UNIVERSIDAD MAYOR DE "SAN ANDRÉS"

Facultad de ingeniería

INFORME PROYECTO LABORATORIO

PROTOCOLO DE COMUNICACIÓN VGA

GRUPO IV

Estudiantes: Cruz Torrez Diego Alejandro

Flores Quispe Nilo

Pacheco Choque Lizandro Shamil

Carrera: Ingeniería electrónica

Docente: Ing. Jorge Mario Leon Gomez

Materia:ETN-821 Sistemas Digitales II

Índice

1. Introducción y antecedentes

En 1988 la empresa IBM comercializo por primera vez una tarjeta gráfica que incorporaba un estándar de video denominado VGA (Video Graphics Array),dicho estándar fue utilizado ampliamente por diversos desarrolladores durante mucho tiempo ,con ligeras modificaciones(número de pixeles, frecuencia de refresco,etc) debido a que establecía los requerimientos mínimos de hardware y software necesario para visualizar imágenes de alta resolución.

Es por eso que surge un interés en estudiar y recrear los mecanismos por los cuales se logra la visualización de imágenes y videos en un monitor mediante dicho estándar VGA con fines didácticos.

En este proyecto aborda los mecanismos de funcionamiento de las señales VGA, el desarrollo de hardware necesario para generarlas y un sistema que permita diversas formas de visualizar una secuencia de imágenes

En este trabajo se abordó el principio de una transmisión de datos a través de un módulo hardware implementado en laboratorio cuya función es trabajar como una tarjeta de video (baja resolución) de un computador permitiendo que datos de este sean enviados a una pantalla sin necesidad de una PC para la comunicación.

Los datos de una imagen son obtenidos de y codificados por medio de Phyton para luego grabarlos una memoria EPROM y estos ser transmitido por un protocolo de comunicación VGA visualizándose así dicha imagen.

2. Planeamiento del problema

La problemática de este trabajo es enviar información almacenada en una EPROM a un monitor utilizando el protocolo VGA sin intermediarios (PC).

Además, se debe establecer un control por teclado que permita controlar diversas funciones, como ser: visualización de una secuencia a una frecuencia determinada, durante un tiempo definido,etc.

El diseño debe incorporar un oscilador de 10M[Hz] que determinará el modo de visualización en el monitor, bits de color o resolución de la información de imagen que será transmitido por el protocolo VGA.

El diseño del circuito entonces contara con una parte de control para el barrido horizontal, contadores y operacionales para la sincronía del monitor, una parte de diseño vertical, el cual cuenta con contadores operadores para la sincronía vertical con respecto a la pantalla del monitor, dos bits destinados a la sincronía y los bits de salida de los contadores tanto vertical como horizontal destinados a la parte de control de la EPROM, y un diseño adicional que se encargara en la parte de la forma de funcionamiento y tiempo de visualización.

De la misma manera se debe enviar información relacionada al color de un pixel almacenada en una EPROM de acuerdo a una codificación de colores preestablecida.

El control por teclado se realizará mediante un circuito combinacional diseñado para tal fin.

Al conocer las especificaciones para trabajar con VGA, se puede proceder al desarrollo del diseño, la imagen creada incorporada en una EPROM será transmitida a un monitor sin el uso de una PC para el procesamiento de datos es aquí donde es necesario el diseño completo, la cual cumplirá la función de una tarjeta de video del monitor destinado, la imagen es un simple mapa de bits, JPEG ,PNG o cualquier otro formato, para nuestro propósito se usara diversas imágenes codificadas para luego ser insertadas en la EPROM

3. Objetivos

3.1 Objetivo General

Diseñar e implementar un circuito que permita controlar diversas funciones de visualización de imágenes en un monitor utilizando el protocolo VGA.

Dicho circuito se implementará utilizando componentes TTL[1] aplicando los conocimientos adquiridos a lo largo del semestre.

3.2 Objetivos Específicos

- Conocer el funcionamiento de un monitor (como se logran visualizar imágenes).
- Conocer las características del protocolo VGA para él envió de datos de imagen a un monitor.
- Diseñar un circuito que cumpla los requisitos temporales de dicho protocolo, así como características impuestas como resolución de una imagen (pixeles) y la frecuencia de reloj a utilizar.
- Diseñar un circuito que permita seleccionar una variedad de funciones tales como: visualizar una imagen en específico, visualizar una sucesión de imágenes durante un tiempo específico, desplazamiento manual entre un conjunto de imágenes, etc.
- Hallar una manera de lograr un desplazamiento omnidireccional de una imagen.
- Plantear software capaz de realizar la tarea anteriormente descrita.

3.3 Alcances y limitaciones

- Debido a que utilizaran circuitos integrados TTL[2] cuya máxima frecuencia de trabajo es de 32 MHz, la cantidad de pixeles a visualizar se reduce significativamente.
- De la misma manera la cantidad de imágenes a visualizar está limitada al espacio que la EEPROM posee, que permite el grabado de hasta 8 imágenes diferentes.
- Como consecuencia este proyecto no involucra el desplazamiento de una imagen en pantalla, ni la visualización de secuencias largas[3].

4. Justificación del proyecto

El presente proyecto se enfocará en el estudio del protocolo VGA pues al ser este un tema con la dificultad adecuada para ser abordado con los conocimientos actuales de un estudiante, representa una oportunidad perfecta para poner en práctica todo lo aprendido a lo largo del

semestre. Este proyecto permitirá demostrará el conocimiento y las habilidades que adquiridas en la presente materia y evaluar el desempeño grupal. Además, ofrecerá un vistazo integral a posibles mejoras y expansiones del proyecto para futuras implementaciones de otros estudiantes en el futuro.

5. Desarrollo del proyecto

5.1 Marco Teórico

5.1.1 Monitor

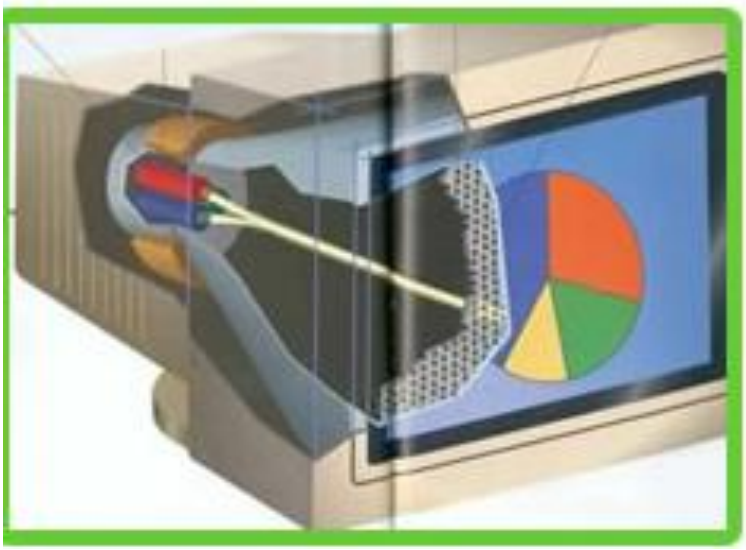
Un monitor es un equipo electrónico que funciona como dispositivo de salida en ordenadores, siendo el responsable de mostrar de forma gráfica todas las imágenes y textos que se generan en la computadora.

En pocas palabras, este es un hardware creado para visualizar el contenido creado por el procesador de una computadora gracias a la tarjeta gráfica.

5.1.1.1. Monitor CTR

Figura 1

Estructura interna del monitor CTR



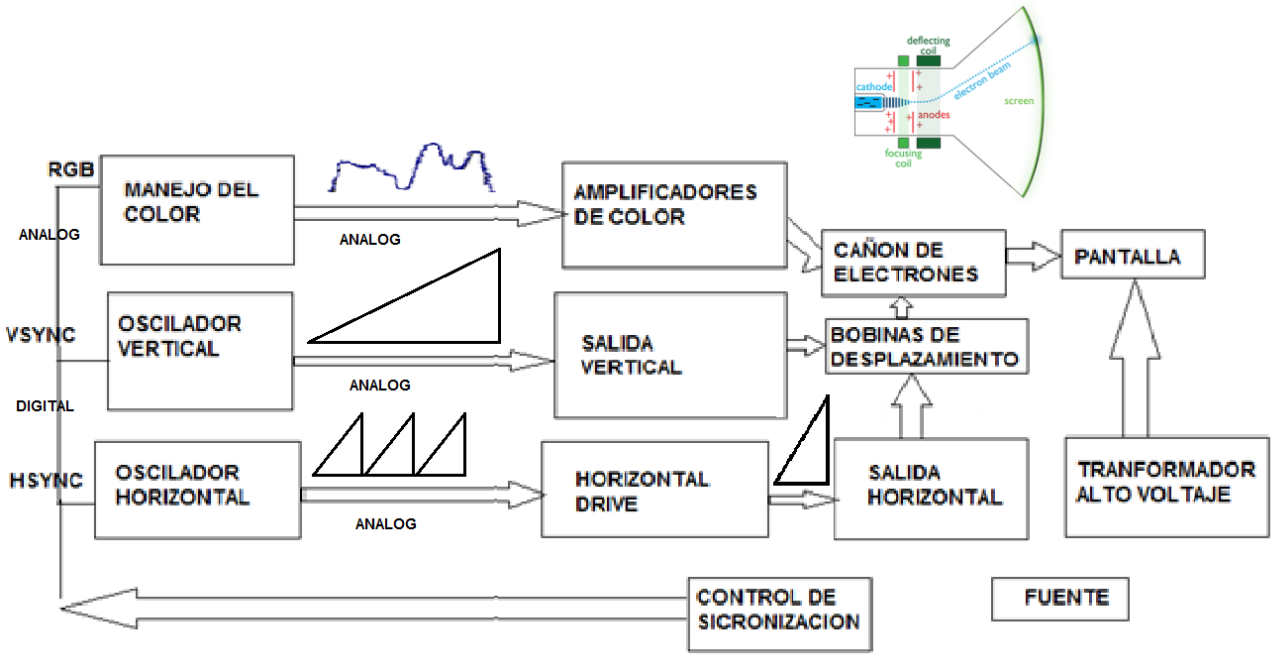
El monitor CTR (Tubo de Rayos Catódicos) representa visualmente las imágenes generadas por el ordenador a través de un puerto de video que se encuentra conectado a los circuitos del monitor. La formación de colores en este tipo de pantallas se genera a partir de la combinación de los tres colores básicos. La mayoría del espacio está ocupado por un tubo de rayos catódicos en el que se sitúa un cañón de electrones. Este cañón dispara constantemente un haz de electrones contra la pantalla, que está recubierta de fósforo (material que se ilumina al entrar en contacto con los electrones). Esta pantalla debe poseer un alto voltaje para generar un campo eléctrico lo suficientemente fuerte para atraer los

electrones En los monitores en color, cada punto o píxel de la pantalla está compuesto por tres pequeños puntos de fósforo: rojo, azul y verde. La intensidad del haz de electrones está determinada por un nivel de voltaje específico. Iluminando estos puntos con diferentes intensidades, puede obtenerse cualquier color.

Su funcionamiento se basa en que el cañón de electrones activa el primer punto de la esquina superior izquierda y, rápidamente, activa los siguientes puntos de la primera línea horizontal. Después sigue pintando y rellenando las demás líneas de la pantalla hasta llegar a la última y vuelve a comenzar el proceso. Este desplazamiento se logra mediante el uso de bobinas de desplazamiento convenientemente ubicadas en el “cuello de botella” a la salida del cañón de electrones que desvían los electrones según la posición requerida. Esta acción es tan rápida que el ojo humano no es capaz de distinguir cómo se activan los puntos por separado, percibiendo la ilusión de que todos los píxeles se activan al mismo tiempo por el efecto de persistencia.

Figura 2

Diagrama de bloques de un monitor CTR



Donde:

Manejo del color. Recibe las 3 señales correspondientes a los colores RGB (0,7 Vpp).

Amplificadores de color. Amplifican las señales a un nivel de voltaje requerido por el cañón de electrones.

Oscilador vertical. Recibe la señal VSYNC (pulso) realizando una operación de integración generando rampas de voltaje en su salida.

Salida vertical. Recibe las rampas de voltaje y las conduce hacia la bobina de desviación vertical

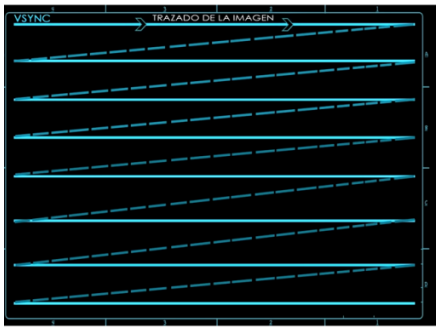
Oscilador horizontal. Recibe la señal HSYNC (impulso respecto a la señal vertical) realizando una operación de integración generando rampas de voltaje en su salida.

Horizontal drive. Amplifica las rampas de voltaje.

Salida horizontal. Recibe las rampas de voltaje y las conduce hacia la bobina de desviación horizontal.

Figura 5

Combinación de la acción de las bobinas da como resultado el barrido de la pantalla



Bobinas de desplazamiento. Son dos bobinas (vertical y horizontal) que forman un sistema de magnetización ubicado en la salida del cañón de electrones que dependiendo de las señales de sincronización desviaran el haz de electrones a una posición especifica.

Transformador de alto voltaje. Transforma el voltaje de alimentación hasta unos 20000V con el fin de generar un campo eléctrico lo suficientemente fuerte para atraer los electrones.

Fuente. Recibe la alimentación domiciliaria AC y la transforma a los diferentes voltajes requeridos.

Control de sincronización. Recibe las señales de control HSYNC, VSYNC y las señales RGB en el caso de una entrada VGA o SVGA.

Tipos de display CRT

1. Monocromático: un solo color, ambar, blanco o verde, combinado con negro.
2. Policromatico: varios colores

    -CGA (Color GraphicAdapter): 8 colores

    -VGA (Video GraphicAdapter): 16 colores

    -SVGA (Super Video GraphicAdapter): 256 colores

-UVGA (Ultra Video GraphicAdapter): 16,000,000 colores

Características del monitor CTR

Tamaño: es la distancia que existe entre la esquina superior derecha y la esquina inferior izquierda de la pantalla de vidrio, por lo que no se considera la cubierta de plástico que la contiene. La unidad de medida es la pulgada. Los más comunes son de 14", 17" y 19 pulgadas.

Color / monocromático: es el tipo de iluminación que puede mostrar. Monocromático solamente mostrará la escala de grises ó solamente un color verde claro, mientras que a color puede mostrar hasta 16 millones de colores distintos.

Control digital ó analógico: es analógico si para encender es necesario un botón rígido que cambia de posición al ser oprimido y los controles de la pantalla utilizan un resistor mecánico (especie de cilindro que se gira a la izquierda o derecha ajustando la pantalla). Será digital si solamente cuenta con botones para controlar el ajuste de la pantalla y estos al ser oprimidos regresan a su estado inicial.

