UNIVERSIDAD DE SONORA

DIVISIÓN DE CIENCIAS EXACTAS Y NATURALES
DEPARTAMENTO DE INVESTIGACIÓN EN FÍSICA
INGENIERÍA EN TECNOLOGÍA ELECTRÓNICA

"CONTROL DE UN MODULADOR ESPECIAL DE LUZ EMPLEANDO UN FPGA"

T E S I S

Para obtener el título de:

LICENCIADO EN TECNOLOGÍA ELECTRÓNICA

Presenta:

ROBERTO SARABIA COHEN

Directores de Tesís:

Dr. Luis Alfredo González López
Dra. Alicia Vera Marquina

Hermosillo, Sonora **Mayo de 2007**

Universidad de Sonora

Repositorio Institucional UNISON

UNIVERSIDAD DE SONORA

DIVISIÓN DE CIENCIAS EXACTAS Y NATURALES

DEPARTAMENTO DE INVESTIGACIÓN EN FISICA

INGENIERIA EN TECNOLOGIA ELECTRONICA

"CONTROL DE UN MODULADOR ESPACIAL DE LUZ EMPLEANDO UN FPGA"

TESIS

PARA OBTENER EL TITULO DE:

INGENIERO EN TECNOLOGIA ELECTRONICA

PRESENTA:

EL SABER DE MIS HIJOS
HARÁ MI GRANDEZA

ROBERTO SARABIA COHEN

DIRECTORES DE TESIS:

DR. LUIS ALFREDO GONZALEZ LOPEZ
DRA. ALICIA VERA MARQUINA

Hermosillo, Sonora. Mayo de 2007

Con todo mi amor dedico este trabajo

… A **Dios**, por todo lo que me ha brindado y permitirme vivir este momento.

… A **Sandy**, por ser la mujer de mi vida y compartir su vida conmigo.

…Y a mi **Padre** y mi **Madre** por el apoyo incondicional que me han dado toda la vida.

Agradecimientos.

A los maestros de la carrera de Ingeniería en Tecnología Electrónica de la Universidad de Sonora por sus enseñanzas, en especial al Dr. Luis González, la Dra. Alicia Vera y el Dr. Fernando Mendoza.

A CONACYT por el apoyo brindado con la beca de tesis del proyecto 44071-A1 "Investigación y desarrollo de sistemas óptico-digitales para el proceso de información tridimensional".

A mis asesores de tesis Dr. Luis González y Dra. Alicia Vera por su apoyo y enseñanzas durante toda mi carrera.

A mi familia porque sin ellos hoy no seria la persona que soy, en especial a mi Padre y mi Madre, así también a mis hermanos Jesús German y Luis Rey.

A mi otra familia Elba, Héctor Antonio, Esmeralda y Héctor Antonio Tadeo por su apoyo siempre.

A todos mis amigos, en particular a los de la carrera por todos los momentos que compartimos y en especial a Temis, Raúl y Chema.

CONTENIDO

LISTA DE FIGURAS

LISTA DE TABLAS

CAPITULO 1

INTRODUCCION

Además de su uso como sistemas de despliegue de imágenes, las pantallas de cristal líquido (LCDs, por siglas en ingles de *Liquid Cristal Displays*) actualmente se emplean en aplicaciones de procesamiento óptico-digital. Para esto, se adaptan pequeñas LCDs con características especiales que modulan la luz proveniente de una fuente de iluminación (que puede ser generada por un láser) en su fase o en su amplitud. A estos dispositivos de cristal liquido se les conoce como moduladores espaciales de luz. Dentro del proyecto de CONACYT 44071-A1, se propuso el desarrollo de un sistema para llevar a cabo la reconstrucción tridimensional de un objeto empleando un modulador espacial de luz de cristal líquido. En la industria, la reconstrucción tridimensional de objetos es muy importante en los procesos de fabricación y manufactura; es decir, en procesos donde sea necesario comparar un objeto real con una imagen tridimensionada, corroborando si existen diferencias entre estos. Por ejemplo, su utilidad puede ser importante en la industria automotriz para revisar abolladuras en las carrocerías y bordes en piezas mecánicas. Una diferencia sustancial del sistema propuesto en comparación con la mayoría de los sistemas de escaneo láser es que no se requieren motores ni piezas mecánicas en movimiento para formar el haz de luz de escaneo. Esto se debe a que las LCDs se pueden direccionar electrónicamente. Para realizar este direccionamiento comúnmente se han empleado tarjetas de video provenientes de computadoras personales, debido a que emplean el formato SVGA de los monitores de video de las computadoras como medio de direccionamiento. Sin embargo, cuando hay aplicaciones que requieren la generación de gráficos a altas velocidades se hace necesario el uso de tarjetas de alto costo y que implican en algunos casos la disposición de mayores recursos computacionales. Con el avance de la electronica digital, en la actualidad se pueden conseguir dispositivos lógicos programables para controlar, direccionar y procesar información digital en tiempo real. Estos sistemas pueden operar de forma independiente a una computadora personal. Específicamente los FPGAs (por sus siglas en inglés

de *Field Programmable Gate Arrays*) son dispositivos reconfigurables que pueden soportar diseños digitales complejos que emplean operaciones logicas, circuitos secuencial, maquinas de estado o hasta procesadores de propósito específico. En este trabajo de tesis presentamos el diseño y la implementación de una interfaz para un modulador espacial de luz de cristal liquido mediante un FPGA el cual opera en una plataforma XUP Virtex II PRO. El diseño digital se realizó con el lenguaje de descripción de hardware VHDL. El propósito de esta interfaz es controlar al modulador de luz para codificar elementos ópticos que dispersen la luz creando patrones de iluminación láser y de esta manera recuperar la forma de tridimensional de un objeto. A continuación se describe brevemente la organización del contenido de esta tesis

1.1. Organización de Documento

Esta tesis está constituida de 7 capítulos. En el capitulo 1 se planteó la meta y el objetivo por el cual surge la idea de desarrollo de esta tesis; así como los antecedentes y descripción del proyecto. En el capitulo 2 se describe el pricipio de operación de las pantallas de cristal liquido para su utilización como modulador espacial de luz. En el capitulo 3 se hace referencia a los dispositivos lógicos programables así como los FPGA que es el que utilizamos. El modo de video utilizado se explica en el capítulo 4 ya que se aplicó en VGA. El capitulo 5 habla de dispositivos de almacenamiento, es decir, memorias y aquí se utilizó una memoria RAM para el almacenamiento de los datos, y finaliza con los resultados y conclusiones que son los capítulos 6 y 7.

CAPITULO 2

PANTALLAS DE CRISTAL LÍQUIDO

2.1. Introducción.

La LCD es un dispositivo electro-optico que fue inventado por Jack Janning. Se trata de un sistema eléctrico de presentación de datos formado por 2 capas conductoras transparentes y en medio un material especial cristalino (cristal líquido) que tienen la capacidad de orientar la luz a su paso, cuando la corriente circula entre los electrodos transparentes con la forma a representar (por ejemplo, un segmento de un número, como se muestra en la figura 2.1.) el material cristalino se reorienta alterando su transparencia. El material base de un LCD lo constituye el cristal líquido, el cual exhibe un comportamiento similar al de los líquidos y unas propiedades físicas anisotrópicas similares a las de los sólidos cristalinos. Las moléculas de cristal líquido poseen una forma alargada y son más o menos paralelas entre sí en la fase cristalina. Según la disposición molecular y su ordenamiento, se clasifican en tres tipos: nemáticos, esméticos y colestéricos. La mayoría de los cristales liquidos responden con facilidad a los campos eléctricos, exhibiendo distintas propiedades ópticas en presencia o ausencia del campo. El tipo más común de visualizador LCD es el denominado nemático de torsión, término que indica que sus moléculas en su estado desactivado presentan una disposición en espiral. La polarización o no de la luz que circula por el interior de la estructura, mediante la aplicación o no de un campo eléctrico exterior, permite la activación de una serie de segmentos transparentes, los cuales rodean al cristal líquido. Según sus características ópticas, pueden también clasificarse como: reflectivos, transmisivos y transreflectivos.

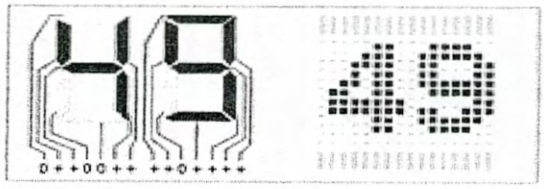


Figura 2.1. Composición de una LCD por segmentos y por píxeles

Las pantallas LCD se encuentran en multitud de dispositivos industriales y de consumo domestico: máquinas expendedoras, electrodomésticos, equipos de telecomunicaciones, computadoras, etc. Todos estos dispositivos utilizan pantallas fabricadas por terceros de una manera más o menos estandarizada.

Cada LCD se compone de una pequeña placa integrada que consta de:

- La propia pantalla LCD.
- Un microchip controlador.
- Una pequeña memoria que contiene una tabla de caracteres.
- Un interfaz de contactos eléctricos, para conexión externa.
- Opcionalmente, una luz trasera para iluminar la pantalla.

El controlador simplifica el uso del LCD proporcionando una serie de funciones básicas que se invocan mediante el interfaz eléctrico, destacando:

- La escritura de caracteres en la pantalla.
- El posicionado de un cursor parpadeante, si se desea.
- El desplazamiento horizontal de los caracteres de la pantalla (*scrolling*).
- Etc.

El circuito de memoria para una LCD implementa un mapa de bits para cada carácter de un juego de caracteres, es decir, cada octeto de esta memoria describe los puntitos o pixeles que deben iluminarse para representar un carácter en la pantalla. Generalmente, se pueden definir caracteres a medida modificando el contenido de esta memoria. Así, es posible mostrar símbolos que no están originalmente contemplados en el juego de caracteres. La interfaz de contactos eléctricos suele ser de tipo paralelo, donde varias señales eléctricas simultáneas indican la función que debe ejecutar el controlador junto con sus parámetros. Por tanto, se requiere cierta sincronización entre estas señales eléctricas. La luz trasera facilita la lectura de la pantalla LCD en cualquier condición de iluminación ambiental. Existen dos tipos de pantallas LCD en el mercado: pantallas de texto y pantallas gráficas. Las LCD de texto son las más baratas y simples de utilizar. Solamente permiten visualizar mensajes cortos de texto. Existen algunos modelos estandarizados en la industria, en función de su tamaño medido en número de líneas y columnas de texto. Existen modelos de una, dos y cuatro filas únicamente. El número de columnas típico es de ocho, dieciséis, veinte y cuarenta caracteres. Las pantallas LCD gráficas permiten encender y apagar individualmente píxeles de la pantalla. De esta manera es posible mostrar gráficos en blanco y negro. No solamente texto. Los controladores más populares son el Hitachi HD61202 y el Samsung KS0108. Los tamaños también están estandarizados y se miden en filas y columnas de píxeles. Algunos tamaños típicos son 128x64 y 96x60. Naturalmente, algunos controladores también permiten la escritura de texto de manera sencilla. Estas pantallas son más caras y complejas de utilizar. Existen pocas aplicaciones donde no baste con un LCD de texto. Se suelen utilizar, por ejemplo, en ecualizadores gráficos [2]. En la figura 2.2. se muestra una pantalla que muestra gráficos a color en la que se pueden representar imágenes.

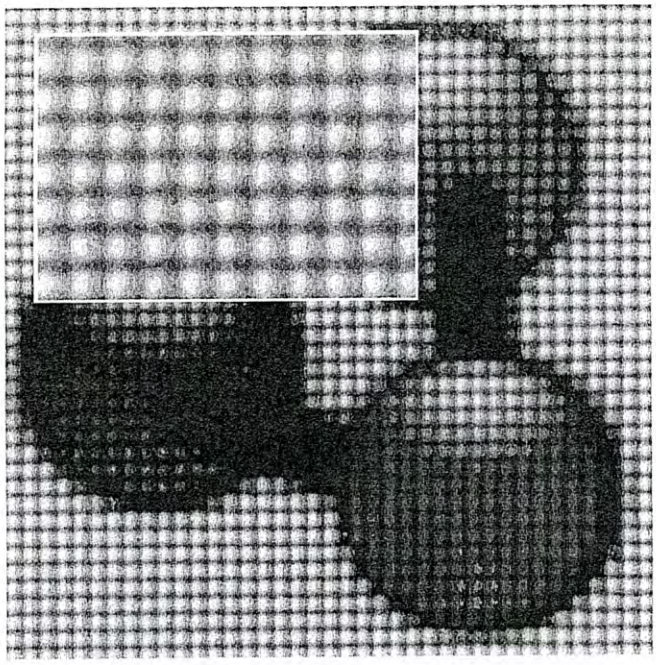


Figura 2.2. Detalle de una pantalla LCD en color

La mayoría de las moléculas de cristal liquido son de forma de barra y se categorizar como termotrópicas y liotrópicas. Los cristales líquidos termotrópicos reaccionan a cambios de temperatura o en algunos casos presión. La reacción de los cristales líquidos liotrópicos que se usan en la fabricación de jabones y detergentes, dependiendo de los solventes que se utilicen. Los cristales líquidos termotrópicos son también isotrópicos o nemáticos. La diferencia principal es que las moléculas en las sustancias de cristal líquido isotrópicas están acomodadas aleatoriamente mientras que los nemáticos tienen un patrón de orden definido. La orientación de las moléculas en la fase nemática se basa en un director, el cual puede ser cualquier cosa de un campo magnético con una superficie con curvas microscópicas. En la fase nemática se clasifican por la orientación de las moléculas una con otra. El arreglo mas común es el esmético, este crea capas de moléculas. Existen muchas variaciones de fase esmética, como la esmética C, en la que las moléculas en cada capa se inclinan a cierto ángulo de la capa anterior. Otra fase muy común es la colestérica, en esta fase las moléculas se tuercen ligeramente de una fase a la otra resultando en una formación espiral. En la figura 2.3 se muestra el cristal líquido en fase esmética.

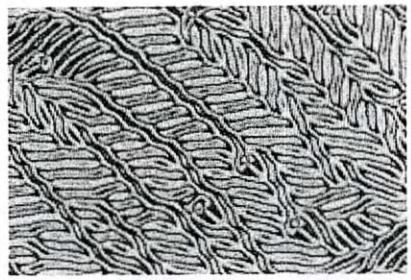


Figura 2.3. Cristal Líquido.

2.2. Fabricación de una LCD

Para construir una LCD se necesita más que simplemente crear una hoja de cristal líquido. La combinación de 4 factores es lo que hace posible una LCD.

- Poder polarizar la luz.
- El cristal líquido pueda cambiar y transmitir la luz polarizada.
- La estructura del cristal líquido pueda cambiar por la corriente eléctrica.
- Existan sustancias transparentes que puedan conducir electricidad.

Para crear una LCD se necesitan dos piezas de cristal polarizado. En el lado del cristal que no tiene película de polarización se instala un polímero rugoso que crea ciertas curvas microscópicas. Las curvas deben estar en la misma dirección que la película de polarización. Entonces se agrega un recubrimiento de cristal líquido nemático a uno de los filtros. Las curvas causaran que la primera capa de moléculas se alinee con la misma orientación del filtro. Luego se agrega la segunda pieza de cristal con la película de polarización en ángulo recto a la primera pieza. Cada capa sucesiva de moléculas TN (twisted nematic) gradualmente se torcerá hasta que la capa de más arriba se encuentre a un ángulo de 90 grados con respecto a la de más abajo, alineando los filtros de cristal polarizado. Al incidir la luz el primer filtro, este se polariza. Las moléculas en cada capa guían la luz que reciben hacia la siguiente capa. Como la luz pasa a través de las capas de cristal líquido, las moléculas también cambian el plano

de vibración de la luz para alinear a su propio ángulo. Cuando la luz alcanza el final de la sustancia de cristal líquido, vibra al mismo ángulo de la capa final de moléculas. Si la capa final se alinea con el segundo filtro de cristal polarizado, la luz pasa a través de el. Si se aplica una carga eléctrica a las moléculas de cristal liquido, se destuercen cuando se expande al salir, cambiando el ángulo de la luz que pasa a través de ellos y el ángulo ya no se alinea con el del filtro polarizado. Por consecuencia no puede pasar luz a través del área de la LCD, lo que hace que esta área sea más oscura que las áreas de alrededor. Construir una LCD simple es muy sencillo. Se empieza con un sándwich de vidrios y cristal liquido y se agregan dos electrodos transparentes. Para crear la LCD mas simple posible con un solo electrodo rectangular. Las capas de la LCD se verían así como se muestra en la figura 2.4.

