
Universidad Autónoma de Querétaro
Facultad de Informática
Maestría en Ciencias de la Computación

# METODOLOGÍA PARA EL ALMACENAMIENTO DE IMÁGENES Y SU PROCESAMIENTO EN HARDWARE, FPGA

TESIS

Que como parte de los requisitos para obtener el Grado de

Maestro en Ciencias de la Computación

**Presenta:**
ING. CARLOS ALBERTO RAMOS ARREGUÍN

**Dirigido por:**
Dr. Jesús Carlos Pedraza Ortega

SINODALES

Dr. Jesús Carlos Pedraza Ortega
Presidente

Dr. Saúl Tovar Arriaga
Secretario

Dr. José Emilio Vargas Soto
Vocal

Dr. Efrén Gorrostieta Hurtado
Suplente

Dr. Marco Antonio Aceves Fernández
Suplente

M. C. Ruth Angélica Rico Hernández
Directora de la Facultad de Informática

Dr. Irineo Torres Pacheco
Director de Investigación y Posgrado

Centro Universitario
Querétaro, Qro.
Octubre 2013
México

**SUMMARY**

This research presents a methodology for image storage hardware and results in external memory of SRAM type. An Altera FPGA manufactured is employ as platform development. The project is composed from the image acquisition, the internal processing of the data to be stored on external memory, then the stored image is displayed on a screen, the image is processed and transformed into a different one. The results are stored in the same memory, the difference, different memory addresses. This will perform three SRAM memory operations: writing, reading and read/write. To carry out such operations, hardware descriptions are designed to allow interaction with memory and display the data on screen. The test images are in color and a size of 256x256 pixels with a color depth of 8 bits. Digital processing is done in software floating point or integer numbers. The hardware is made either integer or fixed point. The results are compared with testing software for fixed-point formats to use. Also, presents he processing times in software and hardware details

(**Key words**: FPGA, SRAM, fixed point)

A veces sientes que no te quedan fuerzas para luchar
A veces, sientes que hasta tu propio corazón se desvanece
Porque ese gran dolor inunda toda tu ser, y también el de los tuyos
Bajas los brazos y te preguntas... ¿Cuánto podré resistir?

¿Sabes? El final de esta historia todavía no está escrito
Debes saber que siempre hay una luz al otro lado de la pared
Cuando te quieras rendir, cuando pienses que todo ha terminado
Déjame que me acerque, déjame que te diga algo al oído
¡Nunca estarás sola!

*Para mi madre por su lucha y superación*
*contra la enfermedad de Cáncer.*

## AGRADECIMIENTOS

A Dios por permitirme seguir en este camino.

A la memoria de Manuela Rodríguez y Candelaria Razo que Dios las tenga en su gloria. En paz descansen.

A mis padres Ma. De la Luz Arreguín y Juan Ramos, quiénes han seguido apoyándome en el transcurso de mi vida profesional y estudiantil, a ellos les debo mi profesión y vida. Gracias.

A mis hermanos Enrique y Juan Manuel por su apoyo a lo largo de mi vida, a mi prometida Sandra Estela Enriquez por su apoyo y cariño incondicional.

A mis amigos Carmen y Juan Carlos, por estar en los buenos y malos momentos de mi vida además de lo académico.

A mis profesores, que además de compartirme su enseñanza, me han apoyado en los malos momentos en todo este camino, a todos los doctores del grupo de trabajo, gracias. En especial a mi asesor Dr. Jesús Carlos Pedraza por sus consejos y apoyo.

Al señor Rector Dr. Gilberto Herrera Ruiz y a la Dra. Rocío Peniche, por todo el apoyo otorgado en el trayecto de mis estudios de maestría.

Al consejo Nacional de Ciencia y Tecnología (CONACYT) por la beca recibida de la convocatoria 290733, número de CV 290796 y número de programa 002950.

**INDICE**

## INDICE DE CUADROS

## INDICE DE FIGURAS

# I. INTRODUCCIÓN

Desde 1964 hasta nuestros días, el campo de procesamiento de imágenes ha crecido enormemente. Las técnicas de procesamiento se usan ahora para resolver una gran variedad de problemas. Aunque a menudo no relacionados, esos problemas requieren comúnmente métodos capaces de realizar y extraer la información contenida en las imágenes para su interpretación y análisis por el hombre ó de manera automática (por computadora). En cualquier caso se contemplan tanto técnicas de mejora de la calidad de las imágenes como relativas a la percepción de la máquina.

Pajares (2008) menciona, que el creciente desarrollo de las tecnologías de adquisición y procesamiento de imágenes, hace que cada día su uso sea más frecuente en el tratamiento e interpretación del contenido de las imágenes. Por otra parte, el desarrollo de equipos cada vez más sofisticados en diversos campos de aplicación, constituyen la base para que el tratamiento de las imágenes sea algo inherente al desarrollo tecnológico.

En el área de procesamiento digital de imágenes, se puede identificar el uso de dos metodologías distintas; primero, tenemos el uso de una metodología genérica para procesamiento digital de imágenes de Gonzalez y Woods (2008), basada en el desarrollo e implementación de algoritmos en computadoras personales e industriales, en el cual, se realiza el procesamiento de los datos de la imagen, el almacenamiento, y la exhibición de los resultados. Además, los lenguajes de alto nivel empleados (Java, C, C++, C#, MATLAB, etc) hacen que esta metodología sea la que se emplea con mayor frecuencia, aunque tiene la desventaja de que solamente puede llevar a cabo un procesamiento secuencial.

La segunda metodología se basa en el uso de hardware de propósito específico, en este caso particular, dispositivos programables en campo **FPGA**. Estos **dispositivos** ofrecen procesamiento de señales en paralelo. Sin embargo, estos dispositivos no tienen la capacidad de almacenar imágenes procesadas, por lo que es necesario el uso de memorias externas tipo RAM. Para lo anterior, es necesario el desarrollo de una interfaz de

comunicación con una PC, para el envío y almacenamiento de resultados, en el caso de no tener una cámara para llevar a cabo la adquisición de las imágenes. En el presente trabajo de investigación, se propone el desarrollo de una metodología para el procesamiento embebido de imágenes en hardware, desarrollando módulos basados en el estándar 1164 del IEEE.

## 1.1. Definición del proyecto de investigación

En este proyecto de investigación se trabaja procesamiento de imágenes en hardware y se guardan los resultados de cada operación en memoria externa, para ello, es necesario establecer algunas técnicas para el almacenamiento de datos. Es importante resaltar el desarrollo de la interfaz para la adquisición de la imagen que va a procesar el sistema. Cabe mencionar que las operaciones de procesamiento pueden llevar intervalos de tiempo en ns, que para el ojo humano es instantáneo.

## 1.2. Objetivos

### 1.2.1. General

Desarrollar una metodología para el almacenamiento de imágenes en memorias RAM, y llevar a cabo pruebas de procesamiento de imágenes en sistemas embebidos utilizando FPGA, bajo los estándares del lenguaje VHDL definido por el IEEE.

### 1.2.2. Específicos

1. Diseñar una descripción de hardware para el almacenamiento de imágenes en una memoria externa y cargarlas al FPGA cuando sea requerido.
2. Diseñar las descripciones de hardware de algunas operaciones de procesamiento de imágenes como pueden ser negativo, escala de grises, entre otros.
3. Comparar la metodología propuesta, con resultados obtenidos en una PC utilizando un software comercial.

### 1.2.3. Alcances y limitaciones

Los alcances de este trabajo de investigación son:

- Adquisición de imagen
- Resolución de la imagen de 256x256 pixeles.
- Almacenamiento de diversas imágenes en un mismo dispositivo de memoria.
- Se realiza el procesamiento a la imagen y el resultado se almacena en memoria.

Las limitaciones en este proyecto son:

- Tiempo de adquisición de imagen relativamente alto.
- Profundidad del color RGB limitado a 8 bit.
- Capacidad de almacenamiento de solamente 2 Mbyte.

## 1.3. Justificación

La rápida evolución de la tecnología de imágenes digitales, acompañado por una gran demanda en el mercado de cámaras y pantallas, presenta un significante cambio para los desarrolladores de dispositivos.

Sofisticados algoritmos de procesamiento de imágenes están disponibles, para llevar a cabo una extracción de imágenes del hardware de captura y exhibición. Desafortunadamente, su implementación está limitada por algunos factores como: la complejidad intrínseca del algoritmo, la presión para reducir el costo de los materiales, la necesidad para soportar una amplia variedad de formatos, y el frecuente requisito para personalizar un ambiente particular de los dispositivos, Tusch (2006).

De acuerdo con Quintero (2006), hoy en día la clave para el desarrollo de dispositivos para realizar procesamiento de señales digitales (especialmente en comunicaciones digitales, así como aplicaciones de procesamiento de imágenes y video), es el uso de dispositivos lógicos programables, en particular los FPGA's (Field Programmable

Gate Arrays). El cual, en la actualidad, ha comenzado a ser muy utilizado en el área de software embebido, ya que son tan eficientes como un ASIC (Application Specific Integrated Circuit) y compiten en costo con los microcontroladores.

Romero (2007) en su libro publica que usualmente, los FPGA's son programados utilizando lenguajes de descripción de hardware. En este caso se trabaja con el lenguaje descriptivo VHDL (Very high speed integrated circuit Hardware Description Language), desconocido para muchos desarrolladores de sistemas embebidos, por lo cual prefieren lenguajes de alto nivel como C, para un rápido desarrollo de alguna aplicación deseada.

En la actualidad, en el campo de procesamiento de imágenes embebido, existe poco trabajo desarrollado, ya que requiere tiempo y especialización por parte del desarrollador de los mismos. Por lo cual, muchos desarrolladores de sistemas embebidos optan por plataformas programadas por medio de lenguajes de alto nivel, como lo son, microcontroladores y DSPs, con el costo de un procesamiento secuencial, lo que se refleja en tiempo de cómputo y costos altos.

Por lo anterior, se tiene un área de oportunidad en el desarrollo de módulos de hardware que puedan ser implementados de una manera sencilla y rápida, explotando al máximo sus principales ventajas:

- Procesamiento paralelo
- Alto número de operaciones realizadas en un segundo
- Reconfigurabilidad
- Protección Intelectual (*IP, Intelectual Propertie*)

Así mismo, en el poco trabajo realizado, se utilizan herramientas de algunos fabricantes de FPGA y fabricantes de tarjetas de desarrollo, empleadas para el almacenamiento temporal de las imágenes a ser procesadas en una memoria externa, ya sea de tipo RAM o Flash. La dimensión de las imágenes que se ha utilizado es de 256x256

pixeles, en escala de grises, este tamaño es bueno para pruebas, sin embargo para realizar un análisis óptimo de las imágenes se requiere al menos una dimensión de 512x512 pixeles.

La ventaja de utilizar FPGA en el trabajo, es que permite crear una arquitectura propia, la cual podrá ser utilizada en descripciones más complejas sin importar el fabricante de FPGA. Así mismo, debido a la diversidad de trabajo en el procesamiento de imágenes, dónde no será necesario utilizar una computadora convencional de propósito general, y al tener un hardware que tenga un propósito específico, nos ayuda en ahorro de costos y podremos obtener una eficiencia mayor por parte del sistema.

## 1.4. Antecedentes

Quintero (2006), muestra metodologías empleadas en un FPGA para realizar el procesamiento de imágenes y el reconocimiento de objetos dentro de dichas imágenes. En las distintas implementaciones se encontró un alto desempeño, medido en cuadros por segundo. Las implementaciones evitaron procesamientos matemáticos complejos, pero sin perder generalidad. El resultado final es un procesador de imágenes cuya principal limitante son los algoritmos iterativos de etiquetado e identificación de objetos.

Bravo (2007), plantea una propuesta para la detección de objetos en movimiento, dentro de una escena, mediante una plataforma de propósito específico FPGA, utilizando un sensor para la captación de imágenes y almacenarlas en el FPGA, las imágenes utilizadas se encuentran en blanco y negro.

Más adelante, Castillo (2008), propone unas prácticas de laboratorio para estudiantes de ingeniería, con FPGA, relacionadas con el área de procesamiento de imágenes, implementando los algoritmos de detección de bordes: Sobel y Canny.

Ramos (2010), propone una metodología para el manejo básico de imágenes, almacenando la imagen como una memoria ROM en el FPGA, sin utilizar herramientas de software del fabricante. Este trabajo permitió, entender cómo llevar a cabo la lectura de los valores RGB de cada pixel de una imagen, para su exhibición en un monitor, aunque,

debido a una limitación en el bus RGB, las imágenes obtenidas, presentan una pérdida en la calidad del color, sin embargo, la apreciación de la imagen obtenida es buena comparándola con la imagen reproducida en una PC.

Así mismo Ramos (2010), propone un sistema básico de procesamiento de imágenes empleando FPGA, en el cual, el almacenamiento de las imágenes, se realiza utilizando una memoria externa RAM, las operaciones básicas realizadas a las imágenes de prueba son: negativo de color, escala de grises y binarización por umbral (blanco y negro). Obteniendo imágenes similares a las obtenidas en una PC, teniendo como limitación el bus RGB, el cual es de 8 bit y solo permite almacenar en memoria una sola imagen.

González (2011), presenta un trabajo que consiste en la adquisición y desplegado de imágenes en escala de grises de 8 bit, la adquisición de la imagen se realiza mediante un decodificador de video compuesto, configurado para recibir una señal de video en estándar NTSC, y convertirla en información de video compuesta, la exhibición de los datos recibidos son almacenados en una memoria externa de tipo SDRAM y exhibidos mediante el puerto VGA.

Ramos (2012), presenta una arquitectura abierta para un controlador VGA, para diversas resoluciones de pantalla, utilizando FPGA, el cual, sirve para exhibir imágenes en monitores CRT o incluso en pantallas LCD.

En los trabajos anteriormente mencionados, se ha observado que se utilizan librerías no estándar, que son del fabricante del FPGA o herramientas del mismo, por ejemplo: Xilinx y las imágenes utilizadas son en escala de grises. En este trabajo se propone trabajar únicamente con los estándares IEEE desarrollando herramientas propias, por ejemplo; la transferencia de los datos de la imagen al FPGA o a una memoria de tipo SDRAM, para trabajar con FPGAs de diversos fabricantes como: Altera, Xilinx, Cypress. De esta forma no será necesario llevar a cabo adaptaciones por cada arquitectura, con excepción de las terminales, empleando imágenes a color y realizando operaciones de procesamiento de imágenes.

Diversas tecnologías son empleadas en sistemas embebidos, con diferentes características, lenguaje de programación, estándares internacionales, se han incorporado con el paso del tiempo (Cuadro 1.1).

Cuadro 1.1. Plataformas de Diseño Digital

| Plataforma de Diseño Digital | Característica principal | Lenguaje de Programación | Estándares Internacionales | Arquitectura | Año |
|---|---|---|---|---|---|
| Microprocesadores | Reconfigurable utilizando software. Bueno para computadoras. Multi-tarea | Ensamblador y C | No | Especificada por fabricante | 1971 |
| Microcontroladores, Procesadores de Señales Digitales (DSP) | Combinación de periféricos y CPU. Aplicación Específica | Ensamblador y C | No | Especificada por fabricante | 1976 |
| Arreglo de Compuertas Programables en Campo (FPGA) | Habilidad de combinar los puntos fuertes del procesador, controlador y ASSP. Interfaz con puertos de lenta y alta velocidad. Aplicación Específica. | VHDL y Verilog | IEEE | Especificada por usuario | 1988 |

Hoy en día, el equipo utilizado para realizar procesamiento de imágenes consta de dos componentes principales, los cuáles son: cámaras digitales; usadas para la adquisición de la imagen y, PCs; para el procesamiento y almacenamiento de la imagen y sus resultados. Así mismo, el procesamiento de la mayoría de los equipos es de manera secuencial, lo cual, afecta en cuestiones de tiempo, debido al alto número de operaciones necesarias para llevar a cabo el procesamiento.

En la actualidad, el procesamiento de imágenes embebido, es realizado obteniendo imágenes en escala de grises, esto se debe, a que, la mayoría de operaciones de procesamiento de imágenes parten de la escala de grises para su análisis. Así, mismo no se describe un algoritmo implementado en el hardware, ya que, existen módulos o librerías ya desarrolladas por el fabricante de tarjetas de desarrollo de FPGA, las cuáles implican un costo y no se desarrollan con las librerías estándar del IEEE para la portabilidad hacia otros fabricantes.

En este trabajo de investigación, se propone desarrollar los algoritmos de procesamiento embebido de imágenes necesarios, para la implementación de algoritmos de algunas operaciones de procesamiento de imágenes, sin necesidad de utilizar una PC, empleando como unidad de procesamiento un FPGA, el cual, lleva a cabo las operaciones

de manera paralela, es decir, todos los procesos se ejecutan al mismo tiempo, obteniendo como resultado un tiempo menor en la ejecución de las operaciones.

## 1.5. Organización de la tesis

La tesis esta organizada por 6 capítulos y uncapítulo de anexos. A continuación se presenta un pequeño resumen de cada capítulo.

Capítulo I. En este capítulo se presenta una introducción del proyecto de investigación, así como, los objetivos, justificación, alcances y limitaciones, y antecedentes relaciuonados con el tema.

Capítulo II. En este capítulo se plazman conceptos relacionados con el trabajo de tesis, como : sistemas embebidos, FPGA y procesamiento de imágenes entre otros.

Capítulo III. Presenta la metodología de trabajo de este proyecto, así como el desarrollo de los diagramas del sistema de acuerdo a los requerimientos. Por último se presentan simulaciones de la implementación del sistema.

Capítulo IV. Este capítulo consta de las pruebas hechas en la implementación del sistema en la tarjeta de desarrollo con un núcleo FPGA, así como, el análisis de algunos algoritmos de procesamiento de imágenes en software.

Capítulo V. Detalle de las conclusiones deltrabajo y propuesta de trabajo a futuro.

Capítulo VI. Aquí se muestra la bibliografía consultada para el desarrollo del proyecto de investigación.

Anexos. En esta sección se anexan dos publicaciones derivadas del proyecto de investigación, las cuáles son: dos publicaciones en congreso internacional, presentados en 2011 y 2012.

# II. MARCO TEÓRICO

## 2.1. Sistemas embebidos

Se conoce como ***sistema embebido*** a un circuito electrónico computarizado que está diseñado para cumplir una labor específica en un producto. La inteligencia artificial, secuencias y algoritmos de un sistema embebido, están residentes en la memoria de una pequeña computadora denominada microcontrolador o dispositivo lógico programable.

De acuerdo con Galeano (2009), a diferencia de los sistemas computacionales de oficina y *laptops*, estos sistemas solucionan un problema específico y están dispersos en todos los ambientes posibles de la vida cotidiana. Es común encontrar sistemas embebidos en vehículos, controlando el sistema de inyección de combustible, sistemas de frenado antibloqueo, control de espejos, sistemas de protección contra impacto (bolsas de aire), alarmas contra robo, sistema de encendido, sistemas de ubicación, etc. Así mismo en lavadoras, estufas, refrigeradores, hornos de microondas; equipos celulares, agendas de bolsillo, PDA, cajeros automáticos, cámaras fotográficas, etc. En consecuencia, un sistema embebido lo encontramos en muchos de los aparatos que utilizamos en nuestra vida cotidiana. En la actualidad un consumidor promedio interactúa con alrededor de 400 sistemas embebidos por día, lo cual tiende a crecer significativamente para los próximos años, considerando que el hardware es cada vez más pequeño, consumen menos energía y el costo es menor gracias a la economía de escala aplicada en la fabricación, aspectos que ayudan a remplazar en mayor proporción los sistemas lógicos, los equipos electromecánicos y en el futuro, se podrán incorporar en los equipos desechables.

Existen distintas tecnologías para la implementación de sistemas embebidos, como: microcontroladores, DSP (Digital Signal Processing), ASIC (Application Specific Integrated Circuit) y FPGA (Field Programmable Gate Array). En este proyecto de investigación la plataforma empleada será FPGA. En la actualidad hay varios fabricantes de esta tecnología; Altera, Xilinx, Cypress, Lattice Semiconductor, por mencionar algunos.

### 2.1.1. FPGA

Citando a Boemo (2005), los circuitos integrados son omnipresentes en gran cantidad de productos industriales. Una de sus alternativas, los circuitos tipo FPGA, presentan una característica única: están al alcance de países con un desarrollo tecnológico medio. El FPGA) aparece en el Mercado en 1984, con una idea central: permitir realizar un circuito integrado a medida, sin los riesgos económicos asociados a las otras opciones tecnológicas. Hoy en día el FPGA está presente en campos tan diversos como la automoción, la electrónica de consumo, o la investigación espacial. La tecnología FPGA tiene una aplicación horizontal en todas las industrias que requieren computación a alta velocidad. Tiene cabida en empresas que realizan las actividades indicadas a continuación: alarmas, arcos de seguridad de bancos, climatización de autobuses, comunicaciones por fibra óptica, conducción automática de trenes, control industrial, control de instalaciones eléctricas, electrónica de potencia, electrónica espacial, electrónica submarina, electrónica aplicada a hoteles, enclavadores eléctricos, ensayo de materiales, equipos de medicina y radiología, equipos de transmisión de TV, equipos ferroviarios, equipos para parking públicos, fabricación de azulejos y cerámicos, herramientas EDA, identificación dactilar, ingeniería nuclear, internet, tarjeta inteligente, telemandos y automatismos para puertas, maquinaria de empaquetamiento de frutas, maquinaria para panaderías, navegación GPS, regulación de tráfico, seguridad electrónica, simuladores de vuelo, sistemas de emergencia tipo 112, sistemas de información a viajeros, sistemas de radiofrecuencia y microondas, tarifación telefónica, telefonía móvil, telemedicina, visión nocturna.

Xilinx es una de las principales compañías que fabrican dispositivos lógicos programables como el CPLD (Complex Programmable Logic Device) y el FPGA. Sus principales y más conocidos chips son las familias SPARTAN 3, SPARTAN 6 y Virtex. Además de construir y vender el chip de FPGA, se desarrollan algunos kits de entrenamiento o desarrollo.

Altera Corporation es el pionero en soluciones de lógica programable, permitiendo a las compañías de semiconductores y sistemas innovar y ganar en sus mercados, de forma rápida y rentable. Ofrece FPGA, FPGA SoC, CPLD y ASIC en

combinación con herramientas de software, propiedad intelectual, procesadores embebidos y soporte al cliente, para ofrecer soluciones programables de alto valor a casi 13, 000 clientes en el mundo. Altera fue fundada en 1983 y tuvo ingresos de $1.9 billones de dólares, está situada en San José, California y emplea a 2, 600 personas en 19 países.

A continuación se mencionan las tarjetas de entrenamiento o desarrollo para sistemas digitales en FPGA, las cuáles, se tienen en la Facultad de Informática de la Universidad Autónoma de Querétaro, se mencionan de menor grado de complejidad y recursos.