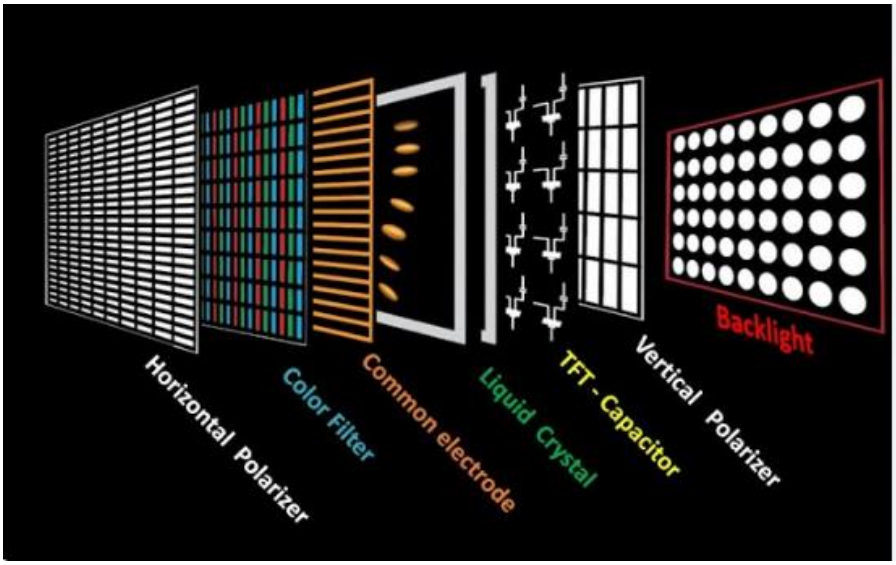
Tecnología: se le conoce como tecnología de barrido, ya que la pantalla se actualiza 25 veces por segundo, lo que a simple vista no se percibe, pero en cambio si puede cansar la vista.

Resolución: se refiere a la cantidad máxima de píxeles que es capaz de utilizar para desplegar una imagen en la pantalla el monitor. Un píxel es cada uno de los puntos que conforman la pantalla y a medida de que tenga mayor cantidad de ellos, se tendrá un mayor detalle de la imagen.

5.1.1.2 Monitor LCD

Figura 6

Estructura interna de un monitor LCD (Liquid Cristal Display)



Es una pantalla plana basada en el uso de una capa de cristal líquido capaz de cambiar sus propiedades a medida que se le aplica un voltaje específico.

Esta capa se encuentra entre 2 electrodos, dependiendo de la carga que se le aplique, podremos cambiar la orientación de las moléculas de cristal, de esta forma, tenemos un filtro polarizado controlado por electricidad.

Tras esta capa de cristal líquido se coloca una capa de vidrio que filtra el color, es decir, convertimos la luz blanca en otro color.

La luz blanca se genera utilizando dos placas de vidrio, una de ellas esta iluminada de la parte trasera por una luz intensa procedente de lámparas CCFL (lámparas fluorescentes de cátodos fríos).

Se utilizan varios filtros de color para crear colores complejos, para cada píxel se utilizan 3 sus píxeles, uno rojo, otro verde y otro azul lo cual permite la visualización de imágenes procedentes de la computadora, por medio del puerto de video hasta los circuitos de la pantalla LCD.

La corriente que le llega a cada capacitor TFT determina la saturación para cada color y asi se genera la gama de colores.

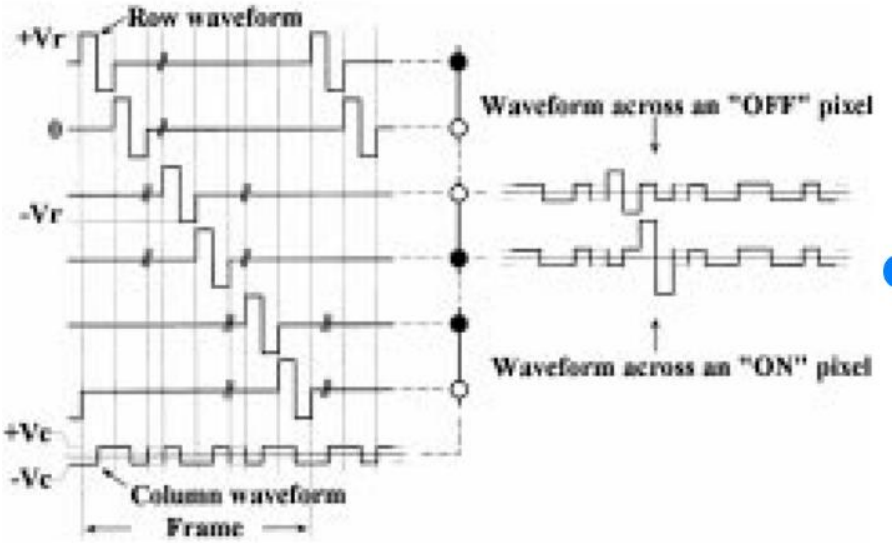
A pesar de utilizar una tecnología completamente diferente a la de los monitores CRT,su principio de funcionamiento es básicamente el mismo pues un monitor LCD utilizara de la misma manera que un CRT el proceso de barrido, encendiendo un pixel a la vez y desplazándose horizontal y verticalmente de acuerdo a las señales de sincronización que

recibe mediante la matriz TFT y un método de multiplexación,y mediante las señales RGB controlando los subpíxeles. Sin embargo, en vez de utilizar bobinas el control de posicionamiento es enteramente digital utilizando microcontroladores especializados

- Los datos son enviados desde la computadora por medio del puerto de video de los circuitos de la pantalla LCD
- Estos dispositivos cuentan con un microprocesador encargado de determinar la posición de cada pixel
- una pantalla LCD cuanta con dos placas de vidrio, una de ellas esta iluminada de la parte trasera por una luz intensa procedente de lámparas CCFL (lámparas fluorescentes de cátodos fríos) lo que permite el brillo de la pantalla
- una vez que se determina el pixel a colorear, la celda cuenta con tres sustancias propensas a recibir corriente y colorearse de algún color básico (verde, rojo, azul) por medio de polarización.
- La corriente que le llega a cada pixel determina la saturación para cada color y asi se genera la gama de colores.
- El proceso se repite cada vez que cambia las imágenes en la pantalla
- El microcontrolador genera señales que implementan lo que se denomina line by line addressing(también denominado multiplexacion) mediante emisión de pulsos analógicos de amplitud modulada

Figura 7

Señales de control Line-by-line addressing



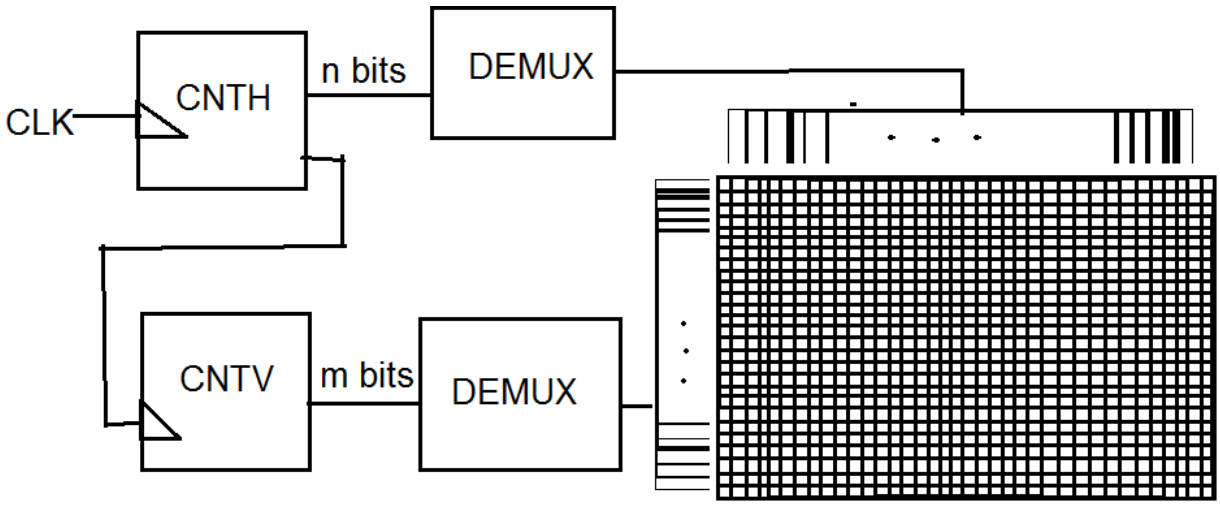
Line by line addressing esta basado en la seleccion de las filas de una matriz de pixeles una a la vez, una línea es seleccionada mediante un pulso de amplitud +Vr mientras que las otras filas están apagadas(a tierra).Los voltajes analógicos aplicados son +Vc para apagar un pixel y -Vc para encenderlo es decir que el estado está determinado por su polaridad (esto para asegurar una larga duración del cristal liquido).

Una vez seleccionadas todas las líneas se termina de mostrar la imagen completa.

El proceso descrito anteriormente se encuentra implementado en el microcontrolador mediante contadores y multiplexores internos de la siguiente manera:

Figura 8

Proceso interno del microcontrolador



Si tenemos un contador horizontal (CNTH) de n bits y un contador vertical (CNTV) de m bits y la cuenta de cada contador actua como seleccionador de datos de un DEMUX la cantidad de posibles salidas es de 2^n y 2^m en horizontal y vertical respectivamente es decir la resolución es de 2^nX 2^m.

Entonces si $CNTH[0]$ y $CNTV[0]$ el Demux muestra en su salida un valor digital que corresponda a ON en el pixel (0,0) y en el resto de los pixeles una señal digital correspondiente a OFF

En el siguiente ciclo de reloj $CNTH[1]$ y $CNTV[0]$ el Demux muestra en su salida un valor digital que corresponda a ON en el pixel (1,0) y en el resto de los pixeles una señal digital correspondiente a OFF

Si se finaliza la línea (n-1) se debe cambiar de fila reiniciando el contador horizontal e incrementando la cuenta del vertical ,entonces: $CNTH[0]$ y $CNTV[1]$ el Demux muestra en su salida un valor digital que corresponda a ON en el pixel (0,1) y en el resto de los pixeles una señal digital correspondiente a OFF y asi hasta habilitar el ultimo pixel (n, m)

Estas señales ON-OFF son luego convertidas a analógicas y moduladas para generar las señales Vr y Vc descritas anteriormente

Sin embargo esta es una sobre simplificación ya que en la implementación real utiliza protocolos de comunicacion serial, SPI, I2C, etc .Sin embargo obedecen el principio presentado anteriormente.

Figura 9

Diagrama de bloques de un monitor LCD

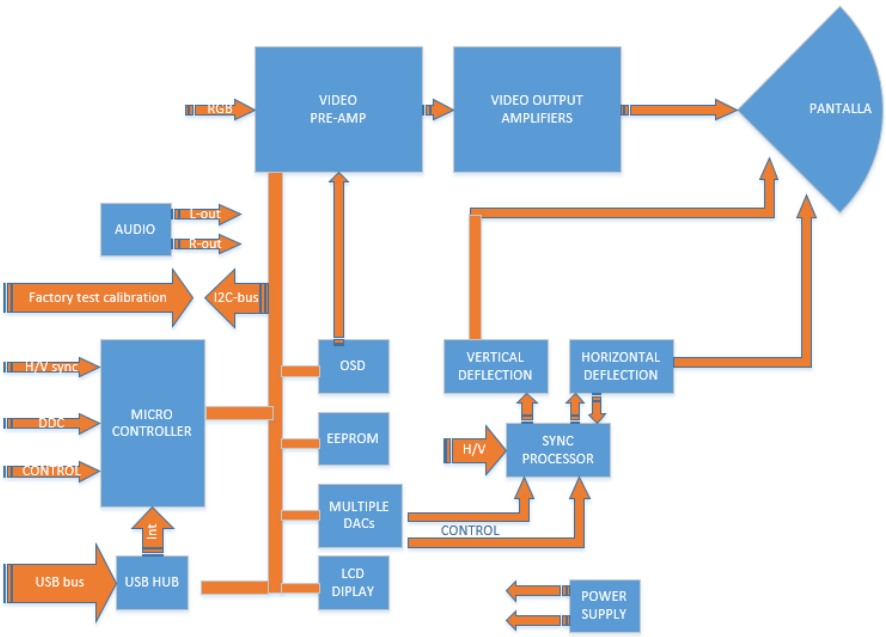


Figura 10

Aspecto externo de un monitor LCD (pantalla plana)

5.1.1.3 Monitor LED

Los monitores LED (*Light Emitting Diode*) utilizan filas de diodos Emisores de Luz (LED) que son pequeños semiconductores que emiten luz visible cuando una corriente eléctrica las atraviesa.

Su principio de funcionamiento es igual al de los monitores LCD, pero en lugar de tubos CCFL para proporcionar la luz de fondo en las pantallas de cristal líquido, las filas de LED proporcionan la luz de fondo.

De esta manera, un monitor LED ofrece un mejor control de la luz, así como una mayor eficiencia pues puede controlar cada LED individualmente.

Puede ser de 2 tipos:

Los monitores LED de borde utilizan diodos emisores de luz que se encuentran en los bordes de la pantalla y se iluminan hacia el centro, lo que proporciona una iluminación más uniforme en toda la pantalla.

Los monitores LED de matriz completa utilizan diodos emisores de luz que cubren toda la superficie de la pantalla y pueden proporcionar una mejor calidad de imagen y un mayor contraste.

Como observamos los conceptos de barrido y escala RGB que utilizan los monitores CRT se trasladan a los monitores más modernos (LCD, LED) permitiendo una clara comprensión del proceso de trazado de imagen.

Figura 11

Estructura interna de un monitor LED

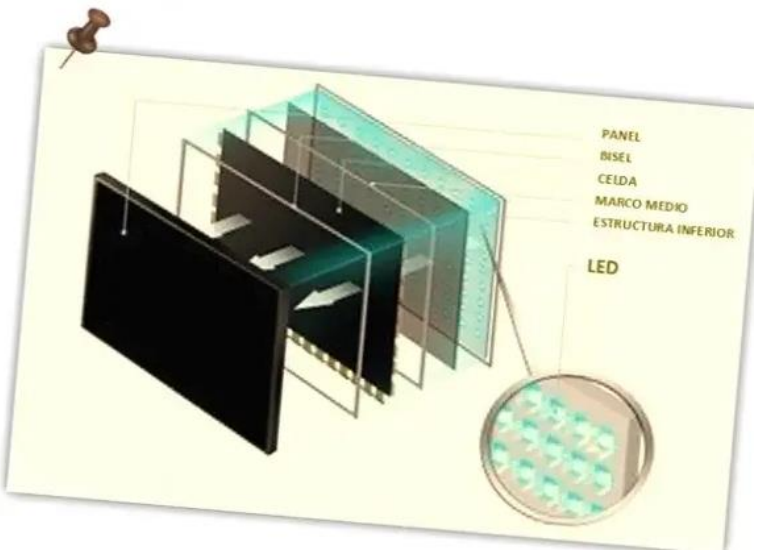


Figura 11

Diferencias entre un monitor LCD y uno LED

5.1.1.4 Otros

Monitor QLED

En lugar del filtro de color se utilizan puntos cuánticos (*Quantum Dot* , QD por sus siglas en inglés) o nanocristales semiconductores que convierten la luz blanca incidente a RGB de acuerdo al diámetro que poseen,este proceso esta estrechamente relacionado con sus niveles de energía

LCD TN

En lugar de utilizar cristales liquidos convencionales este utiliza cristales que trabajan bajo el efecto TN(Twisted Nematic) es decir estos giran de acuerdo a la intensidad de campo eléctrico a la que se someten,dejando pasar mas o menos luz.

LCD IPS

IPS (in-plane switching) utiliza una capa de cristal liquido atrapada entre 2 superficies de vidrio en paralelo con la pantalla, de esta manera se asegura que el cristal liquido se mantenga paralelo al vidrio en todo momento, de esta manera se asegura visión desde diversos angulos

LCD VA

Utilizan el mismo principio que las IPS pero las superficies de vidrio se ubican perpendicularmente a la pantalla

LCD PLS

PLS (Plane to Line Switching) utiliza el mismo principio que el IPS pero la orientación de las superficies de vidrio se encuentran en un angulo preestablecido

OLED

Utiliza diodos OLED (*organic light-emitting diode*) que son un tipo de diodo que se basa en una capa electroluminiscente formada por una película de componentes orgánicos que reaccionan a una determinada estimulación eléctrica, generando y emitiendo luz por sí mismos.