2.3. LCD de Matriz Activa y Matriz Pasiva

Una matriz pasiva de LCD usa una simple rejilla para suministrar la carga a un píxel particular en el display. Crear una rejilla es todo un proceso. Inicia con dos capas de cristal llamadas sustratos. Un sustrato proporciona las columnas y el otro sustrato te proporciona los renglones hechos de un material conductivo transparente. Esto es usualmente oxido de indio-estaño. Los renglones o columnas están conectados a circuitos integrados que controlan cuando una carga se manda a un renglón o columna en particular. El material de cristal esta empaquetado entre dos sustratos de vidrio y se agrega una película de polarizacion al lado externo de cada sustrato. Para encender un píxel el circuito integrado manda una carga a la columna correcta de un sustrato y una tierra activa al renglón correcto del otro. El renglón y la columna intersectan al píxel designado y entrega el voltaje para destorcer al cristal liquido en ese píxel. La simplicidad del sistema de matriz pasiva tiene desventajas importantes, tiempo de respuesta notablemente lento y control de voltaje impreciso. El tiempo de respuesta se refiere a la habilidad de la LCD para refrescar la imagen desplegada. La manera más fácil para observar el tiempo de respuesta lento en una LCD de matriz pasiva es mover el puntero del mouse rápidamente de un lado de la pantalla a otro. Se notara una serie de fantasmas siguiendo al puntero. El control dificulta la

habilidad de la matriz pasiva para influenciar solo un píxel a la vez. Cuando el voltaje se aplica para destorcer un píxel, los píxeles alrededor parcialmente también se destuercen que la imagen parezca borrosa y falta de contraste. Las LCD de matriz activa dependen básicamente de transistores de películas delgadas (TFT), TFT's son pequeños transistores y capacitores que están cambiando como se observa en la Fig.2.4. Ellos están acomodados en una matriz sobre un sustrato de vidrio. Para direccionar a un píxel en particular el renglón apropiado se enciende y la carga es enviada a la columna correcta. Entonces todos los renglones y columnas que intersectan se apagan, solo el capacitor y el píxel asignado recibe una carga. El capacitor es capaz de mantener la carga hasta el siguiente ciclo de actualización. Y si controlamos cuidadosamente la cantidad de voltaje aplicado a un cristal, podemos destorcer solo lo suficiente para permitir cierta luz a través de él. Haciendo estos incrementos pequeños y muy exactos, las LCD's pueden crear una escala de gris. La mayoría de los displays ofrecen 256 niveles de gris por píxel [4].

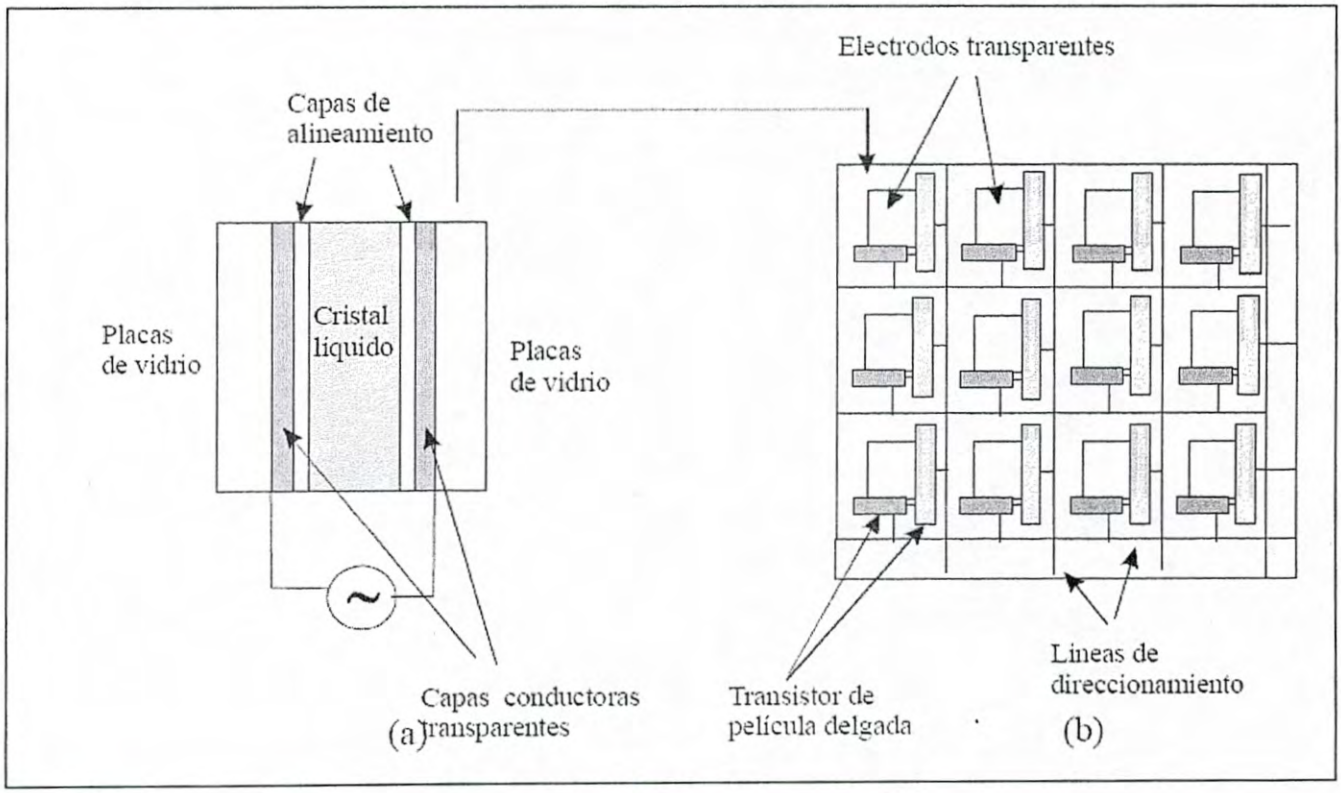


Fig. 2.4. Construcción de una LCD

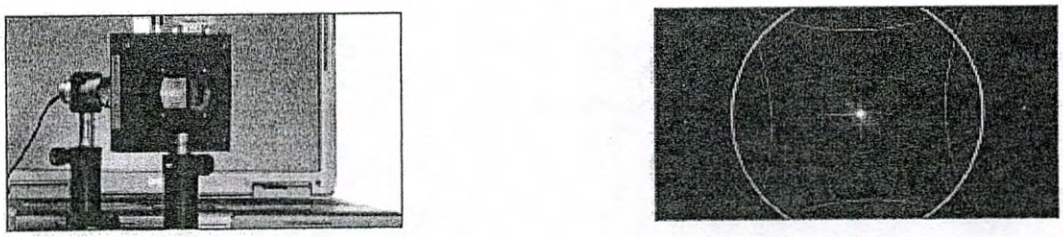
2.4. Modulador Electro-Óptico

El modulador electro-óptico (OEM) es un dispositivo óptico en el cual se usa una señal controlada desplegando efectos electro ópticos para modular un haz de luz. La modulación puede ser impuesta por la fase, modulación, frecuencia o la dirección del haz modulado. Los anchos de banda de modulación extendidos dentro del rango de los gigahertz se pueden lograr con el uso de moduladores de láser controlado.

2.5. Modulador Espacial de Luz Holoeye LC2002

Un modulador espacial de luz se basa en micro-displays de cristal liquido. Estos dispositivos pueden modular la luz espacialmente en amplitud y fase, por lo que pueden actuar como un elemento óptico dinámico. La función óptica o información que se desplegará se toma directamente del diseño óptico o de una imagen fuente y puede ser transferida por medio de una interfase de computadora. Su implementación es muy sencilla debido a su arquitectura de sistema inteligente y por su fácil direccionamiento usando señales VGA o DVI directamente de una tarjeta de gráficos de una computadora. El modulador espacial de luz utilizado en este trabajo fue el LC2002 de la empresa Holoeye, el cual fue más conveniente debido a su tamaño y resolución, además de que por su conexión se le puede cargar el programa generado para crear barras horizontales y poder difractar el rayo láser emitido a través de él. El LC2002 es un sistema modulador espacial de luz fácil de usar basado en un micro-display LC diseñado para investigación y desarrollo de prototipo industrial. Se puede utilizar para modular luz espacialmente, donde la función de modulación puede ser direccionada eléctricamente por una computadora usando un software en sistema operativo Windows. El LC 2002 soporta varios formatos de desplegado con una resolución máxima de 832 x 624 píxeles. Transmisión de alta eficiencia y propiedades de óptimo contraste garantizan excelente calidad óptica. Minimizando las dimensiones del dispositivo habilita la fácil integración en sistemas ópticos, en las figuras 2.5 se

muestra en modulador Holoeye LC2002 y la dispersión del rayo generado como ejemplo [6].



(a) (b)

Figura 2.5. Modulador Espacial de Luz LC2002: Figura 2.2. a). Presentación comercial, b) Dispersión de un rayo a partir de un objeto codificado.

El potencial mas alto de los moduladores espaciales de luz es su uso como un dispositivo de modulación de fase dinámica, el cual actúa como un elemento difractivo direccionable. Además despliega aplicaciones particulares de láser, como dispersión de rayo y forma, óptica difractiva, holografía digital y aplicaciones láser biológicas son las aplicaciones y desafíos mas importantes.

CAPITULO 3

DISPOSITIVOS LOGICOS PROGRAMABLES.

3.1. Introducción

La lógica programable, como el nombre implica, es una familia de componentes que contienen conjuntos de elementos lógicos (AND, OR, NOT, LATCH, FLIP-FLOP) que pueden configurarse en cualquier función lógica que el usuario desee y que el componente soporte. Hay varias clases de dispositivos lógicos programables: ASICs, FPGAs, PLAs, PROMs, PALs, GALs, y PLDs complejos. Anteriormente, los circuitos integrados digitales eran fijos en sus funciones lógicas y no era posible cambiar un diseño sin cambiar el chip en el circuito. La lógica programable ofrece a los diseñadores de circuitos la posibilidad de cambiar las funciones de diseño aun después de construido el circuito.

Un dispositivo lógico programable o PLD puede ser programado, borrado y reprogramado la cantidad de veces que se desee, permitiendo así modificaciones al diseño y al prototipo. El número de paquetes de circuitos integrados requeridos para implementar un diseño con uno o más PLDs son normalmente reducidos comparados con diseños con circuitos estándar de lógica fija.

3.2. ASIC

ASIC significa Circuitos Integrados de Aplicación Específica y son dispositivos definidos por el usuario. Los ASICs, al contrario que otros dispositivos, pueden contener funciones analógicas, digitales, y combinaciones de ambas. En general, son programables mediante máscara y no programables por el usuario. Esto significa que los fabricantes configurarán el dispositivo según las especificaciones del usuario, como se muestra en la figura. Se usan para combinar una gran cantidad de funciones lógicas en un dispositivo. Sin embargo, estos dispositivos tienen un costo inicial alto, por lo tanto se usan principalmente cuando es necesaria una gran cantidad. En la figura 3.1. se muestra la estructura básica de un ASIC.

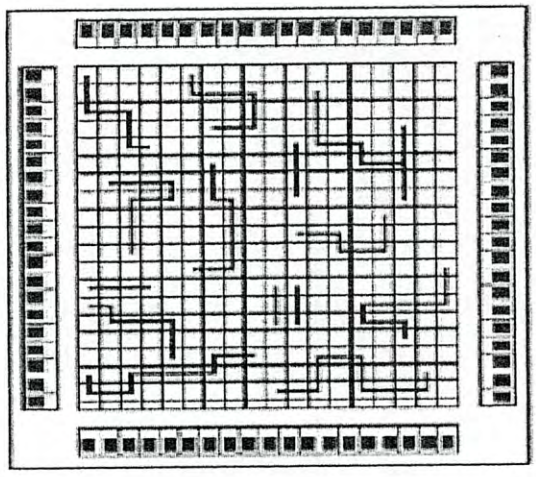


Figura 3.1. Estructura Básica de un ASIC.

3.3. Estructura básica de un PLD

Es un dispositivo programable por el usuario que contiene una arquitectura general predefinida en la que el usuario puede programar el diseño final del dispositivo empleando un conjunto de herramientas de desarrollo. Las arquitecturas generales pueden variar pero normalmente consisten en una o más matrices de puertas AND y OR para implementar funciones lógicas.

Muchos dispositivos también contienen combinaciones de flip-flops y latches que pueden usarse como elementos de almacenaje para entrada y salida de un dispositivo. Los dispositivos más complejos contienen macrocélulas, que permiten al usuario configurar el tipo de entradas y salidas necesarias en el diseño.

Los PLDs son ASICs (Applications Specific Integrated Circuits) configurables o programables por el usuario para realizar una determinada función lógica. Estos dispositivos consisten de una matriz o arreglo de compuertas AND seguida de otra matriz de compuertas OR mediante las cuales se pueden realizar las operaciones lógicas deseadas. [7]

Ya que los PLDs están situados en una zona intermedia en donde pueden sustituir cientos o miles de CIs (Circuitos Integrados) son muy utilizados para realizar operaciones básicas ya que su costo es

menor al de un FPGA.

3.4. PROM

Las PROM son memorias programables de sólo lectura. Aunque el nombre no implica la lógica programable, las PROM, son de hecho lógicas. La arquitectura de la mayoría de las PROM consiste generalmente en un número fijo de términos AND que alimenta una matriz programable OR. Se usan principalmente para decodificar las combinaciones de entrada en funciones de salida.

Por esto la memoria puede ser programada (pueden ser escritos los datos) una sola vez a través de un dispositivo especial, un programador PROM. Estas memorias son utilizadas para grabar datos permanentes en cantidades menores a las ROMs, o cuando los datos deben cambiar en muchos o todos los casos. [8]

Las PROM son excelentes en la implementación de lógica combinacional con un cierto número limitado de entradas y salidas. Son muy útiles en lógica secuencial, dispositivos con reloj externo como flip-flops o microprocesadores. Sin embargo, las PROM son extremadamente lentas por lo que no son muy utilizadas en aplicaciones donde la velocidad sea muy importante.

3.5. PAL

Las PAL son dispositivos de matriz programable. La arquitectura interna consiste en términos AND programables que alimentan términos OR fijos. Todas las entradas a la matriz pueden ser combinadas mediante AND entre si, pero los términos AND específicos se dedican a términos OR específicos. Las PAL tienen una arquitectura muy popular y son probablemente el tipo de dispositivo programable por usuario más empleado. Si un dispositivo contiene macrocélulas, comúnmente tendrá una arquitectura PAL así como en la figura 3.2 se muestra la arquitectura básica de una PAL.

Las macrocélulas típicas pueden programarse como entradas, salidas, o entrada/salida (e/s) usando una habilitación tri-estado. Normalmente tienen registros de salida que pueden usarse o no conjuntamente

con la patilla de e/s asociado. Otras macrocélulas tienen más de un registro, varios tipos de retroalimentación en las matrices, y ocasionalmente realimentación entre macrocélulas.

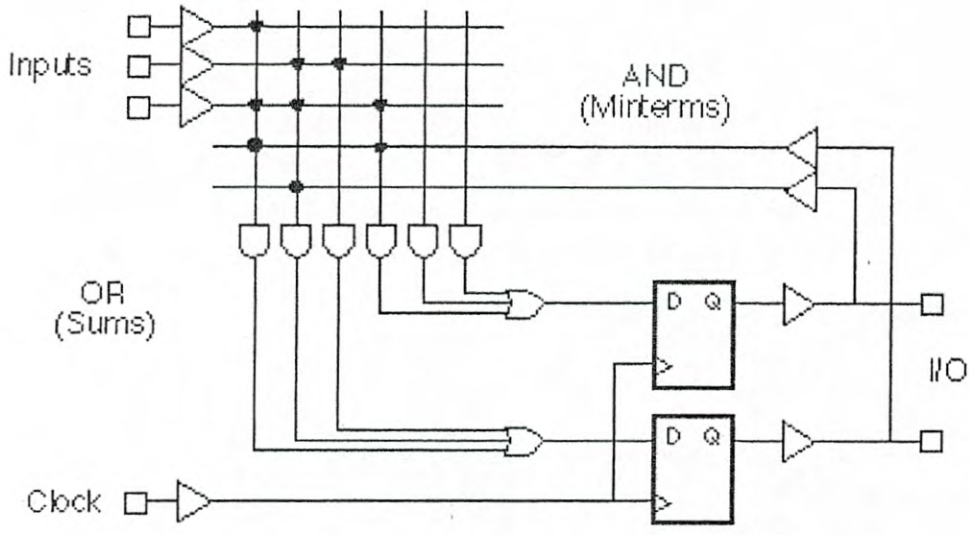


Figura 3.2. Arquitectura básica de PAL.

3.6. GAL

Las GAL son dispositivos de matriz lógica genérica. Están diseñados para emular muchas PAL pensadas para el uso de macrocélulas. Si un usuario tiene un diseño que se implementa usando varias PAL comunes, puede configurar varias de las mismas GAL para emular cada de uno de los otros dispositivos, esto reducirá el número de dispositivos diferentes en existencia y aumenta la cantidad comprada. Comúnmente, una cantidad grande del mismo dispositivo debería rebajar el costo individual del dispositivo. Estos dispositivos también son eléctricamente borrables, lo que los hace muy útiles para los ingenieros de diseño.

3.7. PLA

Las PLA son matrices lógicas programables. Estos dispositivos contienen ámbos términos AND y OR programables lo que permite a cualquier término AND alimentar cualquier término OR. Las PLA

probablemente tienen la mayor flexibilidad frente a otros dispositivos con respecto a la lógica funcional. Normalmente poseen realimentación desde la matriz OR hacia la matriz AND que puede usarse para implementar máquinas de estado asíncronas. La mayoría de las máquinas de estado, sin embargo, se implementan como máquinas síncronas. Con esta perspectiva, los fabricantes crearon un tipo de PLA denominado Secuencial (Sequencer) que posee registros de realimentación desde la salida de la matriz OR hacia la matriz AND como en la estructura básica de la figura 3.3.