**Cool Runner II**

La tarjeta de evaluación Cool Runner II, es una tarjeta de desarrollo completamente energizada por un puerto USB para la plataforma de desarrollo Cool Runner II CPLD de Xilinx. La tarjeta incluye suministros de energía altamente eficientes, un oscilador configurable por el usuario, varios dispositivos de entrada y salida para el usuario, y un puerto USB para energizar la tarjeta y programar el CPLD. La tarjeta incluye 5 conectores de expansión, para 64 señales disponibles del CPLD a circuitos externos para extender su capacidad. Las principales características de la tarjeta son:

*i.* Cool Runner II CPLD XC2C256 de 256 macroceldas en un encapsulado TQ-144.
*ii.* Puerto USB para energizar la tarjeta, programar el CPLD y transferencia de datos del usuario.
*iii.* Convertidor Analógico Digital de 16 bit y 3 canales, que mide en tiempo real $Vcc_{int}$ y dos bancos $Vcc_{IO}$ durante la operación actual.
*iv.* Un oscilador configurable por el usuario a 10 KHz, 100 KHz, 1MHz y un zócalo adicional para un segundo cristal oscilador.
*v.* 64 señales I/O disponibles en los conectores de expansión, 32 en los conectores Pmod y 32 en el conector paralelo.

Lo anterior hace que esta tarjeta de evaluación sea una buena opción para el aprendizaje inicial en este tipo de dispositivos, aunque no sea un FPGA y no cuente con interfaces con memorias RAM o puertos para video, cuenta con buenos recursos para sistemas digitales pequeños. (Figura 2.1.)

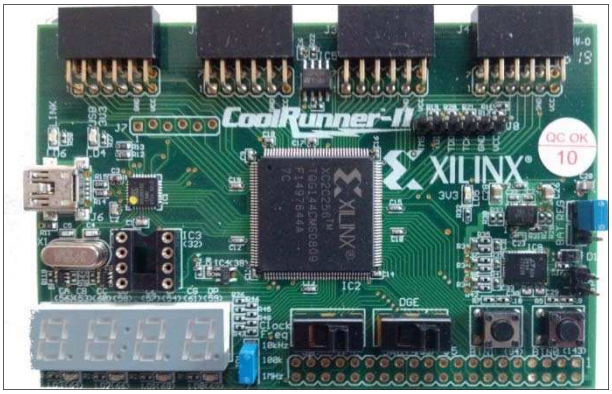



Figura 2.1. Cool Runner II

**Basys 2**

La tarjeta Basys 2 es un circuito diseñado e implementado en una plataforma, que quien desee ganar experiencia en la construcción real de circuitos digitales puede usar. Contiene un FPGA Xilinx Spartan 3E y un controlador USB ATMEL AT90USB2, proporciona hardware completo y listo para usar, adecuado para circuitos de dispositivos básicos hasta controladores complejos. Una gran colección de dispositivos I/O sobre la tarjeta y todo el soporte del FPGA requerido están incluidos, incontables diseños pueden ser creados sin necesidad de otros componentes.

Cuatro conectores estándar de expansión permiten al usuario crear diseños más allá de la tarjeta Basys 2, esto es posible, añadiendo tarjetas universales o Pmods (módulos de I/O análogos y digitales que ofrecen conversión D/A y A/D, controladores de motores, sensores de entrada, y muchas otras características). La Figura 2.2 muestra la tarjeta Basys 2. La tarjeta basys 2 contiene las siguientes características.

*i.* FPGA Xilinx Spartan 3E XC3S100E.
*ii.* Memoria Flash para almacenar las configuraciones del FPGA.

*iii.* 8 leds, 4 displays de 7 segmentos, 4 botones, 8 switches.

*iv.* Puerto PS/2, puerto VGA de 8 bit de color.

*v.* Frecuencia de reloj configurable a 25 MHz, 50 MHz, 100 MHz y un zócalo extra para añadir otro reloj.

*vi.* 4 conectores de expansión de 6 pin.


Figura 2.2. Tarjeta Basys 2

**Spartan 3**

La tarjeta Spartan 3 Starter Kit, ofrece una plataforma poderosa y autónoma de desarrollo de nuevos diseños dirigidos al FPGA Spartan 3 de Xilinx (Figura 2.3).

La tarjeta cuenta con las siguientes características:

*i.* FPGA Xilinx Spartan 3 XC3S200 o XC3S1000

*ii.* 12 multiplicadores de 18 bit, 216 Kb de memoria interna, y velocidades internas de hasta 500 MHz.

*iii.* 2 Mb en memoria Flash incorporada en la tarjeta

*iv.* 8 switches, 4 pushbuttons, 9 leds y 4 display de 7 segmentos

*v.* Puerto serial RS232, puerto PS/2 para mouse o teclado y puerto VGA de 3 bit de color

*vi.* 3 conectores de expansión de 40 pin cada uno

*vii.* 1 MB de almacenamiento en memoria SRAM (256 Kb x 32)

Esta tarjeta es muy buena para comenzar a trabajar con tecnología FPGA, su principal inconveniente es que no posee un puerto USB para la alimentación de la tarjeta y programación del FPGA, ya que, el kit contiene un alimentador de voltaje y un cable para la programación del FPGA por puerto LPT1, sin embargo es posible adquirir el cable USB para la programación.

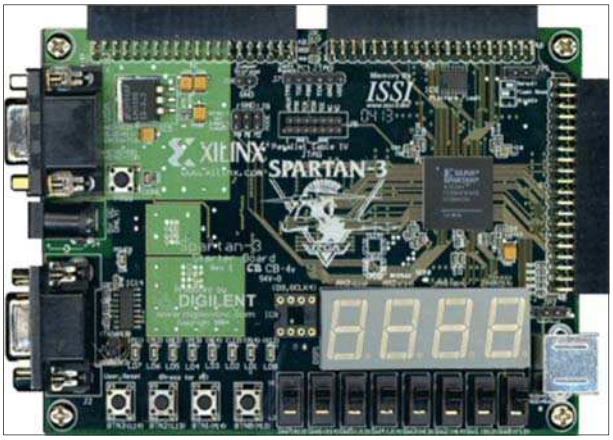



Figura 2.3. Spartan 3 Starter Kit

**Nexys 2**

La tarjeta Nexys 2 es una plataforma poderosa para el diseño de sistemas digitales con un FPGA Spartan 3E integrado. La tarjeta se ilustra en la figura 2.4 y sus características son las siguientes:

*i.* FPGA Xilinx Spartan 3E XC3S500E o XC3S1200E
*ii.* Alimentación, configuración y transferencia de datos de alta velocidad por puerto USB 2.
*iii.* 16 MB de almacenamiento externo PSDRAM
*iv.* 16 MB de almacenamiento flash
*v.* Plataforma Xilinx Flash ROM
*vi.* Switch de encendido/apagado, buena para aplicaciones portátiles
*vii.* Frecuencia de reloj de 50 MHz, zócalo para agregar un reloj de mayor o menor frecuencia.

*viii.* Conectores de expansión

*ix.* Puerto serial RS232, puerto VGA de 8 bit de color

*x.* 8 leds, 4 display de 7 segmentos, 4 pushbuttons, 8 switches

Sus características permiten al usuario tener una mayor cantidad de recursos para el desarrollo de sistemas digitales básicos hasta un microcontrolador (Figura 2.4).


Figura 2.4. Nexys 2

**Spartan 3AN**

La tarjeta Spartan 3AN Starter-Kit está lista para su uso, basta con sacarla de la caja y conectarla a la energía. Los diseños de varios dispositivos I/O almacenados en la memoria externa de tipo Flash, como por ejemplo el puerto VGA y el puerto serial RS232. Así mismo tiene la característica innovadora de ahorro de energía en modo suspensión. En la figura 2.5 se muestra la tarjeta Spartan 3AN, sus principales características son:

*i.* FPGA Spartan 3AN XC3S200AN

*ii.* LCD de caracteres

*iii.* Switch de encendido/apagado

*iv.* Salida para audio audífonos/bocinas

*v.* Puerto PS/2, RS232, VGA de 12 bit de color

*vi.* Programación por puerto USB, alimentación adicional con alimentador de energía.

*vii.* Almacenamiento externo en memoria DDR2-SDRAM de 512 MB (32M x 16)

Por sus características se convierte en una tarjeta ideal para sistemas que requieren almacenar gran cantidad de información como sería el área de procesamiento de imágenes, aunque la resolución en color no es la real de 24 bit, es una buena opción para comenzar a entrar en el campo del estudio de imágenes en sistemas embebidos (Figura 2.5).


Figura 2.5. Spartan 3AN

**DE2-115**

La tarjeta DE2-115 tiene muchas características que permiten a los usuarios implementar un amplio rango de sistemas digitales, desde un simple circuito hasta varios proyectos multimedia.

El hardware proporcionado en la tarjeta DE2-115es el siguiente:

*i.* FPGA Altera Cyclone IV

*ii.* Dispositivo serial EPCS64 Altera

*iii.* USB Blaster (integrado en la tarjeta) para programación; JTAG y AS, ambos modos son soportados.

*iv.* 2 MB SRAM

*v.* 64 MB SDRAM x 2

*vi.* 8 MB Flash

*vii.* SD Card socket

*viii.* 4 push buttons, 18 switches, 9 leds verdes, 18 leds rojos, 8 displays de 7 segmentos, display de caracteres LCD.

*ix.* 4 Osciladores de 50 MHz

*x.* Entradas y salidas de audio y microfono de 24 bit, calidad CD

*xi.* Puerto VGA de 24 bit de color, puerto serial

*xii.* Decodificador de TV (NTSC/PAL/SECAM) y conector de entrada de TV

*xiii.* 2 conectores RJ45 Gigabit Ethernet PHY

*xiv.* Controlador USB maestro/esclavo con conectores USB tipo A y B

*xv.* Transmisor RS232 y conector de 9 pin

*xvi.* Conector PS/2 para mouse y teclado

*xvii.* Receptor IR

*xviii.* 2 conectores SMA I/O para señal de reloj externa

*xix.* Una cabecera de expansión de 40 pin con protección de diodo

*xx.* Un conector HSMC (High Speed Mezzanine Card)

Las características anteriores hacen de esta tarjeta la más completa con mayor número de recursos y hardware integrado, así como la posibilidad de agregar tarjetas con hardware adicional como lo es un cámara digital, una pantalla táctil LCD de 4.3 in. (Figura 2.6.).

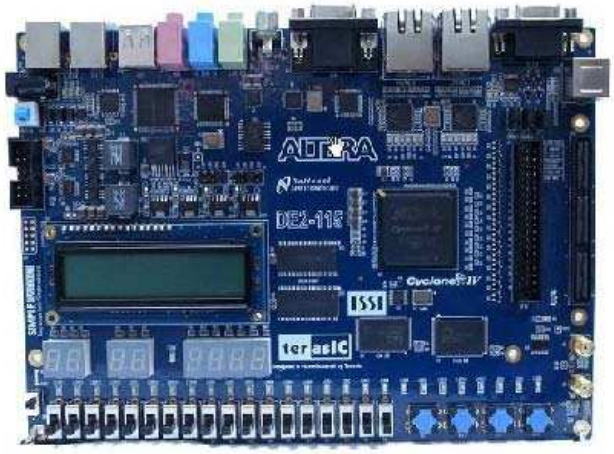



Figura 2.6. Tarjeta DE2-115 de Altera

Esta tarjeta cuenta con la opción de conectar una pantalla LCD táctil (*LTM, LCD Touch panel Module*), y así, sustituir el puerto VGA, y descartar la necesidad de un monitor. Las características de la pantalla se enlistan a continuación:

- Interfaz que soporta 24 bit de color de manera paralela.
- 3 señales de control para exhibir y selección de funcionamiento.
- Modulación de contraste y brillo.
- Convierte las coordenadas X/Y de la membrana táctil a su dato digital correspondiente a través de un convertidor analógico digital.

Cuadro 2.1. Características del módulo LTM

| Item | Descripción | Unidad |
|---|---|---|
| Tamaño de pantalla | 4.3 | Pulgadas (in) |
| Aspecto | 15:9 | - |
| Tipo de pantalla | Transmisivo | - |
| Área activa (HxV) | 93.6 x 56.16 | Milímetros, mm |
| Número de pixeles | (800 x RGB x 480) | Pixeles |
| Distancia entre pixeles (HxV) | 0.039 x 0.117 | Milímetros |
| Número de colores | 16 millones | - |

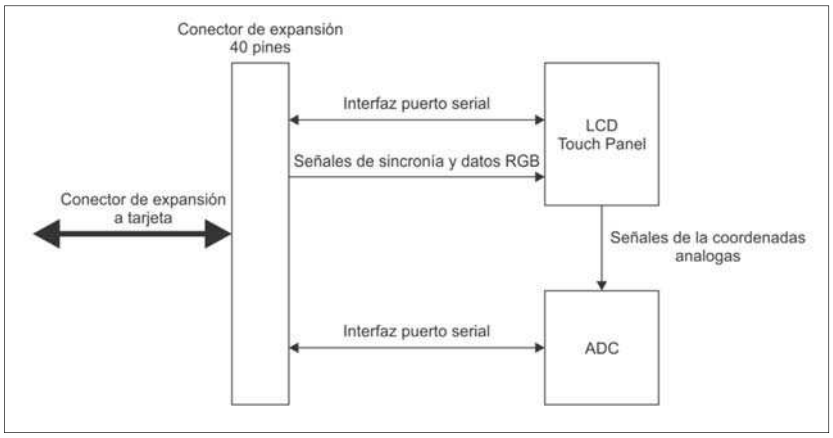



Figura 2.7. Diagrama de Bloques de la pantalla LTM

En este proyecto de investigación cualquiera de las tarjetas mencionadas pueden ser empleadas para la implementación, con una de las limitantes que sería el número de recursos a emplear del FPGA (Cuadro 2.1).

Cuadro 2.2. Tarjetas con tecnología FPGA y características principales

| TARJETA | FABRICANTE FPGA | RECURSOS FPGA | FABRICANTE TARJETA | VGA | RELOJ | ALMACENAMIENTO EXTERNO |
|---|---|---|---|---|---|---|
| Cool Runner II | Xilinx | 256 macroceldas | Digilent | No | 10 KHz<br>100 KHz<br>1 MHz | No |
| Basys 2 | Xilinx | 4908 | Digilent | 8 bit | 25 MHz<br>50 MHz<br>100 MHZ | No |
| Spartan 3 | Xilinx | 9773 | Digilent | 3 bit | 50 MHz | 1 MB SRAM |
| Nexys 2 | Xilinx | 43550 | Digilent | 8 bit | 50 MHz | 16 MB SDRAM |
| Spartan 3AN | Xilinx | 9155 | Digilent | 12 bit | 50 MHz<br>133 MHz | 512 Mb DDR2<br>SDRAM |
| DE2-115 | Altera | 114480 | Terasic | 24 bit | 50 MHz | 2 MB SRAM<br>64 MB SDRAM<br>8 MB Flash memory<br>SD Card Socket hasta 32 GB |

## 2.2. Memorias de tipo RAM

Bailey (2011) publica acerca de las memorias de acceso aleatorio (RAM), son usadas para almacenamiento masivo en un sistema digital, el cual es más simple que almacenamiento en módulos Flip-Flop.

En muchas aplicaciones, es necesario usar memoria externa al FPGA como buffers o bloques de memoria grandes. También, la memoria externa es requerida cuando se ejecuta un procesador embebido con un sistema operativo como Linux. Hay dos tipos de memoria, estática y dinámica, las cuáles son descritas a continuación.

### 2.2.1. RAM Estática

La memoria estática almacena cada bit mediante un latch, por lo cual cada bit de memoria requiere un mínimo de cuatro transistores, con un latch adicional de dos transistores utilizados para acceder a la lectura y escritura. Esto es considerablemente más grande que un transistor utilizado por la memoria dinámica, por lo que requiere mayor costo para fabricar. Su ventaja es la velocidad, haciéndola más utilizada comúnmente para tamaños de memoria intermedios, comúnmente en el rango de los MB, tal como las utilizadas en sistemas de computadoras, como la memoria caché. Casi todas las memorias síncronas de alta velocidad son memorias en pipeline. Por lo tanto, en la operación de lectura, el dato está disponible varios ciclos de reloj después del acceso a la dirección de memoria, aunque la memoria tiene la capacidad de acceder a la memoria una vez por ciclo de reloj. Esto hace un poco más complicado el uso de la memoria externa que usar el bloque RAM interno.

Usualmente, en el ciclo de escritura, la dirección y el dato deben ser accesados al mismo tiempo. En consecuencia, cuando la lectura va seguida de una escritura, hay un tiempo muerto para permitir a la lectura completar el dato antes de que el bus de datos este disponible para proporcionar el dato a escribir. Si se utilizan esas recomendaciones en un diseño, el tiempo muerto debe ser considerado dentro del tiempo total. Mientras que tales recomendaciones pueden ser ajustables para grandes bloques de lectura y escritura, tales

como podrían surgir cuando se transfiere una imagen, a o desde una memoria intermedia, que son inadecuados para diseños de dos fases, donde la lectura y la escritura pueden ser alternadas.

Las memorias con respuesta en cero (ZBT, Zero Bus Turnaround), no tienen tiempos muertos entre la lectura y la escritura. Esto se consigue proporcionando los datos a escribir después de la dirección (Figura 2.8). La capacidad de cambiar entre la lectura y la escritura de un ciclo al siguiente, hace la memoria ZBT más fácil de emplear en la interfaz de una aplicación. Los datos a escribir pueden ser retenidos dentro del FPGA si es necesario mediante el la técnica de pipeline.

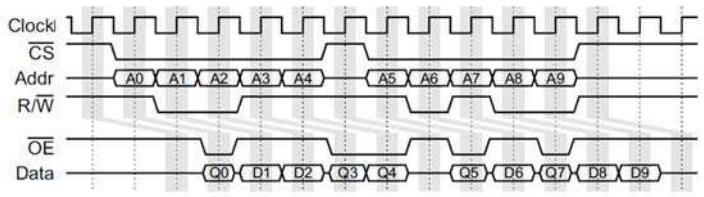



Figura 2.8. Tiempo típico para lectura/escritura para una memoria estática con un periodo de dos ciclos de reloj

### 2.2.2. RAM Dinámica

La memoria dinámica, es la más adecuada para volúmenes grandes de memoria, por su bajo costo. Utiliza un solo transistor por célula de memoria. Debido a que los capacitores tienen fuga de carga, se debe actualizar al menos una vez cada 64 ms para prevenir pérdida de datos. Para lograr esto con memorias de gran capacidad, las memorias dinámicas están estructuradas de manera que un renglón completo (dentro de la matriz de memoria de dos dimensiones) se actualice en el mismo instante de tiempo. La disponibilidad de todo un renglón en un tiempo lleva a una estructura de memoria paginada. La interfaz para un chip es controlado a través de una secuencia de comandos, los cuáles, son proporcionados a través de un conjunto de pines de control. Una secuencia típica de comandos es la siguiente:

1. Un comando de activación selecciona el renglón de memoria a usar. Las líneas de dirección especifican que renglón es seleccionado. La activación de un renglón lee el renglón dentro de la columna de los amplificadores de detección, permitiendo ingresar a los datos dentro de ese renglón. La activación del renglón toma varios ciclos de reloj.
2. Una vez activado el renglón, se puede usar los comandos de lectura y escritura. Las líneas de dirección especifican la columna que será leída o escrita. Una lectura o escritura tiene un periodo de varios ciclos de reloj, sin embargo en pipeline, se permite un acceso por ciclo de reloj al dato en el renglón. Cuando se escribe el dato es proporcionado con la dirección, así que hay un retardo cuando se cambia de lectura a escritura. Operando el modo burst permite varias lecturas o escrituras a sucesivas locaciones de memoria con un solo comando. Esto es explotado con las memodias DDR dando dos lecturas o escrituras por ciclo de reloj, una en cada borde de reloj.
3. Antes de seleccionar otro renglón, es necesario cerrar el actual con un comando de precarga. Esto devuelve a los amplificadores de detección a un estado a un estado de reposo para detectar el siguiente renglón.
4. Un comando de actualización, actualiza un renglón de memoria. Un contador interno se mantiene en lo que los renglones son actualizados en secuencia. El comando interno de actualización activa el renglón, espera el tiempo suficiente para que la carga de los capacitores sea actualizada y luego regresa a la línea de estado de reposo. Mientras, esto es posible realizarlo manualmente, teniendo un comando de actualización que simplifica el controlador de memoria.

Hay periodos de tiempo considerables, especialmente cuando se mueve de un renglón a otro. Para ayudar a arreglar esto, muchos chips DRAM tienen múltiples bancos internos de memoria, así que, un banco puede ser usado mientras espera la activación de un renglón o precarga dentro de otro banco.

Por lo tanto, leer o escribir en una memoria dinámica, no es un asunto sencillo de proporcionar datos y direcciones. La secuencia compleja de operaciones para accesar a una memoria particular y asegurarse de que la memoria se mantiene actualizada requiere un

controlador de memoria. Algunos FPGA nuevos incorporan hardware dedicado como controladores de memoria para simplificar el acceso (Xilinx, 2010c). De lo contrario, es necesario construir el controlador de memoria desde el fabricante del FPGA.

Si se utiliza memoria dinámica para un buffer, funciona mejor en modo streaming, porque realizará muchos accesos secuenciales al mismo renglón. El acceso streaming por columnas (por ejemplo con la FFT), generalmente no es práctico, por las latencias grandes al cambiar de un renglón a otro en la memoria. Esto puede ser superadocreando alguna extensión mapeando entre la memoria lógica y la memoria física, al incrementar el número de secuencias en la misma página. Sin embargo, cada técnica complica el diseño del sistema. Un problema similar, se encuentra con un proceso de acceso aleatorio. En cada sistema, puede ser necesario mantener una memoria estática como bufer extra a la memoria dinámica. Si el patrón de acceso es conocido de antemano, se puede construir una memoria cache pequeña en el FPGA.

DIMMs (Dual in Line Memory Modules) consisten en un número de chips DRAM, montados en un circuito impreso, para dar un ancho más amplio de datos. Esto, comúnmente es muy usado en sistemas computacionales convencionales, pero puede ser usado con FPGA, publicado por Bailey (2011).

## 2.3. Procesamiento de imágenes

La visión, es el sentido más importante y discutible hoy en día, debido al procesamiento y registro de los datos visuales. Las primeras imágenes son dibujos prehistóricos en las paredes de las cuevas o tallados en monumentos de piedra comúnmente asociados a las tumbas de entierro. Tales imágenes consisten de una mezcla de representaciones ilustradas y abstractas. Los avances tecnológicos permiten registrar con más realismo las imágenes, tal como las pintadas por los maestros. Las imágenes registradas de esta manera son indirectas en el sentido que el patrón de la intensidad de luz no es utilizada directamente para producir la imagen. El desarrollo de la fotografía química al principio de 1800 permitió grabar imágenes directamente. Esta tendencia a continuado con la grabación electrónica, primero con sensores analógicos, y posteriormente con

sensores digitales que incluyen el analógico a digital (A/D) en el chip sensor, publicado por Bailey (2011).

Los sensores de imagen no se han limitado a la porción del espectro electromagnético visible para el ojo humano. Han sido desarrollados para cubrir la mayor parte del espectro electromagnético de las ondas de radio a través de los rayos X y los rayos gamma. Otras modalidades de imagen se han desarrollado, incluyendo ultra sonido y la resonancia magnética. En principio, cualquier cantidad que se puede detectar se puede utilizar para imagen incluso los rayos de polvo (Auer, 1982).

### 2.3.1. Procesamiento de imágenes embebido

Un sistema embebido es un sistema de cómputo que es embebido dentro de un producto o un componente. Consecuentemente, son diseñados para para realizar una tarea específica, o un rango pequeño de tareas específicas, a menudo con limitaciones de tiempo real. Un claro ejemplo de sistema embebido para procesamiento de imágenes es una cámara digital. Hay funciones que controlan la exposición, el enfoque, exhibir vista previa, y administrar la compresión y descompresión de imagen.