Se aplica voltaje a través del OLED de manera que el ánodo sea positivo respecto del cátodo. Esto causa una corriente de electrones que fluye en sentido contrario, de cátodo a ánodo. Así, el cátodo da electrones a la capa de emisión y el ánodo los sustrae de la capa de conducción.

Seguidamente, la capa de emisión comienza a cargarse negativamente (por exceso de electrones), mientras que la capa de conducción se carga con huecos (por carencia de electrones). Las fuerzas electrostáticas atraen a los electrones y a los huecos, los unos con los otros, y se recombinan (en el sentido inverso de la carga no habría recombinación y el dispositivo no funcionaría). Esto sucede más cerca de la capa de emisión, porque en los semiconductores orgánicos los huecos se mueven más que los electrones (no ocurre así en los semiconductores inorgánicos).

La recombinación es el fenómeno en el que un átomo atrapa un electrón. Dicho electrón pasa de una capa energética mayor a otra menor, liberándose una energía igual a la diferencia entre energías inicial y final, en forma de fotón.

La recombinación causa una emisión de radiación a una frecuencia que está en la región visible y se observa un punto de luz de un color determinado. La suma de muchas de estas recombinaciones, que ocurren de forma simultánea, es lo que llamaríamos imagen.

Monitor Plasma

Utilizan gases nobles en lugar de cristal liquido para generar colores, además estos no requieren iluminación trasera pues generan luz propia

Los gases xenón y neón en un televisor de plasma están contenidos en cientos de miles de células (píxeles) diminutas entre dos paneles de cristal igual que un emparedado. Los electrodos también se encuentran «emparedados» entre los dos cristales, en la parte frontal y posterior de las células. Ciertos electrodos se ubican detrás de las celdas, a lo largo del panel de cristal trasero, y otros electrodos, que están rodeados por un material aislante, dieléctrico y cubiertos por una capa protectora de óxido de magnesio, están ubicados en frente de la celda, a lo largo del panel de cristal frontal. El circuito carga los electrodos que se cruzan creando diferencia de voltaje entre la parte trasera y la frontal, y provocan que el gas se ionice y cambie su estado al de plasma. Posteriormente, los iones del gas corren hacia los electrodos, donde colisionan emitiendo fotones.

5.1.1.5 Confusión entre monitor LED y monitor con tecnología LED

Existe una gran diferencia entre monitor LED y monitor con tecnología LED

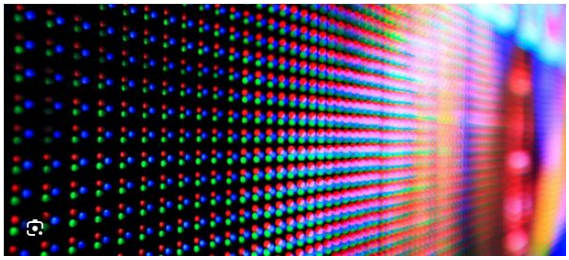
Un monitor LED tiene un microLED individual asignado a cada pixel y uno con tecnología LED utiliza focos LED de tamaño considerable ubicados de manera conveniente en lugar de los tubos CCFL de los monitores LCD

Figura 12

Monitor de tecnología LED

Figura 13

Monitor LED



5.1.1.6 Diferencias entre monitor y televisor

Monitor	**Televisor**
Trabajo y propositos practicos(diseño,etc)	Se utiliza principalmente en entretenimiento
Suele ser de menor tamaño	Se enfoca en el mayor tamaño posible
Tiene un menor angulo de vision	Tiene un amplio angulo de vision
Se enfoca en claridad y precision	Se enfoca en imágenes suaves y definidas
Tiene una amplia variedad de entradas	Tiene pocas entradas
No tiene recepción de ondas	Tiene recepcion de ondas

Si bien ambos pueden utilizar cualquiera de los tipos de pantalla descritos anteriormente estos priorizaran un tipo sobre otro debido al propósito especifico que tienen

Por ejemplo los monitores son mayormente de tecnología LCD pues esta es de menor consumo energético y un angulo de visión estrecho

Mientras que los televisores utilizaran tecnología OLED pues esta permite gran resolución, ángulo de visión y color a expensas de un mayor consumo de energia

5.1.2 VGA

VGA (Video Graphics Array) se trata de un sistema gráfico que presentó la empresa estadounidense International Business Machines (IBM) a fines de la década de 1980 y que se convirtió, por su popularidad, en una especie de estándar para las computadoras (ordenadores) personales.

Las tarjetas gráficas VGA presentan un único circuito integrado que, en sus versiones originales, puede exhibir paletas de 16 y de 256 colores. Estas placas de video tienen una tasa de refresco con un límite de 70 Hz y están en condiciones de mostrar hasta un total de 800 píxeles en sentido horizontal y de 600 líneas.

Tras el auge del VGA, comenzó a desarrollarse una evolución conocida como Super VGA o SVGA que fue lanzado en 1989, diseñado para brindar mayores resoluciones que el VGA.

Aunque esta resolución ha sido superada por otros estándares más modernos, como HDMI o DisplayPort, todavía se utiliza en muchos dispositivos antiguos y en sistemas de retroalimentación de video.

5.1.2.1 Características

Conexión física. VGA utiliza un cable de 15 pines para transferir señales de vídeo analógico. Cada uno de los 15 pines en el cable VGA se utiliza para transmitir diferentes componentes de la señal de vídeo, como los colores primarios (rojo, verde y azul) y los datos de sincronización.

Resolución. Es capaz de transmitir señales de vídeo en una resolución de 640×480 píxeles. Esta resolución se considera baja en comparación con los estándares más modernos, pero sigue siendo adecuada para muchas aplicaciones.

Compatibilidad. Es muy versátil y compatible con una amplia gama de dispositivos, incluyendo computadoras de escritorio y portátiles, consolas de juegos, reproductores de DVD y más. También es compatible con una variedad de sistemas operativos, como Windows, MacOS y Linux.

Calidad de imagen. La transmisión de señales a través de un cable VGA puede ser vulnerable a interferencias electromagnéticas, lo que puede resultar en una calidad de imagen borrosa o con pérdida de detalle.

Frecuencia de actualización. La frecuencia máxima de actualización para una señal VGA es de 85Hz, lo que significa que el monitor puede actualizarse hasta 85 veces por segundo.

Esto puede ser suficiente para muchas aplicaciones, pero es posible que resulte insuficiente para aplicaciones que requieren una frecuencia de actualización más alta, como juegos o videos en tiempo real.

5.1.2.2 Puerto VGA

Figura 14

Cable VGA de 15 pines

Figura 15

Numeración de pines

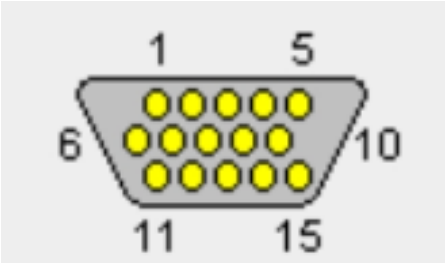


Tabla de Pines

Pin	Nombre	Dirección	Descripcion	Nota	Conexión
1	RED	→	Video(Rojo)	(75 o,0.7V)	✓
2	GREEN	→	Video(Verde)	(75 o,0.7V)	✓
3	BLUE	→	Video(Azul)	(75 o,0.7V)	✓
4	RES	–	Reservado		
5	GND	–	Tierra		✓
6	RGND	–	Tierra(Rojo)		✓
7	GGND	–	Tierra(Verde)		✓
8	BGND	–	Tierra(Azul)		✓
9	5V	→	5 VDC		
10	SGND	–	Sync Ground		✓
11	ID0	←	ID0 del monitor	Opcional(DDC)	
12	SDA	↔	DDC línea de datos serial		
13	HSYNC/CSYNC	→	Sincronización (Horizontal/Compuesta)		✓
14	VSYNC	→	Sincronización (Vertical)		✓
15	SCL	↔	DDC línea de datos reloj		

Nota: *DDC (Display Data Channel):Conjunto de protocolos para la comunicación digital entre una pantalla de ordenador y un adaptador gráfico que permite a la pantalla comunicar sus modos de visualización que permiten al ordenador ajustar ciertos parámetros como el brillo y contraste.

El presente proyecto no involucra la utilización de dichos protocolos.

5.1.2.3 Sincronización (Barrido)[5]

Figura 16

Señal de sincronización

Toda señal de sincronización debe cumplir ciertos parámetros temporales:

Display time (Active video). Tiempo en el cual se envían todos los pixeles de una línea o columna.

Front Porch. Tiempo de envió de pixeles “vacios” antes del pulso de sincronización.

Sync Pulse. Señal de lógica negativa que le indica al monitor que se finalizóélenvió de los pixeles correspondientes.

Back Porch. Tiempo de envió de pixeles “vacios” después del pulso de sincronización.

-Whole line/frame.Duración total de un periodo completo de las señales anteriormente descritas.

Figura 17

Señal de sincronización horizontal HSYNC

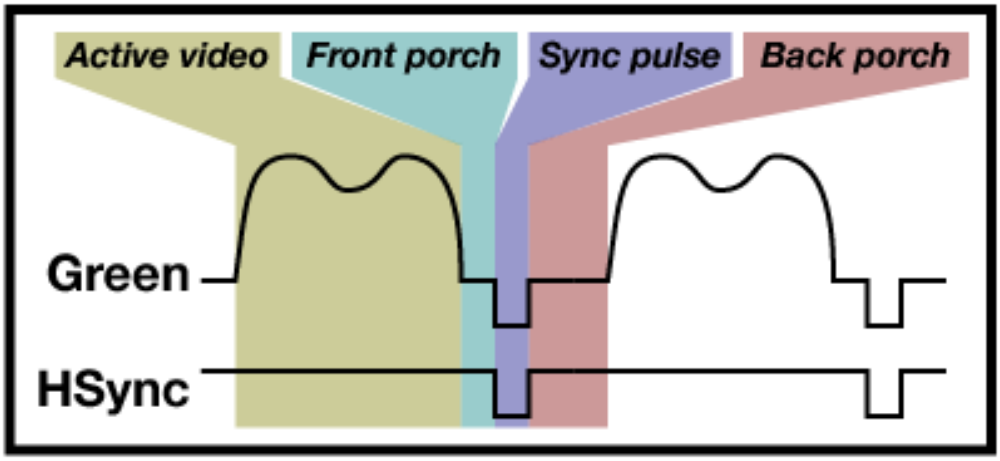


Figura 18

Una composición de las señales de sincronización horizontal y vertical

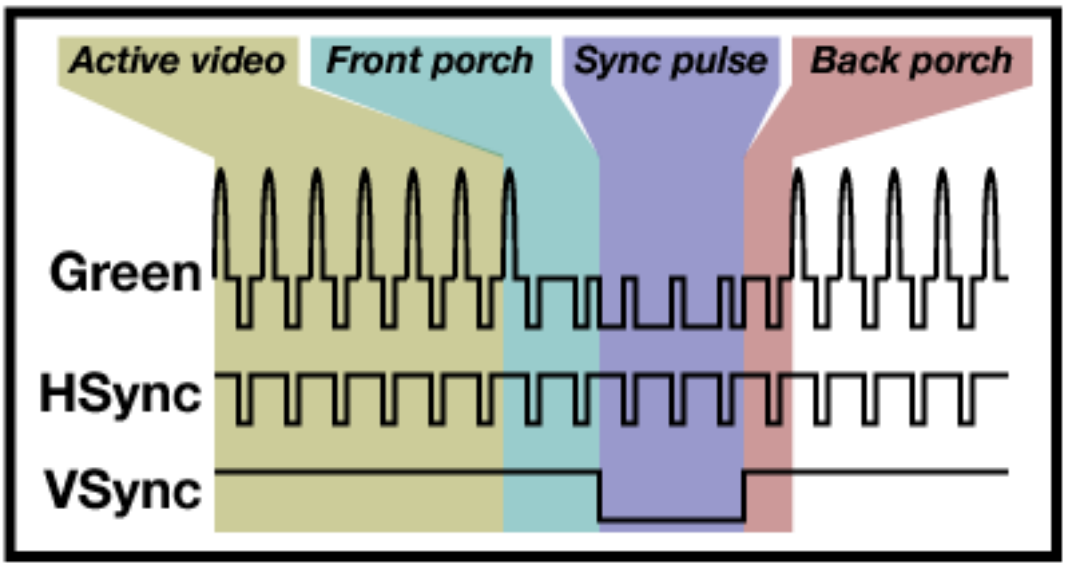
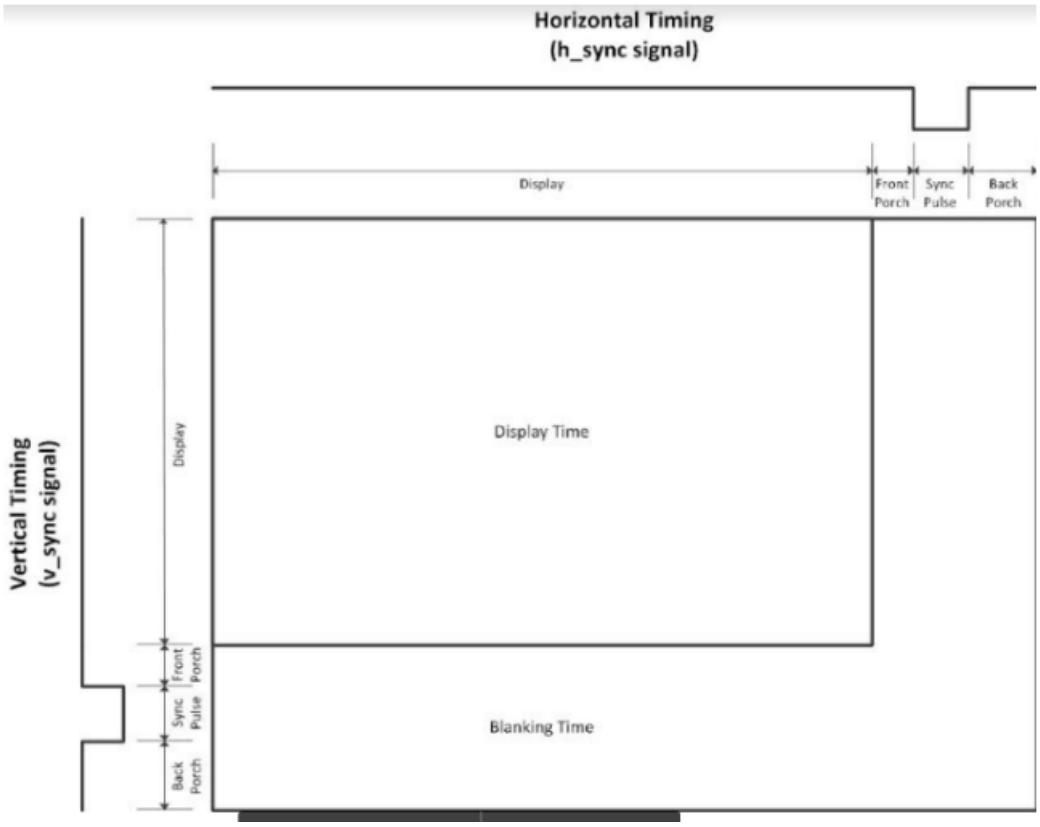


Figura 19

Formación de la imagen en pantalla



La duración de los parámetros previamente mencionados depende de los siguientes factores:

Pixel clock. Frecuencia de pixel

Screen fresh rate. Frecuencia de refresco de la pantalla

Vertical refresh. Frecuencia de refresco de línea

Área de pantalla. (AxB pixeles)

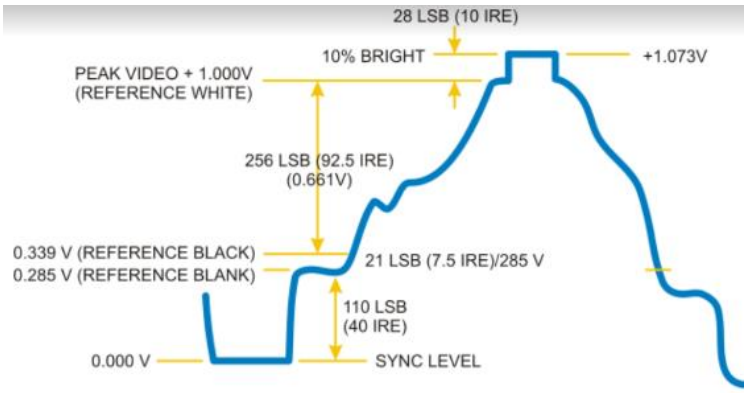
Y son calculados utilizando herramientas computacionales que utilizan la GTF (Generalized Time Formula) provista por VESA (Video Electronics Standars Association).