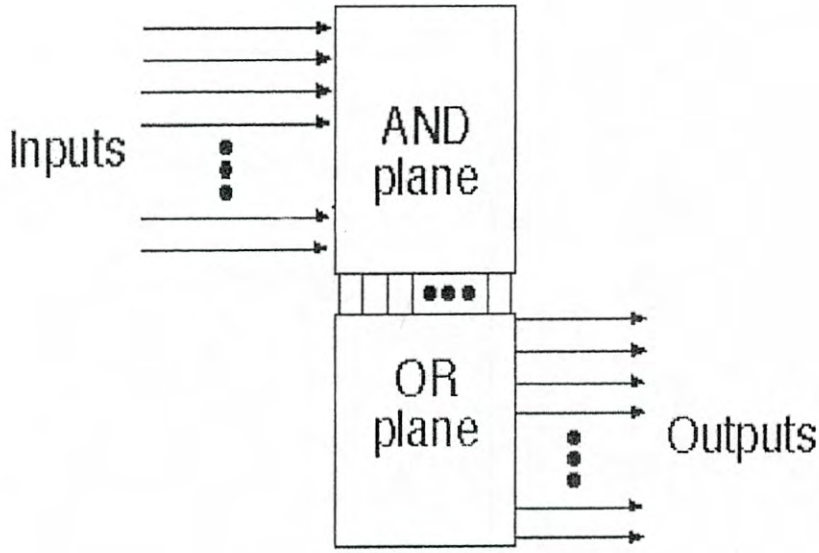


Figura 3.3. Arquitectura básica de PLA.

3.8. PLDs complejos

Los PLDs complejos son lo que el nombre implica, Dispositivos Complejos de Lógica Programable. Se consideran PAL muy grandes que tienen algunas características de las PLA. La arquitectura básica es muy parecida a la PAL con la capacidad para aumentar la cantidad de términos AND para cualquier término OR fijo. Esto se puede realizar quitando términos AND adyacentes o empleando términos AND desde una matriz expandida. Esto permite que cualquier diseño pueda ser implementado dentro de estos dispositivos. Aquí se muestra la estructura básica de un CPLD en la siguiente figura 3.4.

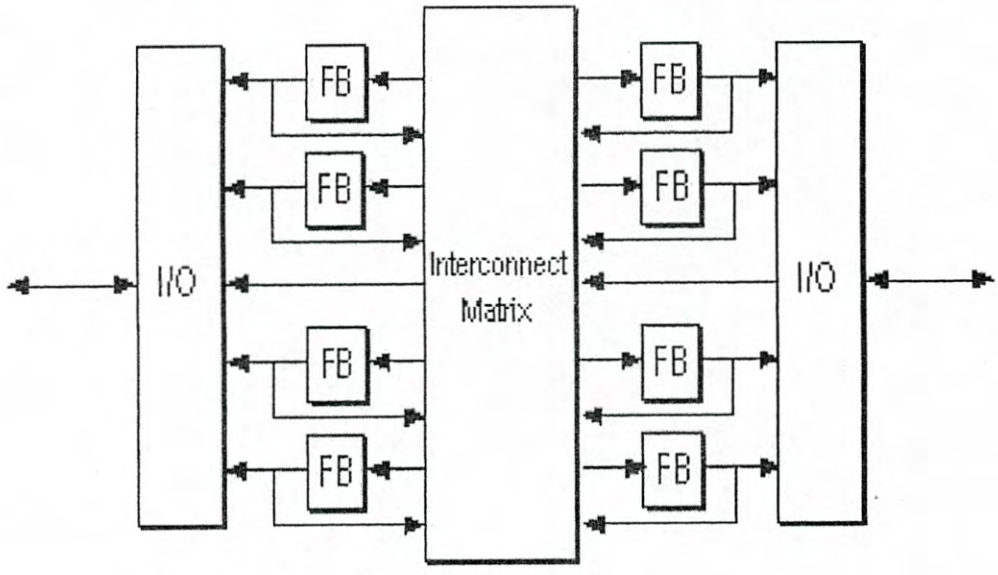


Figura 3.4. Arquitectura de CPLD.

3.9. FPGA

Los FPGA son Campos de Matrices de Compuertas Programables. Simplemente son matrices de puertas eléctricamente programables que contienen múltiples niveles de lógica. Los FPGA se caracterizan por altas densidades de puerta, alto rendimiento, un número grande de entradas y salidas definibles por el usuario, un esquema de interconexión flexible, y un entorno de diseño similar al de matriz de compuertas. No están limitadas a la típica matriz AND-OR. Sin embargo, contienen una matriz interna configurable de relojes lógicos (CLBs) y un anillo de circunvalación de bloques de e/s (IOBs).

Cada CLB contiene lógica programable combinacional y registros de almacenamiento. La sección de lógica combinacional es capaz de implementar cualquier función booleana de sus variables de entrada. Cada IOB puede programarse independientemente para ser una entrada, y salida con control de tres estados o patilla bidireccional. También contiene flip-flops que pueden usarse como buffers de entrada y salida. Los recursos de interconexión son una red de líneas que corren horizontalmente y

verticalmente las filas y columnas entre el CLBS como se muestra en la figura 3.5.

Los interruptores programables conectan las entradas y salidas de IOBS y CLBS a líneas cercanas. Las líneas largas recorren la anchura o longitud entera del dispositivo, estableciendo intercambios para proporcionar una distribución de señales críticas con la mínima demora o distorsión.

Los diseñadores que usan FPGAs pueden definir funciones lógicas en un circuito y revisar estas funciones como sea necesario. Así, los FPGAs pueden diseñarse y verificarse en unos días, a diferencia de las varias semanas necesarias para las matrices de compuerta programables. [9]

En este proyecto se utilizó un FPGA debido a que son sencillos de programar y la capacidad de almacenamiento es suficientemente grande para almacenar en memoria los datos y los diferentes niveles de gris también pueden ser generados.

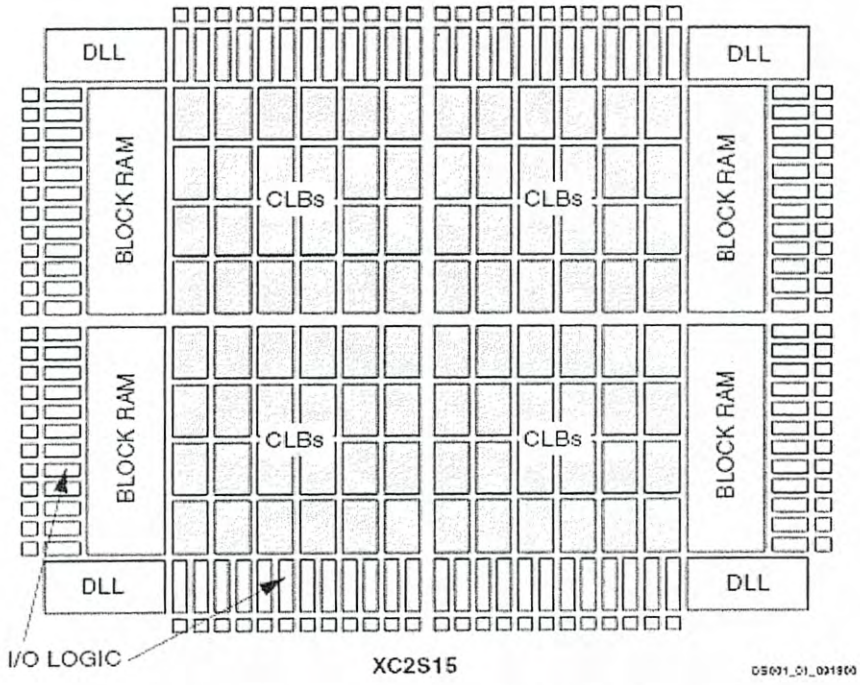


Figura 3.5. Diagrama de bloques del FPGA Spartan:

CAPITULO 4

ESTANDAR SVGA

VGA es una interfase que tiene la característica de separar las señales análogas de video red, green, blue, así como de separar pulsos de sincronización vertical y horizontal, que soporta por lo menos un canal DDC(Display Data Channel), un voltaje de +5V DC y conectores DB-15. Los pines están definidos en el Standard VESA E-DDC.[11]

4.1. Descripción Básica

El hardware de video produce una señal continua en el conector de salida, excepto cuando esta en el modo de reset, donde las salidas de video se mantienen en estado simple. La señal continua se necesita debido a que la información de píxeles solo se despliega por un periodo muy corto de tiempo y se mantiene durante el resplandor del fósforo en el monitor para que la vista pueda apreciar el promedio de la señal y así lograr ver una imagen estable. Esta señal es usualmente una salida de múltiples pines del conector del monitor, además que puede ser una salida compatible con TV. Las pantallas de LCD usan una técnica similar, además el tiempo de señal es más exacto y depende del tipo de panel y su circuitería. La señal incluye los datos de píxel que despliega el monitor así como la información de tiempos y cuadros (frames) que el video utiliza en su circuitería interna.

En la siguiente tabla 4.1. se muestran los tiempos necesarios par poder desplegar imágenes en VGA en las diferentes resoluciones y frecuencias de reloj.

Mode	Vesa					Samsung
Timing	VGA/ 72 Hz 640x480	SVGA/ 56 Hz 800x600	SVGA/ 72 Hz 800x600	EVGA/ 60 Hz 1024x768	EVGA/ 70 Hz 1024x768	S.1280/ 60 Hz 1280x1024
fh(kHz)	37.860	35.156	48.077	48.363	56.476	63.702
A µSec	26.413	28.444	20.800	20.677	17.701	15.698
B µSec	1.270	2.000	2.400	2.092	1.813	1.359
C µSec	4.603	3.556	1.280	2.462	1.920	1.811
D µSec	20.317	22.222	16.000	15.754	13.653	12.075
E µSec	0.762	0.667	1.120	0.369	0.320	0.453
fv(Hz)	72.809	56.250	72.187	60.000	70.069	60.100
O mSec	13.735	17.778	13.853	16.667	14.272	16.640
P mSec	0.079	0.057	0.125	0.124	0.106	0.047
Q mSec	0.740	0.626	0.478	0.600	0.513	0.471
R mSec	12.678	17.067	12.480	15.880	13.599	16.075
S mSec	0.238	0.028	0.770	0.062	0.053	0.047
Clock Freq.MHz	31.500	36.000	50.000	65.000	75.000	106.000
Polarity Horiz.	Negative	Pos/Neg	Positive	Negative	Negative	Negative
Vert.	Negative	Pos/Neg	Positive	Negative	Negative	Negative
Remark						Separate Composite Sync on green

Tabla 4.1. Tiempos para los diferentes tipos de resolución.

Las imágenes de píxel son "escaneadas" en la pantalla de izquierda a derecha, de arriba hacia abajo y es interrumpido por los periodos de punto o píxeles, que son controlados por el reloj. Cada línea de "escaneo" horizontal se le llama refrescado horizontal ya que "refresca" la información en la pantalla en una línea horizontal. Todas estas líneas escaneadas (la cantidad depende del modo de video), de arriba a abajo, hacer un refrescado vertical conocido como "frame" o cuadro. Se hacen muchos refrescados verticales por segundo, es decir, la imagen completa se actualiza muchas veces por segundo, lo cual es imperceptible, donde un refrescado mas rápido produce una imagen mas definida y menos parpadeo, para lograr esto se toman en cuenta los tiempos necesarios para cada tipo de resolución, como se muestra en la siguiente tabla.

4.2 Tiempo Horizontal

Desde el punto de vista del monitor, el tiempo es relativamente sencillo. Detecta el pulso de sincronía horizontal en la línea de hsync, luego basado en la polaridad, frecuencia y duración de esos pulsos inicia el escaneo horizontal a través de la pantalla a la frecuencia deseada. Durante este periodo se despliega continuamente la señal de entrada a los pines de RGB. Es importante conocer los rangos de frecuencia horizontal del monitor, así como la amplitud del pulso de esos rangos de sincronía. Si la amplitud del pulso de sincronía es incorrecto, puede hacer que la información desplegada sea muy grande o muy pequeña. El rango aceptable de amplitud de los pulsos de sincronía y polaridad de una frecuencia específica están dadas en las especificaciones del monitor, como se muestra en la tabla 4.2.

Mode name	Pixel clock	sync pulse	back porch	active time	front porch	whole period	line
	(MHz)	(us)	(pix)	(pix)	(pix)	(pix)	(pix)
VGA 640x480 60Hz	25.175	3.81	96	45	646	13	800
VGA 640x480 72Hz	31.5	1.27	40	125	646	21	832
VGA 720x400 70Hz	28.322	3.81	108	51	726	15	900
VGA 720x350 70Hz	28.322	3.81	108	51	726	15	900
VGA 800x600 56Hz	36	2	72	125	806	21	1024
VGA 800x600 60Hz	40	3.2	128	85	806	37	1056
VGA 800x600 72Hz	50	2.4	120	61	806	53	1040
IBM 640x480 75Hz	31.5	3.05	96	45	646	13	800
MAC 640x480 66Hz	30.24	2.11	64	93	646	61	864

Tabla 4.2. Tiempos de señal Horizontal

4.3. Tiempo Vertical.

La medición de tiempos vertical es casi lo mismo que el horizontal, excepto que controla el movimiento vertical del escaneo de la pantalla, espaciando las líneas de escaneo para hacer la forma de una imagen rectangular. El monitor detecta el pulso de sincronía vertical en la línea de vsync, luego basado en la polaridad, frecuencia y duración de esos pulsos inicia el escaneo vertical a través de la

pantalla a la frecuencia deseada como se muestra en la tabla 4.3. Es necesario conocer el rango de la frecuencia de sincronía vertical del monitor y el rango de amplitud de pulso de sincronía vertical y la polaridad de esos rangos. [12]

Mode name	Lines Total	line width	sync pulse		back porch		active time		front porch		whole frame period	
		(us)	(us)	(lin)	(us)	(lin)	(us)	(lin)	(us)	(lin)	(us)	(lin)
VGA 640x480 60Hz 16683 525	525	31.78	63		2	953	30	15382	484		285	9
VGA 640x480 72Hz 13735 520	520	26.41	79		3	686	26	12782	484		184	7
VGA 720x400 70Hz 14268 449	449	31.78	63		2	1016	32	12839	404		349	11
VGA 720x350 70Hz 14268 449	449	31.78	63		2	1811	57	11250	354		1144	36
VGA 800x600 56Hz 17775 625	625	28.44	56		1	568	20	17177	604			-1*
VGA 800x600 60Hz 16579 628	628	26.40	106		4	554	21	15945	604			-1*
VGA 800x600 72Hz 13853 666	666	20.80	125		6	436	21	12563	604		728	35
IBM 640x480 75Hz 13333 525	525	25.397	51		2	761	30	12292	484		228	9
MAC 640x480 66Hz 14999 525	525	28.57	86		3	1057	37	13827	484		28	1

Tabla 4.3. Tiempos de Señal Vertical.

Una vez programados la frecuencia de sincronía vertical y la amplitud del pulso, el hardware de generación de desplegado tiene que controlar otras salidas, cuando la salida sobrescanea y cuando realiza el blanqueo.

El sobrescaneo es el límite superior e inferior de la pantalla, si se presenta consiste en una o mas líneas en las que el periodo de desplegado activo es remplazado por el color de sobrescaneo.

El blanqueo se usa durante un retrazo vertical y consiste de una o mas líneas en las que el periodo de desplegado y el sobrescaneo se remplazan con el periodo de blanqueo, haciendo toda la línea de blanqueo.

Esto previene intensidades altas en el monitor durante el retrazo vertical cuando regresa hacia arriba al punto inicial del barrido de la pantalla. Intensidades de salida que no sean cero durante este periodo se escriben en un patrón de zig-zag desde en inferior hasta la parte superior de la pantalla. En el generador de desplegado, los tiempos verticales se especifican en la cantidad de tiempos horizontales necesarios para generar un "frame" o barrido completo de pantalla, así como se muestra en la figura 4.1. [13]

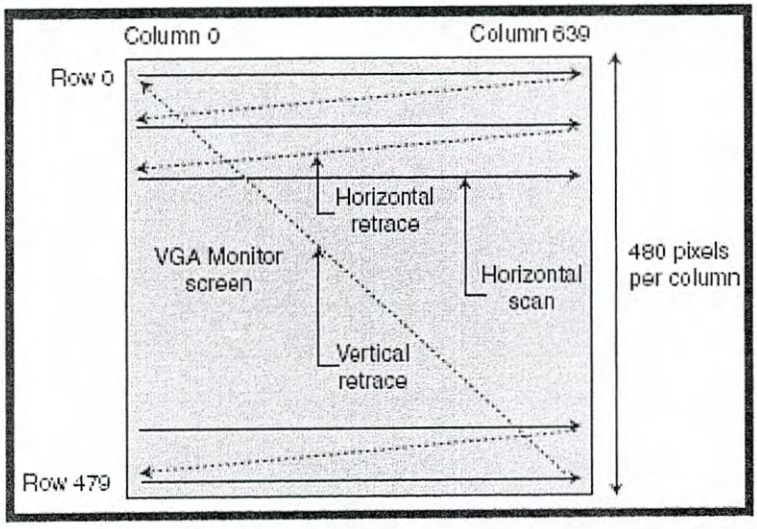


Figura 4.1. El escaneo inicia en el renglón 0 y columna 0 y se mueve a la derecha y hacia abajo hasta llegar al renglón 479 y columna 639.

