA	NOMBRE	FIRMA	ESCUELA POLITÉCNICA SUPERIOR DE LINARES
DIBUJADO	15/06/17	David		
COMPROBADO		Godoy		
		Chiclana		

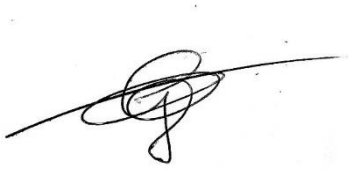
ESCALA:	Controladora para pantalla TFT LCD de 800x600 basada en FPGA para osciloscopios LeCroy 93XX Cara inferior de la PCB	Nº PLANO 5/5
		SUSTITUYE A:
		SUSTITUIDO POR:

Presupuestos

Controladora para pantalla TFT-LCD 800x600 basada en FPGA para osciloscopios LeCroy 93XX

Alumno: David Godoy Chiclana

Tutor: Prof. D. Gregorio Godoy Vilches



Departamento: Ingeniería Electrónica y Automática

Septiembre, 2017

ÍNDICE DE LOS PRESUPUESTOS

ÍNDICE DE TABLAS

1 PARTIDAS

1.1 Partida del Hardware

Apartado	Componente	Precio unidad	Cantidad	Precio total
Sistema de desarrollo basado en FPGA	DE0-CV	106,52	1	106,52
Placa controladora de video	VS-V59AV	20,66	1	20,66
	Chimei INNOLUX de 8.0 pulgadas, 24 bits y 40 pines, TFT-LCD AT080TN52 V.5 SVGA RGB de 800x600	18,34	1	18,34
	AUO B101EW05 de 10.1 pulgadas, TFT-LCD de WXGA de 1280x800	59,36	1	59,36
	Cable VGA	3,74	1	3,74
	Cable LVDS	6,61	1	6,61
Subtotal				215,23

Tabla 1: Hardware

Asciende el presupuesto para Hardware, a la mencionada cantidad de DOSCIENTOS QUINCE EUROS CON VEINTITRÉS CÉNTIMOS.

1.2 Partida de la placa de circuito impreso

Apartado	Componente	Precio unidad	Cantidad	Precio total
Placa de circuito impreso	Doble cara de 100x160mm	2,56	1	2,56
Sujeción	Pilar	0,12	4	0,48
Conversor	AD9057	4,96	1	4,96
Operacionales	AD8002	2	1	2
	7905	0,5	1	0,5
	Mp1584	2,97	1	2,97
	Tracopower TRS 1-2450	4,45	1	4,45
Driver	Ds90c031	0,41	1	0,41
	1x50	0,83	1	0,83
	2x50	1,65	1	1,65
Tira de pines hembra	1x50	0,83	1	0,83
Regleta de conexión	2 contactos y distancia de décima de pulgada	2,28	2	4,56
Jumpers	Décima de pulgada	0,02	3	0,06
Puerto LVDS	2x15	1,24	1	1,24
	100Ω	0,08	5	0,4
	470Ω	0,08	2	0,16
	680Ω	0,08	1	0,08
	1KΩ	0,08	4	0,32
	18KΩ	0,08	1	0,08
	33KΩ	0,08	1	0,08
Condensadores	1µF 50V	0,08	3	0,24
Subtotal				28,86

Tabla 2: Placa de circuito impreso

Asciende el presupuesto para la placa de circuito impreso, a la mencionada cantidad de VEINTIOCHO EUROS CON OCHENTA Y SEIS CÉNTIMOS.

1.3 Partida de mano de obra

Para la realización del presupuesto total final, aparte de todos los elementos que se han adquirido, unido y montado, se ha de tener en cuenta el precio de la mano de obra encargada del diseño, el montaje, las pruebas, las modificaciones, etc…

En la Ley 2/1974, del 13 de febrero, sobre Colegios Profesionales, en su artículo 0número 14 'Prohibición de recomendaciones sobre honorarios' se establece que los Colegio Profesionales y sus organizaciones colegiales no podrán establecer baremos orientativos ni cualquier otra orientación, recomendación, directriz, norma o regla sobre honorarios profesionales, salvo lo establecido en la Disposición adicional cuarta ('Tarifa de primas para la cotización a la Seguridad Social por accidentes de trabajo y enfermedades profesionales').

Por tanto, se requiere hacer una estimación del precio de la mano de obra, para realizarla, se tiene en cuenta el caso de este proyecto, que sería el de un graduado en ingeniería de telecomunicaciones contratado por horas, encargado de todo el trabajo anteriormente mencionado.

Es por lo anteriormente mencionado que, la hora de trabajo, se le estima un precio aproximado de 65,00 € la hora, y teniendo en cuenta que el proyecto ha tenido 300 horas de trabajo.

Sueldo ingeniero/hora	Horas	Subtotal
65,00 €	300	19500,00 €

Tabla 3: Mano de obra

Asciende el presupuesto en mano de obra, a la mencionada cantidad de DIECINUEVE MIL QUINIENTOS EUROS.

2 PRESUPUESTO TOTAL

Concepto	Valor
Hardware	215,23 €
Placa de circuito impreso	28,86 €
Mano de obra	19500,00 €
Subtotal	19744,09 €
IVA	21%
Total	23890,35 €

Tabla 4: Presupuesto total

Asciende el presupuesto total, a la mencionada cantidad de VEINTITRÉS MIL OCHOCIENTOS NOVENTA EUROS CON TREINTA Y CINCO CÉNTIMOS.