La visión embebida, también es muy usada en cámaras inteligentes, donde la cámara no solo captura la imagen, también procesa la extracción de la información como lo requiera la aplicación. Ejemplos donde se utilizan son sistemas de vigilancia, inspección industrial o control, visión robótica, y así sucesivamente.

Un requisito de muchos sistemas embebidos, es que necesitan ser de tamaño pequeño y ligeros en peso. Muchos funcionan con baterías, y por lo tanto, son requeridos para operar con baja energía. Incluso, aquellos que no son operados con batería, usualmente tienen energía disponible limitada, publicado por Bailey (2011).

### 2.3.2. Sistemas de Procesamiento de Imágenes en Hardware

En su libro Bailey (2011), menciona que los primeros sistemas computacionales eran demasiado lentos para procesar imágenes de cualquier tamaño o en cualquier volumen.

La estructura regular, llevó a las imágenes a diseños para sistemas en hardware a aprovechar el paralelismo, ya que, los sistemas en hardware son inherentemente paralelos. Muchos de los primeros sistemas eran basados en tener relativamente un elemento de procesamiento simple (PE, processing element) en cada pixel. Unger en 1958, propone un sistema basado en 4 cuadrículas cuadradas conectadas para procesar imágenes binarias. Aunque el hardware para cada PE era relativamente simple, demostró con operaciones de procesamiento de imágenes de bajo nivel se podían mejorar. Aunque, este primer sistema no fue implementado (Duff, 2000) inspiró otras arquitecturas similares. Golay, propuso un sistema basado en una cuadrícula hexagonal (Landsman, 1965; Golay 1969) con la imagen ciclada a través de un solo PE usando procesamiento de flujo. El CLIP2 (Duff, 1973) usa un arreglo hexagonal de 16x2, aunque, fue capaz de mejorar algún procesamiento básico, no era práctico para aplicaciones reales. El sucesor CLIP4 (Duff, 2000) usó un arreglo de 96x96, y fue capaz de operar con imágenes en escala de grises en un bit de manera serial. Un arreglo similar de 128x128 (el procesador masivamente paralelo), fue hecho por Batcher en 1980; también operaba un bit de manera serial.

Para capturar en una pantalla un video o imágenes, incluso las computadoras seriales requieren un sistema de hardware para la interfaz con la cámara o el display. Estos capturadores de fotogramas interconectados con el sistema de cómputo central a través del acceso directo de memoria (DMA, Direct Memory Acces), o en la memoria compartida que se asigna dentro del espacio de direcciones del host. Un capturador básico consistía de un convertidor A/D, un banco de memoria capaz de mantener al menos una imagen, un convertidor D/Apara exhibir la imagen en display. Muchas operaciones básicas permitidas para aplicar a través de tablas de consulta permitieron capturar o exhibir la imagen. Este pipeline simple fue ampliado a operaciones más sofisticadas y sistemas completamente pipeline de procesamiento de imágenes, el Datacube fue el más notable. Pipeline es menos adecuado para procesamiento de imágenes de alto nivel; se desarrollaron sistemas híbridos consistiendo de hardware en pipeline para el procesamiento y un arreglo de procesadores seriales, para el procesamiento de lato nivel, tal como en el sistema de visión del kiwi (Clist and Valkenburg, 1994).

Los sistemas de hardware iniciales eran implementados con circuitos de baja y media escala de integración. La llegada de los circuitos integrados de gran escala (VLSI) redujo el costo de hardware y dramáticamente incrementó la velocidad y el rendimiento en los sistemas resultantes. Esto llevó a que un amplio rango de arquitecturas de hardware se utilicen en operaciones de procesamiento de imágenes y sean implementados (Offen, 1985). Es interesante notar, como la tecnología a avanzado, al reducir características de tamaño ha impactado en el costo de producción y se ha aumentado significativamente la proporción de costo de una sola vez con el costo de los dispositivos individuales. Consecuentemente, la producción económica de VLSI significa que solo un rango limitado de circuitos sea para uso comercial.

Los sistemas de procesamiento de imágenes basados en hardware, son muy rápidos, pero su gran problema es su relativa inflexibilidad. Una vez configurado realizan su tarea muy bien, pero es complicado, si no imposible, reconfigurar estos sistemas, si la naturaleza de la tarea cambia. En los 1980s, la introducción FPGA, tecnología que abre nuevas posibilidades de hacer diseños con lógica digital. Combina la naturaleza inherente paralela de hardware con la flexibilidad de software en que su funcionalidad puede ser reprogramada o reconfigurada. Los FPGA iniciales eran pequeños en términos de equivalencia en número de compuertas, por lo que se tiende a ser empleados principalmente para proporcionar interfaz de interconexión flexible y lógica (a veces llamado glue logic), y para implementar controladores basados en máquinas de estados finitos. Múltiples FPGA eran requeridos para implementar un cualquier sistema significante de procesamiento de imágenes. Como FPGA creció en poder, se han desarrollado sistemas híbridos donde FPGA es usado como coprocesadores de visión o aceleradores dentro de la computadora principal; ejemplo el sistema Splash-2 (Athanas and Abbot, 1995). Ahora los FPGA modernos tienen suficientes recursos para un amplio número de aplicaciones a implementar en un solo FPGA, haciéndolos una opción ideal para sistemas de visión embebidos en tiempo real.

## 2.4. Representación de Números en Punto Fijo

Citando a Ramos (2010), es importante resaltar que en un sistema binario no existe ningún símbolo que represente el punto decimal, por lo que la separación entre la parte entera y fraccionaria es implícita, y además es fija. Por lo tanto, al representar un número en punto fijo se tienen los siguientes elementos:

- Todos los números representados son con signo.
- La parte entera tiene al menos un bit significativo.
- El punto decimal es implícito y fijo.
- La parte fraccionaria comprende al resto de los bits.

En la Figura 2.9, se observa como representar un número en punto fijo.

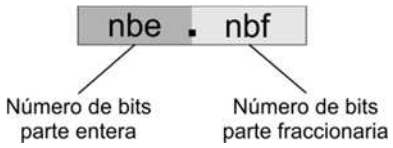



Figura 2.9. Representación de un dato en formato de punto fijo en binario

En un número binario en formato 1.15, los bits se distribuirían de la siguiente manera: nbe = 1 y nbf = 15; es decir la parte entera está formada por 1 bit y la parte fraccionaria por 15 bit; el punto decimal está fijo entre los bit 15 y 14, como se observa en la (Figura 2.10).

$$D_{15}.D_{14}D_{13}D_{12}D_{11}D_{10}D_9D_8D_7D_6D_5D_4D_3D_2D_1D_0$$

Figura 2.10. Representación del valor D en formato binario

La importancia de este tema en el presente trabajo de investigación, es que algunas operaciones requieren operaciones de división; la cual los sistemas digitales no realizan por su naturaleza ya que esta operación se realiza por medio de restas y multiplicaciones, por lo que lo hace un algoritmo complejo y habría que implementarlo. En procesamiento de imágenes cuando se requiere una operación de división se conoce el

número denominador y este puede ser adaptado a un número fraccionario representado por formato de punto fijo.

## 2.5. Metodología para el procesamiento digital de imágenes y diseño digital de sistemas embebidos

El nacimiento de lo que hoy en día llamamos procesamiento digital de imágenes puede ser remontado a la disponibilidad de estas máquinas y al inicio del programa espacial durante ese período. Se combinó esos dos desarrollos para traer el inicio potencial del concepto de Procesamiento de imágenes Digitales desarrollado por Gonzalez y Woods (2008), (Figura 2.11).

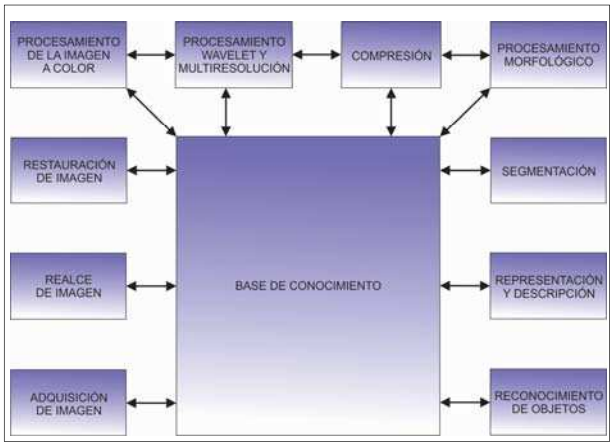



Figura 2.11. Pasos fundamentales para el procesamiento digital de imágenes

*i.* *Adquisición de Imagen*. La adquisición puede ser tan simple como lo es una imagen en forma digital ya dada. Generalmente, esta etapa implica el realizar un reprocesamiento, así como un escalamiento.

*ii.* *Realce de Imagen*. Es una de las áreas más simples y atractivas del procesamiento digital de imágenes. Un ejemplo de realce es cuando se incrementa el contraste de una imagen porque "se ve mejor". Es un área de procesamiento de imágenes muy subjetiva. Se basa en preferencias subjetivas humanas con respecto qué constituye un "buen" resultado de realce.

*iii.* *Restauración de Imagen.* Es un área que también se dedica a mejorar el aspecto de la imagen, a diferencia que está área es objetiva, en el sentido las técnicas de

restauración tienden a ser basadas en modelos matemáticos o de probabilidad de la degradación de la imagen.

iv. *Procesamiento de la Imagen a Color.* Es un área que ha ido ganando importancia debido al aumento significativo en el uso de imágenes digitales sobre el internet. El color es utilizado como base para extraer características del interés en una imagen.

v. *Procesamiento Wavelet y Multiresolución*. Es base para representar imágenes en varios grados de resolución. Particularmente este material se utiliza para la compresión de datos de imagen y representación piramidal, en la cual las imágenes se subdividen sucesivamente en regiones pequeñas.

vi. *Compresión*. Se ocupa de las técnicas para reducir almacenaje requerido para guardar una imagen, o el ancho de banda requerido para transmitirla.

vii. *Procesamiento Morfológico*. Trata de las herramientas para extraer los componentes de la imagen que son útiles en la representación y la descripción de la forma.

viii. *Segmentación*. Los procedimientos de la segmentación reparten una imagen en sus componentes u objetos. La segmentación autónoma es generalmente una de las tareas más difíciles del procesamiento digital de imágenes.

ix. *Representación y Descripción*. Casi siempre siguen la salida de la etapa de segmentación, que es generalmente datos crudos del pixel, el constituir cualquier límite de una región o de todos los puntos en la región de sí mismo. En cualquier caso, es conveniente convertir los datos a una forma que para el tratamiento es necesario para la computadora.

x. *Reconocimiento de Objetos*. Es el proceso que asigna una etiqueta a un objeto basado en sus descriptores.

xi. *Base de Conocimiento*. El conocimiento sobre un dominio del problema se cifra en un sistema de tratamiento de la imagen bajo la forma de base de datos. Este puede estar como regiones de detalle de una imagen don de la información del interés se sabe para ser localizado, así limitando la búsqueda que tiene que ser conducida en buscar esa información.

La metodología anterior, forma una base para el procesamiento de imágenes realizado en software, la cual, ayuda para sentar bases, para cuando se requiera realizar en hardware. Ya que en la actualidad no hay una metodología base para llevarlo a cabo, sin

embargo, este trabajo toma como base una metodología para manejo de imágenes empleando memorias externas de tipo SDRAM, presentado por Ramos (2010). Esa metodología, será empleada como parte inicial del proyecto, la cual, consiste en la implementación de filtros para la extracción de bordes.

Dubey (2009), presenta metodología general empleada en el diseño de sistemas embebidos (Figura 2.12).

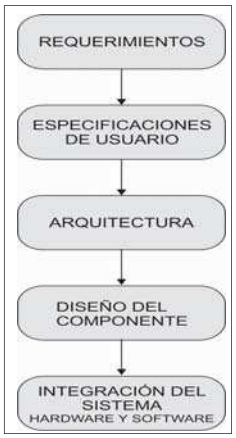



Figura 2.12. Metodología general empleada para diseño de sistemas embebidos

*i.* *Requerimientos*: Se refiere a todo requerimiento funcional y no funcional, como lo pueden ser: tamaño, peso, consumo de energía y costo.

*ii.* *Especificaciones de Usuario*: Detalles de la interfaz de usuario junto con operaciones necesarias para satisfacer la petición de usuario.

*iii.* *Arquitectura*: Hardware (procesador, periféricos, lógica programable y ASSPs) y Software (programas de mando y sus operaciones).

*iv.* *Diseño del Componente*: Componentes prediseñados, componentes modificados y nuevos componentes.

*v.* *Integración del Sistema*: Esquema de verificación para encontrar errores rápidamente.

# III. DESARROLLO DEL SISTEMA E IMPLEMENTACIÓN EN FPGA

## 3.1. Metodología propuesta

Tomando como base las dos metodologías anteriores se propone una metodología para procesamiento digital de imágenes en hardware empleando tecnología FPGA (Figura 3.1).

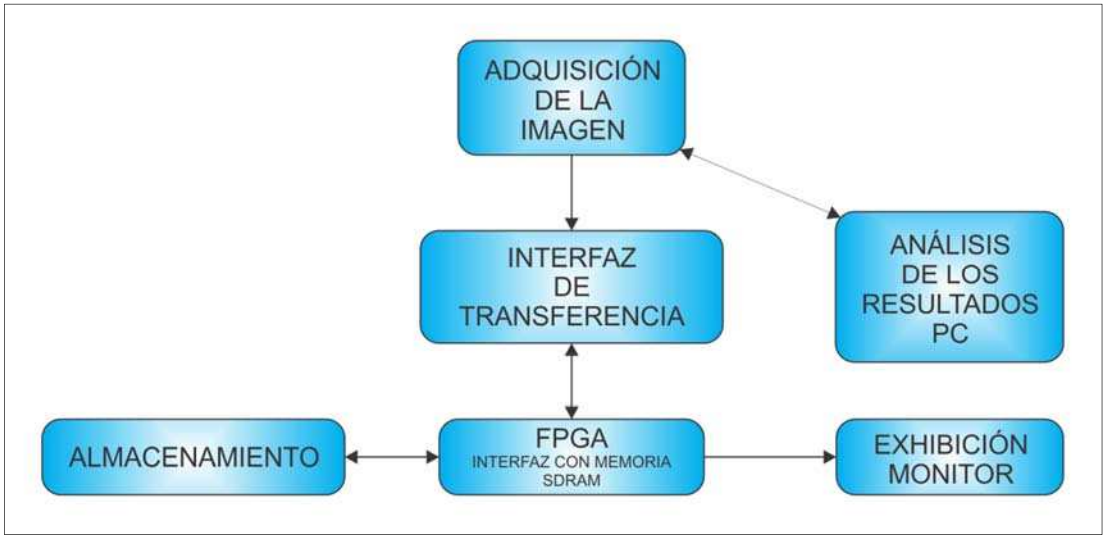



Figura 3.1. Metodología propuesta para procesamiento digital de imágenes en hardware

Las etapas de esta metodología se describen a continuación.

### 3.1.1. Etapa de Adquisición de la Imagen.

Esta etapa comprende en obtener la imagen requerida para procesar. La adquisición puede ser desde una cámara digital o bien, una imagen almacenada en un PC.

### 3.1.2. Etapa de Análisis de resultados en la PC.

En esta etapa es donde el formato de punto fijo es empleado para la realización de las operaciones. Una imagen digital se puede representar por valores enteros de 0 a 255, o

bien, valores entre 0 y 1. En este trabajo de investigación se ocuparán los valores entre 0 y 1, para formatos de punto flotante y punto fijo, utilizando la herramienta de Matlab y en particular para esta etapa.

*Negativo de Color*: este proceso es muy utilizado en la industria de efectos especiales, (Figura 3.2) (ecuación 1).

$$\boldsymbol{img(x,y,z) = 1 - img_{color}(x,y,z)} \quad (1)$$

a) b)

Figura 3.2. a) Imagen a color, b) Negativo de color de la Imagen

*Escala de Grises: e*n una imagen en escala de grises el pixel posee un valor equivalente a una intensidad de gris, que va desde el negro más profundo variando gradualmente la intensidad de gris hasta llegar al blanco. Éstas imágenes se representan con 8 bit para representar cada pixel lo que nos permite únicamente obtener 256 tonalidades diferentes (0 a 1). Cuando se convierte una imagen a color a escala de grises se debe obtener el promedio de las 3 bandas de color de cada pixel (Figura 3.3) (ecuación 2).

$$\boldsymbol{img_{gray} = (img_{color}(x,y,R) + img_{color}(x,y,G) + img_{color}(x,y,B))/3} \quad (2)$$

a) b)

Figura 3.3. a) Imagen a color, b) Imagen a escala de grises

*Blanco y* Negro: si la imagen es en *Blanco y Negro*, se almacena un valor por cada píxel, este valor es el nivel de intensidad o nivel de gris comentado anteriormente. Se suele utilizar un rango de valores para su representación, que generalmente es de 0 a $2^{n-1}$.

Para obtener una imagen en blanco y negro, debemos obtener primero la imagen en escala de grises, y establecemos un valor umbral (ecuación 3), con el cual vamos a definir en qué momento tendremos un valor negro o un valor blanco (Figura 3.4).

$$\boldsymbol{img_{bw}(x,y,z)} = \begin{cases} \boldsymbol{0\ si\ img_{gray}(x,y) \leq\ \lambda} \\ \boldsymbol{1\ si\ img_{gray}(x,y) \geq \lambda} \end{cases} \tag{3}$$

Donde $\boldsymbol{\lambda}$ representa el valor del umbral deseado.

a) b)

Figura 3.4. a) Imagen a color, b) Imagen Blanco y Negro

*Espacios de Color*: únicamente se refiere a la separación de las tres bandas de color R, G y B, (ecuación 4 a ecuación 6).

$$\boldsymbol{img_r(x,y) = img_{color}(x,y,R)} \tag{4}$$

$$\boldsymbol{img_g(x,y) = img_{color}(x,y,G)} \tag{5}$$

$$\boldsymbol{img_b(x,y) = img_{color}(x,y,B)} \tag{6}$$

*Detección de Bordes:* los *puntos de borde*, o simplemente *bordes* son píxeles alrededor de los cuales la imagen presenta una brusca variación en los niveles de gris. El

objetivo consiste en dada una imagen, que puede o no estar corrompida por ruido, localizar los bordes más probables generados por elementos de la escena y no por ruido.

En realidad el borde se refiere a cadenas conectadas de punto de borde, esto es fragmentos de contorno, esto no impide que la imagen también pueda contener puntos aislados que presentan un alto contraste en los niveles de gris. Los puntos de borde a veces son denominados "*edgels*" (procedente de *edge elements*). Existen varias razones que sostienen el interés por los bordes. Los contornos de los objetos sólidos de la escena, las marcas en las superficies, las sombras, todas generan bordes. Además las líneas de las imágenes, las curvas y contornos son características o elementos básicos para muchas aplicaciones tales como calibración, movimiento o reconocimiento.

La detección de bordes es una parte fundamental de la mayoría de sistemas de visión, puesto que el éxito de los niveles siguientes de procesamiento depende fuertemente de la fiabilidad de las características, en este caso bordes. La filosofía básica de muchos algoritmos de detección de bordes es el cómputo de operadores derivada locales (primera o segunda). Esto se hace por que el valor de la magnitud del gradiente no es tan importante como la relación entre diferentes valores. Es decir, se va a decidir si un punto es de borde según que la magnitud del gradiente supere o no un determinado umbral, pues bien sólo es cuestión de ajustar dicho umbral para que el resultado de la extracción de bordes sea el mismo, tanto si se calcula la magnitud del gradiente (ecuación 8) como (ecuación 9), sin embargo, esta última ecuación es mucho más fácil de implementar, particularmente cuando se realiza en hardware. A continuación se menciona uno de estos filtros empleados en la detección de bordes.

*Gradiente de una Imagen*: El gradiente de una imagen *f(x, y)* en un punto *(x, y)* se define como un vector bidimensional (ecuación 7), siendo un vector perpendicular al borde.

$$G[f(x,y)] = \begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} \frac{d}{dx} f(x,y) \\ \frac{d}{dy} f(x,y) \end{bmatrix} \tag{7}$$

Donde el vector *G* apunta a la dirección de variación máxima de *f* en el punto *(x, y)* por unidad de diferencia con la magnitud y dirección dadas por

$$|G| = \sqrt{G_x^2 + G_y^2}; \qquad \emptyset(x,y) = \tan^{-1}\frac{G_y}{G_x} \tag{8}$$

Es una práctica habitual aproximar la magnitud del gradiente con valores absolutos

$$|G| \approx |G_x| + |G_y| \tag{9}$$

Para calcular la derivada (ecuación 7) se pueden utilizar las diferencias de primer orden entre dos pixeles adyacentes; esto es:

$$G_x = \frac{f(x+\Delta x) - f(x-\Delta x)}{2\Delta x} \qquad G_y = \frac{f(y+\Delta y) - f(y-\Delta y)}{2\Delta y} \tag{10}$$

Ésta es la forma más elemental de obtener el gradiente en un punto. La magnitud del gradiente puede obtener cualquier valor real y ángulo también cualquier valor real entre 0° y 360°. No obstante hay otros operadores ampliamente difundidos para implementar el concepto de derivada en un punto y que consideran una vecindad de dimensión 3x3 entorno al punto, como lo es el operador de *Sobel*. (Figura 3.5).

Siguiendo el concepto de derivada (ecuación 10), se puede obtener una imagen binarizada considerando un umbral de *T* para la magnitud del gradiente, es decir:

$$g(x,y) = \begin{cases} 1 \; si \; G[f(x,y)] > T \\ 0 \; si \; G[f(x,y)] \leq T \end{cases} \tag{11}$$

a) b)

Figura 3.5.a) Imagen a color, b) Gradiente de una imagen

*Histograma*: son considerados como medidas estadísticas de la imagen y normalmente son usados como medio de ayuda para evaluar propiedades importantes de

una imagen. Especialmente son los errores producidos en la toma de la imagen los que son fácilmente reconocidos mediante la utilización del histograma.

Son distribuciones que describen la frecuencia con la que se presentan los valores de intensidad (*píxeles*) de la imagen. En el caso más sencillo los histogramas son mejor entendidos por medio de imágenes a escala de grises, Cuevas (2010).

### 3.1.3. Etapa de Interfaz

Comprende la interfaz utilizada para el envío de los datos, desde la adquisición de la imagen hasta la etapa FPGA.

### 3.1.4. FPGA

Se describen los procesos que se le efectuarán a la imagen adquirida, los resultados son exhibidos, almacenados y enviados para su análisis. (Figura 3.6). Así mismo comprende todos los módulos necesarios del sistema.

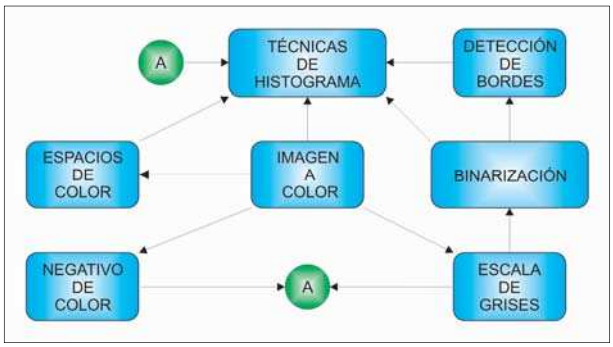



Figura 3.6.Procesos que se ejecutarán dentro del FPGA a la imagen

## 3.2. Requerimientos de un sistema de procesamiento de imágenes

- *Módulo de adquisición de datos.* Este módulo debe recibir los datos desde una interfaz de captura, este puede ser, una cámara digital o en su defecto, desde la PC vía RS232.