4.4. Standard VESA E-DDC.

El propósito de este Standard es definir un canal de comunicaciones entre un display electrónico (CRT, LCD, etc., displays) y un sistema que lo reciba (host). El canal se usa para transportar información de configuración para habilitar el "plug & play" y permitir el óptimo uso del display. El canal también transporta información de control del display. [11]

4.5. Señales de Control.

Los monitores VGA son controlados por cinco señales: red, green, blue, sincronía horizontal y sincronía vertical. Las señales de los tres colores se refieren a la señal RGB, controlan el color de un

píxel en la pantalla. Son señales análogas que trabajan en el rango de 0 a 0,7 V las cuales pasan por un convertidor digital-analógico en cual genera los diferentes niveles de colores como se muestra en la figura 4.2.

Las diferentes intensidades de color se obtienen variando el voltaje, las señales de sincronización horizontal y vertical se utilizan para controlar el tiempo de rango de escaneo.

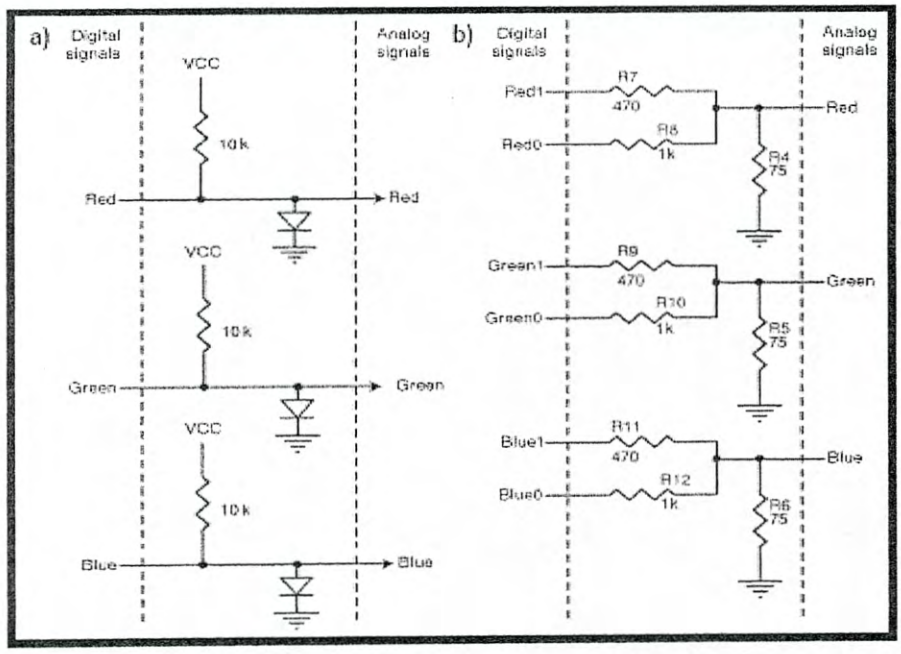


Figura 4.2. Convertidor Digital Analógico que controla las señales RGB. Para 8 colores (a). Para 64 colores (b).

4.6. Especificaciones para el controlador VGA de 640x480.

Para controlar una imagen que se muestra en un monitor VGA es necesario generar las señales de barrido de píxeles horizontal como vertical, con cierto tiempo de duración según las especificaciones. Los píxeles se leen de izquierda a derecha en cada renglón y las columnas de arriba hacia abajo.

Para poder ver correctamente la imagen desplegada es necesario crear señales de sincronía para renglones y columnas ya que estas deben tener tiempos de duración precisos en el estado alto como en

el bajo. Cuando la señal esta en alto, se envían los nuevos datos de colores para cada píxel.

El barrido empieza con la fila 0 hasta contabilizar las 800 columnas, y luego el adicional que permite la generación de sincronía. Así la cuenta en cada renglón va de 0 hasta 1040. Con una señal de reloj a 25.175MHz, es necesario mantener en bajo la señal de sincronía vertical cuando la cuenta este entre 859 y 979 y el resto en alto para generar la sincronía de columnas, esta señal de sincronía se debe generar para cada renglón (figura 4.3.a).

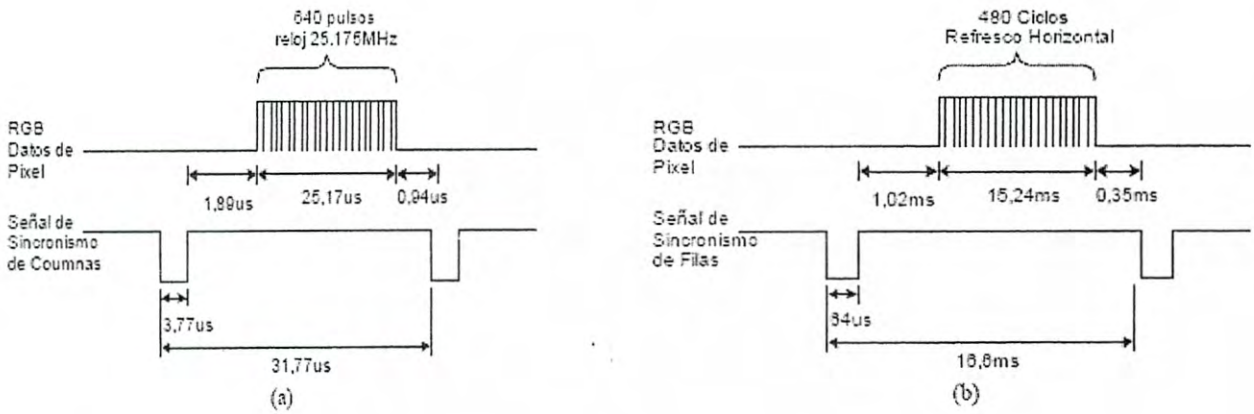


Figura 4.3. Señales de Sincronía.
a) Generación de Sincronía de Columnas. b) Generación de Sincronía de Renglones

Para el caso de las columnas la situación es similar, aunque se especifican 600 se debe de contar de 0 a 806, para poder generar la señal de sincronía correspondiente. La señal de sincronía de renglón (horizontal) debe permanecer en bajo los valores 653 y 659 y el resto en alto, lo cual permite cumplir con los requisitos de la figura 4.3.b.

Puesto que la resolución es de 600 renglones por 800 columnas, el conteo se hace de 806 por 1040 respectivamente, para crear la sincronía con el monitor y así poder generar la imagen deseada como se muestra en la figura 4.4. [14].

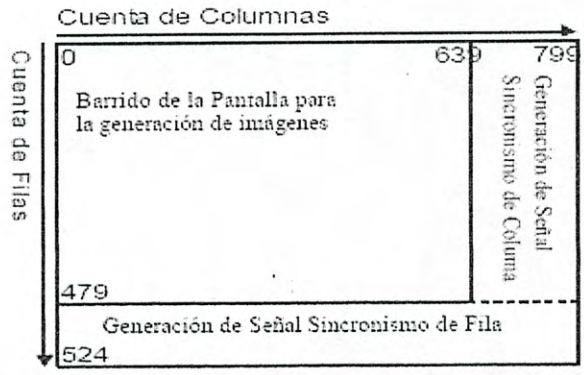


Figura 4.4. Zona de generación de señales de sincronía horizontal y vertical

En caso de que las señales de sincronía no fuesen generadas en forma precisa, con las duraciones de tiempo indicadas en la figura 4.3, no se tendría una correcta resolución con el monitor y resultaría una imagen "nula" (ausencia de color).

VGA es un estándar Industrial en modo de 800 x600 píxeles, que trabaja con una frecuencia de reloj a 50 MHz, con una frecuencia por línea de 31469 Hz y una frecuencia de campo de 59.94 Hz, así como las señales de sincronía vertical y horizontal con una polaridad negativa.

Para generar una línea de 1040 píxeles se crean 8 píxeles para el porch frontal, 96 píxeles para la señal de sincronía horizontal, 40 para el porch posterior, 8 píxeles para el límite izquierdo y 8 para el límite derecho así como 800 píxeles para video.

Para generar un campo se toman 2 líneas de porch frontal, 2 líneas de sincronía vertical, 25 líneas de porch posterior, 8 líneas para el limite superior así como para el inferior y 600 líneas de video.[15]

CAPITULO 5

DISEÑO DEL CONTROLADOR

5.1. Planteamiento del Problema.

Como se ha mencionado con anterioridad el objetivo de esta tesis es realizar el control de un Modulador espacial de luz de cristal liquido que opera con señal de video bajo el estandar SVGA. Para ello diseñamos un circuito digital que genera las señales con las que opera el modulador. En este proyecto se realizo el diseño del sistema con lenguaje VHDL implementado en la tarjeta XUP Virtex II Pro en la cual se reproducen patrones de imagen de 256 diferentes niveles de grises que son desplegados en el Modulador Espacial de Luz (SLM) con una resolución de 800 x 600 píxeles. Se crearon diferentes diseños en los que se puede generar diferentes patrones como son: rejillas multinivel, y un controlador que desplaza lentes difractivas para lograr el desplazamiento de un haz de iluminación apartir de un laser que es difractado por el patron desplegado en el modulador. Esta ultima es la requerida para el escaneo de objetos tridimensionales (que es el objetivo general del proyecto). Los datos de estos patrones fueron almacenados en una memoria ROM.

La propuesta de diseño que sirve para el sistema de escaneo se muestra en la figura 5.1, donde se utiliza como rejilla por lo que el controlador requiere de la señal de reloj (CLOCK) para sincronizar el sistema, una señal de control (A0) proveniente de otro sistema digital que nos dice cuando arrancar con el proceso. Un bloque de una maquina de estados finitos que es la que se encarga de especificar exactamente que evento se llevara a cabo (despliegue de un patron, secuencias de lectura a la memoria y tiempos de despliegue), las señales de salida de la maquina de estados se conectan a las entradas de un multiplexor que es el que se encarga de dividir la señal para los niveles de gris. El ultimo bloque es el que genera las señales de sincronia de VGA que van a las entradas del Convertidor Digital-Analogico (DAC por sus siglas en ingles) proporcionando la señal analogica que necesita el modulador.

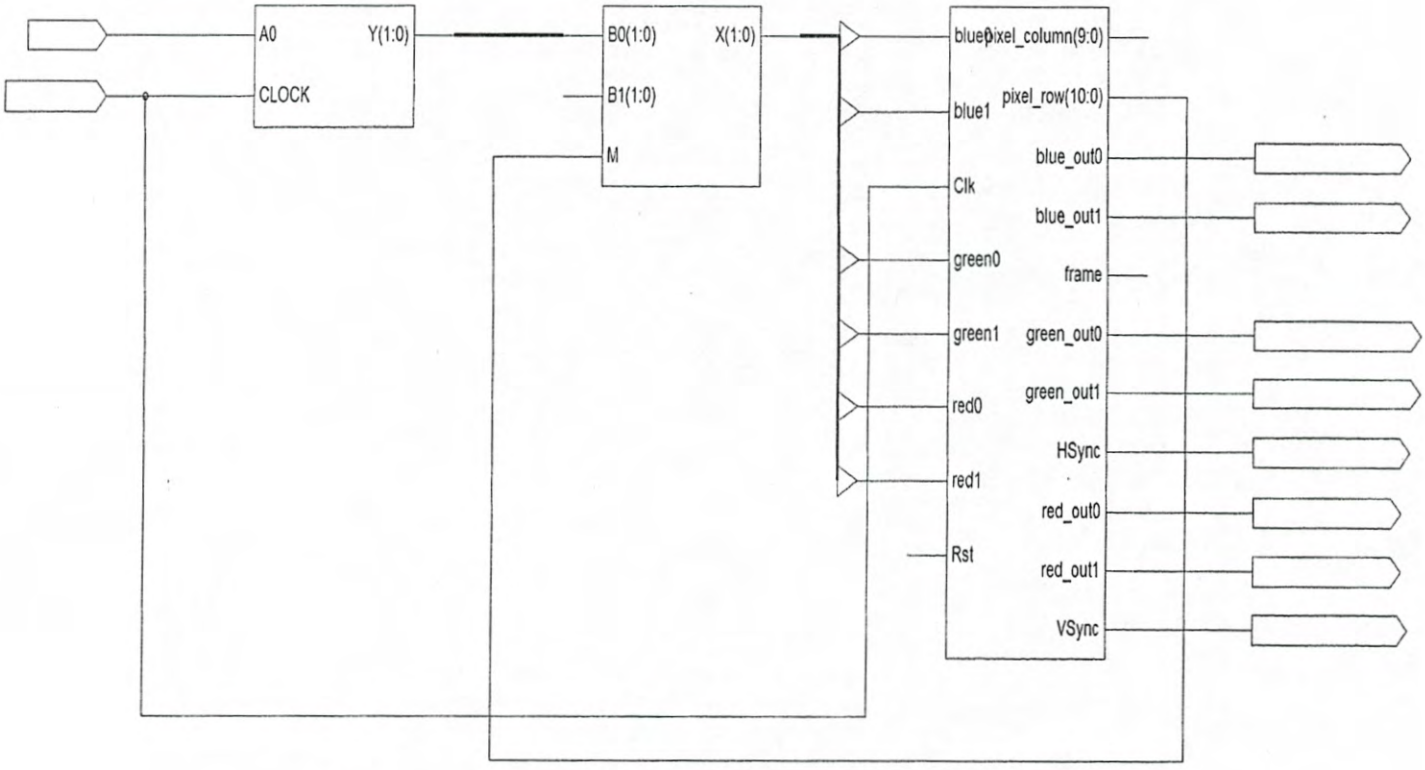


Figura 5.1. Diagrama de Bloques del programa principal

5.2. Controlador de Reloj

Para el controlador de reloj se hizo una adaptación de un proyecto para VGA que se utilizaba para un controlador de VGA de resolución de 640x480 y una velocidad de100 MHz y se adapto para resolución que necesitamos de 800x600 y 50 MHz para el controlador de VGA.

Este controlador esta diseñado en lenguaje Verilog el cual divide la señal de reloj para adaptarla a cualquier velocidad ya sea mayor o menor mediante definiciones de librerías que se encuentran en el programa ISE de Xilinx, aquí se muestra el diagrama en la figura 5.2 en la cual se genera una señal de reloj adaptada para VGA.

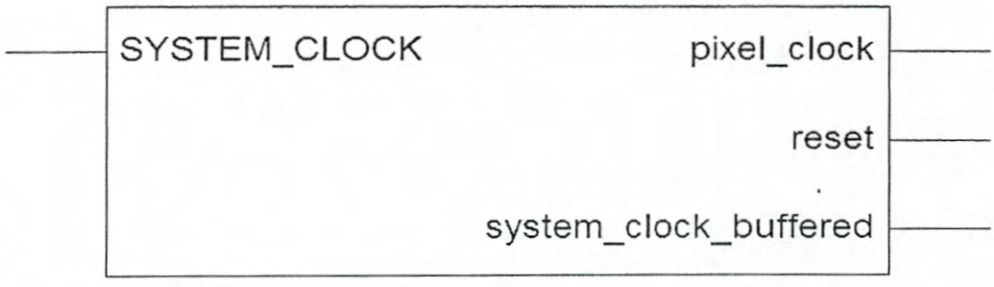


Figura 5.2. Diagrama a bloque de controlador de reloj.

5.3. Controlador de VGA

En este diseño se especifica la resolución a desplegar así como el número de renglones y columnas, se utilizan la señal de reloj, un reset, señales de RGB, sincronía horizontal y vertical. Se declaran señales de habilitación así como constantes para que el desplegado de la imagen quede dentro de la región visible, se hace la asignación de sincronía horizontal y vertical y se realizó un proceso para hacer un ciclo con los conteos vertical y horizontal, además de un proceso controlador del reloj para incrementar el conteo en cada pulso de reloj, en la figura 5.3. a) se muestra el diagrama interno.

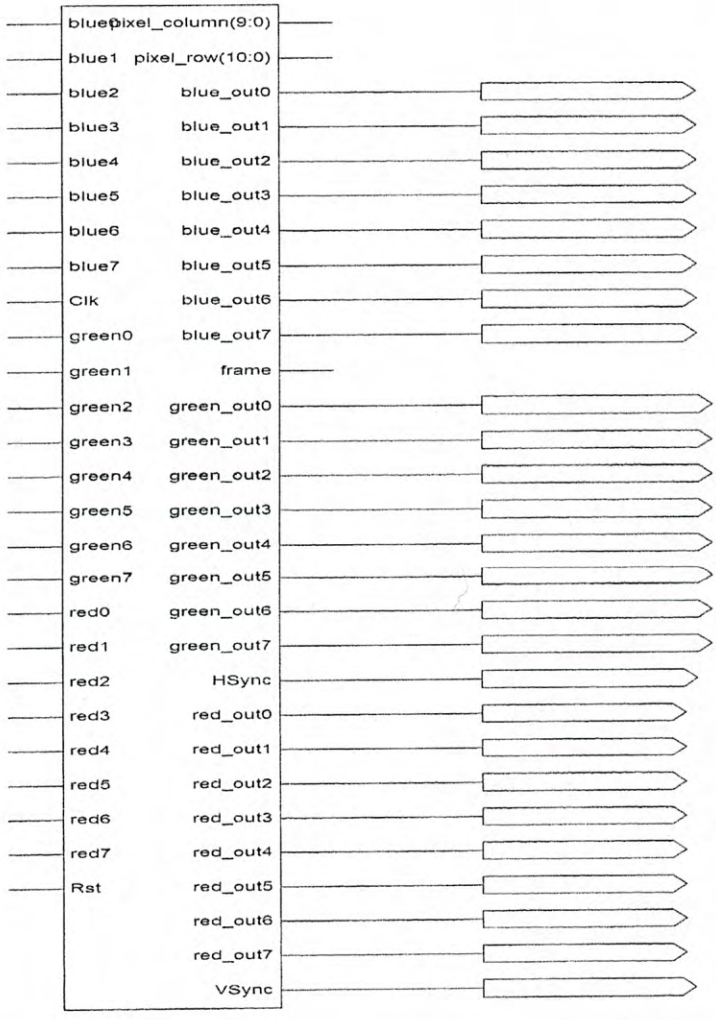


Figura 5.3. a) Diagrama Interno del controlador de VGA generado en ISE.