5.1.3 Otros estándares graficos

-RCA: Mas específicamente el cable amarillo de video,este contiene información relativa a la sincronización y blanqueo) modulada de forma analógica,esta información es descompuesta en un determinado espectro de frecuencias,para posteriormente utilizarla de la manera antes descrita

Figura 20

Señal analógica modulada



Observamos que cuando el voltaje es de (0,285-0,339 V) se indica un periodo de blanqueo (vertical u horizontal)

La primera parte de la señal de voltaje entre (0,339-1 V)se indica el color de los pixeles

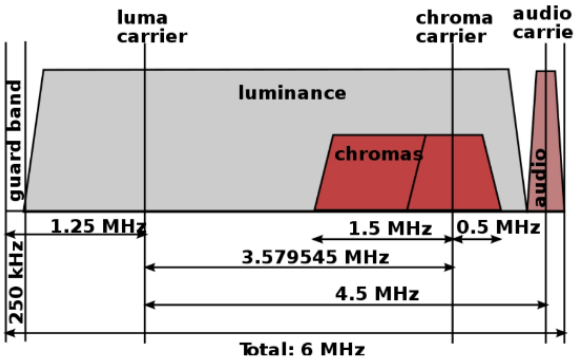
El resto de la señal indica el brillo y el contraste de cada pixel

Figura 21

Señal analógica en un osciloscopio

Figura 22

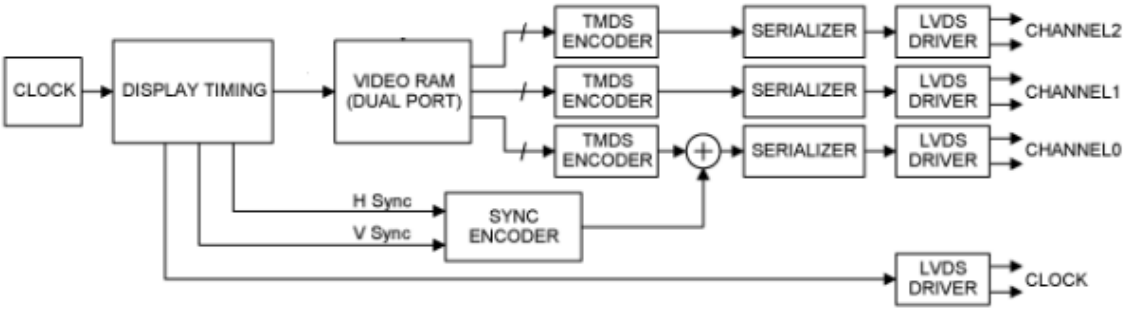
Espectro de frecuencias correspondientes a cada aspecto de un pixel



-DVI:

Figura 23

Diagrama de bloques del estándar DVI



El formato DVI puede ser considerado como una versión digital del VGA. En lugar de utilizar DACs para el RGB,se utilizan:

Codificadores TMDS(Transition-minimized differential signaling): Utilizan un avanzado algoritmo de codificación que reduce sensibilidad a interferencias electromagnéticas y delay del reloj, producidas por la longitud del cable; incrementa el numero de bits recibidos en 2 siendo estos señales de control, y aplica XOR,XNOR a los bits de entrada según convenga.

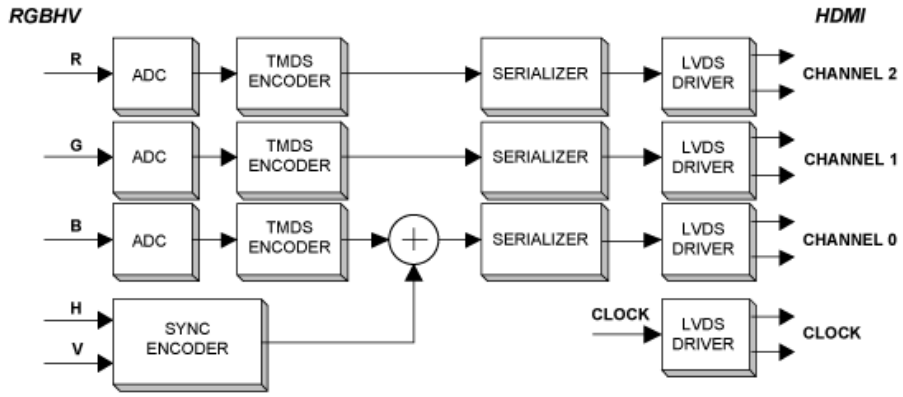
Serializadores: Transforman los bits en paralelo y los transmiten en serie

LVDS Driver(Low-voltage differential signaling): Es un sistema de diferenciacion de señal que transmite informacion analogica de acuerdo a la diferencia de voltajes en un par de terminales

-HDMI:

Figura 24

Diagrama de bloques del estándar HDMI

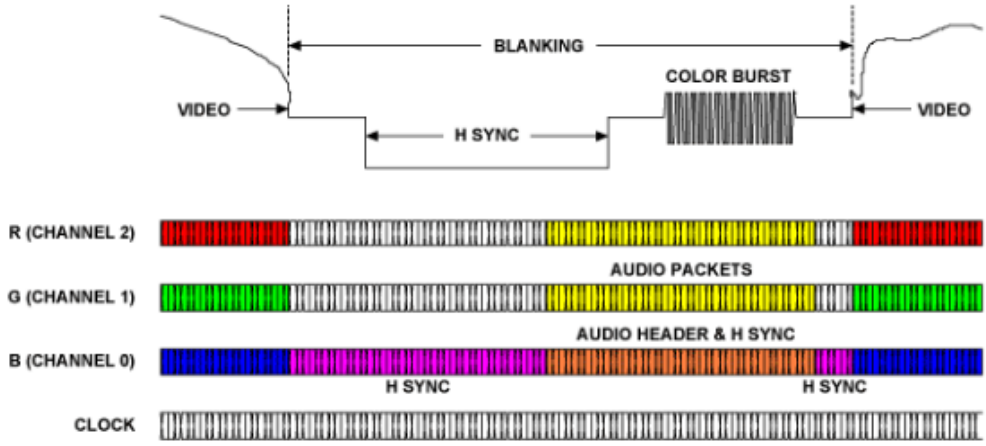


Presenta una combinación de ambos formatos,recibe las señales analógicas RGB y digitales (Hsync y Vsync) para luego convertirlas a digitales y codificadas utilizando el codificador TMDS,para luego sufrir el mismo tratamiento que las señales DVI

Es posible transmitir audio durante el periodo de blanqueo después del Front Porch de la siguiente manera

Figura 25

Diagrama temporal de una señal HDMI

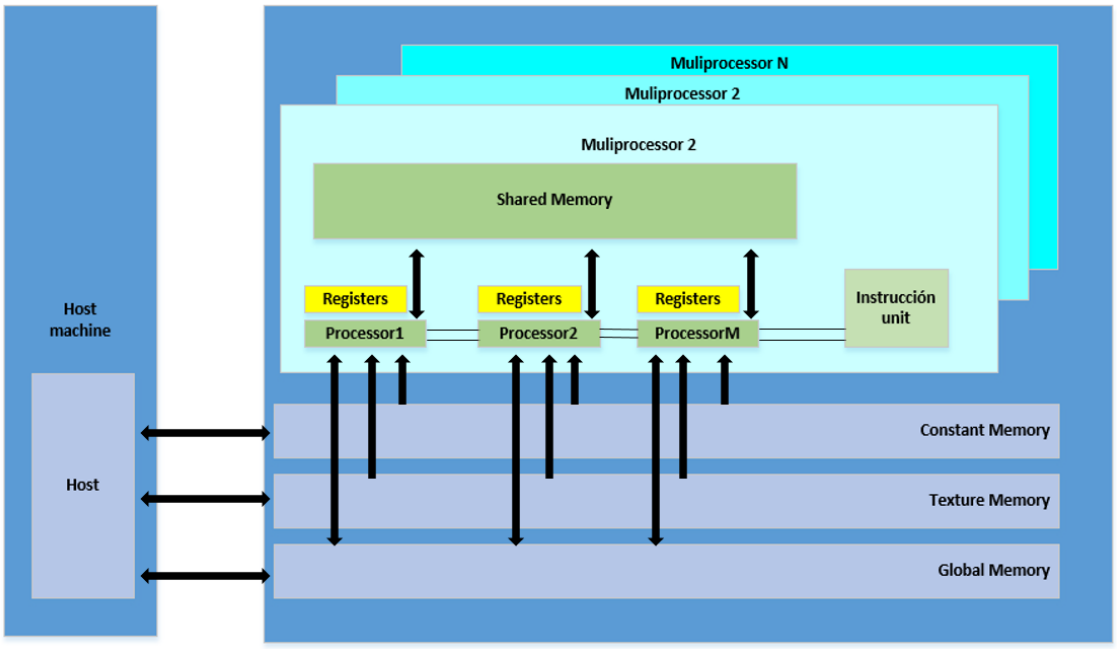


5.1.4 Desplazamiento

Para entender cómo es posible desplazar una imagen a través de la pantalla en cualquier dirección es necesario estudiar el funcionamiento de la GPU (Graphics Processor Unit) que realiza dicha operación en ordenadores modernos.

Figura 26

Diagrama de bloques de un GPU



Esta sección presenta una breve descripción general de la arquitectura GPU, el modelo de programación y la estructura típica de la aplicación. Si bien nos enfocamos en la arquitectura y el entorno de programación (la Arquitectura de dispositivo unificado de cómputo), surgen problemas similares con otros proveedores de GPU o entornos de programación. En una vista de alto nivel, las GPU se componen de varios multiprocesadores SIMD. Cada multiprocesador incorpora una pequeña pero rápida memoria compartida (16 a 48 KB). Todos los procesadores del multiprocesador tienen acceso directo a esta memoria. Además, todos los multiprocesadores tienen acceso a otros tres módulos de memoria a nivel de dispositivo: global, textura, y constante módulos de memoria, a los que también se puede acceder desde el host. La memoria global admite operaciones de lectura y escritura y es la más grande (con un tamaño que oscila entre 256 MB y 4 GB). La textura y las memorias constantes son mucho más pequeñas y solo ofrecen acceso de lectura a los subprocesos de la GPU. Aparte del tamaño, la característica crítica que diferencia a los distintos módulos de memoria es su latencia de acceso. Mientras que acceder a la memoria compartida toma hasta cuatro ciclos, toma de 400 a 800 ciclos acceder a la memoria global. En consecuencia,

para lograr el máximo rendimiento, las aplicaciones deben maximizar el uso de la memoria compartida y los registros del procesador.

Donde

Memoria global. Es una memoria de lectura/escritura y se localiza en la tarjeta del GPU. ***Memoria para constantes.*** Es una memoria rapida (cache) de lectura y se localiza en la tarjeta del GPU.

Memoria para texturas. Es una memoria rapida (cache) de lectura y se localiza en la tarjeta del GPU.

Memoria local. Es una memoria de lectura/escritura para los hilos y se localiza en la tarjeta del GPU.

Memoria compartida. Es una memoria de lectura/escritura para los bloques y se localiza dentro del circuito integrado del GPU.

Memoria de registros. Es la memoria mas rapida, de lectura/escritura para los hilos y se localiza dentro del circuito integrado del GPU.

La memoria compartida y los registros son los mas rapidos, pero su tamaño esta limitado porque es memoria ubicada dentro de un circuito integrado. Por otra parte, la memoria localizada en la tarjeta del dispositivo (local, global, para constantes y para texturas) es grande pero presenta mayor latencia en los accesos, en comparacion con la memoria alojada dentro del circuito integrado.

5.1.4.1 Etapas de procesamiento de una tarea de GPU

Fase	**Operaciones realizadas**
1. pre procesado	Inicialización del dispositivo; asignación de memoria en el host y el dispositivo; pre procesamiento de datos específicos de la tarea en el host.
2. Transferencia de datos	Transferencia de datos desde la memoria del host a la memoria global del dispositivo.

3. procesamiento	Bucle: . transferencia de datos desde la memoria GPU global a la memoria compartida. . Procesamiento del 'núcleo' de la tarea. . Transferencia de resultados a la memoria global.
4. Datos de transferencia	Transferencia de salida del dispositivo a la memoria global a la memoria del host.
5. Post procesamiento	Pos procesamiento específico de la tarea en el host; liberación de recursos [opcional].

Una tarea GPU se compone de cinco etapas:

1.-pre procesamiento

2.-transferencia de datos de host a dispositivo;

3.- procesamiento del núcleo;

4.- transferencia de resultados de dispositivo a host y pos procesamiento.

Como indica la tabla, para una aplicación de datos paralelos, el paso de procesamiento generalmente se repite hasta que se procesan todos los datos de entrada.

El soporte para transferencias de datos eficientes y técnicas de gestión de subprocesos son cruciales para preservar los beneficios de la descarga. Además, surgen preocupaciones específicas del acelerador: por ejemplo, en el caso de una GPU, extraer el paralelismo primitivo de destino e, igualmente importante, emplear técnicas eficientes de gestión de hilos y memoria. La descomposición de tareas, la asignación y administración de memoria o la programación de transferencia de datos incorrectas pueden conducir a una degradación dramática del rendimiento.

Transferencias de datos de host a dispositivo de una copia. Las transferencias de datos hacia y desde el dispositivo requieren transferencias DMA desde (respectivamente a) la memoria del host no paginable. Si una aplicación presenta datos que residen en la memoria paginable para transferirlos a la GPU, la biblioteca de tiempo de ejecución de CUDA primero asigna un nuevo búfer en la memoria no paginable, copia los datos en este nuevo búfer y

finalmente inicia la transferencia DMA. Por lo tanto, para evitar estos gastos generales adicionales (copia de datos y nueva asignación de búfer), la aplicación debe presentar los datos en búferes asignados en la memoria no paginable.

Memoria

El modelo de programación permite el acceso a varios tipos diferentes de memoria, como se muestra en la Figura. Los valores de muestra de píxeles de imagen de entrada y la palabra clave de salida y los valores de longitud se almacenan en la memoria global del dispositivo. Por lo tanto, la memoria del dispositivo global tiene la mayor latencia de todos los tipos de memoria disponibles; Para garantizar transacciones de memoria eficientes, es importante asegurarse de que todas las operaciones de memoria estén fusionadas.