- *Módulo de interacción con la memoria.* Este módulo debe generar las señales adecuadas en el tiempo preciso para la interacción con el dispositivo de memoria.
- *Módulo de registro.* Debe almacenar de manera temporal el dato que será procesado de una dirección de la memoria.
- *Módulo de exhibición de resultados*. Para la exhibición de las imágenes en un monitor o pantalla LCD.
- *Módulo de procesos.* En él se procesan las imágenes con las transformaciones u operaciones deseadas.
- *Contador programable*. Este generará la dirección de memoria a la que se desea escribir o leer.
- *Bancos de memoria.* Estos bancos ayudan a almacenar en distintas localidades de memoria varias imágenes.
- *Activaciones sincronizadas.* Comprende en generar las señales de escritura, lectura de una manera automática en tiempos determinados.
- *Velocidad.* Ya que el procesamiento de imágenes, comprende una gran cantidad de datos, el sistema de ser rápido.
- *Sincronización*. Todos los módulos deben estar perfectamente sincronizados para el correcto funcionamiento y resultados confiables.

Los requerimientos mencionados se pueden describir en un diagrama a bloques, donde, se contemple la interconectividad entre ellos sección 3.3. (Figura 3.7).

## 3.3. Diseño de los módulos que componen el sistema

De acuerdo con Romero (2007), para el diseño de los diagramas descriptivos de hardware se emplea la metodología TOP-DOWN, el diagrama de primer nivel, comprende las entradas y salidas que el sistema requiere. (Figura 3.7).

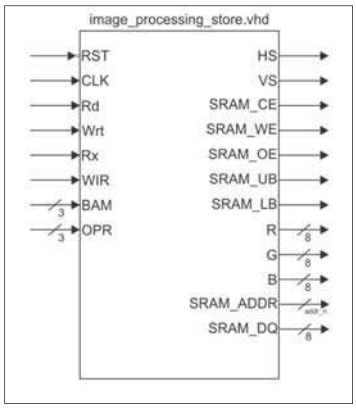



Figura 3.7. Descripción general del sistema, Image_processing_store.vhd

Cuadro 3.1. Señales del módulo general, Image_processing_store.vhd

| **Señal** | **Descripción** |
|---|---|
| RST | Reset maestro del sistema |
| CLK | Reloj fuente del sistema, 50 MHz |
| Rd | Operación de lectura |
| Wrt | Operación de escritura |
| Rx | Recepción del dato vía RS232 |
| WIR | Operación lectura/escritura |
| BAM | Banco de memoria |
| OPR | Operación de procesamiento de imágenes |
| HS | Sincronía Horizontal |
| VS | Sincronía Vertical |
| SRAM_CE | Chip Enable de la memoria se activa en '0' |
| SRAM_WE | Habilitación de escritura. Activa en '0' |
| SRAM_OE | Habilitación lectura. Activa en '0' |
| SRAM_UB | Habilita el byte más significativo del bus de datos de cada dirección. |
| SRAM_LB | Habilita el byte menos significativo del bus de datos de cada dirección. |
| R | Canal de color Rojo. Bus de datos 8 bit |
| G | Canal de color Verde. Bus de datos 8 bit |
| B | Canal de color Azul. Bus de datos 8 bit |

El siguiente paso en el diseño, consiste en realizar un diagrama de bloques de segundo nivel, el cual, presenta la interconectividad de los módulos que componen el sistema (Figura 3.8).

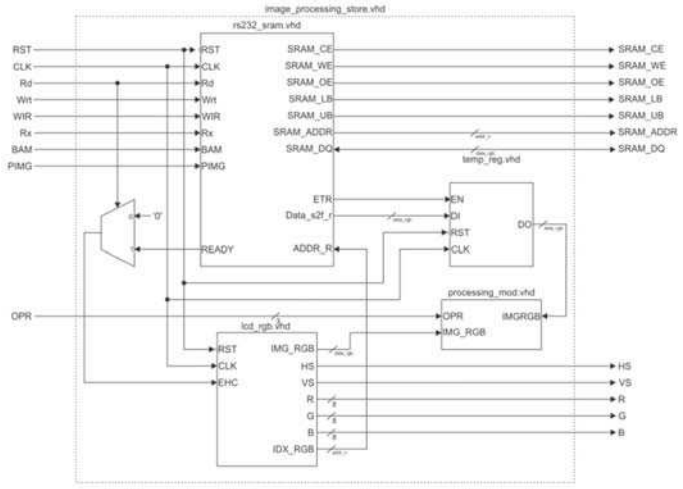



Figura 3.8. Interconectividad de los componentes del módulo image_processing_store.vhd

Es importante recalcar que, para realizar una correcta operación del sistema, cada componente está constituido por sub-módulos ( Cuadro 3.2).

Cuadro 3.2. Componentes del sistema image_processing_store.vhd

| Módulo | Sub Módulos | Componentes | Descripción |
|---|---|---|---|
| rs232_sram.vhd | rs232rx.vhd | | Este módulo es el encargado de recibir el dato proveniente de una computadora por protocolo RS232. |
| | sram.vhd | sram_ctrl.vhd | Controlador de las señales de operación para la memoria externa SRAM |
| | | fsm_sram.vhd | Se encarga de generar las señales de activación del controlador en el tiempo preciso. Así como proporcionar el banco de memoria donde se almacena o lee el dato del pixel deseado |
| | | programable_counter.vhd | Proporciona la dirección en la cual se almacenará el dato o bien en el proceso de lectura/escritura. |
| processing_mod.vhd | color_neg.vhd<br>gray_scale.vhd<br>bin.vhd<br>opr_mux.vhd | | Este módulo realiza las operaciones de procesamiento de imágenes deseadas, como negativo de color, escala de grises, binarización por umbral, separación de color.<br>El componente opr_mux.vhd, administra la salida del dato procesado. |
| lcd_rgb.vhd | driver_lcd.vhd | Genera la sincronía y la exhibición de los datos en pantalla. | |
| tem_reg.vhd | | Este registro se encarga de mantener el dato leído para ser procesado en el módulo processing_mod.vhd | |

Cabe destacar que los módulos importantes para este trabajo de investigación son: *sram_ctrl.vhd*, ya que es el driver de la interacción con la memoria SRAM de la tarjeta y el módulo *processing_mod.vhd*, donde se implementan los algoritmos de procesamiento de imágenes.

### 3.3.1. SRAM DE2-115

Es difícil para un sistema síncrono accesar directamente a un módulo SRAM directamente. Para ello, es necesario un *controlador de memoria* como interface, el cual, toma comandos desde el sistema síncrono principal y después genera señales correctamente medidas en tiempo para tener acceso al SRAM. El controlador brinda al sistema principal del tiempo detallado y hace que el acceso a memoria parezca como una operación síncrona. El funcionamiento de un controlador es medido por el número de accesos a memoria que pueden ser completados en un periodo dado. Mientras que el diseño de un controlador simple de la memoria es directo, la realización del funcionamiento óptimo implica muchas ediciones de la sincronización y es absolutamente complicado.

La tarjeta de desarrollo DE2-115, cuenta con un dispositivo SRAM de 2 MB, el cual tiene un bus de almacenamiento de datos de 16 bit. Puesto que las características de cada dispositivo SRAM son diferentes, el controlador diseñado en esta sección, es aplicable solo para este dispositivo en particular. Sin embargo, el principio de diseño puede ser utilizado para dispositivos similares SRAM. El encapsulado SRAM es un IS61WV102416BLL, organizada en 1, 048, 576 (20 bit) direcciones por 16 bit, con una frecuencia máxima de funcionamiento de 125 MHz (8 ns por ciclo de reloj mínimo). Fabricada por Integrated Silicon Solution, Inc. (Junio, 2009). (Figura 3.9)

Figura 3.9. Conexiones entre FPGA y SRAM

Como ya se mencionó, el uso de la memoria SRAM requiere de un controlador para utilizarla y realizar las operaciones de lectura/escritura correctamente, el módulo (Figura 3.10) representa el controlador para la memoria y en la tabla 3.1 se describe cada señal empleada en el módulo.

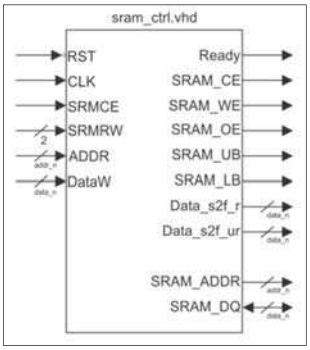



Figura 3.10. Controlador de memoria

Cuadro 3.3. Descripción de las señales del módulo sram_ctrl.vhd

| Nombre de la Señal | Descripción |
|---|---|
| addr_n | Indica el tamaño del bus de direcciones. Máximo 20 bit |
| data_n | Indica el tamaño del bus de datos. Máximo 16 bit |
| RST | Reinicia el sistema a sus condiciones iniciales. Se activa en estado bajo '0' |
| CLK | Señal de reloj |
| SRMCE | Indica cuando el chip enable de la memoria se habilitará |
| SRMRW[1:0] | Indica la operación a realizar lectura/escritura. Bus de datos de 2 bit; "00", "11" La operación lectura/escritura se encuentra inactiva. "01" La memoria se activa para escritura. "10" La memoria se encuentra activa para escritura |
| ADDR[addr_n-1:0] | Indica la dirección a la cual se escribirá o leerá un dato |
| DATA_W[data_n-1:0] | Es el dato que se va a escribir en la memoria |
| Ready | Esta señal indica el momento en que la memoria se encuentra lista para utilizarla |
| SRAM_CE | Señal de chip select habilita la memoria |
| SRAM_WE | Habilita la memoria para operación de escritura |
| SRAM_OE | Habilita la memoria para operación de lectura |
| SRAM_UB | Habilita el byte más significativo para almacenar datos |
| SRAM_LB | Habilita el byte menos significativo para almacenar datos |
| Data_s2f_r[data_n-1:0] | Dato de la memoria al FPGA como registro |

| | |
|---|---|
| Data_s2f_ur[data_n-1:0] | Señal de dato de la memoria al FPGA sin registro |
| SRAM_ADDR[addr_n-1:0] | Dirección de la memoria |
| SRAM_DQ[data_n-1:0] | Señal bidireccional de los datos que serán almacenados o leídos en la memoria. |

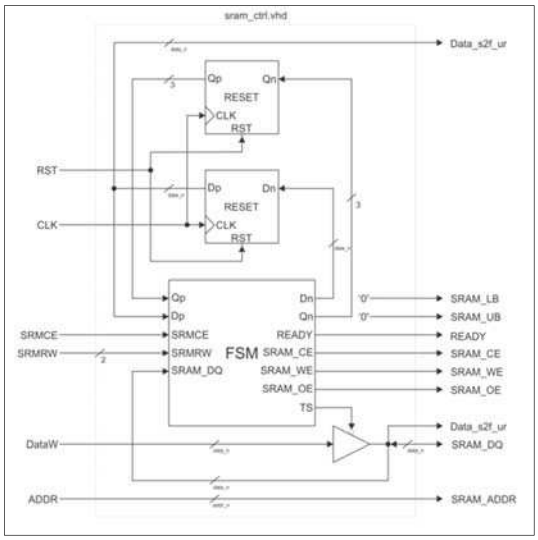



Figura 3.11. Diagrama de Bloques del módulo sram_ctrl.vhd

Se observan dos procesos, los cuáles son: máquina de estados finitos (FSM) y un proceso de reinicio (RESET), estos procesos llevan a cabo el control de las señales que se interconectan al FPGA, (Figura 3.9). El proceso RESET es un módulo Flip-Flop y se activa en estado bajo. El proceso FSM es una máquina secuencial, la cual ayuda a activar las conexiones del FPGA a la SRAM. Se observa que los estados importantes son Q2 y Q4, ya que en ellos se efectúa la escritura y lectura. (Figura 3.12)

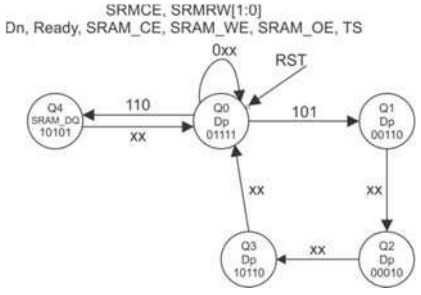



Figura 3.12. Grafo del proceso FSM

Para probar el módulo se realiza una simulación donde se activarán SRMCE y SRMRW en un tiempo aleatorio, se puede comprobar que se almacenan datos ya que se utiliza una memoria implementada en VHDL por Romero (2007) (Figura 3.13).

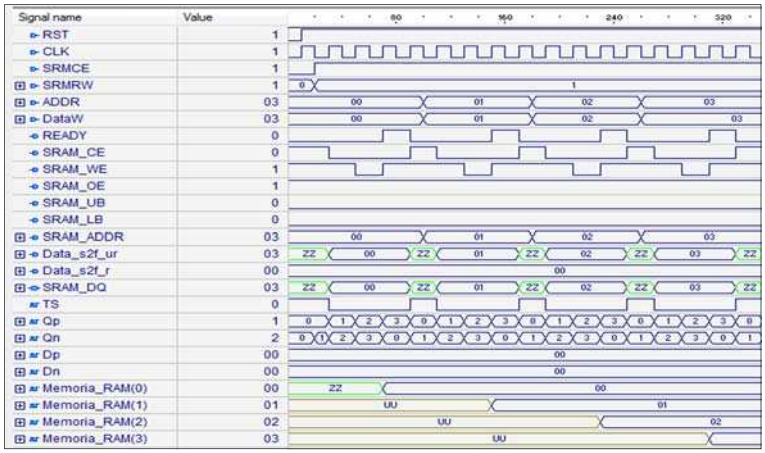



Figura 3.13. Simulación del módulo sram_ctrl.vhd.

En la simulación se observa el tiempo el funcionamiento de las señales del módulo, los valores que toman las señales y el momento en que se almacena el dato en la memoria RAM y en la dirección correspondiente. En esta simulación se almacenan 4 datos en 4 direcciones, la señales SRMCE se activa después de 20 ns y se mantiene hasta almacenar todos los datos, cada ciclo de escritura dura 60 ns, sin embargo, el momento en que el dato se almacena, transcurren 20 ns, el tiempo total de la simulación es de 350 ns, únicamente escritura. La Figura 3.14 muestra la simulación de la operación de lectura, la cual, consiste en leer las direcciones donde ya se ha almacenado algún dato con anterioridad.

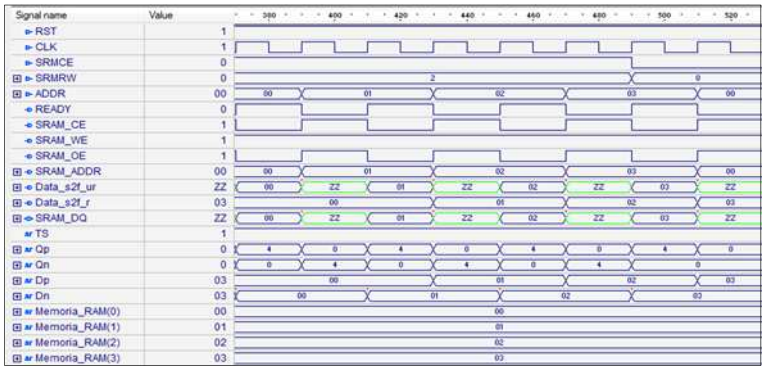



Figura 3.14. Simulación de la operación de lectura

En esta simulación se observa, cuando la señal SRMCE está activa y SRAMRW tiene un valor 2, los datos son almacenados exactamente en el momento en que SRAM_CE y SRAM_OE se activan en estado bajo, cada dato se lee en un tiempo de 20 ns, la dirección de memoria debe estar presente por 40 ns. El tiempo de lectura de las 4 direcciones es de 120 ns. Así mismo se observa que, el tiempo de lectura es menor al de escritura esto se debe a los requerimientos que se necesitan para leer y exhibir el dato de una imagen a través de la pantalla LCD de prueba.

Asignando los valores necesarios a las señales SRMCE y SRMRW, como se observa, los tiempos son muy pequeños. Si esto se implementa en la tarjeta de desarrollo los tiempos serían relativamente grandes, ya que las señales son estimuladas manualmente. Para evitar esto y que las operaciones de almacenamiento y lectura se lleven en los tiempos adecuados es necesario realizar una máquina secuencial que genere estas señales cada cierto tiempo.

### 3.3.2. Diseño de los Algoritmos de Procesamiento de Imágenes en Hardware

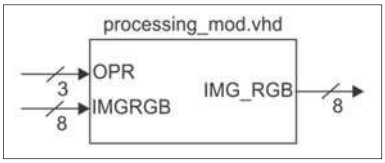



Figura 3.15. Processing_mod.vhd

El módulo *processing_mod.vhd* contiene en su interior los algoritmos de procesamiento de imágenes, (Figura 3.16).

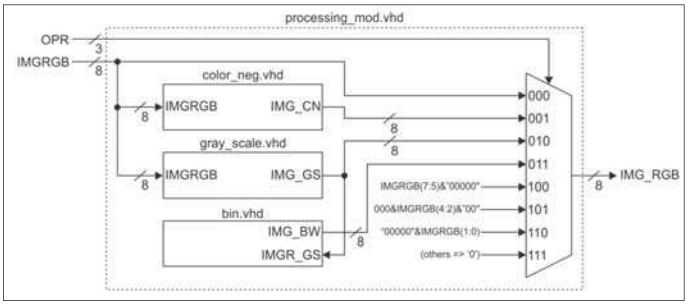



Figura 3.16. Interconectividad de los componentes del módulo processing_mod.vhd

En este módulo se observa un multiplexor este se encarga de conmutar la salida que se exhibirá o almacenará dependiendo de la operación a realizar por el sistema. En el multiplexor se observa, la separación del color, este se realiza únicamente con el canal de color que se desee obtener ya sea rojo (R), verde (G) o azul (B).

De acuerdo con la teoría de la sección 3.1.2, donde se mencionan los algoritmos de procesamiento de imágenes se establecen las descripciones de hardware de cada uno de ellos.

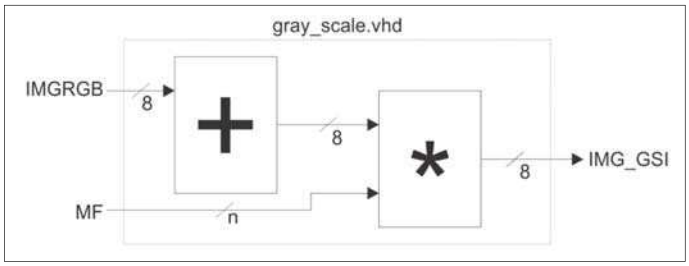



Figura 3.17. Gray_scale.vhd

La descripción para obtener la escala de grises consta en realizar la suma de los tres canales y multiplicarlo por un factor constante, este valor es obtenido realizando las pruebas necesarias en el software.

El diagrama para obtener el negativo de una imagen, consta en tomar el valor máximo que se puede obtener del bus de datos de cada canal. La señal *IMGRGB* que lleva los datos de los 3 canales de color al módulo, es dividida en el interior del módulo, para realizar la operación de resta. Ya realizada la operación los 3 canales se unen para formar el bus de datos del tamaño reguerido (Figura 3.18).

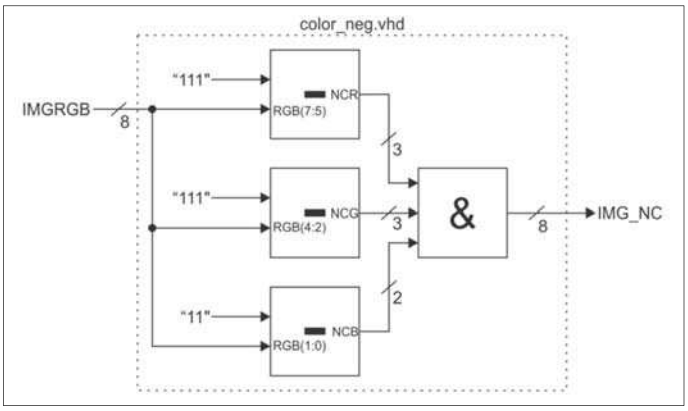



Figura 3.18. Módulo color_neg.vhd

El módulo de binarización de la imagen *bin.vhd*, consta únicamente en fijar un valor como umbral y realizar una comparación del dato obtenido en escala de grises con el umbral y si este es menor que el umbral enviar los datos de salida en '0' y si es mayor en '1'.

Una vez implementado todos los módulos anteriormente mencionados para la obtención del módulo principal, se realiza una simulación de las tres operaciones a realizar con la memoria, (Figura 3.19 a Figura 3.21).

La primer simulación es de la operación de la recepción de los por protocolo rs232. En ella se observa el momento en que las señales SRAM_CE y SRAM_WE se activan ('0') y se almacena el dato recibido en la dirección correspondiente. Cabe mencionar que para la simulación se reciben 4 datos para almacenar en las primeras 4 direcciones los cuáles son: $C5_{16}$ ($197_{10}$), $D5_{16}$ ($213_{10}$), $B1_{16}$ ($177_{10}$) y $F0_{16}$ ($240_{10}$), en las direcciones 0 a 3, el banco de memoria es el 0. (Figura 3.19).

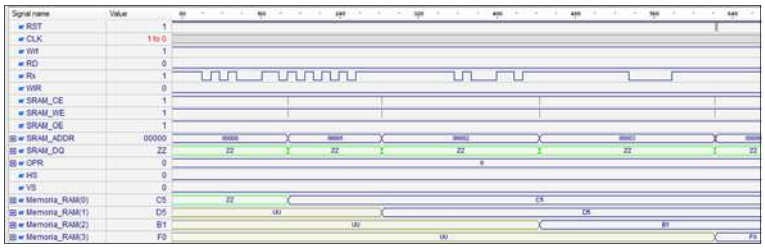


Figura 3.19. Simulación operación de escritura

La operación de lectura se realiza en intervalos de tiempo menores a la escritura, esto se debe a que el módulo de exhibición requiere de 40 ns para exhibir el dato en pantalla (Figura 3.20).

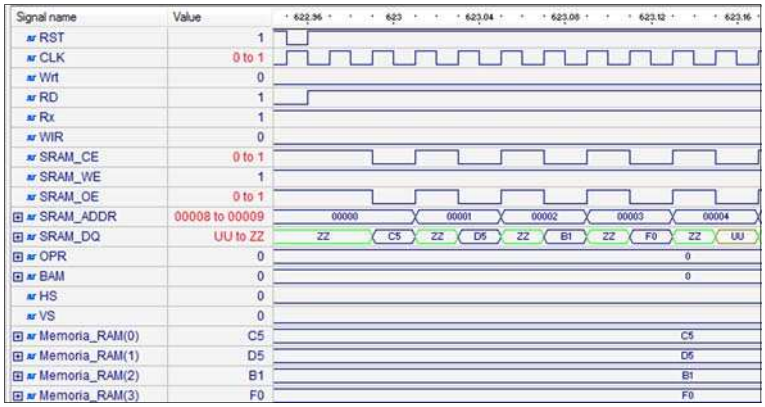



Figura 3.20. Simulación operación de lectura

La operación de lectura/escritura, consta en leer primero la dirección del banco en el que se encuentra almacenada la imagen a procesar, una vez leído el dato, este es procesado en el módulo *processing_mod.vhd*, y por último, almacenado en un banco de memoria distinto. Esta operación tiene un periodo de tiempo de 200 ns por pixel (Figura 3.21).