El programa cuenta con entrada de reloj, una entrada para controlar los niveles de gris y salida de

buffer de 8 bits para generar los 256 estados diferentes. Se generan señales para cada uno de los estados, además de una señal para el estado presente y otra para el estado siguiente, después de esto se genera un proceso el cual controla el reloj para que el estado presente cambie en cada pulso de reloj al estado siguiente cada vez que cambie de nivel de gris.

El segundo proceso controla los estados o niveles de gris generados mediante la activación de la entrada de niveles de gris.

El siguiente diagrama explica el funcionamiento del código de VGA que se utiliza, el cual trabaja con un reloj de 50MHz con una resolución de 800x600 píxeles, donde la primera parte se refiere al conteo horizontal y al incremento del mismo hasta llegar al máximo de 1040 píxeles.

La segunda parte del diagrama se refiere al conteo vertical donde cada vez que el conteo horizontal llegue al máximo el conteo vertical se incrementara uno, hasta llegar al máximo del conteo vertical y así se generara un cuadro completo de pantalla y se reiniciará de nuevo.

La tercera parte se refiere al control de las señales de video para que se encuentren dentro del rango especificado de 800x600 píxeles.

La cuarta y quinta parte se encarga del control del pulso de sincronía horizontal y vertical para que los pulsos activos se encuentren dentro del rango visible y así proyectar la imagen deseada.

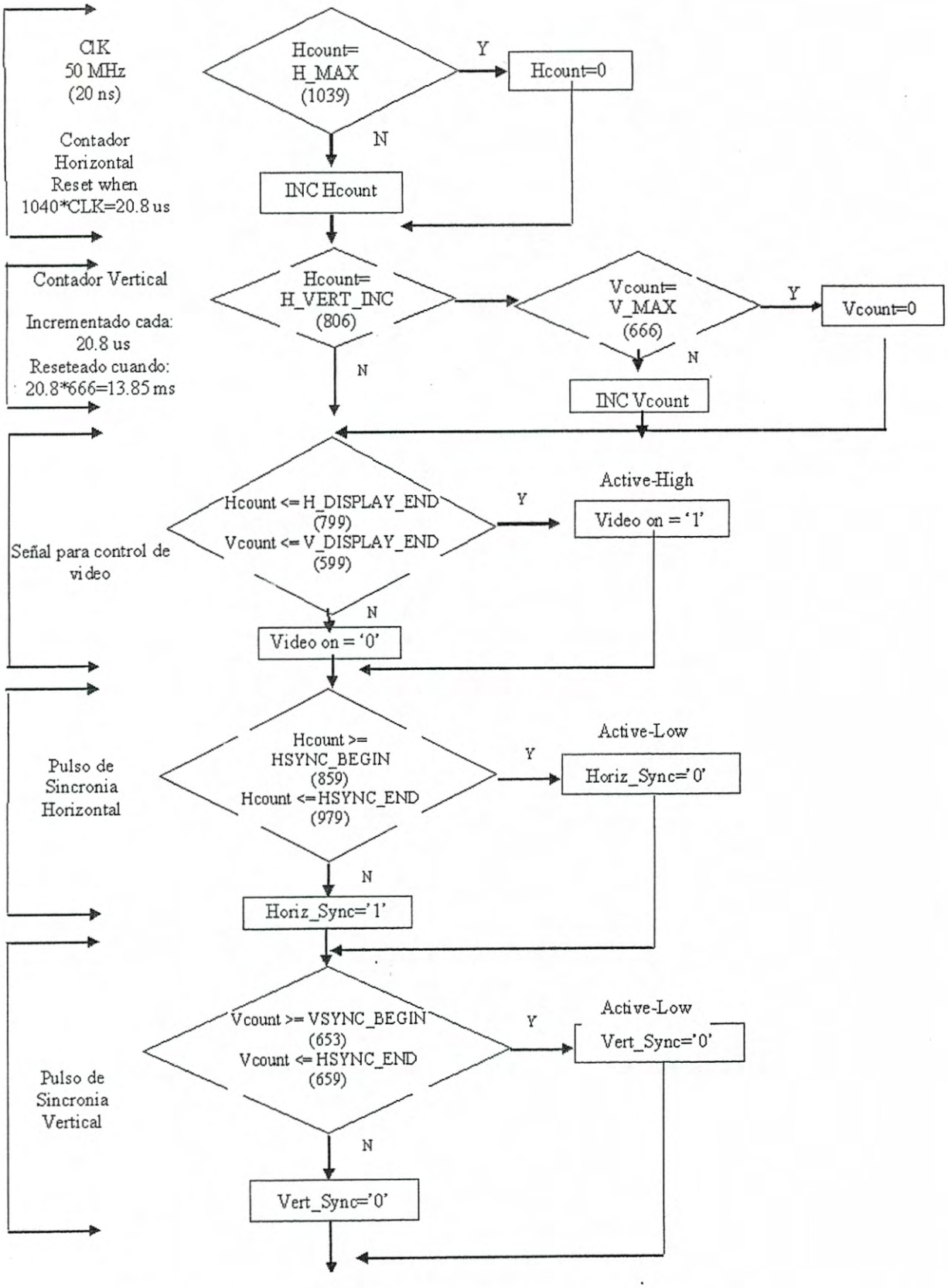


Figura 5.3. b) Diagrama de flujo del controlador de VGA

En la figura 5.3. c) se muestra el diagrama de tiempos del controlador de VGA para una resolución de 800x600.

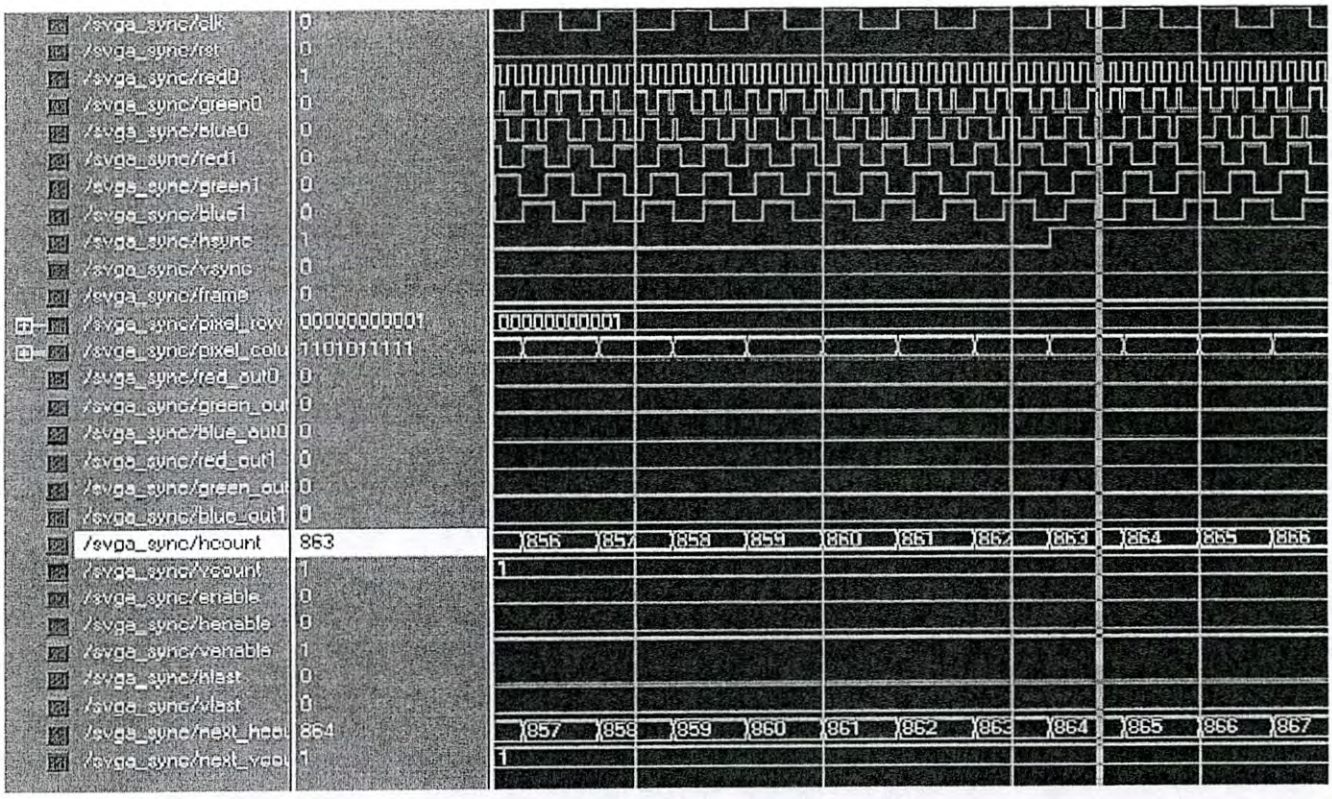


Figura 5.3. c) Diagrama de tiempos del controlador de VGA.

5.4. Multiplexor

Se generó un multiplexor de dos entradas y una salida, de 8 bits para tomar como primer entrada la salida de la maquina de estados y la segunda entrada se fijó a "00000000" (el nivel mas oscuro, es decir, negro) y con la selectora se cambiara cada nivel en la primer entrada, así también la selectora se asignó al controlador de VGA para generar franjas horizontales de 2 píxeles en el controlador, por lo que en la LCD se proyectan en toda la pantalla franjas horizontales una de color negro ("00000000") y otra con el nivel respectivo de gris.

El programa cuenta con un proceso que mediante la entrada selectora asigna a la salida el nivel de gris que este activado en la entrada proveniente de la maquina de estados, mientras la otra entrada permanece en ceros siempre, en la siguiente figura 5.4.a) se muestra el diagrama a bloques generado en

el programa ISE.

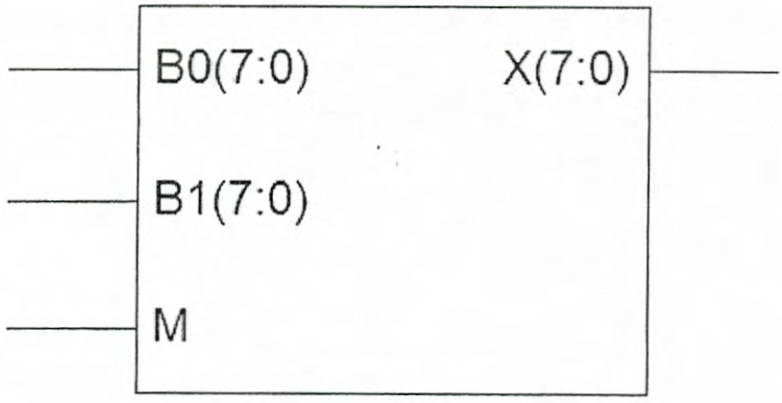


Figura 5.4. a) Diagrama a bloques del Multiplexor.

En la figura 5.4. b) se muestra el diagrama de tiempos generado para lograr los diferentes niveles de salida.

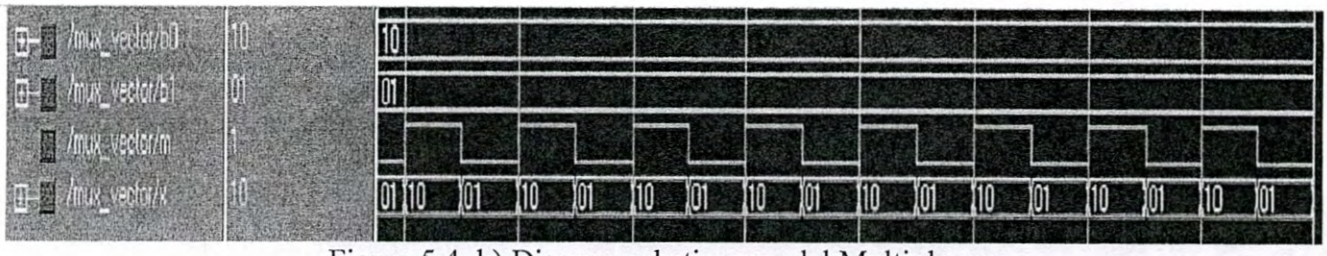


Figura 5.4. b) Diagrama de tiempos del Multiplexor

5.5. Memoria

Esta es una memoria ROM que se genero unidimensionalmente para asignar las valores de niveles de gris y la secuencia que debe seguir al desplegar en pantalla. Naddress es el valor tomado de píxel_row de las señales de sincronía de VGA y este es el valor que se le asigna a la señal change la cual esa determinada por el contador para cada salto en píxeles, en la siguiente figura 5.5. a) se muestra el diagrama de bloques de la memoria ROM.

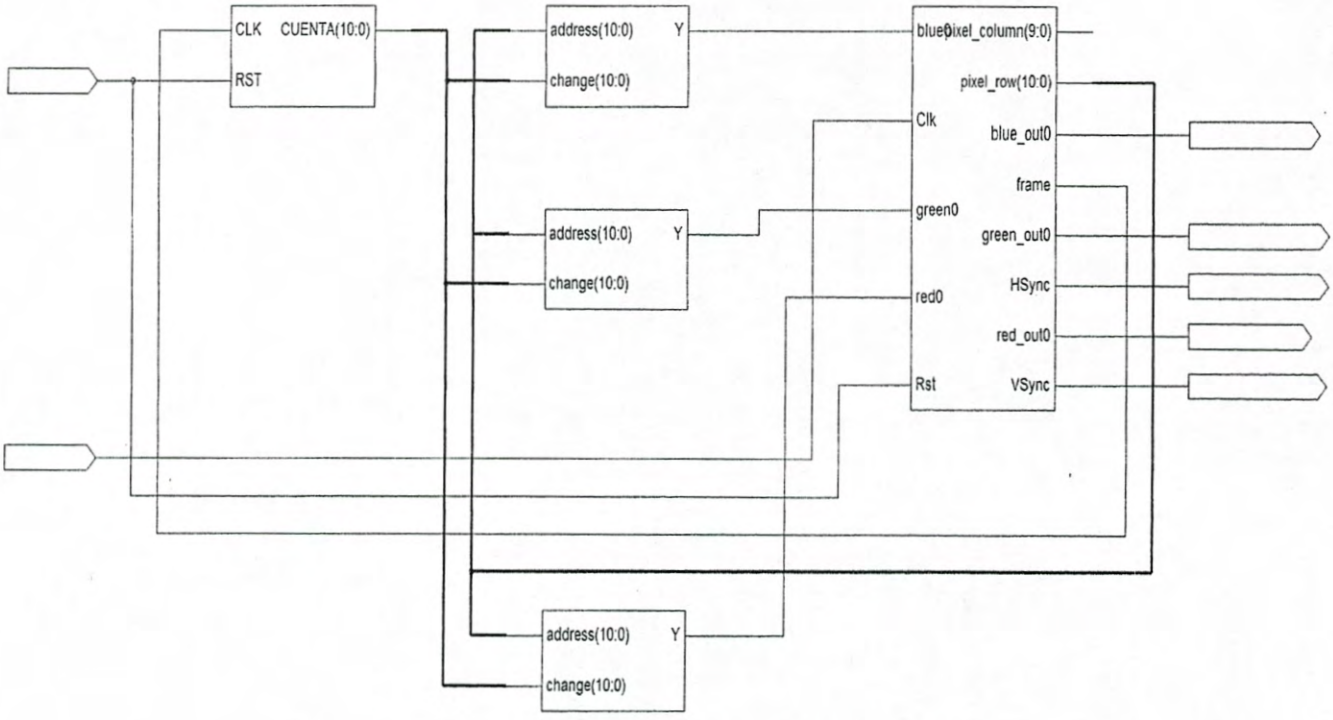


Figura 5.5. Símbolo esquemático de memoria generada.

5.6. Maquina de Estados

Se realizó un programa que trabaja como maquina de estados para generar cada uno de los niveles de gris como una secuencia por medio de un botón externo de 256 estados. En la figura 5.6. a) se muestra el grafo de la maquina de estados finitos generada.