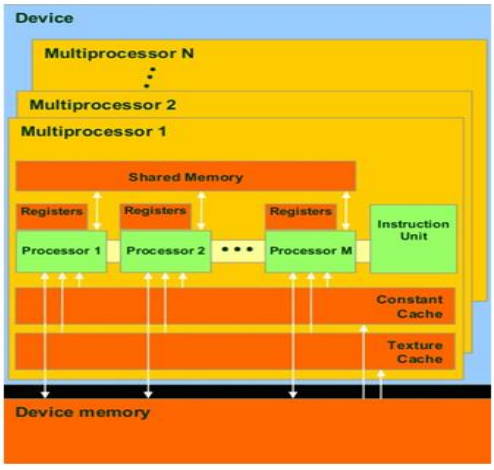
La memoria compartida es local para cada bloque de subprocesos, por lo que facilita la cooperación entre subprocesos y el intercambio de datos. En nuestro kernel CCSDS-123, utilizamos la memoria compartida de baja latencia en el chip para el almacenamiento de los valores de diferencia local y vector de peso.

Los registros son el tipo de memoria más rápido disponible en la GPU, son locales para cada subproceso y tienen la misma duración que el subproceso. Todos los valores de cálculo intermedios se almacenan en los registros del procesador. El número de registros utilizados por el subproceso utilizado por el kernel es 72 cuando se compila para SM 5.0 arquitectura GPU. Cuando el análisis de perfiles se realiza en nuestra implementación se encontró que este relativamente alto en número de registros utilizados por el factor limitante para el rendimiento del núcleo.

Como el número de registros por GPU son fijos, el número de registros utilizados limita el número de subproceso y el número de bloques que se pueden ejecutar en paralelo.

Figura 27

Diagrama de bloques del GPU



THRUST Suma Inclusiva

THRUST es una biblioteca de algoritmos paralelos de código abierto y proporciona una interfaz flexible de alto nivel para GPU de rutinas optimizadas. Cuenta con un abundante dato de primitivos paralelos de datos, como escanear, ordenar y reducir, que son componentes clave de muchos algoritmos complejos. La biblioteca se incluye con el kit de herramientas CUDA y la documentación extensa está disponible en línea. El algoritmo THRUST se puede llamar desde el código del host y del dispositivo y, además, se puede ejecutar en cualquier ubicación donde se proporcionen diferentes políticas de paralización. En esta implementación, se utiliza el algoritmo THRUST Inclusive Sum para calcular las ubicaciones de desplazamiento en el flujo de bits para las palabras de código variables generadas a partir del algoritmo CCSDS-123. Al calcular este desplazamiento como una suma inclusiva de las longitudes de las palabras de código individuales, podemos realizar el empaquetado de las palabras de código en el flujo de bits final sin dependencias seriales

Núcleo de empaquetador de bits

El empaquetador de bits ha sido diseñado para admitir las dos matrices de salida del núcleo. El primero contendrá los datos binarios de la palabra clave de longitud variable, representados en forma de número entero, que se empaquetarán en un flujo de bits de salida. El segundo contendrá las longitudes acumulativas sumadas, por el kernel de suma inclusiva de THRUST, de las palabras de código que se empaquetarán. Para las imágenes en mosaico, esto se vuelve un poco más complicado que el caso clásico en mosaico. Cuando se haya realizado el mosaico, el empaquetador de bits también deberá calcular el desplazamiento adicional requerido para garantizar que el extremo de cada mosaico se rellene para el relleno de bits para garantizar que los mosaicos estén alineados con los límites de bytes.

La operación del empaquetador de bits se puede realizar en todo el número de píxeles de la imagen, ya que no habrá dependencias de datos. Por lo tanto, cada subproceso será responsable del empaquetado de la palabra clave comprimida de un solo píxel en el flujo de bits final. Como cada subproceso funcionará de forma independiente, es importante asegurarse de que no haya dos subprocesos que intenten escribir en la misma ubicación de memoria al mismo tiempo, por lo tanto, se utiliza la operación atómica. Los tamaños comprimidos para cada mosaico individual también deberán almacenarse con el flujo de bits para permitir que el decodificado identifique el inicio y el final de cada mosaico para permitir la descompresión posterior.

Se descubrió, a través de la experimentación, que la configuración del kernel de la cuadrícula de bloques y bloques de subprocesos está mejor configurada para depender del hardware de la GPU para maximizar los recursos de la GPU en lugar de depender de las dimensiones de la imagen de entrada. Como no hay dependencias de datos en la función, todos los datos se pueden tratar de forma independiente para obtener el máximo rendimiento.

El algoritmo de compresión estipula que se garantizará que el flujo de bits de salida no sea mayor que el tamaño de la imagen original, por lo tanto, podemos reutilizar la memoria asignada para almacenar los valores de píxeles de entrada para el almacenamiento del flujo de bits de salida, lo que elimina una gran operación de asignación de memoria adicional.

5.2 Diseño del circuito

El presente proyecto utilizara una configuración de 800x600 pixeles con una frecuencia de refresco de la pantalla de 60 Hz.

Utilizando la formula[7] anteriormente mencionada obtenemos:

Figura 28

Temporización VGA para el modo 800x600 60 Hz

General timing

	Teorico	Experimental
Screen refresh rate	60 Hz	60.3152 Hz
Vertical refresh	37.878787878788 k Hz	37.8778 k Hz
Pixel freq	10.0 M Hz	10.0 M Hz

Horizontal timing (line)

		Teorico	Experimental
Scanline part	Pixels	Time[us]	Time[us]
Visible area	200	20	19.8
Front porch	10	1	1.16
Sync pulse	32	3.2	3.24
Bank porch	22	2.2	2.24
Whole line	264	26.4	26.2

Vertical timing (frame)

		Teorico	Experimental
Frame part	Lines	Time[ms]	Time[ms]
Visible area	600	15.84	15.6
Front porch	1	0.0264	0.028
Sync pulse	4	0.1056	0.108
Bank porch	23	0.6072	0.604
Whole frame	628	16.5792	16.6

Como observamos se cambió la frecuencia de reloj de 40MHz a 10MHz por las siguientes razones:

-Los CI (contadores) tienen una frecuencia de trabajo máxima inferior a 40 MHz

-Se tendrá un periodo de reloj de 0.1µs lo que facilita el cálculo enormemente

$$20\mu s \rightarrow 200\ pulsos\ de\ 0{,}1\mu s$$

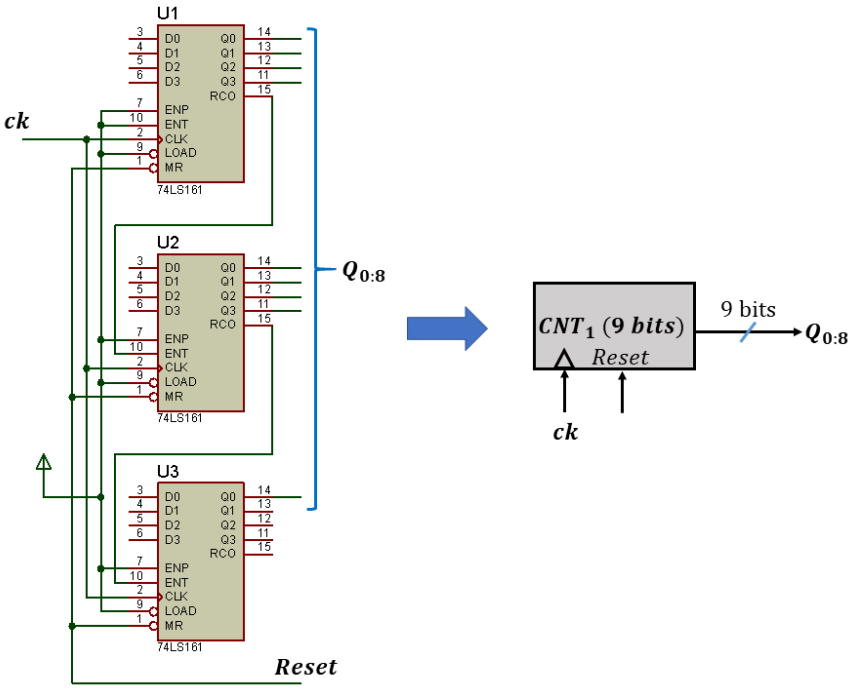
5.2.1 Barrido Horizontal y Vertical

Diseñaremos un circuito que cuente estos pixeles para saber en qué pixel estamos como ser el pixel 0, pixel 200, pixel 210, pixel 242 o el pixel 264, todo esto para la parte de temporización horizontal.

Entonces necesitamos un circuito contador que cuando llegue a 264 se restablezca y vuelva a contar, para poder contar hasta el 264 necesitamos $\log_2 264 = 9$ bits, el CI 74LS161 es un contador de 4 bits, conectaremos 3 de estos CI en cascada.

Figura 29

Contador 9 bits



Ahora queremos saber cuándo llegaremos al 200, 210, 242 y 264 que en binario están dados de la siguiente forma:

$$200_{10} = 0\ 1100\ 1000_2$$

$$210_{10} = 0\ 1101\ 0010_2$$

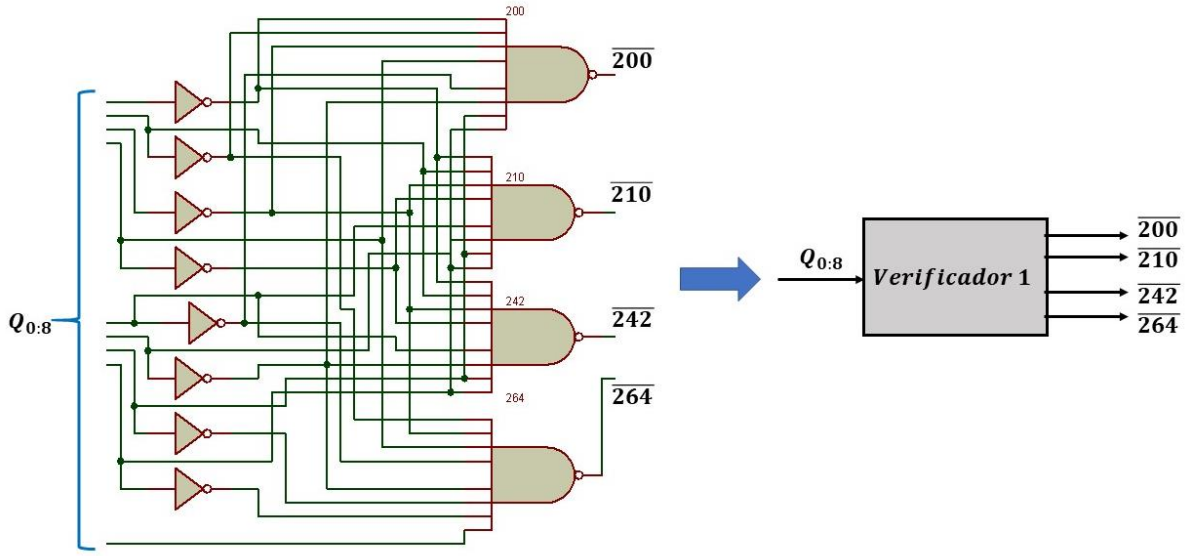
$$242_{10} = 0\ 1111\ 0010_2$$

$$264_{10} = 1\ 0000\ 1000_2$$

Para verificar que estamos en esos números diseñaremos un circuito verificador, compuesto por 8 compuertas NOT (2 CI 74LS04) y 4 compuertas NAND de 8 entradas (4 CI 74LS30).

Figura 30

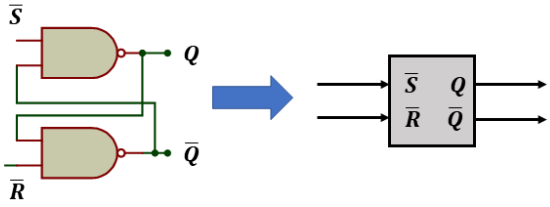
Circuito verificador



Una vez llegado aquí nuestro objetivo es mostrar una señal que indique cuando nos encontramos en el tiempo de visualización (display time) y otra señal que nos indique cuando nos encontremos en el pulso de sincronización horizontal (Sync. Pulse) para lograr esto, haremos el uso de dos flip-flop SR, el cual podemos construir a partir de 2 compuertas NAND de 2 entradas (CI 74LS00), conectamos las entradas del FF SR a los terminales $\overline{264}$ y $\overline{200}$ para el tiempo de visualización y las entradas del otro FF SR entre los terminales $\overline{210}$ y $\overline{242}$ para el pulso de sincronización horizontal.

Figura 31

Latch SR



Realizamos exactamente el mismo análisis para la parte de temporización vertical.

5.2.2 Lectura de datos de la memoria

Para mostrar una imagen para ello necesitamos almacenar los datos de pixeles de una imagen en una memoria, entonces usaremos una memoria EEPROM para almacenar la imagen.

La cantidad de direcciones limita la cantidad de imágenes que podemos grabar (como se observara posteriormente

Para nuestro diseño inicial (1 imagen) necesitaríamos una memoria EEPROM de 19 líneas de dirección Además, que con el cambio que realizamos en la frecuencia de reloj, obtenemos hasta este momento una resolución de 200x600 en lugar de 800x600, la cual es muy rara, con estas observaciones cambiaremos la resolución.

Para tener las proporciones correctas no haremos uso del bit menos significativo del área visible del contador de la parte de temporización horizontal y tampoco usaremos los tres bits menos significativos del área visible del contador de la parte de temporización vertical, como podemos ver a continuación.

- $X: 0 - 200$

$$0_{10} = 0000000\mathbf{0}_2$$

$$1_{10} = 0000000\mathbf{1}_2$$

$$2_{10} = 0000001\mathbf{0}_2$$

$$\vdots$$

$$199_{10} = 1100011\mathbf{1}_2$$

$$200_{10} = 1100100\mathbf{0}_2$$

Eliminando el bit menos significativo tendremos:

$$0_{10} = 0000000_2$$

$$0_{10} = 0000000_2$$

$$1_{10} = 0000001_2$$

$$1_{10} = 0000001_2$$

$$\vdots$$

$$99_{10} = 1100011_2$$

$$99_{10} = 1100011_2$$

$$100_{10} = 1100100_2$$

- $Y: 0 - 600$

$$0_{10} = 0000000\mathbf{000}_2$$

$$1_{10} = 0000000\mathbf{001}_2$$

$$2_{10} = 0000000\mathbf{010}_2$$

$$\vdots$$

$$599_{10} = 1001010\mathbf{111}_2$$

$$600_{10} = 1001011\mathbf{000}_2$$

Eliminando los tresbits menos significativos tendremos:

$$0_{10} = 0000000_2$$

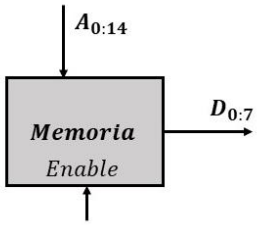
$$\vdots$$

$$75_{10} = 1001011_2$$

Con estos cambios tendremos una resolución de 100x75, usaremos entonces 14 líneas de dirección, 7 para X y 7 para Y.

Figura 32

Señales de entrada y salida de la memoria a utilizar



5.2.3 Código de colores

En la memoria tenemos 8 salidas, la interfaz VGA tiene tres pines asignados para los colores rojo, verde y azul y cada uno espera un voltaje entre 0 voltios y 0.7 voltios, dependiendo de en qué voltaje se encuentren estos pines, se determinara el color del pixel.