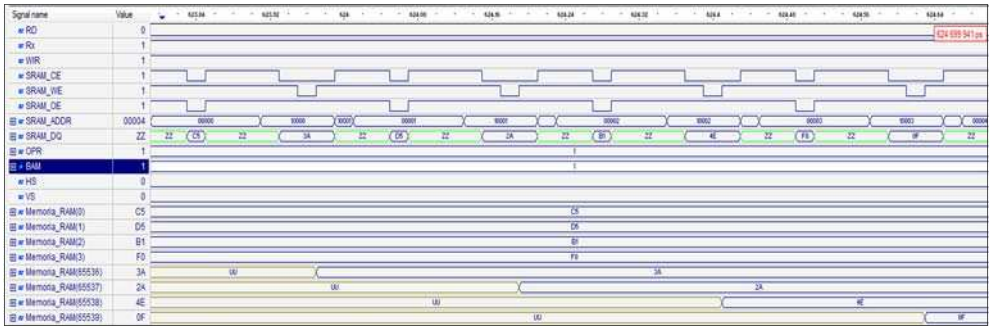


Figura 3.21. Simulación operación lectura/escritura

# IV. PRUEBAS Y RESULTADOS

La tarjeta seleccionada para las pruebas del sistema es la tarjeta DE2-115. Las pruebas realizadas son:

1. Almacenamiento de la imagen en memoria SRAM.
2. Exhibición de la imagen almacenada.
3. Aplicar alguna operación de procesamiento de imágenes como: negativo de color, escala de grises, binarización por umbral, separación del color, etc.
4. Comparar los resultados obtenidos con una imagen en punto flotante con una imágen en punto fijo.

## 4.1. Almacenamiento de la imagen en la memoria SRAM

En la sección 3.3 se menciona una señal que indica el banco de memoria donde se almacenará la imagen o del cual se leerá. Físicamente la memoria SRAM, no tiene la característica de proporcionar bancos de memoria. Sin embargo, esto es posible, ya que la imagen de prueba mide 256x256 pixeles o 65,5335 pixeles, es decir, la imagen requiere de 65,535 direcciones de memoria. La memoria SRAM de la tarjeta DE2-115 cuenta con un bus de direcciones de 20 bit, de los cuáles se necesitan al menos 16 bit para almacenar una imagen, restando los 4 bit más significativos para almacenar otras 16 imágenes de la misma dimensión, de los cuáles, se necesitan 3 bit para almacenar 6 imágenes adicionales a la del banco 0. Para este trabajo en particular la profundidad del color es de 8 bit, es decir, 3 bit para el canal R y G cada uno, y 2 bit para el canal B. el bus de datos de la memoria es de 16 bit.

El sistema para las pruebas, se compone por una laptop, la tarjeta DE2-115, pantalla LTM y el puerto RS232. (Figura 4.1).

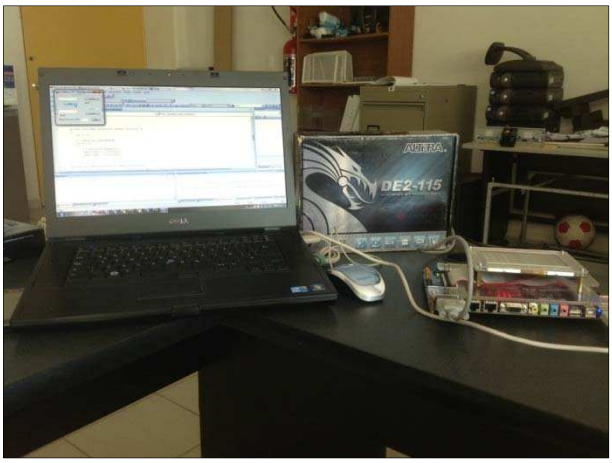



Figura 4.1. Sistema de prueba

Por lo anterior podemos adquirir la imagen desde una PC por protocolo RS232. La prueba del sistema en la tarjeta. La pantalla se encuentra apagada en esta operación, los leds rojos indican la dirección donde se almacena el dato recibido y el display de 7 segmentos , indica el banco de memoria donde se guardará la imagen (Figura 4.2).

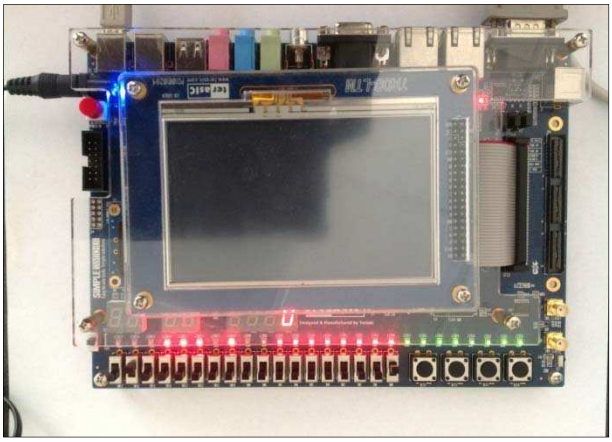


Figura 4.2. Recepción de los datos RS232

La velocidad de transferencia utilizada en esta etapa es de 115200, lo que nos permite almacenar en 65536 direcciones, 8 bit por dirección, en un tiempo de 14.06 s.

Una vez almacenada la imagen, se realiza el proceso de lectura y exhibición para apreciar si los datos fueron almacenados correctamente (Figura 4.3).

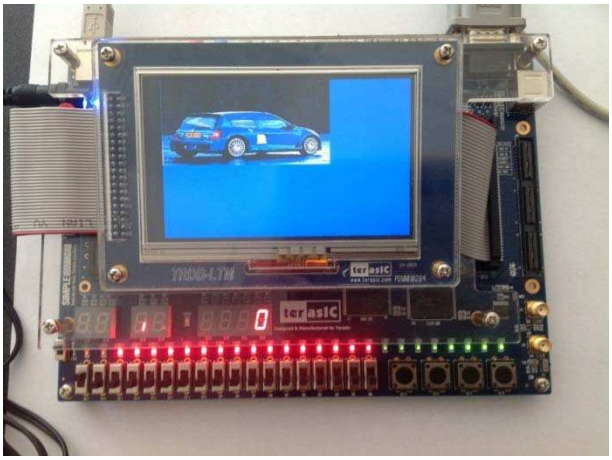


Figura 4.3. Lectura y exhibición de la imagen almacenada

Ya que cada vez que se lee un dato de la memoria este no debe pasar los 40 ns, esto quiere decir, que todos los datos de la memoria, son leídos en un periodo de 2.62 ms.

Cuando se aprecia que la imagen se ha guardado correctamente por medio de la exhibición, se realiza cualquiera de las operaciones de procesamiento de imágenes. En este punto se puede almacenar el resultado en otro banco de memoria, deshabilitando la lectura y habilitando la operación de lectura/escritura. O bien, aplicar el proceso y exhibirlo sin almacenarlo, este proceso lleva el mismo tiempo de lectura y exhibición anteriormente mencionado (Figura 4.4 a Figura 4.9).

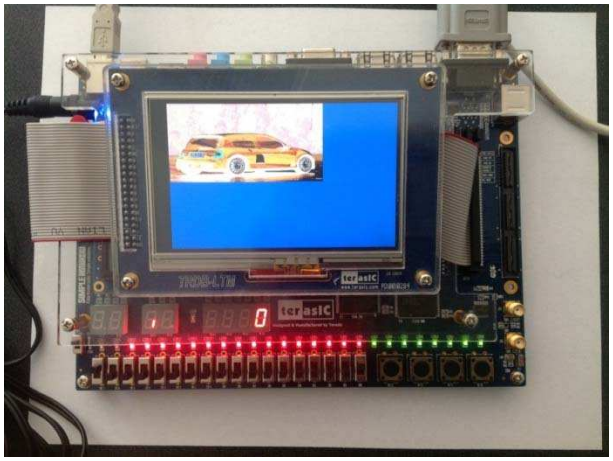


Figura 4.4. Operación negativo de color

Para la escala de grises se define el formato de punto fijo el valor constante de multiplicación (4.2)

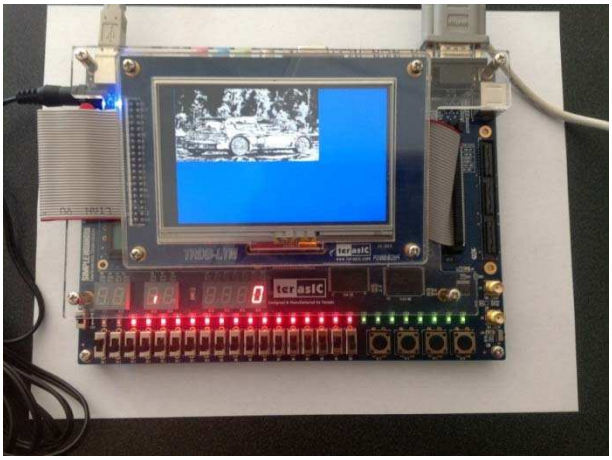


Figura 4.5. Escala de grises

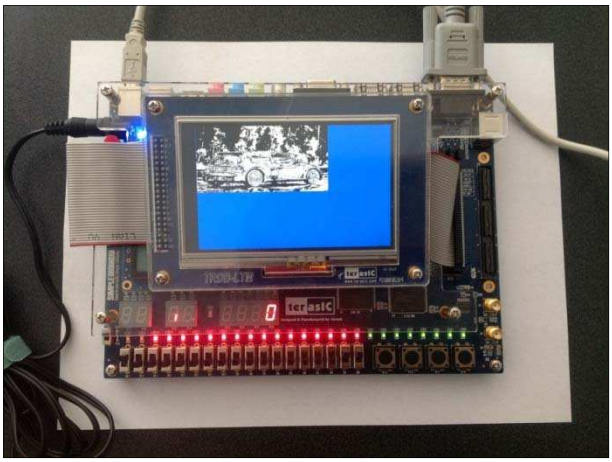


Figura 4.6. Binarización por umbral

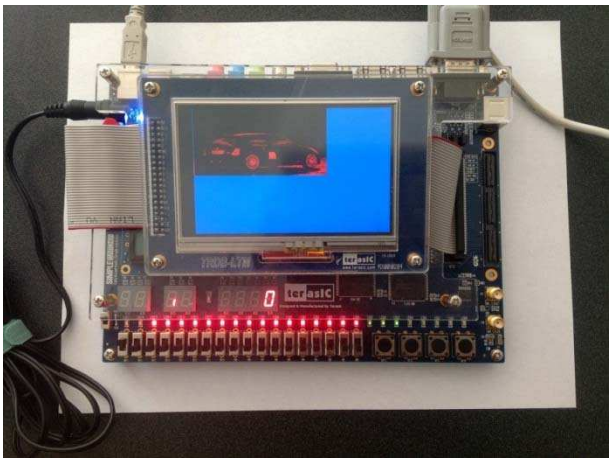


Figura 4.7. Separación del color rojo

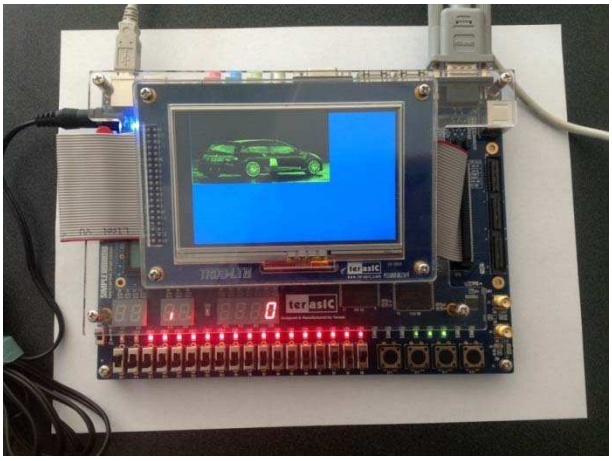


Figura 4.8. Sepearación del color verde

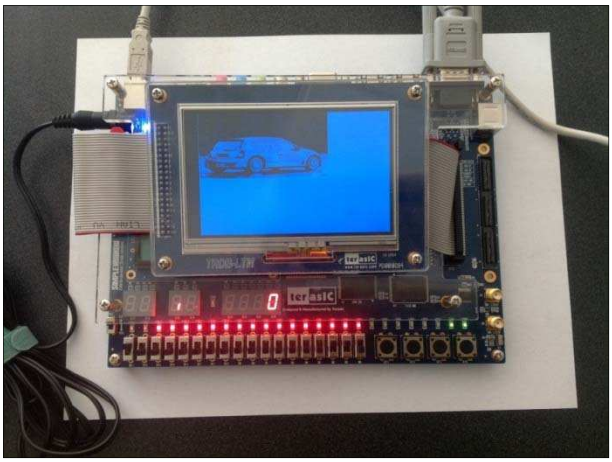


Figura 4.9. Separación del color azul

Para almacenar algún resultado, se cambia el banco de dirección (bancos disponibles 7), y se selecciona la operación deseada, este proceso toma un tiempo de 200 ns, desde que se lee el dato de la memoria hasta que se almacena, para almacenar las 65536 datos de la imagen procesada, toma un tiempo total de 0.013 s (Figura 4.10).

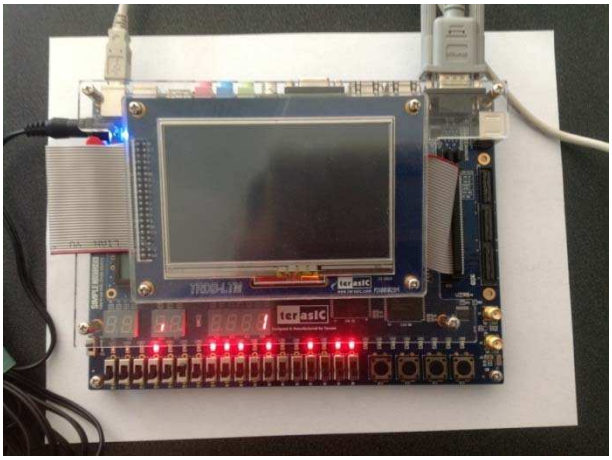


Figura 4.10. Almacenar resultados

Los bancos de memoria, se leen en el momento en que se indique en el sistema, cabe aclarar que, si en el banco no se ha almacenado datos, la exhibición no es la esperada (Figura 4.11). Sin embargo, si se guardan datos antes de leer el banco de memoria, el resultado puede ser el esperado (Figura 4.12).

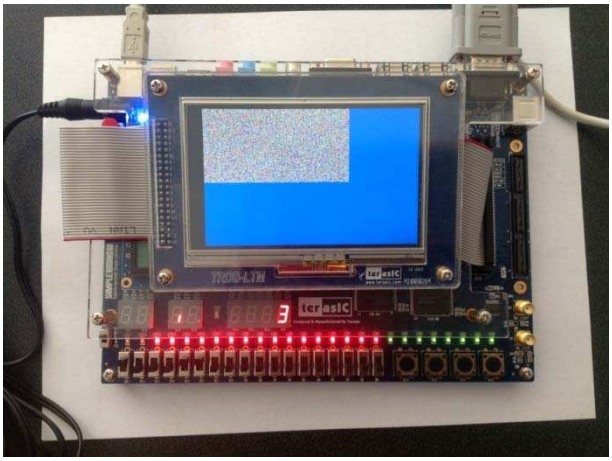


Figura 4.11. Banco de memoria sin datos de imagen

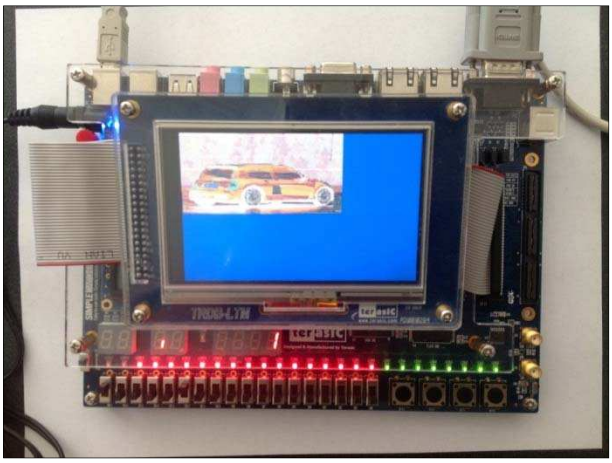


Figura 4.12. Banco de memoria con datos de imagen.

## 4.2. Resultados esperados en MATLAB

Para comparar los resultados obtenidos en el FPGA es necesario tener un punto de comparación el cual es posible obtener por medio de MATLAB, realizando un programa con los algoritmos de procesamiento de imágenes que se deseen implementar en el FPGA. Estas operaciones o procesos se realizan a imágenes de 24 bit y se realiza un programa con el cual las imágenes se ajustan a una resolución de 8 bit empleando formato de punto fijo, el cual es empleado para realizar alguna comparación con los datos obtenidos en el FPGA. Para ello se realiza la siguiente serie de pasos:

1. Cargar la imagen que será procesada.
2. Obtener los datos de la imagen, es decir, los valores RGB de cada pixel componente de la imagen.
3. Se convierte la imagen original de 24 bit a una imagen en formato de punto fijo, a la resolución deseada como: 3, 8, 12 o 24 bit. Esta imagen es utilizada como la imagen original en punto fijo.
4. Se realiza las operaciones de procesamiento de imágenes requerido, en punto fijo y en punto flotante.
5. Se compara la imagen en punto fijo y en punto flotante al finalizar los procesos.
6. Se lleva a cabo un análisis de resultados para validar el formato empleado.

7. Se corrigen errores en caso de existir o comprobación de que no hay errores en el código.

(Figura 4.13) Ilustra el proceso realizado en MATLAB para realizar procesamiento de imágenes en punto fijo y punto flotante.

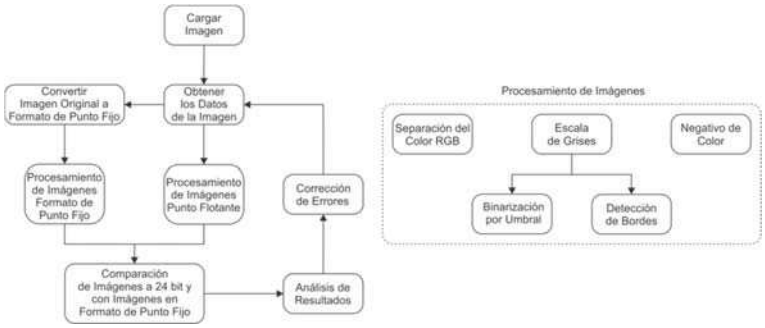



Figura 4.13. Procesamiento de imágenes en MATLAB

El formato de punto fijo empleado para esta prueba fue de 3:3, es decir; 3 bit por la parte entera y 3 bit para la parte fraccionaria, la resolución de las imágenes que se utilizarán en el FPGA es de 256x256 pixeles, con una profundidad de color de 8 bit. Al ejecutar los códigos anteriores, se obtiene cada resultado de los procesos realizados a una imagen en 24 bit y como se debe observar una imagen normalizada en 8 bit con el formato de punto fijo ya mencionado (Figura 4.14 a Figura 4.17).

a) b)

Figura 4.14. a) Imagen punto flotante, b) imagen punto fijo.

Similitud de imágenes en color: 87.53 %, error: 0.1247.

a) b)

Figura 4.15. a) Negativo en punto flotante, b) negativo punto fijo

Similitud de imágenes en negativo: 95.98%, error: 0.04014

a) b)

Figura 4.16. a) Escala de grises punto flotante, b) Escala de grises punto fijo

Similitud de imágenes en escala de gris: 87.77%, error: 0.122221.

a) b)

Figura 4.17. a) Binarización punto flotante, b) binarización punto fijo

Similitud de imágenes en escala de gris: 97.39%, error: 0.026042.

# V. CONCLUSIONES Y TRABAJO A FUTURO

En las pruebas realizadas en software y hardware, se observa una pérdida de calidad en el color con respecto a la imagen original, esto se debe a la normalización de la imagen empleada a 8 bit de color. Sin embargo, la calidad de la imagen resultante es aceptable, ya que se distinguen la mayoría de los detalles en la imagen.

En cuanto a tiempo, el realizar estas operaciones de procesamiento de imágenes a la imagen en punto flotante (24 bit), lleva un total de 1.826474 s, y en formato de punto fijo, lleva un total de 84.199225 s, al ejecutar los dos en conjunto, lleva un tiempo de 85.900365 s, el cual ya es un tiempo considerable de espera para los resultados.

Sin embargo al realizar estas operaciones empleando FPGA, como la unidad de procesamiento, los procesos son realizados en unos cuantos ciclos de reloj dependiendo la operación que se realice.

Incorporar el sistema de adquisición por medio de una cámara, para realizar procesamiento de imágenes en tiempo real.

Aumentar el tamaño de la imagen así como la profundidad de color a 24 bit, para reducir el error de similitud en el análisis de las imágenes.

# VI. REFERENCIAS BIBLIOGRÁFICAS


Bailey Donald G. 2011. Design for Embedded Image Processing on FPGA. John Wiley & Sons, IEEE, New Zealand. ISBN: 978-0-470-82849-6.

Boemo Scalvinoni Eduardo. 2005. Estado del Arte de la Tecnología FPGA. Cuaderno Tecnológico N0. 1 Microelectrónica, Instituto Nacional de Tecnología Industrial, Argentina. http://www.inti.gob.ar/electronicaeinformatica/instrumentacion/utic/publicaciones/cuadernilloUE/CT_Microelectronica17_FPGA.pdf; última consulta 30 de octubre de 2012.

Branislav Kisacanin, Shuvra S. Bhattacharyya. 2009. Embedded Computer Vision, Springer, Londrés. ISSN 1617-7916, ISBN 978-1-84800-303-3.

Bravo Múñoz Ignacio. 2007. Arquitectura Basada en FPGA's para la detección de objetos en movimiento, utilizando visión computacional y técnicas PCA, Tesis Doctoral, Universidad de Alcalá, España,.

Carlos Alberto Ramos-Arreguín, Juan Carlos Moya-Morales, Juan Manuel Ramos-Arreguin, Jesus Carlos Pedraza-Ortega, Saul Tovar Arriaga, Marco Antonio Aceves Fernandez, Jose de Jesus Rangel-Magdaleno. Mayo 2012. FPGA Open Architecture Design for a VGA Driver. The 2012 Iberoamerican Conference on Electronics Engineering and Computer Science, Guadalajara, Jalisco, Procedia Technology, Elsevier, Volume 3 pages 324-333; ISSN: 2212-0173 Mexico.

Castillo A., Vázquez J., Ortegón J. y Rodríguez C. Junio, 2008. Prácticas de Laboratorio para Estudiantes de Ingeniería con FPGA; IEEE Latin America Transactions, Vol 6, No. 2.

Cuevas Erik, Zaldívar Daniel, Pérez Marco. 2010. Procesamiento Digital de Imágenes con MATLAB y Simulink, Alfaomega. México. ISBN: 978-607-707-030-6.

Dubey Rahul. 2009. Introduction to Embedded System Design using Field Programmable Gate Arrays. Springer-Verlag London Limited, India. ISBN 978-1-84882-015-9.

Galeano Gustavo. 2009. Programación de Sistemas Embebidos en C. Alfaomega, 1ª Edición, México DF. ISBN: 978-958-682-770-6.