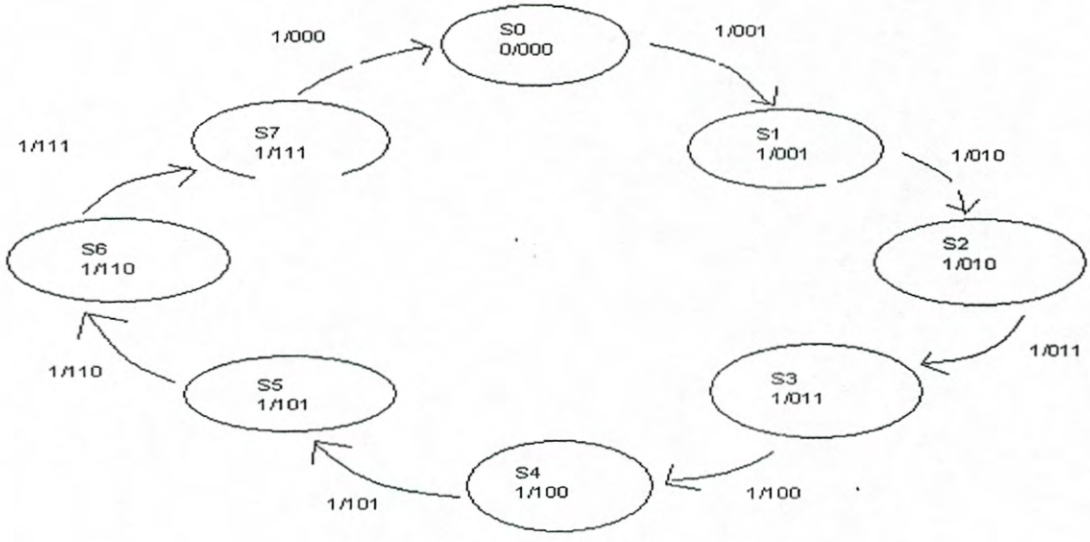


Figura 5.6. a) Grafo de la Maquina de Estados Finitos

Esta maquina utiliza dos procesos, uno para trabajar con cada flanco de reloj y el otro con un CASE para generar cada uno de los estados, este CASE trabaja con el botón para que a cada pulsación cambie de estado y la salida se envía a un Multiplexor, en la figura 5.6. a) se muestra el diagrama generado en ISE, así como el diagrama de tiempos de la maquina de estados finitos.

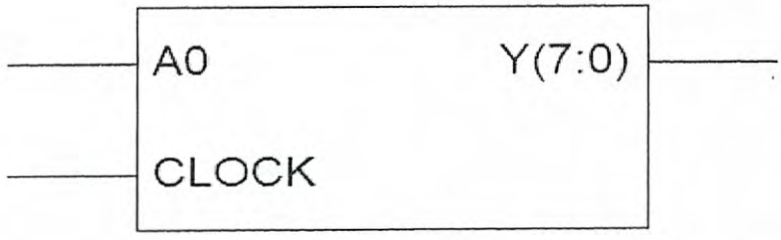


Figura 5.6. b) Diagrama a bloque de Maquina de Estados.

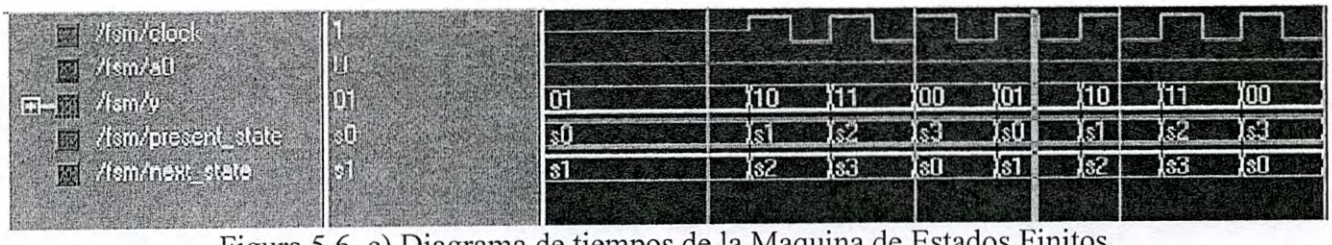


Figura 5.6. c) Diagrama de tiempos de la Maquina de Estados Finitos.

5.7. Bloque Principal

Este programa se encarga de hacer el llamado de todos los componentes del programa y se encarga de realizar las conexiones entre ellos y también de generar los puertos de entrada y salida de la tarjeta.

Este programa se tomó de un proyecto anterior realizado y se le hicieron las modificaciones necesarias para adaptarlo a las necesidades de este proyecto, en el cual se generan niveles de gris en una resolución de 640x480 y se adaptó para la resolución de 800x600, así como la velocidad del reloj ya que trabajaba a 50Mhz y se modificó a 100Mhz para adaptarlo a la tarjeta XUP Virtex.

Así también se modificaron la cantidad de niveles de gris debido a que esta tarjeta cuenta con un DAC de 8 bits que genera 256 niveles diferentes.

CAPITULO 6

RESULTADOS

Como una prueba del controlador diseñado se realizó una adecuación del sistema para desplegar rejillas (barras horizontales) con los distintos niveles de gris antes mencionados en VGA en 600x800. Para cambiar los niveles de gris desplegados se empleo como entrada la señal proveniente de un botón externo En este diseño se utiliza una maquina de estados para enviar las salidas a un multiplexor para generar barras horizontales de 2 píxeles. Las salidas del multiplexor se asignan como entradas al de sincronía de VGA para generar las salidas de VGA al área visible de un monitor de computadora. El resultado de la implementación de este sistema se muestra en la Figura 6.1., donde barras horizontales son desplegadas

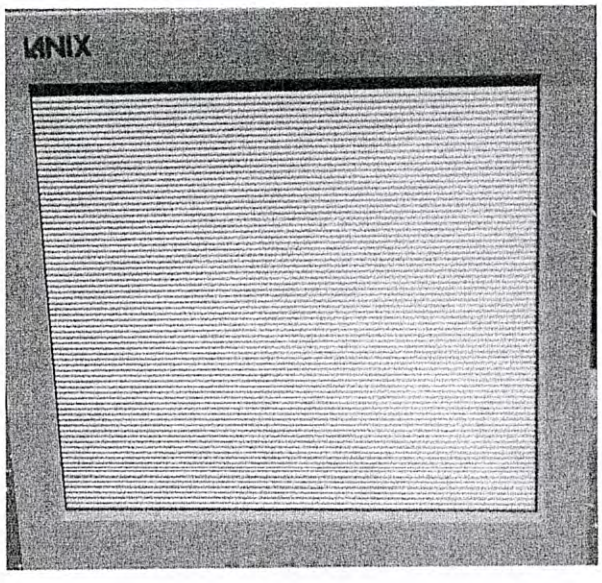


Figura. 6.1. Despliegue de barras horizontales

Para comprobar la operación del controlador desplegando lentes difractivas se colocó al modulador de luz en un sistema optoelectrónico como el que se muestra en la figura 6.2, donde un haz de luz láser incide sobre el modulador (LCOS-SLM)

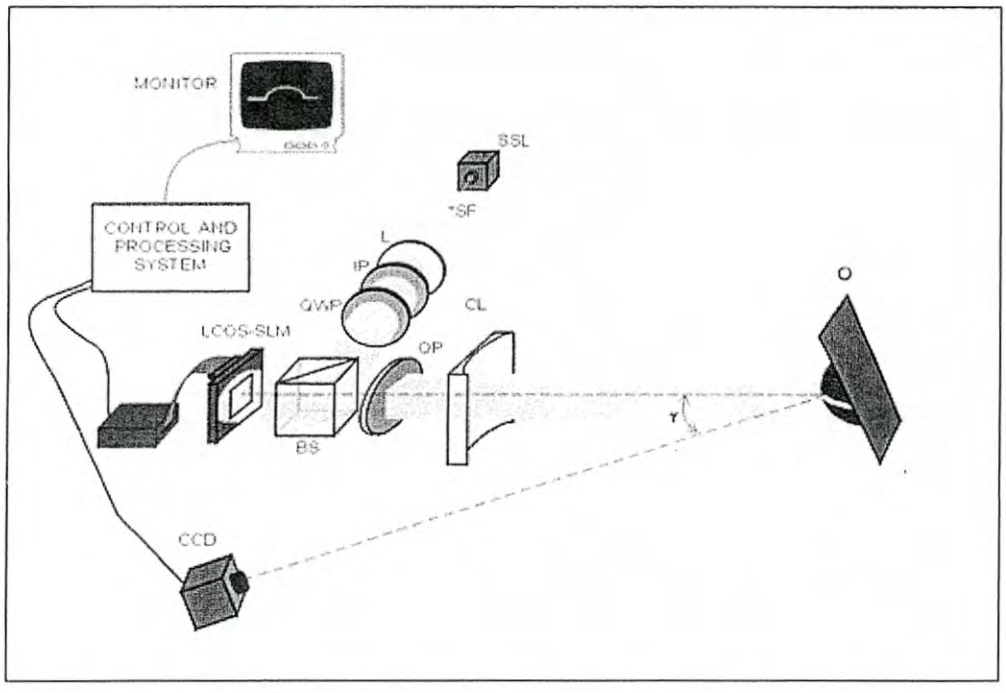


Fig. 6.2 Sistema para la recuperación de formas de objetos 3-D

En la figura 6.3 (a) vemos el patrón de una lente difractiva desplegada en niveles de gris con un monitor. Los datos de la lente difractiva, previamente diseñada, fueron guardados en la memoria del sistema y enviados al modulador para formar una línea de iluminación láser como se observa con más detalle en la Figura 6.3. (b).

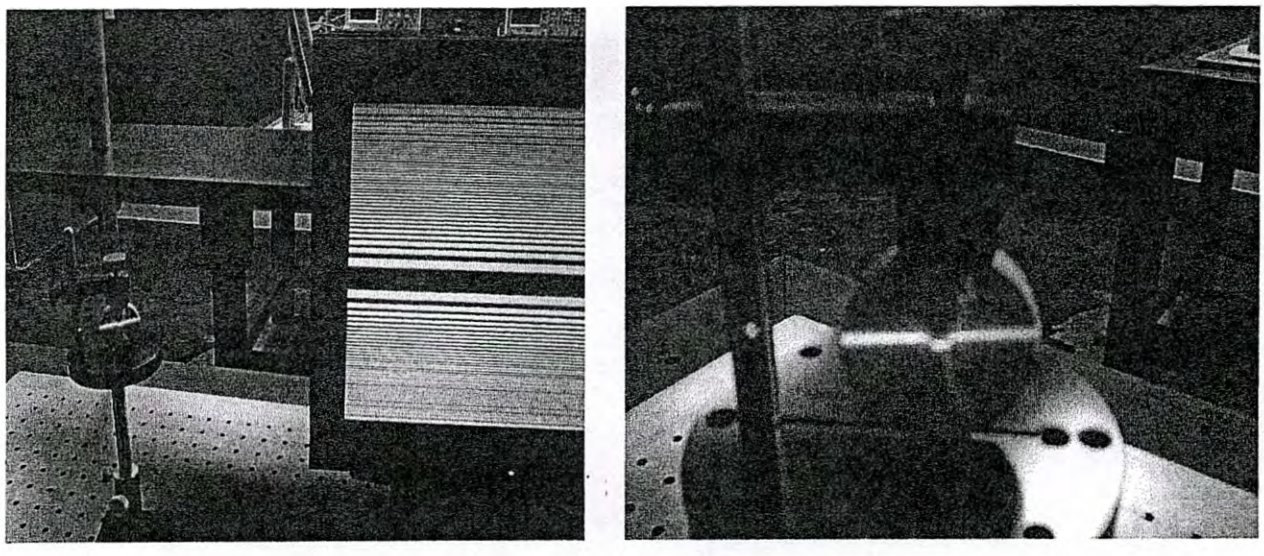


Figura. 6.3. Despliegue de una lente difractiva a) muestra en un monitor, b) Línea horizontal de láser generada.

Para el escaneo del objeto se desplaza el patron de lente difractiva a petición de una señal de control proveniente de una computadora. En la figura 6.4 se muestra el desplazamiento de este patrón. Una secuencia de barrido con el haz generado por la lente difractiva sobre un objeto se muestra en figura 6.5. Finalmente, la reconstrucción del objeto mostrado en la figura 6.6 se realizo mediante un algoritmo de proyección geométrica en la computadora.

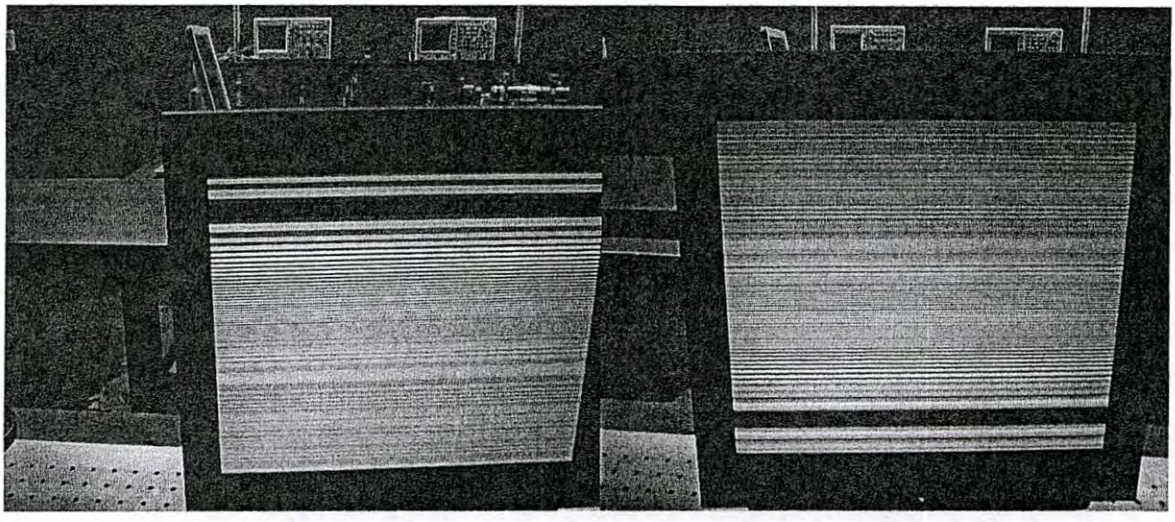


Figura. 6.4. Codificación de una lente difractiva y su desplazamiento vertical.

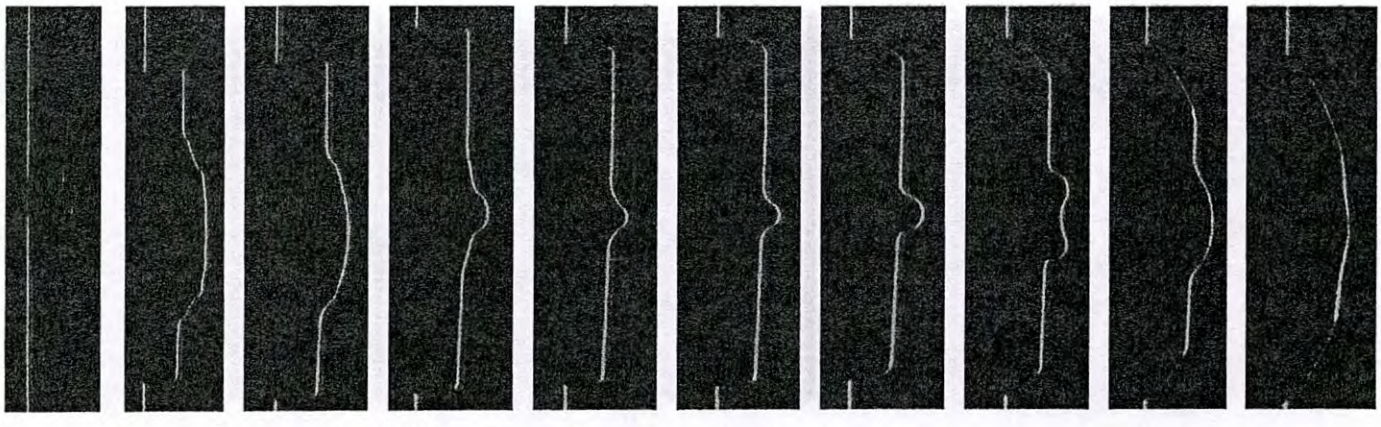


Figura 6.5. Barrido del haz generado con la lente difractiva desplegada en el modulador

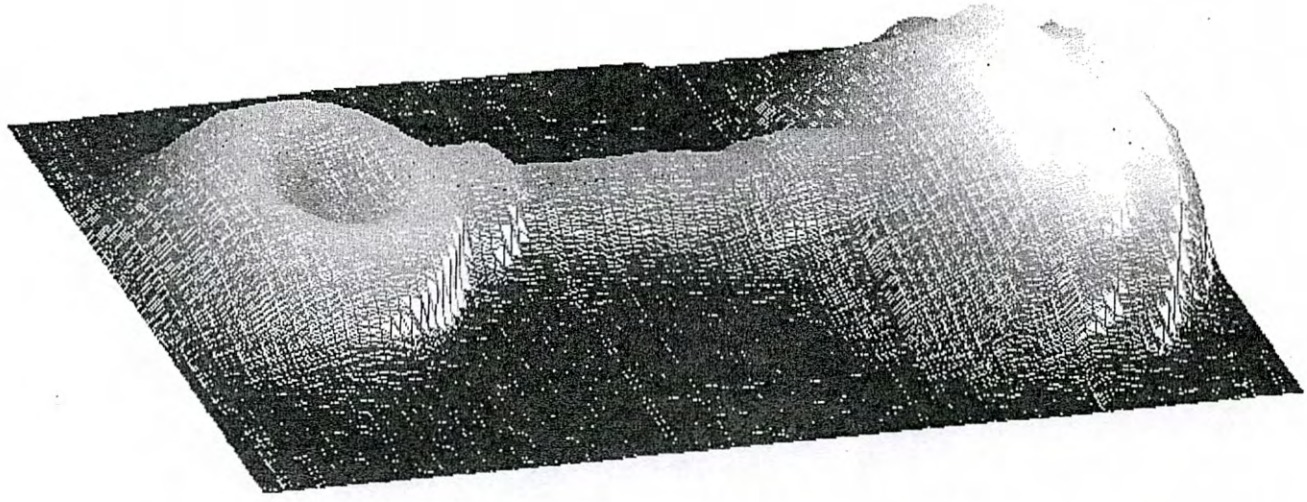


Figura 6.6. Objeto reconstruido por medio de un algoritmo.