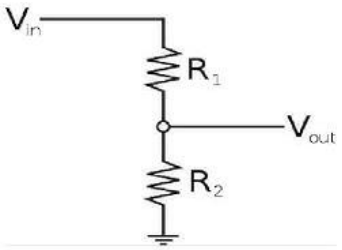
En los pines de salida de la memoria solo podemos obtener 0 voltios o 5 voltios, por lo tanto, usaremos un divisor de voltaje, tomando en cuenta que la impedancia en los pines del VGA es 75Ω.

Con esta consideración y supongamos inicialmente que queremos un voltaje de 0.7 voltios, tendríamos lo siguiente:

$$V_{IN} = 5V \quad V_{OUT} = 0.7VR_2 = 75\Omega$$

Figura 33

Divisor de tensión



$$V_{OUT} = \frac{R_2}{R_1 + R_2} V_{IN}$$

Calculando R_1:

$$R_1 = 460.7\Omega$$

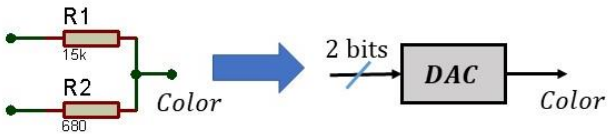
Ahora si quisiéramos cuatro voltajes diferentes desde 0 voltios a 0.7 voltios espaciados uniformemente, tendremos 4 valores diferentes para R_1.

00	$0\ V$	$Circuito\ abierto$
01	$0.23\ V$	1555.4Ω
10	$0.47\ V$	722.9Ω
11	$0.7\ V$	460.7Ω

Para esto usaremos 2 resistencias en paralelo, como podemos ver en la figura.

Figura 34

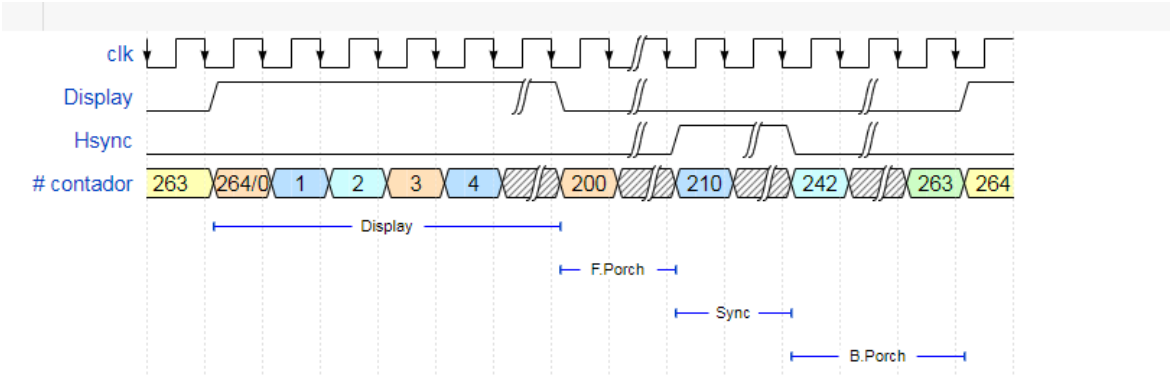
DAC de 2 bits



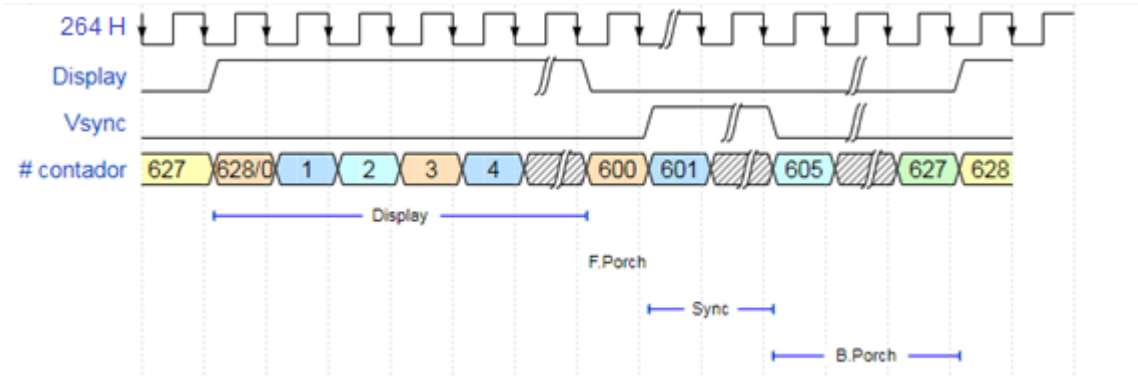
Con esto podemos ver que usaremos 6 bits de datos saliendo de la EEPROM para obtener cuatro tonos diferentes de rojo, verde y azul que dan como resultado 64 colores en total[8].

El diagrama de tiempos resultante es:

Sincronización horizontal



Sincronizacion Vertical



5.2.4 Diseño Final

Para expandir el número de imágenes posible solo se deben utilizar más direcciones de la memoria, en este caso la implementación se realizará utilizando 3 direcciones adicionales lo que resulta en 8 imágenes grabadas[9].

Para lograr esto usamos un teclado con 16 teclas, donde cada una de ellas cumple una función, las cuales son las siguientes:

Tecla	Función
1	Velocidad 1 a la que se muestran las 4 imágenes en bucle
2	Velocidad 2 a la que se muestran las 4 imágenes en bucle
3	Velocidad 3 a la que se muestran las 4 imágenes en bucle
A	Muestra la imagen 1
4	Velocidad 4 a la que se muestran las 4 imágenes en bucle
5	Monoestable tiempo 0
6	Monoestable tiempo 1
B	Muestra la imagen 2
7	Monoestable tiempo 3
8	Monoestable tiempo 4
9	Manual
C	Muestra la imagen 3
*	Diapositiva, información del grupo
0	Pone en pause en una de las 4 imágenes en bucle (Mario)
#	Muestra 4 imágenes en bucle (Mario)
D	Muestra la imagen 4

Con esto ya tenemos nuestro circuito completamente diseñado, el circuito final es el siguiente:

Figura 35

Circuito de control

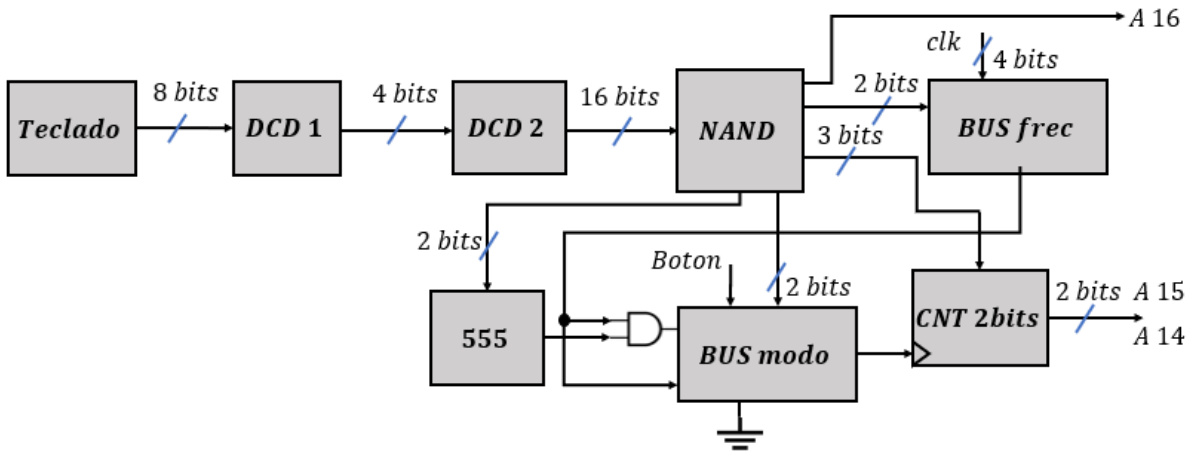
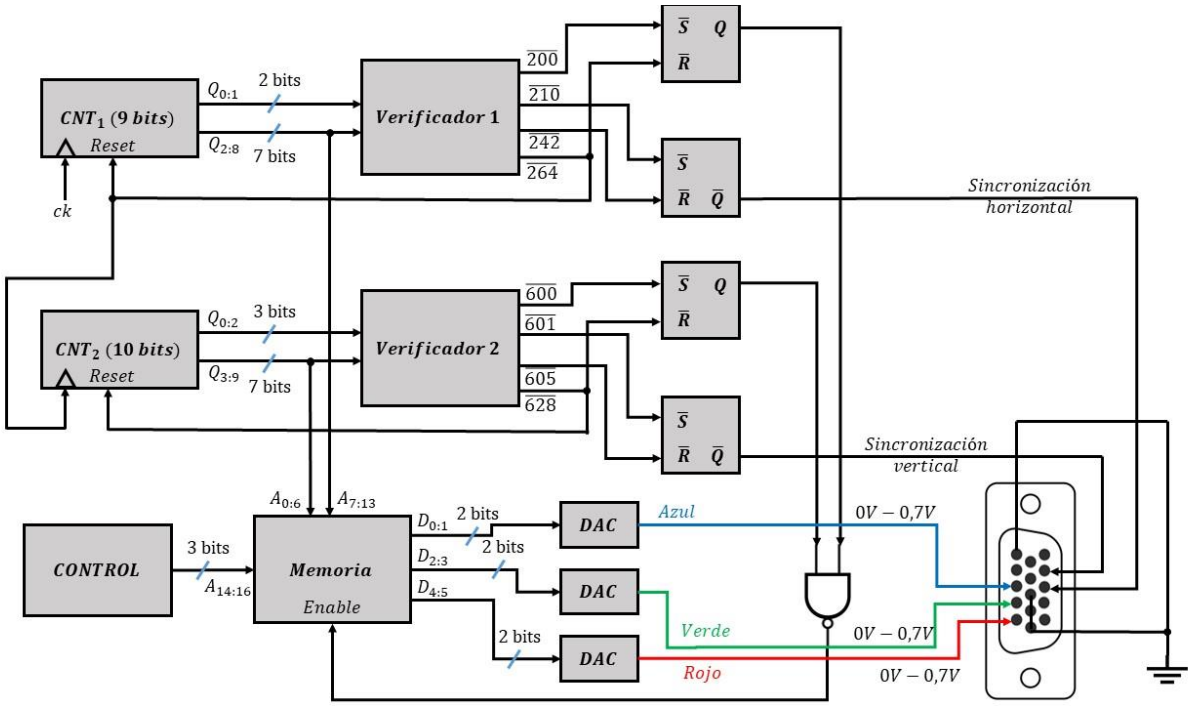


Figura 36

Circuito final



5.2.5 Expansion: posible circuito de desplazamiento

Tras estudiar el modelo de funcionamiento de una GPU se identificaron los siguientes componentes principales:

-Unidad de instrucciones: Como un símil de una SIC controla los registros, memorias, contadores, etc.

-Memoria compartida: Una GPU utiliza múltiples procesadores que comparten una memoria, en este caso nos servirá para organizar los pixeles

-Memoria global: Ubicación de los datos de los pixeles de la imagen

-ALU: Permite el uso de diversas operaciones aritméticas y lógicas, en este caso suma y resta

Algoritmo de reescritura

Si observamos: El desplazamiento de una imagen de 4x4 pixeles tenemos:

0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15

Siendo cada numero la hipotética posición en la memoria a leer (0-15)

Si quisiéramos desplazar hacia la derecha obtenemos:

3	0	1	2
7	4	5	6
11	8	9	10
15	12	13	14

Es decir se debe regrabar en la memoria los datos de la siguiente manera

Si D[i]=Dato correspondiente al anterior arreglo

Si N[i]=Dato correspondiente al nuevo arreglo

$$\mathrm{N}[0] = \mathrm{D}[3]$$

$$\mathrm{N}[1] = \mathrm{D}[0]$$

……

$$\mathrm{N}[15] = \mathrm{D}[14]$$

Podemos establecer entonces que:

$$\mathrm{N}[\mathrm{i}] = \begin{cases} \mathrm{D}[\mathrm{i}+3]\,, \mathrm{i} = 0,4,8,12 \\ \mathrm{D}[\mathrm{i}-1], \text{otros casos} \end{cases}$$

Extendiendo este razonamiento al presente proyecto tenemos:

$$\mathrm{N}[\mathrm{i}] = \begin{cases} \mathrm{D}[\mathrm{i}+99]\,, \mathrm{i} = \text{primer pixel de cada columna} \\ \mathrm{D}[\mathrm{i}-1], \text{otros casos} \end{cases}$$

De la misma manera para desplazar en otras direcciones tenemos:

Arriba

$$N[i] = \begin{cases} D[i-384] \text{ , } i = \text{pixeles de la ultima fila} \\ D[i+128], \text{otros casos} \end{cases}$$

Izquierda

$$N[i] = \begin{cases} D[i-99] \text{ , } i = \text{ultimo pixel de cada columna} \\ D[i+1], \text{otros casos} \end{cases}$$

Abajo

$$N[i] = \begin{cases} D[i+384] \text{ , } i = \text{pixeles de la primera fila} \\ D[i-128], \text{otros casos} \end{cases}$$

Notamos que existen 4 números recurrentes en cada desplazamiento: 1, 99,384 y 128,números a utilizarse en operaciones aritméticas(ALU)

El hecho de que se requiera regrabar datos en la memoria implica el uso de memorias RAM (Memoria Global MG, Memoria Compartida MC) y EEPROM (MI)

Descripcion del circuito

El funcionamiento del circuito puede considerarse una implementación del algoritmo de reescritura, pues en el proceso de lectura de la memoria global MG también se escriben esos datos en la memoria compartida MC en direcciones determinadas por el algoritmo,una vez que se dejan de leer datos(front porch) se realiza la escritura directa de los contenidos de MC a MG en direcciones determinadas por numero de fila o columna actual

La memoria de instrucciones MI funciona con un PC(que sincronizara su funcionamiento) y una entrada I que definirá la dirección de desplazamiento, generando diversas señales de control internas que controlan procesos de carga de contadores, función de la ALU, habilitación de lectura/escritura, etc.