Galeano Gustavo. 2009. Programación de Sistemas Embebidos en C. Alfaomega, 1ª Edición. México DF. ISBN: 978-958-682-770-6.

Ganssle Jack, Noerggard Tammy, Eady Fred, Edwards Lewin, Katz David J., Gentile Rick, Arnold Ken, Hyder Kamal, Perrin Bob, Huddleston Creed. 2008. Embedded Hardware. Springer. ISBN 978-0-7506-8584-9.

González Aguirre Marco Antonio, Morales Velázquez Luis, Osornio Ríos Roque Alfredo, Morales Hernández Luis Alberto. Noviembre, 2011. IP Core Genérico para Adquisición de Imágenes en Plataforma Basada en FPGA. 10° Congreso Nacional de Mecatrónica, Puerto Vallarta, Jalisco, México. pp 315-319, ISBN: 978-60795347-5-2.

Gonzalez Rafael C., Woods Richard E. 2008. Digital Image Processing. Prentice Hall, 3ª Edición, USA.ISBN: 0-201-18075-9.

Pajares Gonzalo, Sanz Martin. 2008. Visión por Computador, Imágenes Digitales y Aplicaciones. Alfaomega, 2ª Edición. ISBN: 978-84-7897-831-1.

Pong P. Chu; FPGA Prototyping by VHDL Examples Xilinx Spartan – 3 Version, Wiley Interscience; New Jersey, 1ª Edición, USA, 2008; ISBN 978-0-470-18531-5.

Quintero M. Alexander, Vallejo R. Eric. 2006. Image Processing Algorithms using FPGA. Revista Colombiana de Tecnologías de Avanzada. Vol. 1; No. 7. Pags. 11 a 16, ISSN: 1692-7257.

Ramos Arreguín Carlos Alberto, Cora Gallardo Orlando Marcos, Ramos Arreguín Juan Manuel, Pedraza Ortega Jesús Carlos, Canchola Magdaleno Sandra Luz, Vargas Soto José Emilio. 2010. Metodología para Manejo de Imágenes en FPGA. 6º Congreso Internacional de Ingeniería. Querétaro, Qro. México. ISBN: 978-607-7740-39-1.

Ramos Arreguín Carlos Alberto, Moya Morales Juan Carlos, Ramos Arreguín Juan Manuel, Canchola Magdaleno Sandra Luz, Vargas Soto José Emilio. Noviembre 2010. Metodología de una Etapa Básica de un Sistema de Procesamiento de Imágenes Basado en FPGA. 9º Congreso Nacional de Mecatrónica, Puebla, Puebla. pp 235-240, ISBN: 978-607-95347-2-1.

Ramos Arreguín Juan Manuel, Aceves Fernández Marco Antonio. Septiembre 2010. Lógica Digital; Universidad Autónoma de Querétaro. ISBN978-607-7740-43-8.

Romero Troncoso René de Jesús. 2007. Electrónica Digital y Lógica Programable. Universidad de Guanajuato, 2ª Edición; Guanajuato, México. ISBN: 968-864-449-8.

Tusch Michael. 2006. High-Performance Image Processing on FPGAs. Xcell Journal, Xilinx, pages 42 to 44.

Walls Colin. 2006. Embedded Software: The Works. Elsevier, USA. ISBN 0-7506-7954-9.

Altera Corporation, www.altera.com

Digilent, Basys 2 reference manual, www.digilentinc.com

Digilent, Cool Runner II reference manual, www.digilentinc.com

Digilent, Nexys 2 reference manual, www.digilentinc.com.

Digilent, Spartan 3 reference manual, www.digilentinc.com

Digilent,Spartan 3AN reference manual, www.digilentinc.com

Integrated Silicon Solution Inc. Junio 2009. IS61WV102416ALL/BLL IS64WV102416BLL Data sheet. www.issi.com, última consulta 1 de Enero de 2013.

Xilinx, www.xilinx.com

Terasic DE2-115 reference manual, www.terasic.com

Terasic LTM user manual, www.terasic.com

# ANEXOS

**CIINDET 2011**

***IX Congreso Internacional sobre Innovación y Desarrollo Tecnológico, 23 al 25 de noviembre de 2011, Cuernavaca Morelos, México.***

# Metodología para Almacenamiento de Imágenes en Memorias externas de tipo RAM, empleando FPGA.

C. A. Ramos Arreguín Student Member IEEE, J. C. Moya Morales student member IEEE, J. M. Ramos Arreguín member IEEE, J. C. Pedraza Ortega member IEEE, M. A. Aceves Fernández, J. E. Vargas Soto member IEEE, S. Tovar Arriaga member IEEE.

**Resumen:** En el presente trabajo se propone una metodología para almacenar diversas imágenes en memorias externas de tipo RAM (Memorias de Acceso Aleatorio), y procesarlas en FPGA (Arreglo Programable de Compuertas en Campo), para exhibir los resultados en un monitor. Se presentan descripciones generales del proyecto, ya que la aplicación puede variar dependiendo la necesidad del usuario.
Palabras Clave: Metodología, FPGA, RAM.

**Abstract**: In the present work, a methodology to storage different images in RAM (Random Access Memory) is proposed, also the image processing in the FPGA is shown (Field Programmable Gate Array), later, the results can be observed on a display. General descriptions of the work are presented, because the application could vary depending on the needs of the user.
Keywords: Methodology, FPGA, RAM.

## Introducción

En la actualidad, en el área de procesamiento de imágenes, el uso de FPGA, es cada vez más frecuente debido a las ventajas que conlleva, como lo son: portabilidad de código, bibliotecas de código reutilizables, herramientas de programación baratas (algunas gratuitas, ISE de Xilinx, versión Web Pack), bajo costo, reconfigurabilidad, procesamiento paralelo, capacidad de interactuar con interfaces de alta o baja velocidad, protección y reutilización de la propiedad intelectual [1]. Así mismo, al trabajar en el área de procesamiento digital de imágenes, se requiere guardar la imagen adquirida, la cual es procesada por el sistema. Así mismo, los resultados de algunas operaciones realizadas a la imagen, requieren ser almacenados como resultados intermedios de otras operaciones de procesamiento de imágenes, como filtros pasa alta y pasa baja. En el FPGA es posible guardar una imagen, la desventaja es que el almacenamiento llega a ocupar el 100% de los recursos del FPGA, dependiendo del encapsulado y la resolución de la imagen. Esto tiene el inconveniente de que la síntesis de la aplicación requiere demasiado tiempo (aprox. 5 hrs.) para la generación del archivo .bit para la programación del dispositivo [2]. Debido a lo anterior, surge la necesidad de realizar un almacenamiento externo de manera temporal utilizando memorias de tipo RAM.

En la actualidad, existen diversos trabajos de procesamiento de imágenes en FPGA, empleando memorias RAM para el almacenamiento temporal de las imágenes, empleando librerías de fabricantes de tarjetas de desarrollo como es el caso de MemUtil de Digilent [3]. Otra manera, es desarrollar un software para el envío de la información al FPGA y a la memoria RAM, o bien, empleando una interfaz con una cámara para la adquisición de la imagen. Como en algunos trabajos realizados con anterioridad, donde se emplean memorias de tipo RAM para almacenar resultados y analizarlos. Sin embargo no muestran una metodología para el control de la interfaz con la memoria [4-8], además las tarjetas empleadas en algunos casos, tienen un núcleo FPGA de la familia VIRTEX de Xilinx, Cyclone de Altera, los cuáles, son de costo alto y poseen gran capacidad para el procesamiento de imágenes debido a sus características, a su vez[9-13].

Este trabajo se enfoca, en la propuesta de una metodología, para el almacenamiento de imágenes en una memoria RAM y ser leídas y exhibidas en monitor

Ing. Carlos Alberto Ramos Arreguín. Facultad de Informática U.A.Q., Querétaro Qro, México. (cramos06@alumnos.uaq.mx).
Ing. Juan Carlos Moya Morales. Facultad de Informática U.A.Q., Querétaro Qro, México. (moyajc@gmail.com).
Dr. Juan Manuel Ramos Arreguín. Facultad de Informática U.A.Q., Querétaro Qro, México. (jramos@mecamex.net).
Dr. Jesús Carlos Pedraza Ortega. Facultad de Informática U.A.Q., Querétaro Qro, México. (caryoko@yahoo.com).
Dr. Marco Antonio Aceves Fernández. Facultad de Informática U.A.Q., Querétaro Qro, México. (marco.aceves@uaq.mx).
Dr. José Emilio Vargas Soto. Facultad de Informática U. A. Q., Querétaro Qro., México. (emilio@mecamex.net).
Dr. Saúl Tovar Arriaga. Facultad de Informática U. A. Q., Querétaro Qro, México. (saulotov@yahoo.com.mx).
*Agradecemos especialmente al Consejo de Ciencia y Tecnología del Estado de Querétaro (CONCYTEQ) y al Gobierno del Estado de Querétaro, por el apoyo proporcionado para la publicación de este artículo.*

empleando FPGA, únicamente utilizando los estándares del IEEE. Ya que hoy en día no se cuenta con una metodología general para la utilización de memorias RAM, librerías estándar y libres, ya que las existentes tienen un costo adicional, si se desea utilizar módulos ya existentes, las cuáles, son desarrolladas por las empresas fabricantes de tarjetas de desarrollo o FPGA.

## Metodología General de Diseño de Sistemas Embebidos

En la figura 1, se muestran las etapas de una metodología general empleada para el diseño de sistemas embebidos [1].


*Figura 1. Metodología general empleada para Diseño de Sistemas Embebidos*

A continuación se describe cada etapa de la metodología de la figura 1:

*Requerimientos*: Se refiere a todo requerimiento funcional y no funcional, como lo pueden ser: tamaño, peso, consumo de energía y costo.

*Especificaciones de Usuario*: Detalles de la interfaz de usuario junto con operaciones necesarias para satisfacer la petición de usuario.

*Arquitectura*: Hardware (procesador, periféricos, lógica programable y ASSPs) y Software (programas de mando y sus operaciones).

*Diseño del Componente*: Componentes prediseñados, componentes modificados y nuevos componentes.

*Integración del Sistema*: Esquema de verificación para encontrar errores rápidamente.

Basándose en la metodología anterior, se propone en este trabajo una metodología para almacenar información de distintas imágenes en memorias de tipo RAM, utilizando el estándar del IEEE para el lenguaje descriptivo de hardware VHDL.

## Metodología Propuesta

En la figura 2 se ilustra el diagrama de la metodología propuesta en este trabajo.

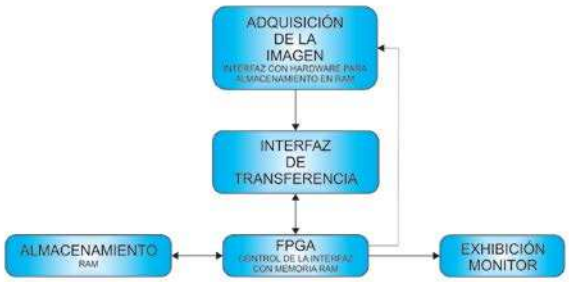



*Figura 2. Metodología propuesta*

*Adquisición de la Imagen*: esta etapa comprende en obtener la imagen requerida para procesar. La adquisición puede ser desde una cámara digital o bien, una imagen almacenada en un PC.

*Interfaz de Transferencia*: Comprende la interfaz utilizada para el envío de los datos, de la etapa de adquisición de la imagen a la etapa de almacenamiento, a través del FPGA. La interfaz puede ser un sistema digital para el procesamiento de la información de la imagen proveniente de una cámara digital, o bien, vía puerto RS232, cuando la información de la imagen es enviada desde una PC. Para este proyecto se usa el puerto RS232.

*FPGA*: Interfaz de control para el almacenamiento en memoria RAM y procesamiento de la imagen.

*Almacenamiento*: Se almacenan los datos de cada pixel de la imagen recibida de las etapas anteriores, o bien, se almacenan los resultados obtenidos en la etapa de arquitectura y cuando se requiera se enviarán los datos guardados a la etapa anterior, para llevar a cabo el procesamiento deseado.

*Exhibición*: Se muestran los resultados obtenidos en un monitor.

## Memorias de tipo RAM

La memoria de acceso aleatorio (RAM), es usada en un sistema digital para almacenamiento masivo de información o datos, una célula RAM es mucho más simple que una célula FF (Flip Flop). El tipo de memoria comúnmente empleado es el SRAM (asynchronous static RAM). Para un sistema síncrono el acceso directo a una memoria SRAM es complicado, para esto es necesario implementar un *controlador de*

*memoria* para la interacción entre el FPGA y la memoria, el cual es activado desde el sistema síncrono principal, el cual genera señales correctamente medidas en tiempo para accesar a la memoria, ya sea para escritura o lectura. El controlador brinda al sistema principal el tiempo detallado y hace que el acceso a memoria parezca una operación síncrona [8]. Existen diversos tipos de memorias RAM, sin embargo, el funcionamiento básico es el mismo para todos, cambiando algunas características que las hacen más eficientes o con mayor funcionalidad.

## Desarrollo del Sistema

En la figura 3, se muestra una descripción para un controlador genérico de una memoria RAM, el cual se compone de tres módulos, los cuáles son:

*Divisor de frecuencia*, consiste en generar un pulso el tiempo necesario para cumplir con los tiempos de escritura/lectura de la memoria.

*Máquina Secuencial*, donde se determina la operación y el tiempo de reposo para que se lleve a cabo cada operación, figura 4.

*RAM*: a este módulo llegan los datos a ser almacenados y la dirección donde se almacenarán, el bus de salida HAB, agrupa 5 señales de habilitación, que son: *escritura, lectura, dispositivo* (*chip select*)*, byte más significativo* y *byte menos significativo*.

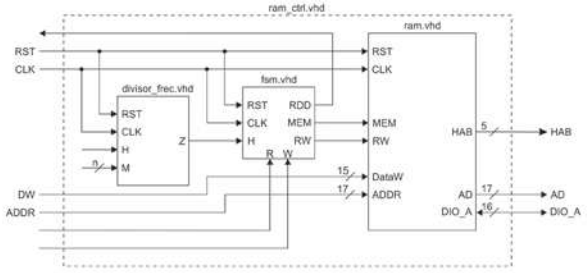



*Figura 3. Diagrama de bloques del control para utilizar la memoria RAM.*

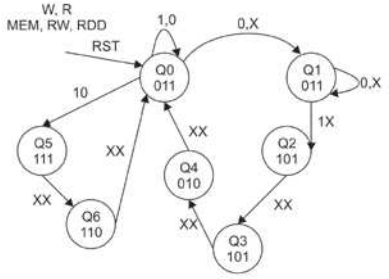



*Figura 4. Maquina secuencial del controlador RAM*

El módulo controlador, figura 5, funciona en conjunto con el módulo de adquisición de la imagen. El módulo de adquisición puede utilizar cualquier protocolo como RS232, USB, directamente de una cámara digital, entre otros.

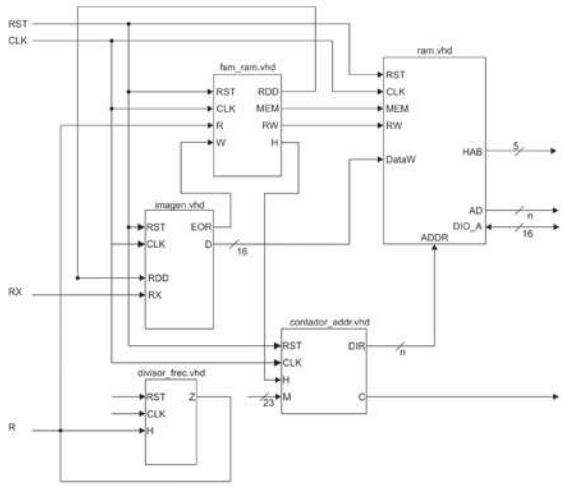



*Figura 5. Control RAM incorporado con la interfaz de recepción de datos vía RS232*

En el módulo *imagen.vhd* de la figura 5, se reciben los datos correspondientes a la imagen, la señal EOR indica la finalización de la recepción de los datos y envía una señal de habilitación para la escritura en el módulo *fsm_ram.vhd*. Así mismo, este módulo habilita al módulo *contador_addr.vhd*, en el cual se incrementa la dirección para almacenar el siguiente dato. La señal C indica cuando la última dirección de la memoria ha sido escrita, es decir, ya no se cuenta con más direcciones para realizar almacenamiento de datos.

En el diagrama de la figura 6, se describe el sistema para almacenamiento de imágenes y su exhibición. Donde se integran los módulos *vgasync.vhd* y *vga_rgb.vhd*, propuestos en trabajos previos [2], [7]. El módulo *procesos.vhd*, consta de las operaciones que requiera aplicar a la imagen, como lo son: escala de grises, binarización, filtros pasa baja y pasa alta, entre otros. También se utiliza el módulo de control RAM, descrito anteriormente en la figura 7, y es modificado para almacenar en distintas direcciones de la memoria.

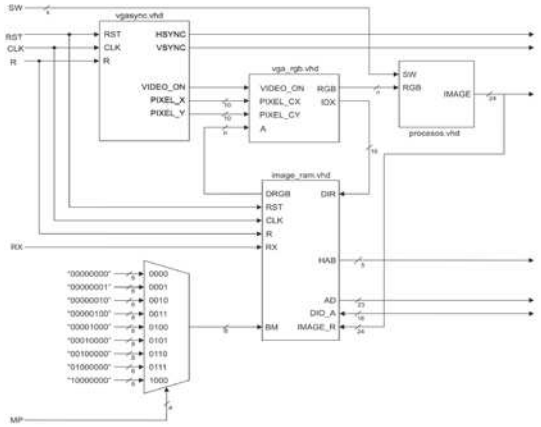



*Figura 6. Descripción del sistema para almacenamiento de imágenes y su exhibición.*

## Tarjeta de Experimentación Nexys 2

En la figura 7 se muestra la tarjeta empleada en el proyecto y sus características son las siguientes [14]:

- FPGA Spartan 3E de Xilinx XC3SE1200 FG320.
- SDRAM Micron 16 MB (128 Mb).
- Frecuencia de reloj 50 MHz.
- Puerto VGA de 8 bit.
- Comunicación RS232.

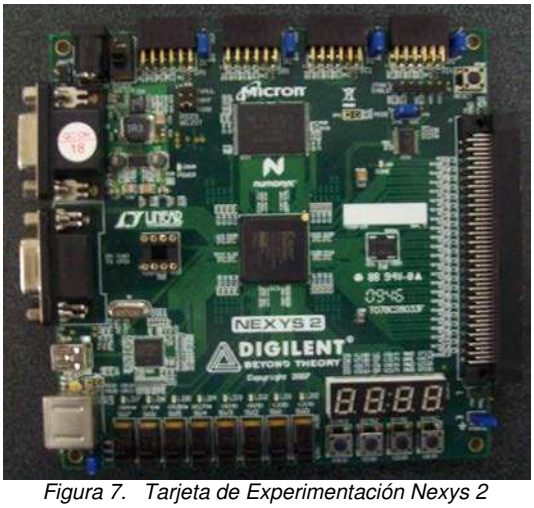



*Figura 7. Tarjeta de Experimentación Nexys 2*

El dispositivo RAM, empleado en este proyecto es una SDRAM MICRON MT45W8MW16BGX, el cual es un dispositivo CMOS de alta velocidad, memoria de acceso pseudo-estática desarrollada para aplicaciones portables de baja potencia, puede operar como una SRAM asíncrona típica. Este dispositivo contiene un núcleo SDRAM de 128 Mbit, organizado con 8M direcciones cada una de 16 bit para almacenamiento de datos. En la figura 8, se muestra el diagrama de bloques del funcionamiento interno de la memoria [15].

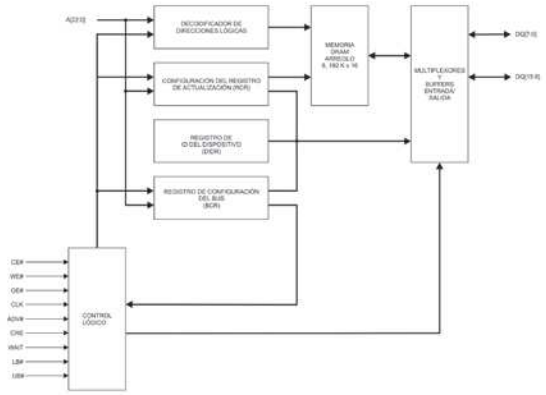



*Figura 8. Diagrama simplificado de la operación de la memoria SDRAM de la tarjeta Nexys 2.*

La operación de lectura (Read) es iniciada con un estado '0' en las señales CE (chip enable), OE (output enable), LB (Lower Byte), UB (Upper Byte) y WE (Write Enable) en estado '1'. Los datos son enviados después del acceso en el tiempo especificado, el ciclo de trabajo se presenta en la figura 9.

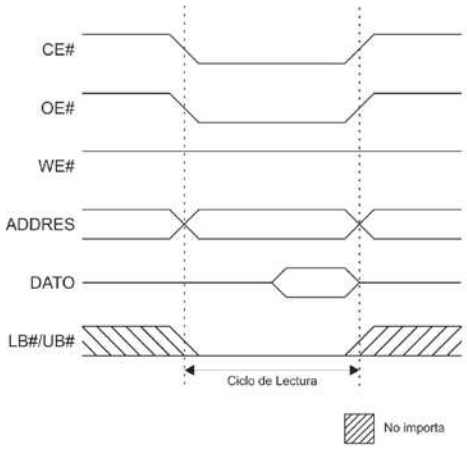



*Figura 9. Ciclo de lectura*

La operación de escritura (Write), ocurre cuando CE, WE, LB y UB, se encuentran en estado '0' y OE en estado '1'. ADV puede estar en estado '0' para cualquiera de las dos operaciones. Su ciclo de trabajo se ilustra en la figura 10.

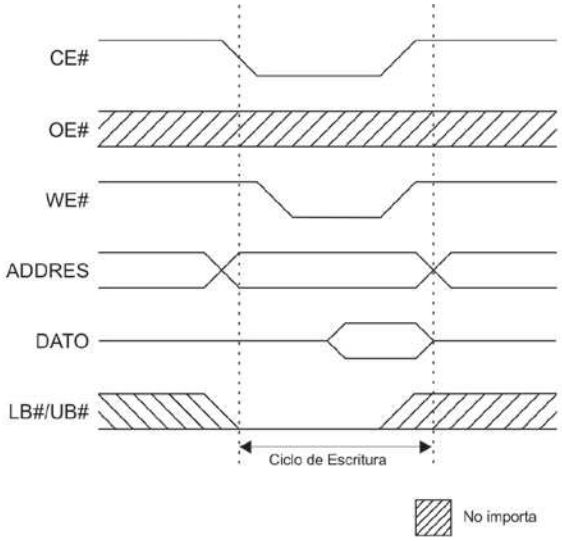
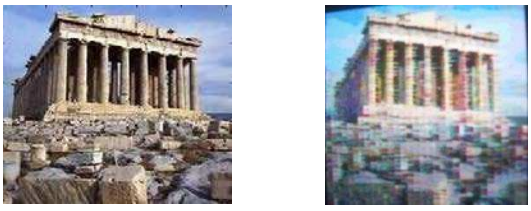



*Figura 10. Operación de Escritura*

## Pruebas y Resultados

Para implementar el sistema en la tarjeta de experimentación es necesario adaptar el módulo de mapeo de las direcciones dependiendo de la cantidad de imágenes que requieran ser almacenadas. En este trabajo se almacenan: imagen original, escala de grises, negativo de color y binarización. Por lo que el multiplexor debe ser ajustado para 4 imágenes. Así mismo, la imagen es normalizada de 24 bit de color a 8 bit [2, 7]. En las figuras 11 a 14, se muestran los resultados de las imágenes, reproducidas en FPGA y exhibidas en un monitor de tipo CRT.