CAPITULO 7

CONCLUSIONES.

Se presentó el diseño y la implementación de una interfaz digital para un modulador espacial de luz que opera básicamente mediante el despliegue de imágenes en escala de 256 niveles de grises y una resolución de 800x600 pixeles. El propósito de emplear este dispositivo es crear un sistema óptico-digital para la obtención de la forma de un objeto. La interfaz digital consta de bloques o circuitos digitales cuyo diseño fue realizado mediante el lenguaje de descripción de hardware VHDL. Para la implementación del diseño se dispuso de un FPGA que se encuentra insertado en la plataforma de desarrollo XUP Virtex II Pro de la compañía Xilinx.. Como prueba preliminar del control de la modulador de cristal liquido se logro desplegar líneas horizontales con diferentes niveles de gris en formato VGA. Para cumplir con el propósito por el cual se demandaba esta interfaz se diseño un sistema para el despliegue de lentes difractivas en el modulador de luz. La función principal de estas lentes es generar un patrón de iluminación (o línea láser) para escanear la forma de un objeto. De esta manera el sistema de escaneo no requiere motores ni espejos que formen el haz láser.

Este proyecto es de mucha utilidad ya que al lograr obtener la forma de un objeto se podrá comprobar si existen discrepancias entre objetos que aparentemente son similares. Por lo que se puede utilizar para producción en serie en la que suelen salir piezas o partes dañadas poco notorias que se descubrirían más fácilmente al emplear este sistema.

Una ventaja de esta interfase es que esta preparada para desplegar cualquier tipo de patrón de gráficos unidimensionales en el modulador espacial de luz por lo que se le puede emplear para otro tipo de aplicaciones.

En el futuro, se planea realizar un sistema mas complejo empleando los microprocesadores integrados en la tarjeta utilizada, combinando la programación en lenguaje C y periféricos o circuitos de función de procesamiento en VHDL.

REFERENCIAS

[1] Sistemas Electrónicos Digitales. Fundamentos y diseños de Aplicaciones. Autor Enrique Sanchos Peris. Publicado 2002 Publ. Universitat de Valencia. 511 páginas. ISBN 8437055172.

[2] http://es.wikipedia.org/wiki/LCD

[3] http://Howstuffworks_Creating_an_LCD.htm

[4] http://electronics.howstuffworks.com/lcd2.htm

[5] Comparación de aspectos visuales entre pantallas. Autor. Fco. Miguel Martínez Verdú. Publicado. Septiembre de 2004. Publ. Universidad de Alicante.

[6] www.holoeye.com

[7] Fundamentals of Logic Design. Autor. Charles H. Roth, Jr, Ed. PWS Publishing Company. Publicado.1995. 770 páginas. ISBN 0534954723

[8] http://es.wikipedia.org/wiki/PROM

[9] http://www.ilustrados.com/publicaciones/EpZEEVlkAVCpQevBHL.php

[10] XUP Virtex II Pro User Guide. **www.xilinx.com**

[11] www.vesa.org

[12] http://www.epanorama.net/documents/pc/vga_timing.html

[13] http://www.osdever.net/FreeVGA/vga/vga.htm#intro

[14] Controlador de un monitor VGA con resolución 640x480 sobre un CPLD FLEX10K Abstract. Autor. Ing. Gerard Santillán Quiñónez. Publicado. 2000. Publ. Pontifica Universidad Católica de Perú.

[15] Built a VGA Monitor Controller. Enoch Hwang. November 2004.

[16] Single Port Block Memory. Xilinx Logicore. April, 2005.

APENDICE A.

Tarjeta XILINX UNIVERSITY PROGRAM VIRTEX II PRO DEVELOMENT SYSTEM

La tarjeta de sistema de desarrollo XUP VIRTEX II PRO provee una plataforma de hardware que consiste de una plataforma FPGA de alto desarrollo Virtex II Pro rodeada de una completa colección de componentes periféricos que se usan para crear un sistema complejo y para demostrar la capacidad de la plataforma FPGA Virtex II Pro.

En la figura A-1. se muestra el diagrama de bloques de la tarjeta de sistema de desarrollo XUP Virtex II Pro.

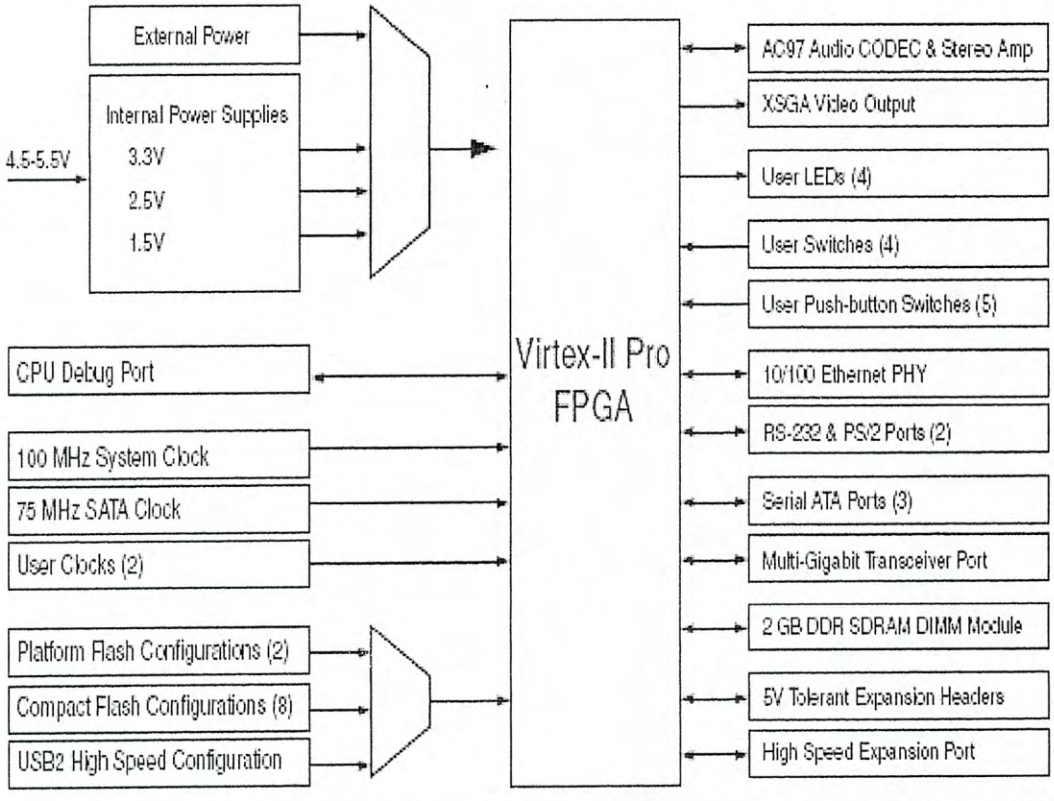


Figura A-1. Diagrama de bloques de la tarjeta de sistema de desarrollo XUP Virtex II Pro.

Componentes de la Tarjeta XUP Virtex II.

Aquí en la figura A-2. se describen algunos de los componentes más importantes de la tarjeta de desarrollo XUP Virtex II Pro.

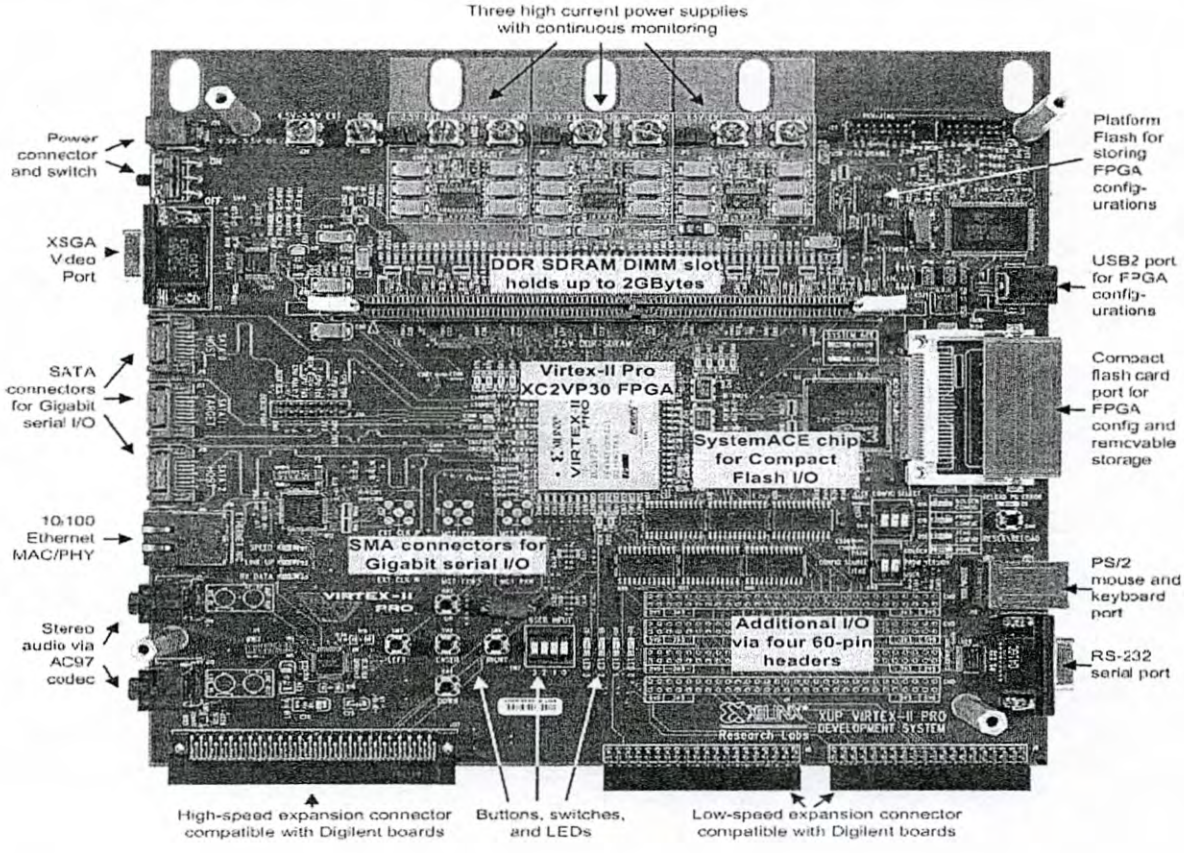


Figura A-2. Fotografía de la tarjeta de desarrollo XUP Virtex II Pro.

FPGA Virtex II PRO

Esta tarjeta utiliza un FPGA Virtex II PRO que tiene un empaquetado FF896 BGA. La tarjeta de desarrollo XUP Virtex II PRO puede usar dos diferentes tipos de FPGA sin cambiar su funcionalidad. En la siguiente tabla A-1. se muestran algunas características de FPGA y en la figura siguiente A-3. se muestra la ubicación de los dispositivos periféricos.

Features	XC2VP20	XC2VP30
Slices	9280	13969
Array Size	56 x 46	80 x 46
Distributed RAM	290 Kb	428 Kb
Multiplier Blocks	88	136
Block RAMs	1584 Kb	2448 Kb
DCMs	8	8
PowerPC RISC Cores	2	2
Multi-Gigabit Transceivers	8	8

Tabla. A-1. Características de los dispositivos FPGA XC2VP20 Y XC2VP30.

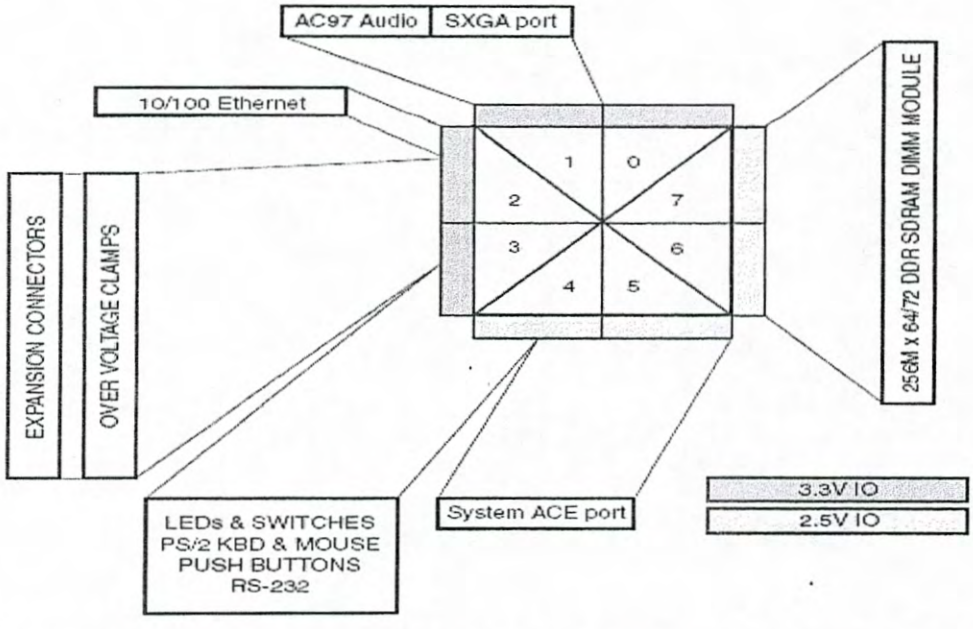


Figura A-3. Dispositivos periféricos y bancos de conexión de entrada y salida.

Voltajes de alimentación y configuración de FPGA.

La tarjeta de desarrollo XUP Virtex II Pro es alimentada con una fuente de 5V regulados. Los switch de alimentación de voltaje de la tarjeta generan 3.3V, 2.5V y 1.5V para el FPGA, los componentes periféricos y multi-gigabit transceivers (MGT).

La XUP Virtex II PRO tiene medidores de corriente para todas las entradas de voltaje del FPGA, así como aplicaciones para alimentación externa si se excede la capacidad de los switch de la tarjeta, cuenta tambien con varios métodos de configuración para el FPGA Virtex II Pro. La configuración de datos se puede originar desde la plataforma interna Flash PROM (2 tipos de configuración), la compactFlash interna (8 tipos de configuración) y configuraciones externas a través del cable USB o interfase de puerto paralelo.

RAM

La tarjeta XUP Virtex II Pro esta provista para la instalación de un modulo de memoria RAM Dinámica Sincronía de doble dato. La tarjeta soporta módulos de memoria de buffer y sin buffer con una capacidad de hasta 2GB o menos en organizaciones de 64 y 72 bits.

Controlador de Compact Flash Sistema ACE

El controlador del sistema de medio de configuración avanzada (System ACE) maneja los datos de configuración. El controlador provee una interfase inteligente entre el FPGA y varias fuentes de configuración. El controlador tiene varios puertos: el puerto de la Flash, el puerto de configuración JTAG, el puerto del microprocesador (MPU) y el puerto de prueba JTAG. La XUP Virtex II Pro soporta un controlador simple System ACE. Los puertos de configuración JTAG conectan al FPGA y a los puertos de expansión. El puerto de prueba JTAG conecta al puerto de JTAG y a la interfase USB del CPLD, y los puertos MPU conectan directamente al FPGA.

Interfase Ethernet

La XUP Virtex II Pro provee una conexión Fast Ethernet que soporta aplicaciones 10BASE-T y 100 BASE-TX. Soporta operaciones de full-duplex a 10Mb/s y 100Mb/s con auto negociación. La PHY provee una Interfase Media Independiente (MII) para la unión del Controlador de Acceso a la Media (MAC) implementada en el FPGA.

Puertos Seriales

La XUP Virtex II Precuenta con tres puertos seriales, un puerto RS232 y dos puertos PS/2. El puerto RS 232 se configura como un DCE usando un conector DB9 estándar. Este conector se usa comúnmente para comunicaciones a una computadora usando un cable serial estándar de 9 pines conectada a un puerto COM. Los dos puertos PS/2 se pueden usar para conectar un teclado y un Mouse a la tarjeta. Todos los puertos están equipados con circuitos de corrimiento de nivel, porque la tarjeta no puede hacer la interfase directamente a los niveles de voltaje requeridos para RS 232 y PS/2.

Leds, Switches y Botones

La tarjeta cuenta con un total de 4 LEDS para propósitos del usuario, cuando el FPGA envía un cero lógico, el led correspondiente se enciende. Un DIP switch de 4 posiciones simples y 5 botones para utilizarlos como entradas. Si el DIP switch esta arriba, cerrado o encendido, o el botón presionado, el FPGA envía un cero lógico, de otra manera enviara un uno lógico.