Un registro se encargara de detectar el número de cuenta de los contadores e iniciar el proceso de escritura de acuerdo a la dirección (vertical, horizontal)

Otro registro se encargara de sincronizar el proceso de lectura/escritura bidireccional entre las memorias RAM, siendo controlado por buffers tri-state activados por la memoria de instrucciones MI

Se utilizaran bloques WR que se encargaran de los procesos de lectura/escritura de acuerdo al funcionamiento de la RAM a utilizar

Los bloques resaltados en rojo son bloques provenientes del anterior diseño (NANDS, contadores)

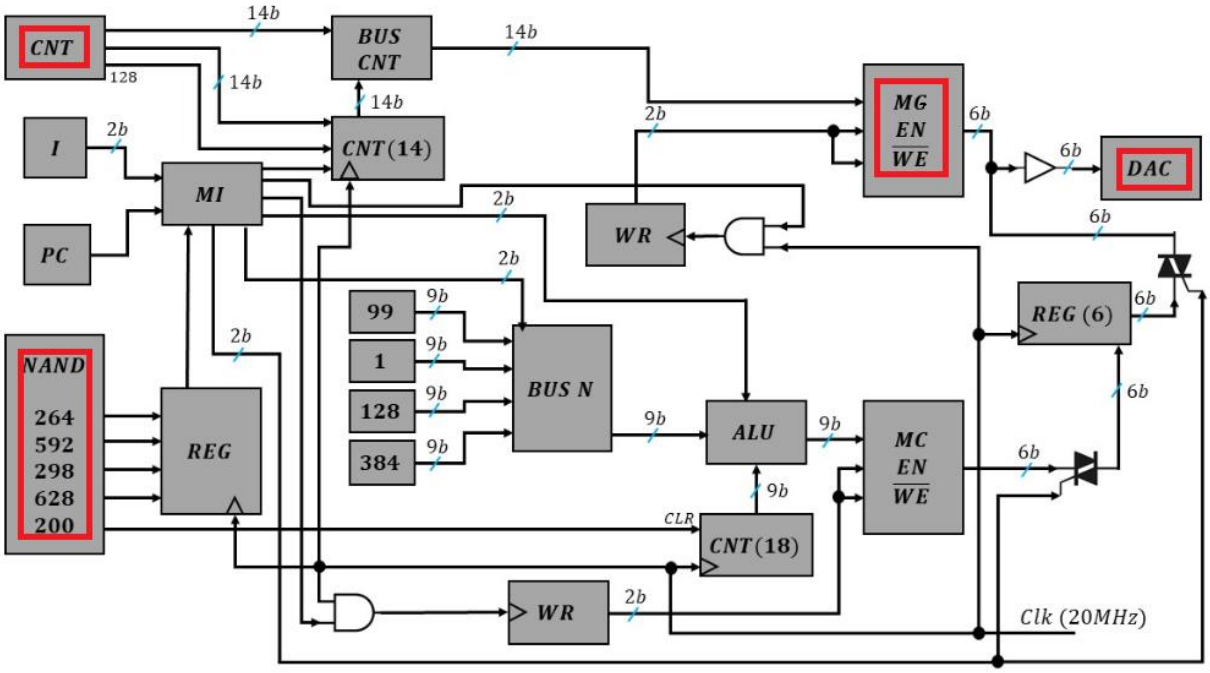
Se utilizan dos buses:

-BUSCNT: Direcciona las direcciones de la memoria MG de acuerdo al proceso que se ejecuta (Lectura, escritura)

-BUSN: Direcciona los números a sumar/restar (mediante la ALU) del algoritmo de reescritura

Figura 37

Circuito de desplazamiento propuesto



6. Conclusiones y recomendaciones.

Se aprendió el proceso de visualización de imágenes en un monitor, a partir del envió de datos mediante el protocolo VGA.

Se logro diseñar un circuito que cumple los requisitos temporales del protocolo VGA además de otras características, como ser la resolución de una imagen y la frecuencia de reloj, todo esto con ayuda de lo aprendido en la materia de Sistemas digitales 2.

Además, se diseñó e implemento un circuito de control capaz de seleccionar entre visualizar una imagen en específico, visualizar una sucesión de imágenes durante un determinado tiempo, desplazamiento de imágenes, etc.

Se logro diseñar e implementar el proyecto de manera satisfactoria, obteniendo los resultados esperados como se puede ver en la parte de anexos.

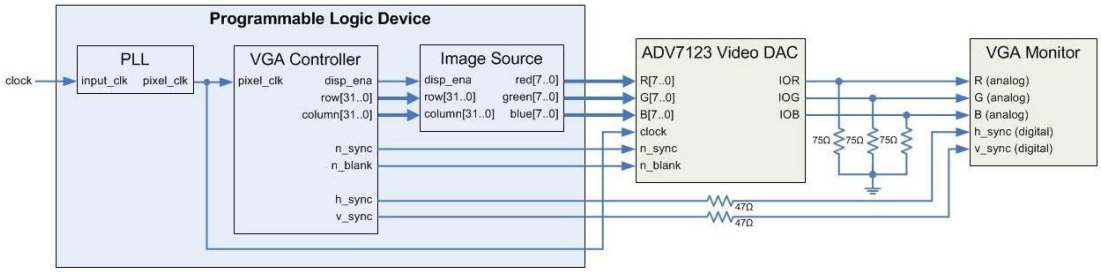
Si bien podemos mejorar tanto la resolución como la cantidad de imágenes mostradas por el monitor, esto implica realizar un circuito de mayor magnitud, mayor cantidad de circuitos

integrados, mayor capacidad de memoria, sin embargo, es posible realizar dicha ampliación utilizando la misma idea mostrada en el presente proyecto.

Una posible implementación utilizando el FPGA (Cyclone II) utilizando el software Altera Quartus II, puede realizarse de la siguiente manera:

Figura 38

Diagrama de bloques del controlador VGA implementado utilizando Quartus II



Donde:

Clk puede ser introducido externamente mediante un pin dedicado o generado internamente

Figura 39

Tabla de señales de entrada y salida

Port	Width	Mode	Data Type	Interface	Description
pixel_clk	1	in	standard logic	user logic	Pixel clock at the frequency specified for the desired VGA mode.
reset_n	1	in	standard logic	user logic	Asynchronous active low reset.
h_sync	1	out	standard logic	VGA monitor	Horizontal sync signal.
v_sync	1	out	standard logic	VGA monitor	Vertical sync signal.
disp_ena	1	out	standard logic	user logic	Display enable; 1 = display time, 0 = blanking time.
row[31..0]	32	out	integer	user logic	Y pixel coordinate (i.e. row); 0 = top row, number of rows - 1 = bottom row.
column[31..0]	32	out	integer	user logic	X pixel coordinate (i.e. column), 0 = leftmost column, number of columns - 1 = rightmost column.
n_blank	1	out	standard logic	video DAC	Determines if direct blanking is used. This example permanently sets this bit to '1', so no direct blanking is used.
n_sync	1	out	standard logic	video DAC	Determines if sync-on-green is used. This example permanently sets this bit to '0', so no sync-on-green is used.

En la figura podemos observar las diversas señales a generar mediante el FPGA, no ahondaremos en su significado pues ya se estableció anteriormente.

El diseño se realizará para una imagen de resolución 1920X1200 @60Hz con una frecuencia de reloj =193,16MHz.

Programando las señales de acuerdo a las especificaciones necesarias obtenemos:

Figura 40

Codigo VHDL:Definicion de las variables a utilizar:Generic(señales internas),port(señales de entrada y salida)

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

ENTITY vga_controller IS
  GENERIC(
    h_pulse  : INTEGER := 208;    --horiztonal sync pulse width in pixels
    h_bp     : INTEGER := 336;    --horiztonal back porch width in pixels
    h_pixels : INTEGER := 1920;   --horiztonal display width in pixels
    h_fp     : INTEGER := 128;    --horiztonal front porch width in pixels
    h_pol    : STD_LOGIC := '0';  --horizontal sync pulse polarity (1 = positive, 0 = negative)
    v_pulse  : INTEGER := 3;      --vertical sync pulse width in rows
    v_bp     : INTEGER := 38;     --vertical back porch width in rows
    v_pixels : INTEGER := 1200;   --vertical display width in rows
    v_fp     : INTEGER := 1;      --vertical front porch width in rows
    v_pol    : STD_LOGIC := '1'); --vertical sync pulse polarity (1 = positive, 0 = negative)
  PORT(
    pixel_clk : IN   STD_LOGIC;  --pixel clock at frequency of VGA mode being used
    reset_n   : IN   STD_LOGIC;  --active low asycnchronous reset
    h_sync    : OUT  STD_LOGIC;  --horiztonal sync pulse
    v_sync    : OUT  STD_LOGIC;  --vertical sync pulse
    disp_ena  : OUT  STD_LOGIC;  --display enable ('1' = display time, '0' = blanking time)
    column    : OUT  INTEGER;    --horizontal pixel coordinate
    row       : OUT  INTEGER;    --vertical pixel coordinate
    n_blank   : OUT  STD_LOGIC;  --direct blacking output to DAC
    n_sync    : OUT  STD_LOGIC); --sync-on-green output to DAC
END vga_controller;
```

```
ARCHITECTURE behavior OF vga_controller IS
  CONSTANT h_period : INTEGER := h_pulse + h_bp + h_pixels + h_fp; --total number of pixel clocks in a row
  CONSTANT v_period : INTEGER := v_pulse + v_bp + v_pixels + v_fp; --total number of rows in column
BEGIN

  n_blank <= '1';  --no direct blanking
  n_sync <= '0';   --no sync on green

  PROCESS(pixel_clk, reset_n)
    VARIABLE h_count : INTEGER RANGE 0 TO h_period - 1 := 0;  --horizontal counter (counts the columns)
    VARIABLE v_count : INTEGER RANGE 0 TO v_period - 1 := 0;  --vertical counter (counts the rows)
  BEGIN

    IF(reset_n = '0') THEN    --reset asserted
      h_count := 0;             --reset horizontal counter
      v_count := 0;             --reset vertical counter
      h_sync <= NOT h_pol;      --deassert horizontal sync
      v_sync <= NOT v_pol;      --deassert vertical sync
      disp_ena <= '0';          --disable display
      column <= 0;              --reset column pixel coordinate
      row <= 0;                 --reset row pixel coordinate

    ELSIF(pixel_clk'EVENT AND pixel_clk = '1') THEN
```

Figura 41

Codigo VHDL:Conmutacion de acuerdo a un numero en especifico

```
      --counters
      IF(h_count < h_period - 1) THEN    --horizontal counter (pixels)
        h_count := h_count + 1;
      ELSE
        h_count := 0;
        IF(v_count < v_period - 1) THEN  --veritcal counter (rows)
          v_count := v_count + 1;
        ELSE
          v_count := 0;
        END IF;
      END IF;

      --horizontal sync signal
      IF(h_count < h_pixels + h_fp OR h_count >= h_pixels + h_fp + h_pulse) THEN
        h_sync <= NOT h_pol;     --deassert horiztonal sync pulse
      ELSE
        h_sync <= h_pol;         --assert horiztonal sync pulse
      END IF;

      --vertical sync signal
      IF(v_count < v_pixels + v_fp OR v_count >= v_pixels + v_fp + v_pulse) THEN
        v_sync <= NOT v_pol;     --deassert vertical sync pulse
      ELSE
        v_sync <= v_pol;         --assert vertical sync pulse
      END IF;
```

```
      --vertical sync signal
      IF(v_count < v_pixels + v_fp OR v_count >= v_pixels + v_fp + v_pulse) THEN
        v_sync <= NOT v_pol;     --deassert vertical sync pulse
      ELSE
        v_sync <= v_pol;         --assert vertical sync pulse
      END IF;

      --set pixel coordinates
      IF(h_count < h_pixels) THEN  --horiztonal display time
        column <= h_count;           --set horiztonal pixel coordinate
      END IF;
      IF(v_count < v_pixels) THEN  --vertical display time
        row <= v_count;              --set vertical pixel coordinate
      END IF;

      --set display enable output
      IF(h_count < h_pixels AND v_count < v_pixels) THEN  --display time
        disp_ena <= '1';                                    --enable display
      ELSE                                                --blanking time
        disp_ena <= '0';                                    --disable display
      END IF;

    END IF;
  END PROCESS;

END behavior;
```

Para poder visualizar una imagen es necesario tener los datos necesarios, estos se obtienen a través de una memoria externa o interna, sin embargo es posible generar formas básicas utilizando Vhdl de la siguiente manera:

Donde se codifican los colores en 8 bits es decir (256 colores)

Figura 42

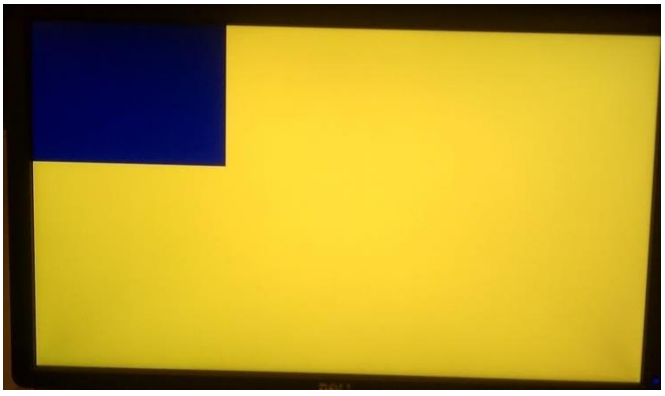
Generación de una imagen de prueba

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

ENTITY hw_image_generator IS
  GENERIC(
    pixels_y :  INTEGER := 478;   --row that first color will persist until
    pixels_x :  INTEGER := 600);  --column that first color will persist until
  PORT(
    disp_ena :  IN   STD_LOGIC;  --display enable ('1' = display time, '0' = blanking time)
    row      :  IN   INTEGER;    --row pixel coordinate
    column   :  IN   INTEGER;    --column pixel coordinate
    red      :  OUT  STD_LOGIC_VECTOR(7 DOWNTO 0) := (OTHERS => '0');  --red magnitude output to DAC
    green    :  OUT  STD_LOGIC_VECTOR(7 DOWNTO 0) := (OTHERS => '0');  --green magnitude output to DAC
    blue     :  OUT  STD_LOGIC_VECTOR(7 DOWNTO 0) := (OTHERS => '0')); --blue magnitude output to DAC
END hw_image_generator;

ARCHITECTURE behavior OF hw_image_generator IS
BEGIN
  PROCESS(disp_ena, row, column)
  BEGIN

    IF(disp_ena = '1') THEN        --display time
      IF(row < pixels_y AND column < pixels_x) THEN
        red <= (OTHERS => '0');
        green  <= (OTHERS => '0');
```

```
        blue <= (OTHERS => '1');
      ELSE
        red <= (OTHERS => '1');
        green  <= (OTHERS => '1');
        blue <= (OTHERS => '0');
      END IF;
    ELSE                           --blanking time
      red <= (OTHERS => '0');
      green <= (OTHERS => '0');
      blue <= (OTHERS => '0');
    END IF;

  END PROCESS;
END behavior;
```

Figura 43

Imagen de prueba



Como observamos de esta manera es posible mejorar enormemente la calidad de imagen utilizando un FPGA,además si se ahonda en la programación es posible lograr el proceso de desplazamiento de imágenes omnidireccional

7.Referencias

-*What is VGA (Video Graphics Array)?* (s.f.). Computer Hope's Free Computer Help. https://www.computerhope.com/jargon/v/vga.htm

-*VGA (VESA DDC) pinout diagram @ pinoutguide.com*. (s.f.). Handbook of hardware schemes, cables layouts and connectors pinouts diagrams @ pinouts.ru. https://pinouts.ru/Video/VGAVesaDdc_pinout.shtml

-Yu, B., & Guang, Y. (2014). The Design and Verification of VGA Controller. *Advanced Materials Research*, *981*, 82–85. https://doi.org/10.4028/www.scientific.net/amr.981.82

-*VGA Signal Timing*. (s.f.). Microcontroller VGA Interface projects. http://www.tinyvga.com/vga-timing

-*VGA Controller (VHDL)*. (s.f.). Electronic Component and Engineering Solution Forum - TechForum | Digi-Key. https://forum.digikey.com/t/vga-controller-vhdl/12794

-Ferraro, R. F. (1994). *Programmer's guide to the EGA,VGA, and Super VGA cards: [includes graphics accelerators]* (3ª ed.). Addison-Wesley.

-*Uso de los componentes basicos de la computadora*. (s.f.). http://cca.org.mx/cca/cursos/cucfc/modulo4/tema2-09.html

-*¿Cómo Funciona un Monitor CRT?* (s.f.). Ordenadores y Portátiles. https://ordenadores-y-portatiles.com/monitor-crt/

-Gian Explica. (2016, 10 de septiembre). *¿Cómo Funcionan LosMonitores? (CRT)* [Video]. YouTube. https://www.youtube.com/watch?v=3fMegsebOUc

-Moran, R. (1994). *The changing display industry: CRT and flat panel*. Business Communications Co.