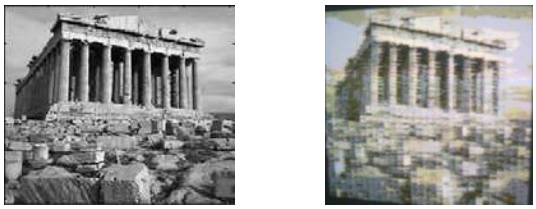


a) b)

*Figura 11. a) Imagen original a 24 bit, b) Imagen normalizada almacenada en la memoria SDRAM*

a) b)

*Figura 12. a) Negativo de color, obtenido en software, b) Resultado en Hardware*



a) b)

*Figura 13. Escala de Grises, obtenida en software, b) Resultado en hardware*

a) b)

*Figura 14. Imagen Binarizada, obtenida en software, b) Binarización en FPGA.*

Cabe mencionar que la diferencia o pérdida de calidad en las imágenes reproducidas en el hardware, es debido a la normalización efectuada antes de almacenar la primera imagen en la memoria SDRAM.

## Conclusiones y trabajo a futuro

Es factible utilizar memorias externas para almacenamiento de datos, como imágenes, por la gran cantidad de datos que se puede almacenar temporalmente, ya que realizar el almacenamiento directamente en el FPGA, resulta costoso por el alto número de recursos consumidos, y tiempo de síntesis alto. Además, los recursos empleados del FPGA, son mínimos (1%).

La dimensión de prueba de cada imagen es de 256x256 pixeles, es decir, se ocupan 65536 direcciones y en cada dirección se almacenan 8 bit de datos, en la memoria SDRAM para almacenar una imagen.
El ciclo de trabajo de la memoria RAM es especificada por el fabricante, la metodología permite adaptar diferentes ciclos de operación, para diversos dispositivos RAM de distintos fabricantes. Así mismo, puede ser implementada a FPGAs de distintos fabricantes, debido a que únicamente se utilizan librerías estándar del IEEE.
La aportación de este trabajo de investigación es una metodología estándar para el manejo de memorias externas RAM, donde se describe un controlador básico para llevarlo a cabo, el cual, puede ser extendido fácilmente en funcionamiento dependiendo la necesidad del usuario.
Como trabajo a futuro, se propone trabajar con imágenes de 24 bit de color, para obtener una imagen de la misma calidad a la obtenida en software (PC). Implementar algoritmos de procesamiento de imágenes de mayor complejidad, detección de bordes, filtros pasa baja y pasa alta, transformada de Fourier, entre otros.
Desarrollar e implementar un sistema digital, para la adquisición de imágenes provenientes de una cámara digital.

## Referencias

[1] Dubey Rahul, Introduction to Embedded System Design Using Field Programmable Gate Arrays, Springer-Verlag London Limited, ISBN: 978-1-84882-015-9, Londres, 2009.

[2] Ramos Arreguín Carlos Alberto, Cora Gallardo Orlando Marcos, Ramos Arreguín Juan Manuel, Pedraza Ortega Jesús Carlos, Canchola Magdaleno Sandra Luz, Vargas Soto José Emilio, Metodología para Manejo de Imágenes en FPGA, 6º Congreso Internacional de Ingeniería, pp. 219-226, ISBN: 978-607-7740-39-1, Querétaro, Qro., Abril 2010.

[3] Software MemUtil de Digilent Incorporated para almacenamiento de datos en memorias de tipo RAM incorporadas en algunas tarjetas de desarrollo, www.digilentinc.com, última consulta 29 de abril 2010.

[4] Quintero M. Alexander, Vallejo R. Eric, Image Processing Algorithms using FPGA, Revista Colombiana de Tecnologías de Avanzada, Vol. 1; No. 7; pp. 11-16, ISSN: 1692-7257, 2006.

[5] Bravo Muñoz Ignacio, Arquitectura basada en FPGA para la Detección de Objetos en Movimiento, utilizando Visión Computacional y Técnicas PCA, Tesis Doctoral, Departamento de Electrónica de la Escuela Politécnica de la Universidad de Alcalá, España 2007.

[6] A. Castillo, J. Vázquez, J. Ortegón, C. Rodríguez, Prácticas de Laboratorio para Estudiantes de Ingeniería con FPGA, IEEE Latin America Transactions, pp. 130-136, Junio 2008.

[7] Ramos Arreguín Carlos Alberto, Moya Morales Juan Carlos, Ramos Arreguín Juan Manuel, Pedraza Ortega Jesús Carlos, Metodología de una Etapa Básica de un Sistema de Procesamiento de Imágenes basado en FPGA, 9º Congreso Nacional de Mecatrónica, pp. 235-240, ISBN: 978-607-95347-2-1, Octubre 2010.

[8] Pong P. Chu, FPGA Prototyping by VHDL Examples Xilinx Spartan 3 Version, Wiley Interscience, ISBN: 978-0-470-18531-5, USA 2008.

[9] Kalomiros J. A., Lygouras J., Design and evaluation of a hardware/software FPGA-based system for fast image processing, Microprocessors and Microsystems, Elsevier, pp. 95 – pp. 106, doi:10.1016/j.micpro.2007.09.001.

[10] Chaikalis D., Sgouros N. P., Maroulis D., A Real Time FPGA Architecture for 3D reconstruction from integral images, Journal of Visual Communication & Image Representation, Elsevier, pp. 9 - pp. 16, doi:10.1016/j.jvcir.2009.09.004.

[11] Siéler L., Tanougast C., Bouridane A., A scalable and embedded FPGA architecture for efficient computation of gray level co-ocurrence matrices and Haralick textures features, Microprocessors and Microsystems, Elsevier, pp. 14 - pp. 24, doi:10.1016/j.micpro.2009.11.001.

[12] Krill B., Ahmad A., Amira A., Rabah H., An efficient FPGA-based dynamic partial reconfiguration design flow and environment for image and signal processing IP cores, Signal Processing: Image Communication, Elsevier, pp. 377 - pp. 387, doi:10.1016/j.image.2010.04.005.

[13] Satake Shin-ichi, Sorimachi Gaku, Masuda Nobuyuki, Ito Tomoyoshi, Special-purpose computer for particle image velocimetry, Computer Physics Communications, Elsevier, pp. 1178 – pp. 1182, doi:10.1016/j.cpc.2011.01.022.

[14] Datasheet Nexys 2, www.digilentinc.com, última consulta 29 de Abril 2010.

[15] Micron Technology, Inc. Datasheet MT45W8MW16BGX, última consulta 29 de Abril de 2010.

## Currículo corto de los autores

**Carlos Alberto Ramos Arreguín**
Ingeniero en Computación egresado de la Facultad de Informática, Universidad Autónoma de Querétaro en 2011. Actualmente es estudiante de Maestría en Ciencias de la Computación en la Facultad de Informática, Universidad Autónoma de Querétaro y su línea de investigación es procesamiento digital de imágenes en hardware.

**Juan Carlos Moya Morales**
Ingeniero en Computación egresado de la Facultad de Informática, Universidad Autónoma de Querétaro en 2011. Actualmente es estudiante de Maestría en

Ciencias de la Computación en la Facultad de Informática, Universidad Autónoma de Querétaro y su línea de investigación es en procesamiento digital de imágenes.

**Juan Manuel Ramos Arreguín**

Ingeniero en Electrónica egresado de la Facultad de Ingeniería Mecánica, Eléctrica y Electrónica en 1994, Universidad de Guanajuato. Maestría en Ingeniería Eléctrica en 1996, en la Facultad de Ingeniería Mecánica, Eléctrica y Electrónica, Universidad de Guanajuato. Sus estudios de Doctorado fueron en Ciencia y Tecnología con especialidad en Mecatrónica en el Centro de Investigación y Desarrollo Industrial CIDESI, Querétaro en 2008. Actualmente es profesor-investigador en la Facultad de Informática, Universidad Autónoma de Querétaro y su línea de investigación es sistemas embebidos.

**Jesús Carlos Pedraza Ortega**

Ingeniero en Electrónica egresado del Instituto Tecnológico de Celaya en 1993. Maestría en Ingeniería Eléctrica en la Facultad de Ingeniería Mecánica, Eléctrica y Electrónica, Universidad de Guanajuato. Sus estudios de Doctorado fueron en Ingeniería Mecánica en la Universidad de Tsukuba, Japón en 2002. Actualmente es profesor-investigador en la Facultad de Informática, Universidad Autónoma de Querétaro y su línea de investigación es en procesamiento digital de imágenes.

**Marco Antonio Aceves Fernández**

Ingeniero en Telemática egresado de la Universidad de Colima en 2000. Cursó la Maestría y Doctorado en la Universidad de Liverpool, Inglaterra en el 2006. Actualmente es Profesor-Investigador en la Facultad de Informática de la Universidad Autónoma de Querétaro y su línea de investigación es en sistemas embebidos.

**José Emilio Vargas Soto**

Ingeniero Mecánico con mención honorífica por la Universidad Nacional Autónoma de México. Maestría en Ingeniería en el área de Control por la Universidad Politécnica de Madrid, España. Doctor en Ciencias Físicas, en el área de Informática y Automática por la Universidad Compútense de Madrid. Post-Doctorado por sus investigaciones sobre la locomoción libre de robots caminantes, por The Electrocommunications University of Tokio, Japón. Desde el 2010 a la fecha, es profesor-investigador en la Facultad de Informática, Universidad Autónoma de Querétaro y su línea de investigación es en inteligencia artificial y procesamiento digital de imágenes.

**Saúl Tovar Arriaga**

Ingeniero en Electrónica egresado del Instituto Tecnológico de Querétaro. Maestría en Mecatrónica por la Universidad de Siegen, Alemania 2006. Sus estudios de Doctorado fueron Biología Humana (equivalente a Ingeniería Biomédica) por el Instituto de Física Médica por la Universidad Friederich-Alexander en Nuremberg, Alemania en 2009. Desde el 2010 es profesor-investigador en la Facultad de Informática, Universidad Autónoma de Querétaro y su línea de investigación es en robótica aplicada a la medicina y procesamiento digital de imágenes.

Available online at www.sciencedirect.com

SciVerse ScienceDirect

Procedia Technology 00 (2012) 000–000

Procedia Technology

www.elsevier.com/locate/procedia

Conference title

# FPGA Open Architecture Design for a VGA Driver

[a]Carlos Alberto Ramos-Arreguín*, [a]Juan Carlos Moya Morales, [a]Juan Manuel Ramos-Arreguín, [a]Jesús Carlos Pedraza-Ortega, [a]Saúl Tovar-Arriaga,[a]Marco Antonio AcevesFernandez,[b]José de Jesús Rangel-M*gdaleno

[a]*Faculty of Computer Science, Universidad Autónoma de Querétaro*
[b]*Universidad Politécnica de Puebla*
*Av. De las Ciencias s/n. C.P., 76230Juriquilla Querétaro, Qro., México*
*San Mateo Cuanalá, Juan C. Bonilla, Puebla, México*

**Abstract**

This work presents an open architecture proposal for a VGA (Video Generic Array) controller to be used into embedded systems based in FPGA. Several developers of hardware design, which use some hardware description language, have a video library to be used with VHDL or Verilog language, but in most of cases, we need to buy an expensive annual license, and can be used only for the manufacturer hardware. This controller is developed using only VHDL based in the IEEE standards, to ensure the portability with any manufacturer, and this is part of the contribution of this work. The controller designed is generic, due that it can be used for any resolution used by a commercial monitor, including the widescreen monitor. The generic controller will be used for image processing research. The work presents two kinds of test: first, we display the eight basics color generated with the RGB (Red, Green, Blue) values; second, an image is stored into an external memory RAM (Random Access Memory), and the FPGA reads and display the image into a CRT (Cathode Ray Tube) and LCD (Liquid Crystal Display) monitor.



*Keywords:* FPGA; VGA; resolution.

## 1. Introduction

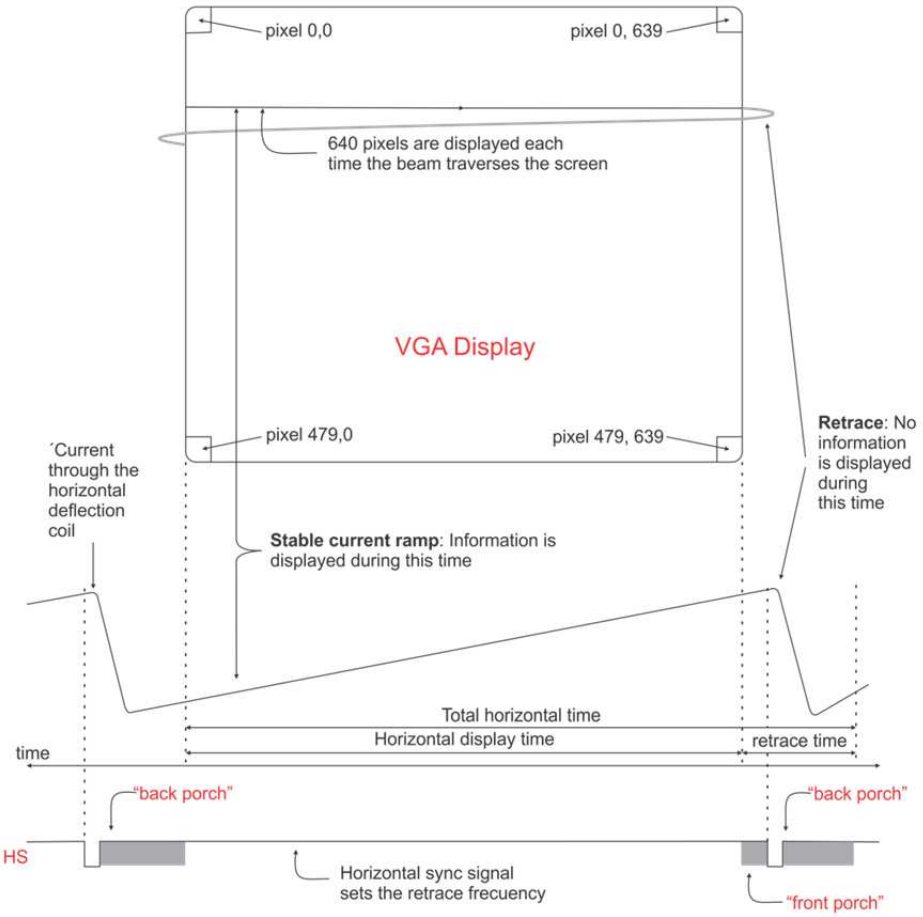
At present, the use of FPGA in research and development of applied digital systems for specific tasks is increasing. This is due to the advantages FPGAs has over other programmable devices. These advantages are: high clock frequency, high operations per second, code portability, code libraries reusability, low cost, parallel processing, capability of interacting with high or low interfaces, security and Intellectual Property (IP) retention, among others [1].

* Corresponding author. Tel.: +52 442 192 1200 ext 5941.
*E-mail address*: c.ramos@ieee.org.

One of the topics where FPGA has been used is image processing, attributable to the high degree of parallelism in comparison with a PC or a microcontroller where processing is sequential. Nowadays, there is much work carried out in the field of image processing based on FPGA, where image processing based algorithms have been widely implemented [2 - 11].
In order to be able to see the results obtained from the implemented algorithms on an FPGA, a standard VGA (Video Graphics Array) display may be chosen. However, a video controller is necessary to show the results in the display. This controller must be designed based on the datasheets of the FPGA's developing board, where the synchronization timing is specified. The VGA resolution corresponds to 640x480 pixels, created by IBM PCs in the 80's decade [12], which is widely support by both CRT and LCD technologies.

In this work, the video controller is proposed, showing from the standard VGA theory, to the results obtained in the development board for FGPA's Spartan 3 [13], Spartan 3AN [14], Nexys 2 [15], and Basys 2 [16].

## 2. Signal Timing for a 60 Hz, 640x480 VGA Display




CRT-based VGA displays use amplitude-modulated, moving electron beams (or cathode rays) to display information on a phosphor-coated screen. LCD displays use an array of switches that can impose a voltage across a small amount of liquid crystal, thereby changing light permittivity through the crystal on a pixel-by-pixel basis. LCD displays have evolved to use the same signal timings as CRT displays. Within a CRT display, current waveforms pass through the coils to produce magnetic fields that deflect electron beams to transverse the display surface in "raster" pattern, horizontally from left to right and vertically from top to bottom. Fig. 1 shows an example on exhibit time and vertical and horizontal sync on a CRT monitor [13 - 17].

Fig 1. CRT Display Timing VGA Example.

The timing needed for horizontal and vertical (HS and VS, respectively) VGA port sync of a monitor with a clock frequency of 25MHz with display resolution of 640-pixel by 480 row, are shown on table 1.

Table 1.640x480 Mode VGA Timing

| Symbol | Parameter | Vertical Sync | | | Horizontal Sync | |
|---|---|---|---|---|---|---|
| | | Time | Clocks | Lines | Time | Clocks |
| $T_S$ | Sync pulse time | 16.7 ms | 416, 800 | 521 | 32 µs | 800 |
| $T_{DISP}$ | Display time | 15.36 ms | 384, 000 | 480 | 25.6 µs | 640 |
| $T_{PW}$ | Pulse width | 64 µs | 1, 600 | 2 | 3.84 µs | 96 |
| $T_{FP}$ | Front porch | 320 µs | 8, 000 | 10 | 640 ns | 16 |
| $T_{BP}$ | Back porch | 928µs | 23, 200 | 29 | 1.92 µs | 48 |

The timing of VGA driver according to table 1 is shown on Fig. 2.

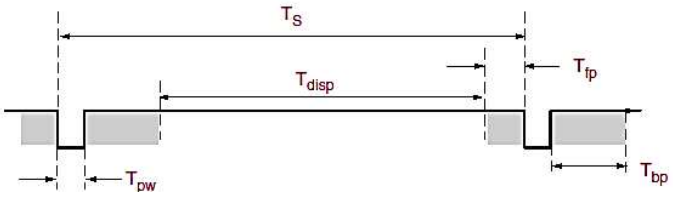



Fig. 2. VGA Control Timing

Digilent® provides in the reference manual of the development boards Basys-2® and Nexys-2®, the block diagram of the VGA controller, which is shown on Fig. 3.

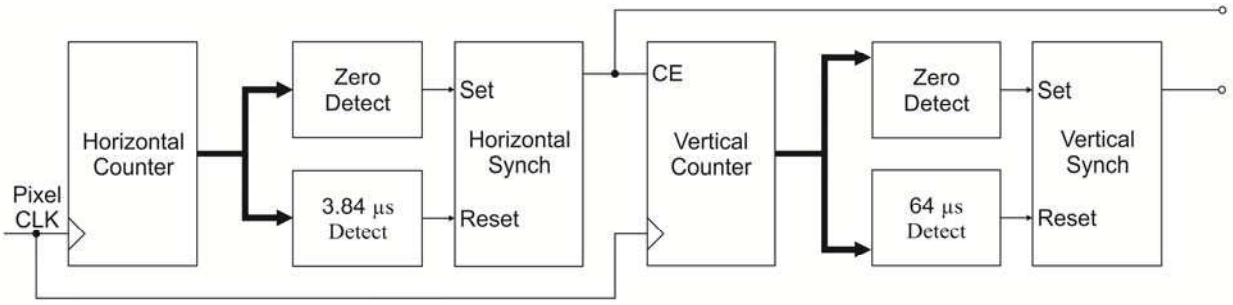



Fig. 3. Digilent VGA Sync General Block Diagram.

The VGA driver ought to generate HS and VS signals and coordinates the delivery of the video stream, based on the Pixel CLK (25MHz), this clock defined the needed time to display the pixel information. The signal VS defines the frequency of the display refresh rate, or the frequency in which all the information of the display is re-drawn. The controller decodes the output of the horizontal counter module to generate the HS signal time. This counter may be used to locate the pixel of a given row. Also, the output for the module Vertical Counter that increases the HS pulse may be used to generate the VS output time and this counter may be used to locate any row [15].

## 3. Development of the Proposed Controller

The controller proposed in this work is also based on table 1. The main difference is that instead of using the times for each part, the counting is carried out by the number of lines. The resolution values used for 640x480 are shown on table 2.

Table 2.Used resolution values for the proposed controller for a 640x480.

| Symbol | Parameter | Vertical Sync Lines | Horizontal Sync Lines |
|---|---|---|---|
| $T_S$ | Sync Pulse | 521 | 800 |
| $T_{DISP}$ | Display Time | 480 | 640 |
| $T_{PW}$ | Pulse Width | 2 | 96 |
| $T_{FP}$ | Front porch | 10 | 16 |
| $T_{BP}$ | Back porch | 29 | 48 |

The VGA Driver architecture is obtained following the methodology TOP-DOWN [17]. Fig. 4 presents a TOP-DOWN methodology diagram, where a 1st level design (main entity) is presented, the second level consists in specify the main entity components; in this case, every module don't have internal modules, in last level we find the module description of operation.

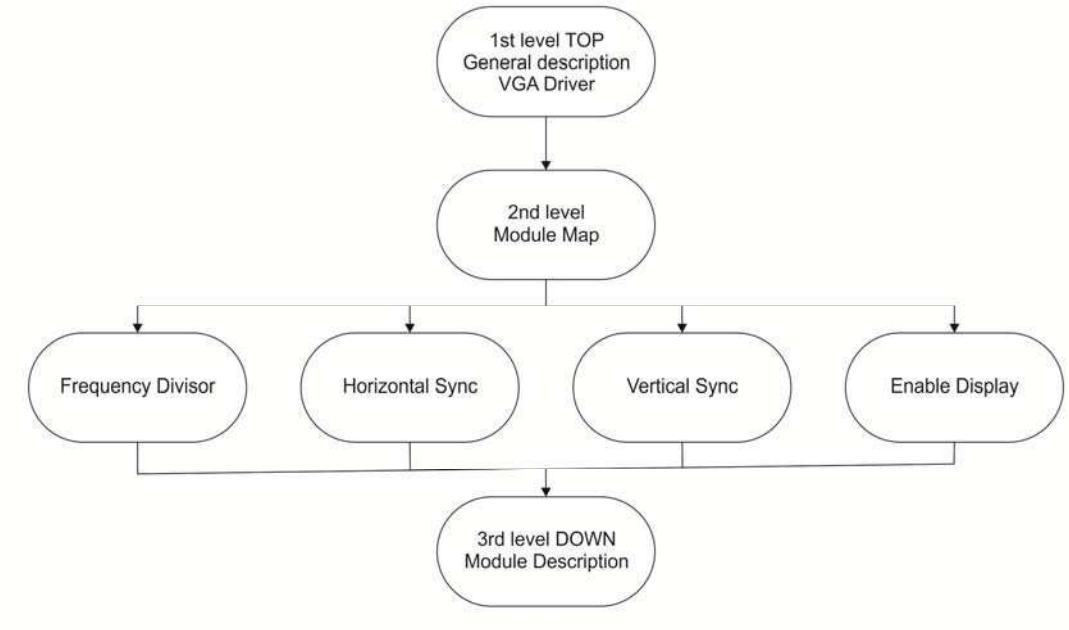



Figure 4. Design with TOP-DOWN Methodology.

Therefore, the general block diagram (called "entity") is shown on Fig 5.