Conectores de Expansión

El Virtex II Pro cuenta con un total de 80 pines en dos conectores de 40 pines cada uno para asignación por el usuario. Cuenta también con uno de 60 pines que esta diseñado para aceptar conectores de cable plano.

Salida XSGA

La tarjeta incluye un convertidor digital-analógico (DAC) de video y un conector de 15 pines que soporta salidas XSGA. El DAC de video puede operar con un reloj de píxel de hasta 180MHz, esto permite salidas compatibles para VESA de 1280 x 1024 a 75Hz y una resolución máxima de 1600 x 1200 a 70Hz.

CODEC de audio AC97

Se incluyen también un CODEC de audio y un amplificador de potencia que provee alta calidad de audio y todas las funcionalidades análogas en un sistema de audio de una PC. Cuenta con un ADC y DAC en modo full-duplex, con un mezclador análogo, combinando entradas lineales, entrada de micrófono y datos PCM.

Interfase de programación USB 2

La tarjeta cuenta con un microcontrolador embebido USB 2.0 capaz de comunicarse con USB hosts a velocidad de full-duplex (transmisión y recepción al mismo tiempo) y high speed (alta velocidad). Esta interfase se utiliza para programación y configuración del FPGA Virtex II Pro en modo Boundary-scan (IEEE 1194.1/IEEE1532). Las velocidades de reloj de la tarjeta son seleccionadas de 750 Khz. a 24 MHz. El microcontrolador se conecta a una laptop o una PC con un cable USB off-the-shelf high-speed A-B.[10]

APENDICE B. Listado de diseños para el control del SLM en lenguaje VHDL.

Programa principal
El cual realiza la asignacion de patillas en la tarjeta y el llamado de los componentes.

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.numeric_std.all;

entity chipLens is

        --Declaracion de entradas y salidas
  port(
        pin_Reset     :       in std_logic;    -- Reset del sistema
    pin_sysclk  : in std_logic;        -- Reloj principal externo
        -- vga port connections
    pin_vga_red_S0    : out std_logic;  --Salida 0 de VGA
    pin_vga_green_S0  : out std_logic;  --Salida 0 de VGA
    pin_vga_blue_S0   : out std_logic;  --Salida 0 de VGA
    pin_vga_red_S1    : out std_logic;  --Salida 1 de VGA
    pin_vga_green_S1  : out std_logic;  --Salida 1 de VGA
    pin_vga_blue_S1   : out std_logic;  --Salida 1 de VGA
    pin_vga_red_S2    : out std_logic;  --Salida 2 de VGA
    pin_vga_green_S2  : out std_logic;  --Salida 2 de VGA
    pin_vga_blue_S2   : out std_logic;  --Salida 2 de VGA
    pin_vga_red_S3    : out std_logic;  --Salida 3 de VGA
    pin_vga_green_S3  : out std_logic;  --Salida 3 de VGA
    pin_vga_blue_S3   : out std_logic;  --Salida 3 de VGA
    pin_vga_hsync_n  : out std_logic;          --Sincronia Horizontal
    pin_vga_vsync_n  : out std_logic;          --Sincronia Vertical
        switch1: in std_logic                  --Boton
    );
end chipLens;

architecture arch of chipLens is

        -- Declaracion de Señales para el llamado a los componentes
        signal sysClk   : std_logic;  -- system clock
        signal Sframe   : std_logic;
        signal sysReset : std_logic;  -- system reset
       signal vga_pixel_row : std_logic_vector(10 downto 0); --Renglones
        signal vga_pixel_col : std_logic_vector(10 downto 0); --Columnas
        signal pin_vga_red, pin_vga_green, pin_vga_blue:  std_logic_vector(3 downto 0);
--Señales RGB
        signal B0, B1 : std_logic_vector(3 downto 0);
```

```
--Llamado a componentes

component svga_sync
   generic (
    Row_Wid : integer;
    Col_Wid : integer;
    Vdisp   : integer;
    Vpw     : integer;
    Vfp     : integer;
    Vbp     : integer;
    Hdisp   : integer;
    Hpw     : integer;
    Hfp     : integer;
    Hbp     : integer);
   port (

   Clk    : in  std_logic;
   Rst    : in  std_logic;
   red0, green0, blue0 : in std_logic;
   red1, green1, blue1 : in std_logic;
   red2, green2, blue2 : in std_logic;
   red3, green3, blue3 : in std_logic;
   HSync  : out std_logic;
   VSync, Frame : out std_logic;
   pixel_row    : out std_logic_vector(Row_Wid downto 0);
   pixel_column : out std_logic_vector(Col_Wid-1 downto 0);
   red_out0, green_out0, blue_out0 : out std_logic;
   red_out1, green_out1, blue_out1 : out std_logic;
   red_out2, green_out2, blue_out2 : out std_logic;
   red_out3, green_out3, blue_out3 : out std_logic
   );
        end component;

component FSM is
        port(
                A0: in std_logic;
                CLOCK: in std_logic;
                Y: out std_logic_vector(3 downto 0)
                );
        end component;

component MUX_vector is
 port(
        B0, B1: in std_logic_vector(3 downto 0);
        M: in std_logic;
        X: out std_logic_vector(3 downto 0));
```

```
        end component;


begin

  -- define Reloj del sistema y reset
  sysClk   <= pin_sysclk;
  sysReset <= '0';


  -- Creacion de componentes

        svga_sync_1 : svga_sync
  generic map (
    Row_Wid => 10,
    Col_Wid => 10,
    Vdisp   => 600,
    Vpw     => 6,
    Vfp     => 37,
    Vbp     => 23,
    Hdisp   => 800,
    Hpw     => 120,
    Hfp     => 56,
    Hbp     => 64)
  port map (
    Clk         => sysClk,                    --Asignacion de Señales
    Rst         => sysReset,
        Frame       => Sframe,
    HSync       => pin_vga_hsync_n,
    VSync       => pin_vga_vsync_n,
        pixel_row    => vga_pixel_row,
        pixel_column => vga_pixel_col,
        red0         => pin_vga_red(0),
        red1         => pin_vga_red(1),
        red2         => pin_vga_red(2),
        red3         => pin_vga_red(3),
        green0       => pin_vga_red(0),
        green1       => pin_vga_red(1),
        green2       => pin_vga_red(2),
        green3       => pin_vga_red(3),
        blue0        => pin_vga_red(0),
        blue1        => pin_vga_red(1),
        blue2        => pin_vga_red(2),
        blue3        => pin_vga_red(3),
        red_out0     => pin_vga_red_S0,
        red_out1     => pin_vga_red_S1,
        red_out2     => pin_vga_red_S2,
        red_out3     => pin_vga_red_S3,
```

```
        green_out0    => pin_vga_green_S0,
        green_out1    => pin_vga_green_S1,
        green_out2    => pin_vga_green_S2,
        green_out3    => pin_vga_green_S3,
        blue_out0     => pin_vga_blue_S0,
        blue_out1     => pin_vga_blue_S1,
        blue_out2     => pin_vga_blue_S2,
        blue_out3     => pin_vga_blue_S3
        );


   Maquina: FSM
   port map (
    A0 => switch1,                          --Entrada A0 a boton
        CLOCK => sysclk,
         Y => B0                            --Salida de maquina a señal B0
          );

        multiplexor: mux_vector
        port map(
                B0 => B0,                   --Señal B0 a entrada de mux
                B1 => "0000",               --A B1 se le asignan "00"
                M => vga_pixel_row(1),      --Al selector se le asigna el renglo (1)
para generar barras de 2 pixeles, (0 para 1 pixel)
                X => pin_vga_red            --A la salida del mux se le asigna como
entrada a VGA sync RGB
                );
end arch;
```

Sincronía VGA

Programa que realiza la sincronía de VGA para resolución de 800x600.

```
---------------------------------------------------------------------
--Este programa genera video en modo de 600x800 con un reloj a 50 MHz
---------------------------------------------------------------------
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;

entity svga_sync is

 generic (
        --Declaracion en pixeles
    Row_Wid : integer := 10;  --amplitud de renglon
    Col_Wid : integer := 10;  --amplitud de columna
    Vdisp :   integer := 600; --Desplegado en vertical
    Vpw :     integer := 6;       --Amplitud de porsh vertical
```

```
    Vfp :    integer := 37;  --Porsh frontal vertical
    Vbp :    integer := 23;  --Porsh posterior vertical
    Hdisp :  integer := 800; --Desplegado en horizontal
    Hpw :    integer := 120; --Amplitud de porsh horizontal
    Hfp :    integer := 56;  --Porsh frontal horizontal
    Hbp :    integer := 64       --Porsh posterior horizontal
        );
  port (
   Clk   : in  std_logic;
   Rst   : in  std_logic;
   red0, green0, blue0          : in     std_logic;
   red1, green1, blue1          : in     std_logic;
   red2, green2, blue2          : in     std_logic;
   red3, green3, blue3          : in     std_logic;
   HSync  : out std_logic;
   VSync : out std_logic;
   frame : out std_logic;
   pixel_row    : out std_logic_vector(Row_Wid downto 0);
   pixel_column : out std_logic_vector(Col_Wid-1 downto 0);
   red_out0, green_out0, blue_out0 : out std_logic;
   red_out1, green_out1, blue_out1 : out std_logic;
   red_out2, green_out2, blue_out2 : out std_logic;
   red_out3, green_out3, blue_out3 : out std_logic

   );

end svga_sync;

architecture Behavior of SVGA_sync is

  --Declaracion de constantes
  constant HPulseStart : integer := Hdisp + Hbp;
  constant HPulseEnd   : integer := HPulseStart + Hpw;
  constant HTotal      : integer := HPulseEnd + Hfp;
  constant VPulseStart : integer := Vdisp + Vbp;
  constant VPulseEnd   : integer := VPulseStart + Vpw;
  constant VTotal      : integer := VPulseEnd + Vfp;

  --Declaracion de señales
  signal HCount : integer range 0 to HTotal-1;
  signal VCount : integer range 0 to VTotal-1;
  signal Enable : std_logic;
  signal HEnable : std_logic;
  signal VEnable : std_logic;
  signal HLast   : std_logic;
  signal VLast   : std_logic;
  signal Next_HCount : integer range 0 to HTotal;
  signal Next_VCount : integer range 0 to VTotal;
```

```
begin


 -- Informacion de salida de columna y renglon como vectores
pixel_column <= conv_std_logic_vector(HCount, Col_Wid);
pixel_row <= conv_std_logic_vector(VCount, Row_Wid + 1);

 -- Esta dentro de la region visible?
 HEnable <= '1' when HCount < Hdisp else '0';
 VEnable <= '1' when VCount < Vdisp else '0';
  -- Aqui se establecen los datos de RGB
 frame<=Henable;
 Enable  <= HEnable and VEnable;

 red_out0 <= red0 AND Enable;
 green_out0 <= green0 AND Enable;
 blue_out0 <= blue0 AND Enable;
 red_out1 <= red1 AND Enable;
 green_out1 <= green1 AND Enable;
 blue_out1 <= blue1 AND Enable;
 red_out2 <= red2 AND Enable;
 green_out2 <= green2 AND Enable;
 blue_out2 <= blue2 AND Enable;
 red_out3 <= red3 AND Enable;
 green_out3 <= green3 AND Enable;
 blue_out3 <= blue3 AND Enable;

 -- Llego al final del ciclo
 HLast <= '1' when (HCount = HTotal-1) else '0';
 VLast <= '1' when (VCount = VTotal-1) else '0';

 -- Asignacion de sincronia horizontal y vertical
 HSync <= '1' when HCount >= HPulseStart-1 and Hcount < HPulseEnd-1 else '0';
 VSync <= '1' when VCount >= VPulseStart-1 and Vcount < VPulseEnd-1 else '0';


  -- Decide que hacer con los conteos de vertical y horizontal
 Make_Signals : process (HCount, HLast, VCount, VLast)
 begin
  if (HLast = '1') then
    Next_HCount <= 0;
    if (VLast = '1') then
      Next_VCount <= 0;
    else
      Next_VCount <= VCount + 1;
    end if;
  else
    Next_HCount <= HCount + 1;
    Next_VCount <= VCount;
```

```
    end if;
  end process Make_Signals;

  Registers : process (Clk, Rst)
  begin
    if Rst = '1' then
      HCount <= 0;
      VCount <= 0;
    elsif Clk'event and Clk = '1' then
      HCount <= Next_HCount;
      VCount <= Next_VCount;
    end if;
  end process Registers;
end Behavior;
```

Maquina de Estados.

Programa de la Maquina de Estados que controla los niveles de grises.

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.numeric_std.all;


ENTITY fsm IS
PORT(
        CLOCK : IN STD_LOGIC;                          -- Reloj
        A0 : IN STD_LOGIC;                             -- Entrada de boton
        Y : OUT STD_LOGIC_VECTOR(3 DOWNTO 0)           -- Salida de vector de 4 bits
        );
END fsm;
-----------------------------------------------
ARCHITECTURE behaviour OF fsm IS
   --Declaracion de Señales
        TYPE state_type IS (s0,s1,s2,s3,s4,s5,s6,s7,s8,s9,s10,s11,s12,s13,s14,s15);
        SIGNAL present_state,next_state : state_type;

        BEGIN
--Process secuencial para el estado de reloj
state_reg:PROCESS
        BEGIN
        WAIT UNTIL clock'EVENT AND clock = '1';
        present_state <= next_state;
        END PROCESS;
--Process combinacional para logica de estados
fb_logic:PROCESS(present_state,A0)
        BEGIN
                CASE present_state IS
                WHEN s0 =>
```

```
IF A0 = '1' THEN Y <= "0000"; next_state <= s0;    --Aqui se genera la maquina de estados
            ELSE Y <= "0001"; next_state <= s1; --para generar los 16 diferentes niveles
            END IF;                                  -- de grises
      WHEN s1 =>
            IF A0 = '0' THEN Y <= "0001"; next_state <= s1;
                  ELSE Y <= "0010"; next_state <= s2;
            END IF;
      WHEN s2 =>
            IF A0 = '1' THEN Y <= "0010"; next_state <= s2;
                  ELSE Y <= "0011"; next_state <= s3;
            END IF;
      WHEN s3 =>
            IF A0 = '0' THEN Y <= "0011"; next_state <= s3;
                  ELSE Y <= "0100"; next_state <= s4;
            END IF;
      WHEN s4 =>
            IF A0 = '1' THEN Y <= "0100"; next_state <= s4;
                  ELSE Y <= "0101"; next_state <= s5;
            END IF;
      WHEN s5 =>
            IF A0 = '0' THEN Y <= "0101"; next_state <= s5;
                  ELSE Y <= "0110"; next_state <= s6;
            END IF;
      WHEN s6 =>
            IF A0 = '1' THEN Y <= "0110"; next_state <= s6;
                  ELSE Y <= "0111"; next_state <= s7;
            END IF;
      WHEN s7 =>
            IF A0 = '0' THEN Y <= "0111"; next_state <= s7;
                  ELSE Y <= "1000"; next_state <= s8;
            END IF;
  WHEN s8 =>
            IF A0 = '0' THEN Y <= "1000"; next_state <= s8;
                  ELSE Y <= "1001"; next_state <= s9;
            END IF;
      WHEN s9 =>
            IF A0 = '0' THEN Y <= "1001"; next_state <= s9;
                  ELSE Y <= "1010"; next_state <= s10;
            END IF;
      WHEN s10 =>
            IF A0 = '0' THEN Y <= "1010"; next_state <= s10;
                  ELSE Y <= "1011"; next_state <= s11;
            END IF;
      WHEN s11 =>
            IF A0 = '0' THEN Y <= "1011"; next_state <= s11;
                  ELSE Y <= "1100"; next_state <= s12;
  END IF;
      WHEN s12 =>
            IF A0 = '0' THEN Y <= "1100"; next_state <= s12;
```

```
                    ELSE Y <= "1101"; next_state <= s13;
                END IF;
            WHEN s13 =>
                IF A0 = '0' THEN Y <= "1101"; next_state <= s13;
                    ELSE Y <= "1110"; next_state <= s14;
                END IF;
            WHEN s14 =>
                IF A0 = '0' THEN Y <= "1110"; next_state <= s14;
                    ELSE Y <= "1111"; next_state <= s15;
                END IF;
            WHEN s15 =>
                IF A0 = '0' THEN Y <= "1111"; next_state <= s15;
                    ELSE Y <= "0000"; next_state <= s0;
                END IF;
            END CASE;
    END PROCESS;
END behaviour;
```

Multiplexor.

El programa de multiplexor selecciona las salidas de la maquina de estados para la generacion de los niveles de gris.

```
--Declaracion de librerias
library IEEE;
use IEEE.std_logic_1164.all;

entity MUX_vector is
 port(
        B0, B1: in std_logic_vector(3 downto 0);                --Entradas de 4 bits para la salida de
la maquina de estados
        M: in std_logic; --Entrada selectora
        X: out std_logic_vector(3 downto 0));                   --Salida para asignarla como entrada de
RGB.
end MUX_vector;

architecture Vector_1 of MUX_vector is
 begin
        process(B0, B1, M)
          begin
                case M is
                        when '1' => X <= b0;
                        when others => X <= b1;
                end case;
        end process;
end Vector_1;
```