-Contributor, T. (2019a, 13 de septiembre). *What is LCD (Liquid Crystal Display)?* WhatIs.com. https://www.techtarget.com/whatis/definition/LCD-liquid-crystal-display

-Cabading, Z. (2020, 11 de mayo). *Differences Between An LED Display And LCD Monitor | HP® Tech Takes*. Laptop-Computer, Desktops, Drucker, Tinte und Toner | HP® Deutschland. https://www.hp.com/us-en/shop/tech-takes/differences-between-led-display-and-lcd-monitor

-*RGB Color Codes Chart ðŸŽ¨*. (s.f.). Online Calculators & Tools - RapidTables.com. https://www.rapidtables.com/web/color/RGB_Color.html

-*ALLDATASHEET.COM - Electronic Parts Datasheet Search*. (s.f.). ALLDATASHEET.COM - Electronic Parts Datasheet Search. https://www.alldatasheet.com/

-*Lets build a video card!* (s.f.). Ben Eater. https://eater.net/vga

---

[1] Como se verá más adelante existen diversas razones para establecer esta frecuencia
[2]Más específicamente los contadores 74LS163

[3] Sin embargo se abordara este tema de forma puramente teórica

[4]A partir de este punto VGA será un término equivalente a SVGA

[5]El estudio de señales no incluye a las señales DDC

[6] Esta información será la referencia utilizada en recomendaciones

[7]No se ahondara en método utilizado para hallar los valores presentados posteriormente

[8] Ver Anexos "*Escala de 64 colores utilizada"*

[9] La memoria utilizada tiene 19 bits de direcciones

8. Anexos

Tiempos de sincronía horizontal y vertical

General timing

	Teorico	Experimental
Screen refresh rate	60 Hz	60.3152 Hz
Vertical refresh	37.878787878788 k Hz	37.8778 k Hz
Pixel freq	10.0 M Hz	10.0 M Hz

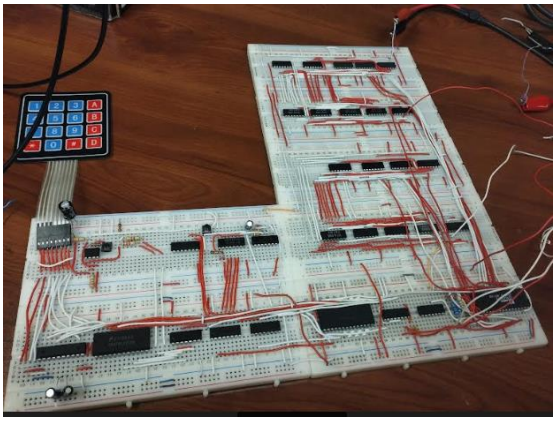
Horizontal timing (line)

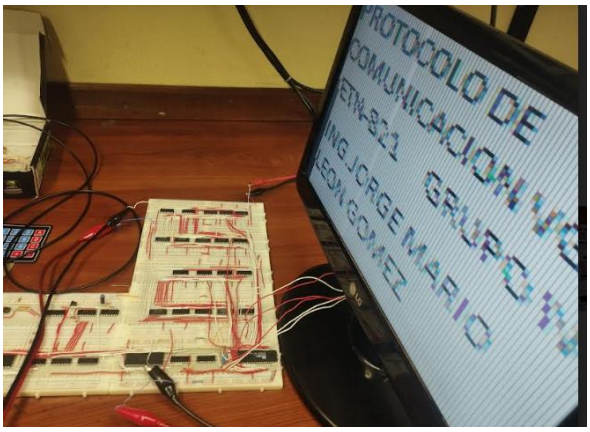
		Teorico	Experimental
Scanline part	Pixels	Time[us]	Time[us]
Visible area	200	20	19.8
Front porch	10	1	1.16
Sync pulse	32	3.2	3.24
Bank porch	22	2.2	2.24
Whole line	264	26.4	26.2

Vertical timing (frame)

		Teorico	Experimental
Frame part	Lines	Time[ms]	Time[ms]
Visible area	600	15.84	15.6
Front porch	1	0.0264	0.028
Sync pulse	4	0.1056	0.108
Bank porch	23	0.6072	0.604
Whole frame	628	16.5792	16.6

Fotos de la implementación





Escala de 64 colores utilizada

El sistema de color **RGB**, construye todos los colores de la paleta a partir de la combinación de los colores **R**ed(Rojo), **G**reen(Verde) y **B**lue(Azul).

La codificación se realiza utilizando 8 bits por color lo que significa que pueden tener los valores decimales desde 0 hasta 255, el 0 indica una ausencia del color y el 255 indica el color en su "brillo" máximo, cualquier valor en medio expresa una presencia parcial.

Estos 8 bits se expresan en un numero hexadecimal de 2 digitos, por ejemplo:

$$0101\ 0101 \rightarrow 55$$

Como la codificación se realizo utilizando 2 bits (salida de la memoria) por color significa que estamos limitados a 4 valores distintos de color, siendo estos:

$$00{,}55, AA, FF$$

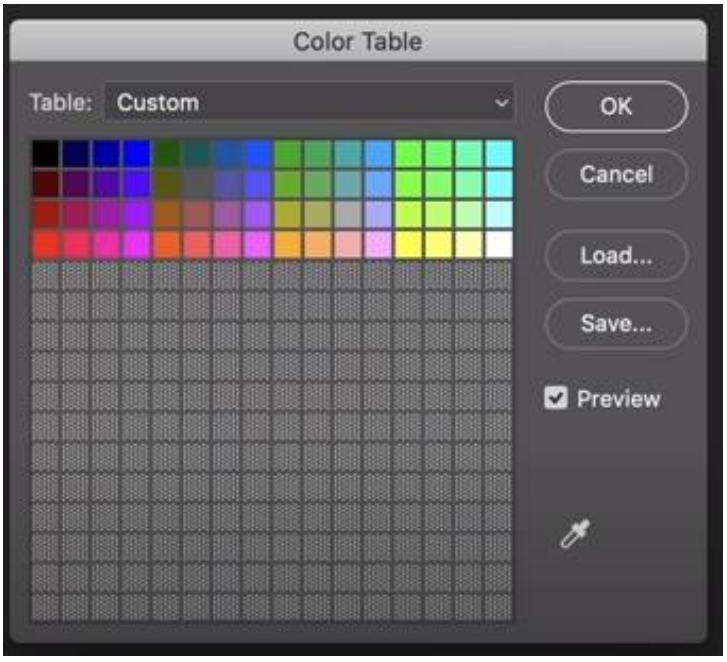
Realizando todas las posibles combinaciones obtenemos

BIN	HEX	BIN	HEX	BIN	HEX	BIN	HEX
000000	#000000	010000	#550000	100000	#AA0000	110000	#FF0000
000001	#000055	010001	#550055	100001	#AA0055	110001	#FF0055

000010	#0000AA	010010	#5500AA	100010	#AA00AA	110010	#FF00AA
000011	#0000FF	010011	#5500FF	100011	#AA00FF	110011	#FF00FF
000100	#005500	010100	#555500	100100	#AA5500	110100	#FF5500
000101	#005555	010101	#555555	100101	#AA5555	110101	#FF5555
000110	#0055AA	010110	#5555AA	100110	#AA55AA	110110	#FF55AA
000111	#0055FF	010111	#5555FF	100111	#AA55FF	110111	#FF55FF
001000	#00AA00	011000	#55AA00	101000	#AAAA00	111000	#FFAA00
001001	#00AA55	011001	#55AA55	101001	#AAAA55	111001	#FFAA55
001010	#00AAAA	011010	#55AAAA	101010	#AAAAAA	111010	#FFAAAA
001011	#00AAFF	011011	#55AAFF	101011	#AAAAFF	111011	#FFAAFF
001100	#00FF00	011100	#55FF00	101100	#AAFF00	111100	#FFFF00
001101	#00FF55	011101	#55FF55	101101	#AAFF55	111101	#FFFF55
001110	#00FFAA	011110	#55FFAA	101110	#AAFFAA	111110	#FFFFAA
001111	#00FFFF	011111	#55FFFF	101111	#AAFFFF	111111	#FFFFFF

Cada código hexadecimal corresponde a un color específico del sistema RGB

Mediante el software Photoshop obtenemos los 64 colores:



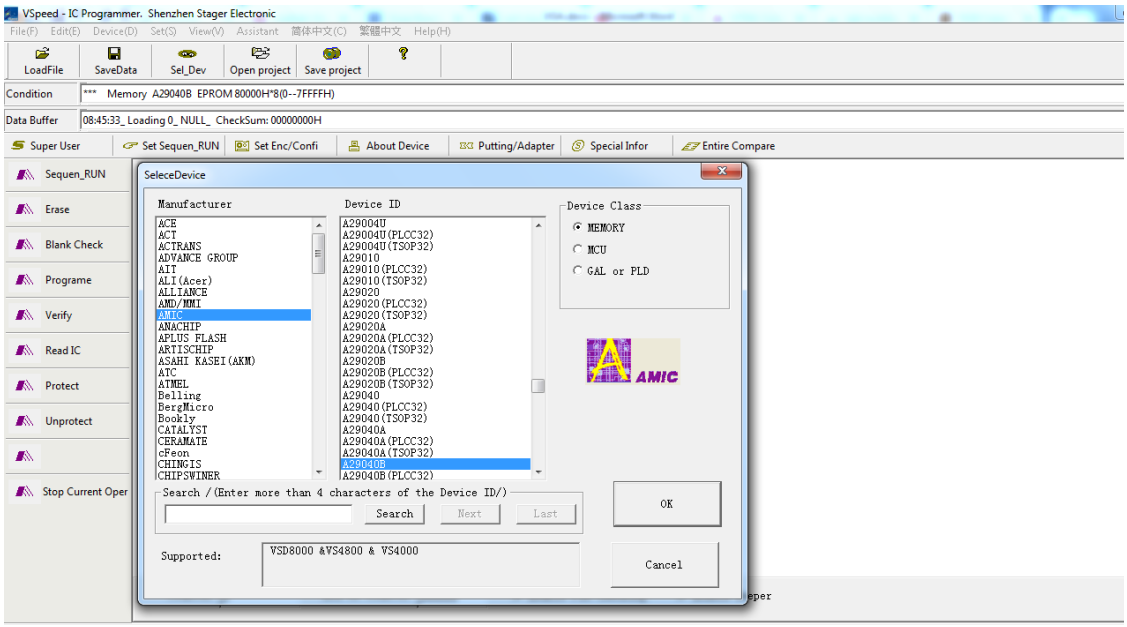
Programa Python para crear el archivo.bin necesario para grabar imágenes en la EEPROM

```
from PIL import Image   %Libreria que convierte una imagen a binario
image = Image.open("image1.png") %Conviertiendo una imagen a binario
pixels = image.load()%Depositando cada pixel en un arreglo
image1 = Image.open("image2.png")
pixels1 = image1.load()
image2 = Image.open("image3.png")
pixels2 = image2.load()
image3 = Image.open("image4.png")
pixels3 = image3.load()
image4 = Image.open("disp1.png")
pixels4 = image4.load()
image5 = Image.open("disp2.png")
pixels5 = image5.load()
image6 = Image.open("disp3.png")
pixels6 = image6.load()
image7 = Image.open("disp4.png")
pixels7 = image7.load()
out_file = open("inch4.bin", "w") %Abriendo un archivo.bin que almacena cada dato
%Escaneando cada imagen
for y in range(128):
for x in range(128):
try:
%Repetimos el proceso
for y in range(256,384):
for x in range(128):
try:
if x>=100:
out_file.write(chr(0))
else:
out_file.write(chr(pixels2[x, y-256]))
except IndexError:
out_file.write(chr(0))
if x>=100: %Una vez alcanzados los 100 pixeles grabar 0's
out_file.write(chr(0))
else:
out_file.write(chr(pixels[x, y])) %Si no grabar el dato binario
except IndexError:
out_file.write(chr(0)) %Si ya no hay pixeles grabar 0's
%Repetimos el proceso
for y in range(128,256):
for x in range(128):
try:
if x>=100:
out_file.write(chr(0))
else:
out_file.write(chr(pixels1[x, y-128]))
except IndexError:
out_file.write(chr(0))
```

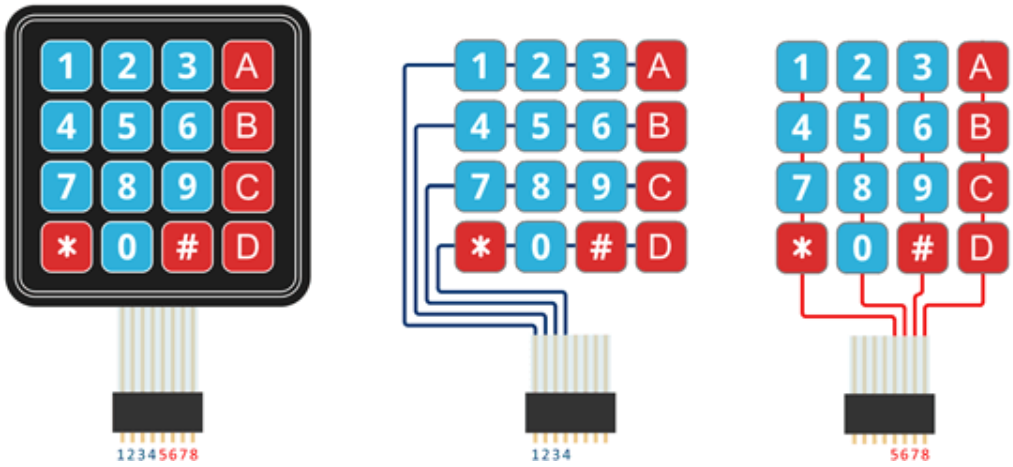
```
%Repetimos el proceso
for y in range(384,512):
for x in range(128):
try:
if x>=100:
out_file.write(chr(0))
else:
out_file.write(chr(pixels3[x, y-384]))
except IndexError:
out_file.write(chr(0))
%Repetimos el proceso
for y in range(512,640):
for x in range(128):
if (y==639 and x>=124):
continue
try:
if x>=100:
out_file.write(chr(0))
else:
out_file.write(chr(pixels4[x, y-512]))
except IndexError:
out_file.write(chr(0))
%Repetimos el proceso
for y in range(640,768):
for x in range(128):
if (y==767 and x>=117):
if x>=100:
out_file.write(chr(0))
else:
out_file.write(chr(pixels7[x, y-1024]))
except IndexError:
out_file.write(chr(0))
continue
try:
if x>=100:
out_file.write(chr(0))
else:
out_file.write(chr(pixels5[x, y-640]))
except IndexError:
out_file.write(chr(0))
%Repetimos el proceso
for y in range(768,896):
for x in range(128):
if (y==895 and x>=123):
continue
try:
if x>=100:
out_file.write(chr(0))
else:
out_file.write(chr(pixels6[x, y-768]))
except IndexError:
out_file.write(chr(0))
%Repetimos el proceso
for y in range(896,1024):
for x in range(128):
if (y==1023 and x>=126):
continue
try:
if x>=100:
out_file.write(chr(0))
else:
out_file.write(chr(pixels7[x, y-896]))
except IndexError:
out_file.write(chr(0))
%Repetimos el proceso
for y in range(1024,1152):
for x in range(128):
try:
```

Software necesario para grabar datos en la memoria

VSpeed IC Programmer



Pines del teclado matricial



Decodificador de teclado MM74C923