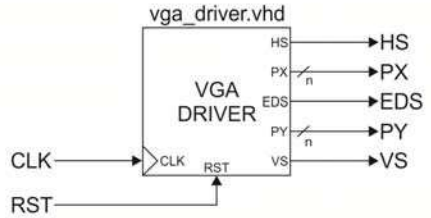



Fig. 5. General VGA sync controller diagram with clock frequency of 25MHz.

For the Fig. 5, the principal signals on the entity are shown in table 3, including a brief description.

Table 3. Principal System Signals

| Signal | Description |
|---|---|
| CLK | Main Clock signal coming from a 50MHz clock from the developing board Nexys 2. |
| RST | System´smainreset |
| HS | Horizontal synchronization signal |

| | |
|---|---|
| PX | Indicates the actual position from the horizontal line |
| EDS | Enables the actual pixel display on the monitor, the pixel is displayed when EDS is on high state. |
| PY | Indicates the actual pixel position on the vertical line |
| VS | Vertical synchronization signal |
| N | Indicates the bit size of the signal. This is set according to the resolution specifications. For a 640x480 pixel resolution, it requires a zise 10 data bus. |

The internal architecture of the VGA controller, consists of four modules to generate the output signals shown on Fig. 5, to get a successful synchronization with the monitor. The interconnections with the other modules (Frecuency Divisor, Horizontal Sync, Vertical Sync and Enable Display) are shown on Fig. 6.

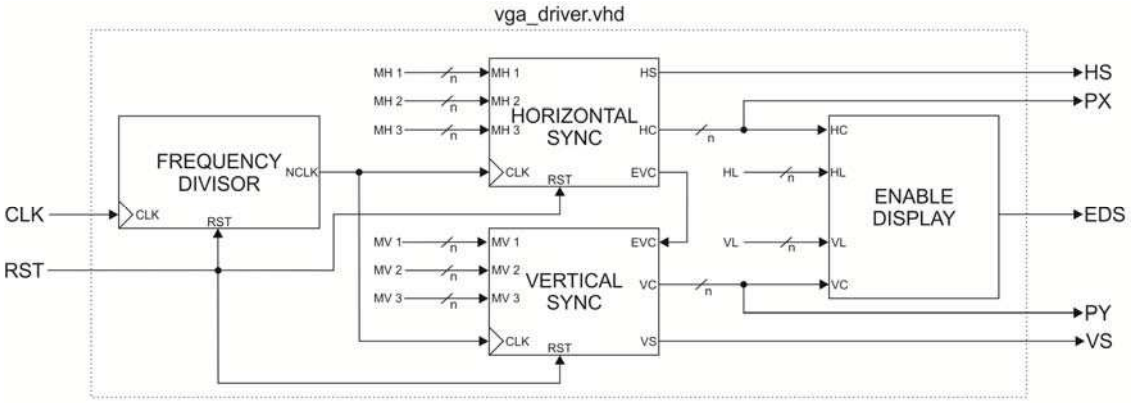



Fig. 6. Component Interconnection of the VGA Driver.

The module called Frequency Divisor generates a frequency of 25MHz to work with a 640x640 resolution. The output signal NCLK corresponds to the 25MHz signal, which is received by the modules Horizontal_Sync and Vertical_Sync to generate both sync signals needed for monitor display process. This module description is shown on Fig. 7.

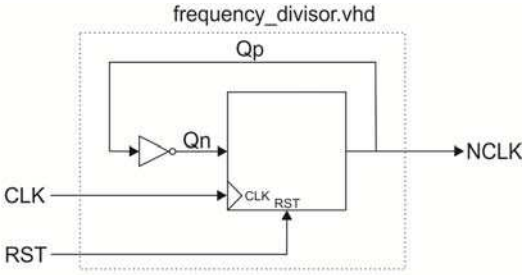



Fig. 7. Frequency Divider Module.

The 25MHz signal, NCLK, is connected to the ***vertical_sync.vhd*** y ***horizontal_sync.vhd*** modules such as the clock signal. The modules used to generate the horizontal sync long with its signals are shown on Fig. 8.

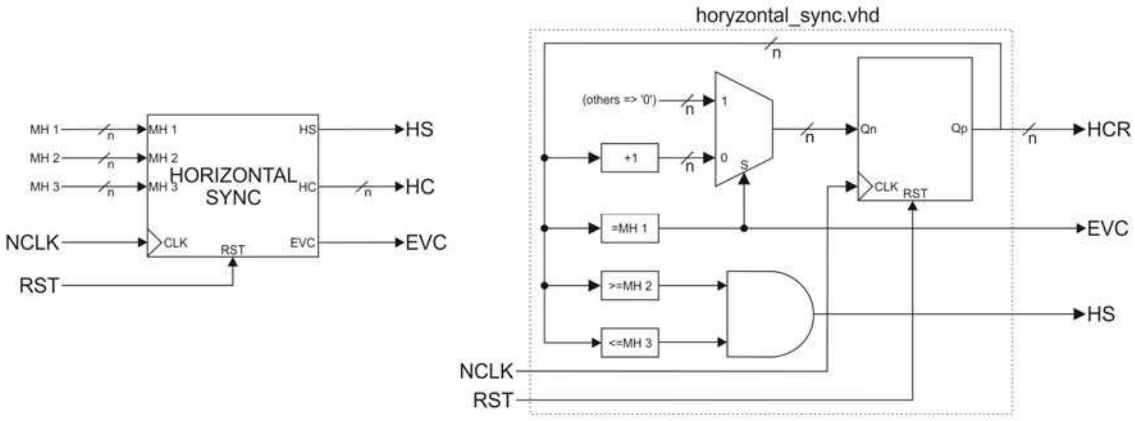



Fig. 8. (a) Entity defined for the Horizontal_sync module; (b) Architecture of the Horizontal_sync module.

The signals **MH1**, **MH2**, **MH3** y **HC** are data buses of size n and their values are obtained from equations 1-4.

$$MH1 = T_S - 1 \quad (1)$$
$$MH2 = T_{DISP} + T_{FP} - 1 \quad (2)$$
$$MH3 = T_{DISP} + T_{FP} + T_{PW} - 1 \quad (3)$$
$$HC = T_{DISP} \quad (4)$$

Fig. 8 (b) shows the internal connection from the module **Horizontal Sync,** where a 2-1 multiplexor is used. The signal **S** indicates whether the value is incremented or reset to 0. Afterwards, the output is sent to a D Flip-Flop (FF) synchronized to 25HMz. This FF is used to store the actual data until the following data is received. The signal **HC** indicates the horizontal position of the actual refresh position. The signal **EVC**, indicates the moment in which the count reached its peak value (**MH1**) and enables the count for module V**ertical Sync.** The signal **HS** allows the horizontal sync with the monitor interface when its value is high. This value is generated while the current count is greater or equal than **MH2** and **MH3**.

Fig. 9 shows the module description that generates the vertical sync and its corresponding signals.

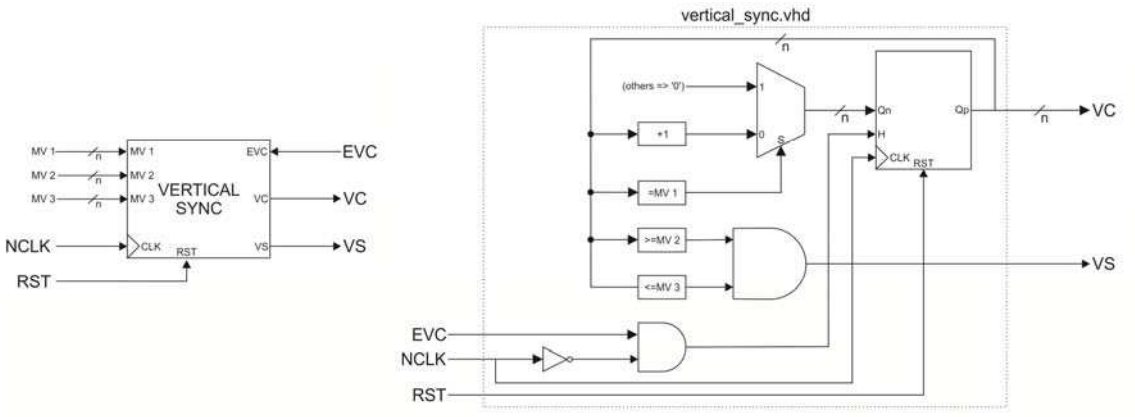



Fig. 9. (a) Vertical Sync Description; (b) Vertical Sync Internal Description.

The signals **MV1, MV2, MV3** and **VC** are data buses of size **n** and its values are obtained with respect to the indications of table 2 from the vertical sync, its values are obtained from a similar way to the equations 1-4 previously mentioned. These calculations are shown on equations 5-8.

$$MV1 = T_S - 1 \quad (5)$$
$$MV2 = T_{DISP} + T_{FP} - 1 \quad (6)$$
$$MV3 = T_{DISP} + T_{FP} + T_{PW} - 1 \quad (7)$$
$$VC = T_{DISP} \quad (8)$$

In Fig. 9 (b) the internal wiring of **Vertical Sync** is shown. It is observed that the descriptions is similar to Fig. 8 (b), with the difference that the register where the actual count is stored, requires an activate signal **H** generated with the signal **EVC** and the signal **NCLK** complemented. This means that when its value is high, the value of the signal **VC** is updated increase by 1 or reset to 0. The sync signal **VS** is generated while the value of the count **VC** is less or equal than M**V2** or greater or equal than **MV3**.

In Fig. 10, the comparator module that generates the signal **EDS** is observed. This module indicates the moment in which the current pixel will be displayed on the monitor in the case it is situated within the resolution range of 640x480 pixels.

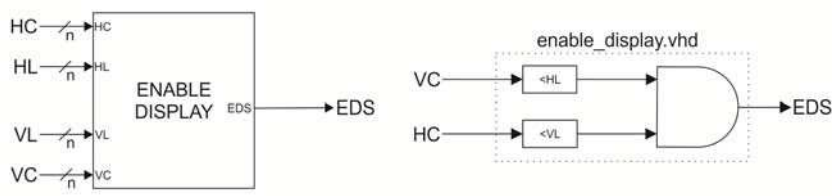



Fig. 10. Enable pixel component.

## 4. Testing and Results

To test the proposed driver, an 8 basic RGB colors display were carried out. Table 3 shows the color code from each of the development boards mentioned previously [13 – 16].

Table 3. Color code from each of the development boards used in the test.

| Spartan 3 3 bit RGB | | | Spartan 3AN 12 bit RGB | | | Nexys 2 8 bit RGB | | | Basys 2 8 Bit RGB | | | Color |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | G | B | R | G | B | R | G | B | R | G | B | |
| 0 | 0 | 0 | 0000 | 0000 | 0000 | 000 | 000 | 00 | 000 | 000 | 00 | Black |
| 0 | 0 | 1 | 0000 | 0000 | 1111 | 000 | 000 | 11 | 000 | 000 | 11 | Blue |
| 0 | 1 | 0 | 0000 | 1111 | 0000 | 000 | 111 | 00 | 000 | 111 | 00 | Green |
| 0 | 1 | 1 | 0000 | 1111 | 1111 | 000 | 111 | 11 | 000 | 111 | 11 | Cyan |
| 1 | 0 | 0 | 1111 | 0000 | 0000 | 111 | 000 | 00 | 111 | 000 | 00 | Red |
| 1 | 0 | 1 | 1111 | 0000 | 1111 | 111 | 000 | 11 | 111 | 000 | 11 | Magenta |
| 1 | 1 | 0 | 1111 | 1111 | 0000 | 111 | 111 | 00 | 111 | 111 | 00 | Yellow |
| 1 | 1 | 1 | 1111 | 1111 | 1111 | 111 | 111 | 11 | 111 | 111 | 11 | White |

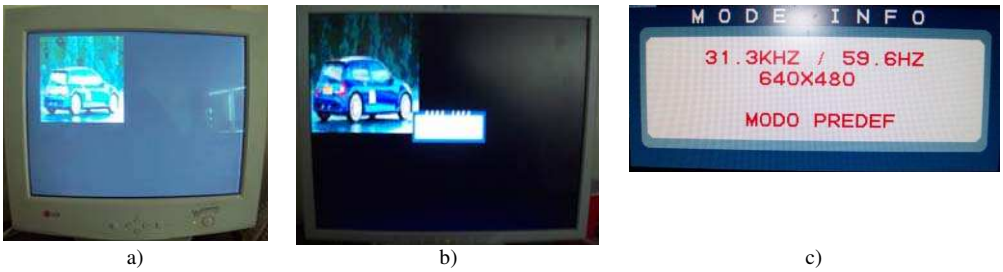
The colors displayed with the controller used in the test are shown on table 4. The monitor used is a VGA port CRT. However, the system can be connected to any VGA port LCD monitor.

Table 4. Example showing the monitor with the proposed driver.

Black Blue Cyan Green

Red Magenta Yellow White

Another example employed to test the performance of the proposed controller, consist in reading an image stored on an external RAM memory [19]. The result of this test is shown on Fig. 11.




a) b) c)

Fig. 11. Image stored on an external RAM memory and displayed on a monitor; (a) CRT Monitor, (b) LCD Monitor, (c) Information of the operation mode.

The Fig. 12 shows a time diagram, obtained by the hardware simulation using Active-HDL, it shows the timing of each main signal on VGA Driver in one second. On Fig. 12 (a) VS signal is in “high” when the vertical count finished and restarts to 0, one pulse on high of VS signal is generated in 16.7 ms, it means that in one second, we have 60 screens displayed on monitor, like it see in Fig 11 (c). Fig. 12 (b) shows the cycle time to generate every horizontal synchronous (HS signal) each 26.21 µs.

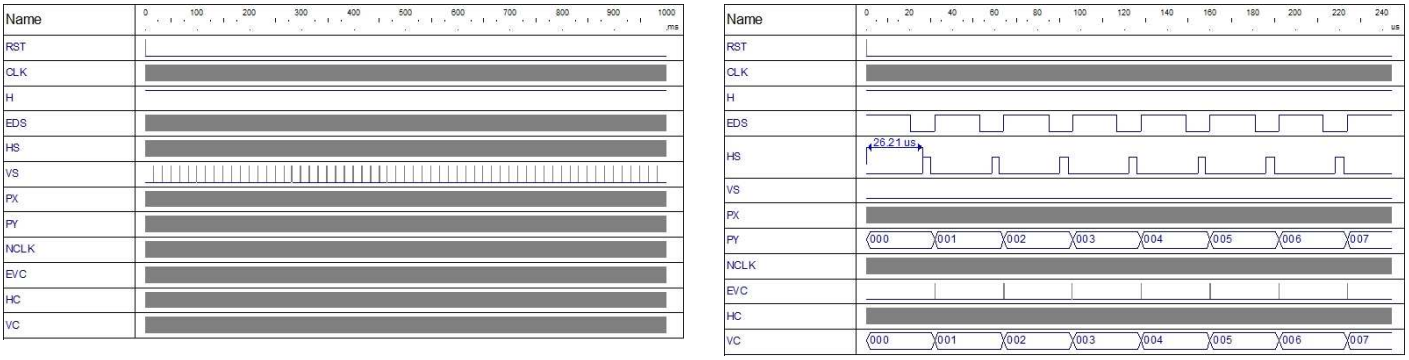



Figure 12. VGA Driver Simulation using Ative-HDL, (a) VS Signal Timing, (b) HS Signal Timing.

The proposed driver allows modification to other resolutions which are shown on table 4. The video controller proposed in this work can be used for several video resolutions [20]. To do that, a change for MH1, MH2, MH3, HC, MV1, MV2, MV3 and VC are needed, according with the specifications of the resolution. A limitation for the video controller is the clock frequency, because with higher resolution, a higher clock frequency is needed. The user has to compute the value for the frequency divider and get the frequency needed. If the system has the exact frequency needed, the Frequency Divider module can be omitted. In figure 10 it is observed a good synchrony and display of an image in the monitor regardless the type of monitor (CRT or LCD), even though the projection scheme is different. Also, the operation mode indicates the resolution and the current frequency of the monitor, whose resolution is 640x480 with a refresh rate of 59.6Hz., the horizontal path per column last 31.97µs, whereas the total path of the monitor is16.64ms.

Table 4. VGA resolutions parameters.

| SVGA 800x600 72Hz 50MHz | | | | | | |
|---|---|---|---|---|---|---|
| Symbol | Parameter | Vertical Sync | | | Horizontal Sync | |
| | | Time | Clocks | Lines | Time | Lines |
| $T_S$ | Sync Pulse | 13.85 ms | 692, 500 | 666 | 20.79 µs | 1040 |
| $T_{DISP}$ | Display Time | 12.5 ms | 625, 000 | 600 | 16 µs | 800 |
| $T_{PW}$ | Pulse Width | 124.8 µs | 6, 240 | 6 | 2.39 µs | 120 |
| $T_{FP}$ | Front porch | 769.6 µs | 38, 480 | 37 | 1.27µs | 64 |
| $T_{BP}$ | Back porch | 478.4 µs | 23, 920 | 23 | 1.11 µs | 56 |
| XGA 1024x768 60Hz 65MHz | | | | | | |
| $T_S$ | Sync Pulse | 16.66 ms | 1, 082, 900 | 806 | 20.67 µs | 1344 |
| $T_{DISP}$ | Display Time | 15.87 ms | 1, 031, 550 | 768 | 15.75 µs | 1024 |
| $T_{PW}$ | Pulse Width | 124.06 µs | 8, 064 | 6 | 2.09 µs | 136 |
| $T_{FP}$ | Front porch | 62.03 µs | 4, 032 | 3 | 0.36 µs | 24 |
| $T_{BP}$ | Back porch | 599.63 µs | 38, 976 | 29 | 2.46 µs | 160 |

## 5. Conclusions

The contribution of this work is an open architecture and standardization of a video VGA controller with all the required signals for correct display and performance and the architecture of how to obtain each signal. This architecture may be used in any FPGA device regardless of the brand or model and is scalable to any resolution such as: SVGA, XGA or WUXGA. Furthermore, this architecture minimizes the developing time of a controller according to the datasheet, with respect to the design and planning of the controller. Also, a worth noticing that there are IP cores created by each FPGA maker. Nevertheless, to use this modules or libraries requires an extra cost and are no portable to every FPGA.

The block diagram used in this work can help to anybody who wants to implement a VGA driver to get a successful result. Major of designs provided in the open literature only shows the code in VHDL, Verilog, etc, but a graphical description is not presented. The Block Diagram Hardware Description presented in this work, let us to implement the driver VGA controller easily with any hardware description language.

## References

[1] Dubey Rahul, Introduction to Embedded System Design Using Field Programmable Gate Arrays, Springer-Verlag London Limited, ISBN: 978-1-84882-015-9, Londres, 2009.

[2] Ramos Arreguín Carlos Alberto, Cora Gallardo Orlando Marcos, Ramos Arreguín Juan Manuel, Pedraza Ortega Jesús Carlos, Canchola Magdaleno Sandra Luz, Vargas Soto José Emilio, Metodología para Manejo de Imágenes en FPGA, 6º Congreso Internacional de Ingeniería, pp. 219-226, ISBN: 978-607-7740-39-1, Querétaro, Qro., Abril 2010

[3] Quintero M. Alexander, Vallejo R. Eric, Image Processing Algorithms using FPGA, Revista Colombiana de Tecnologías de Avanzada, Vol. 1; No. 7; pp. 11-16, ISSN: 1692-7257, 2006.

[4] Bravo Muñoz Ignacio, Arquitectura basada en FPGA para la Detección de Objetos en Movimiento, utilizando Visión Computacional y Técnicas PCA, Tesis Doctoral, Departamento de Electrónica de la Escuela Politécnica de la Universidad de Alcalá, España 2007.

[5] A. Castillo, J. Vázquez, J. Ortegón, C. Rodríguez, Prácticas de Laboratorio para Estudiantes de Ingeniería con FPGA, IEEE Latin America Transactions, pp. 130-136, Junio 2008.

[6] Ramos Arreguín Carlos Alberto, Moya Morales Juan Carlos, Ramos Arreguín Juan Manuel, Pedraza Ortega Jesús Carlos, Metodología de una Etapa Básica de un Sistema de Procesamiento de Imágenes basado en FPGA, 9º Congreso Nacional de Mecatrónica, pp. 235-240, ISBN: 978-607-95347-2-1, Octubre 2010.

[7] Kalomiros J. A., Lygouras J., Design and evaluation of a hardware/software FPGA-based system for fast imageprocessing, Microprocessors and Microsystems, Elsevier, pp. 95– pp. 106, doi:10.1016/j.micpro.2007.09.001.

[8] ChaikalisD., Sgouros N. P., Maroulis D., A Real Time FPGAArchitecture for 3D reconstruction from integral images,Journal of Visual Communication

& Image Representation,Elsevier, pp. 9 - pp. 16, doi:10.1016/j.jvcir.2009.09.004.

[9] SiélerL., Tanougast C., Bouridane A., A scalable andembedded FPGA architecture for efficient computation of graylevel co-ocurrence matrices and Haralick textures features,Microprocessors and Microsystems, Elsevier, pp. 14 - pp. 24,doi:10.1016/j.micpro.2009.11.001.

[10] Krill B., Ahmad A., Amira A., Rabah H., An efficient FPGAbaseddynamic partial reconfiguration design flow andenvironment for image and signal processing IP cores, SignalProcessing: Image Communication, Elsevier, pp. 377 - pp. 387,doi:10.1016/j.image.2010.04.005.

[11] Satake Shin-ichi, SorimachiGaku, Masuda Nobuyuki, ItoTomoyoshi, Special-purpose computer for particle imagevelocimetry, Computer Physics Communications, Elsevier, pp.1178 – pp. 1182, doi:10.1016/j.cpc.2011.01.022.

[12] Pong P. Chu, FPGA Prototyping by VHDL Examples XilinxSpartan 3 Version, Wiley Interscience, pag. 257 – pag. 266, ISBN: 978-0-470-18531-5, USA 2008.

[13] Xilinx, Spartan-3 FPGA Starter Kit Board User Guide, www.xilinx.com.

[14] Xilinx, Spartan-3A/3AN FPGA Starter Kit Board User Guide, www.xilinx.com.

[15] DigilentNexys 2 Board Reference Manual, www.digilentinc.com.

[16] DigilentBasys 2 Board Reference Manuel, www.digilentinc.com.

[17] Romero Troncoso René de Jesús; Electrónica Digital y Lógica Programable; Universidad de Guanajuato, 2ª. Edición; ISBN: 968-864-449-8; Guanajuato México, 2007.

[18] Marco Antonio Aceves Fernández, Juan Manuel Ramos Arreguín; Fundamentos de Sistemas Embebidos, Mediante Lenguajes Descriptivos de Hardware; Asociación Mexicana de Mecatrónica A. C.; ISBN: 978-607-95347-4-5; Santiago de Querétaro, Qro., México, 2011.

[19] C. A. Ramos Arreguín, J. C. Moya Morales, J. M. Ramos Arreguín, J. C. Pedraza Ortega, M. A. Aceves Fernández, J. E. Vargas Soto, S. Tovar Arriaga; Metodología para almacenamiento de imágenes en Memorias externas de tipo RAM, empleando FPGA; IX Congreso sobre Innovación y Desarrollo Tecnológico, CIINDET 2011; ISBN: 978-607-95255-3-8; Cuernavaca, Morelos, México, 2011.

[20] http://tinyvga.com/vga-timing, last consult 12/08/11.