
**INSTITUTO TECNOLÓGICO DE TUXTLA GUTIÉRREZ**


# Sistema de visión implementado en FPGA para el seguimiento de una trayectoria definida

**Autor:**
José Adrian Pérez Cueto

**Directores:**
Dr. Madaín Pérez Patricio
Dr.Héctor Hernández de león

**Residencia profesional**

Agosto-Diciembre-2009

# RESUMEN

En esta residencia se plantea una propuesta para la detección o seguimiento de trayectoria u objetos dentro de una escena, mediante una plataforma de propósito específico construida en base a una arquitectura de tipo FPGA (*Field Programmble Gate Array*). Este dispositivo además de encargarse de la captura y análisis de las imágenes en escala de grises procedente de un sensor CMOS, de resolución 1.3M píxeles, realiza la detección o seguimiento de trayectoria aplicando filtros detección de bordes, además a eso se envía la imagen segmentada a una pantalla TFT-LCD, para la visualización y corregimiento de errores de los resultados obtenidos.

Los problemas más importantes que se plantean en una solución como la descrita en esta residencia son los relacionados con la implementación de la visión artificial de forma integral en una FPGA. El elevado número de operaciones que demanda las técnicas de visión conlleva elevados tiempos de ejecución si se utilizan plataformas de procesamiento convencionales, normalmente de tipo secuencial. Por esta razón se ha desencadenado la adaptación del procesamiento de imágenes digitales para hardware reconfigurable. Para ello, se ha paralelizado la ejecución de las diferentes fases que forman el procesamiento digital de imágenes: Captura de la imagen, Preprocesado, Segmentación, Medición e Interpretación. Con la solución propuesta e implementada en esta residencia se ha conseguido un sistema segmentado que alcanza un alto ratio de imágenes procesadas por segundo (aproximadamente 30 imágenes/sg).

Todo el diseño implementado en la FPGA ha sido codificado en Verilog HDL, sin emplear ningún *core* comercial. Esto permite la portabilidad del sistema diseñado a cualquier FPGA.

La plataforma hardware desarrollada en esta residencia está basada en un sensor CMOS de alta velocidad (hasta 30 imágenes/segundos con la máxima resolución) cuyo interfaz de control se realiza en el FPGA. Además la plataforma dispone de un banco de memoria externa de tipo SDRAM, que es gestionado desde la misma FPGA, permitiendo almacenar en él las imágenes capturadas, procesadas o datos temporales y por último la visualización de los resultados se realiza utilizando un panel TFT-LCD que se controla desde el mismo FPGA. Estos periféricos dotan al sistema de una alta versatilidad y flexibilidad, con lo que su empleo en diferentes algoritmos de visión para FPGAs es factible. En nuestro caso se utiliza un FPGA Cyclone II el cual viene integrada en el kit DE2 de Altera.

# ÍNDICE

# ÍNDICE DE FIGURAS

## ÍNDICE DE TABLAS

# 1 Planteamiento del Problema

## 1.1 INTRODUCCIÓN

Uno de los principales objetivos de las aplicaciones de visión artificial, es la descripción de forma automática de una determinada escena. Así, se entiende como descripción, a la identificación y localización de los objetos que la forman en base a sus características (Ratha & Jain, 1999). Frecuentemente, el principal problema de las aplicaciones de visión artificial es el tiempo de ejecución de los algoritmos y el coste de los equipos. La mayor demanda de prestaciones de los algoritmos de procesamiento de imágenes, unido a la mayor resolución espacial, hace que cada vez sean mayores las demandas de carga computacional. Si además se desea trabajar en tiempo real, las exigencias de los tiempos de ejecución de los algoritmos son aún más críticas.

Habitualmente, las plataformas elegidas para realizar estos algoritmos son las basadas en programas secuenciales. Sin embargo, éstas no son las óptimas en muchas aplicaciones desde un punto de vista de rendimiento. Por tanto, se justifica la búsqueda de nuevos sistemas segmentados para procesamiento de imágenes en paralelo. Así, según (Hamid, 1994), en base a las características de computación, la mayoría de los sistemas de visión se pueden dividir en tres niveles tal y como se muestra en la Figura 1.1: en el nivel inferior (bajo nivel), se incluyen operaciones a nivel de píxel, tales como filtrado o detección de bordes. Este nivel se caracteriza por manejar un elevado número de datos (píxeles) y de operaciones sencillas, como sumas y multiplicaciones. El procesado de nivel medio, se caracteriza por realizar operaciones más complejas de bloques de píxeles. Dentro de este nivel se pueden incluir operaciones de segmentación o etiquetado de regiones. Por último, las tareas pertenecientes al nivel alto, se caracterizan por una alta complejidad en la algoritmia. Se puede observar en la Figura 1.1, cómo a medida que crece la altura de la pirámide, el volumen de datos a manejar disminuye, incrementándose la complejidad computacional de los algoritmos. La Tabla 1.1 muestra un resumen de las características de cada nivel (Ratha & Jain, 1999).
Por tanto, es importante seleccionar el sistema hardware necesario, en función del nivel de complejidad de procesamiento requerido. Esto permitirá explotar al máximo el rendimiento del sistema.

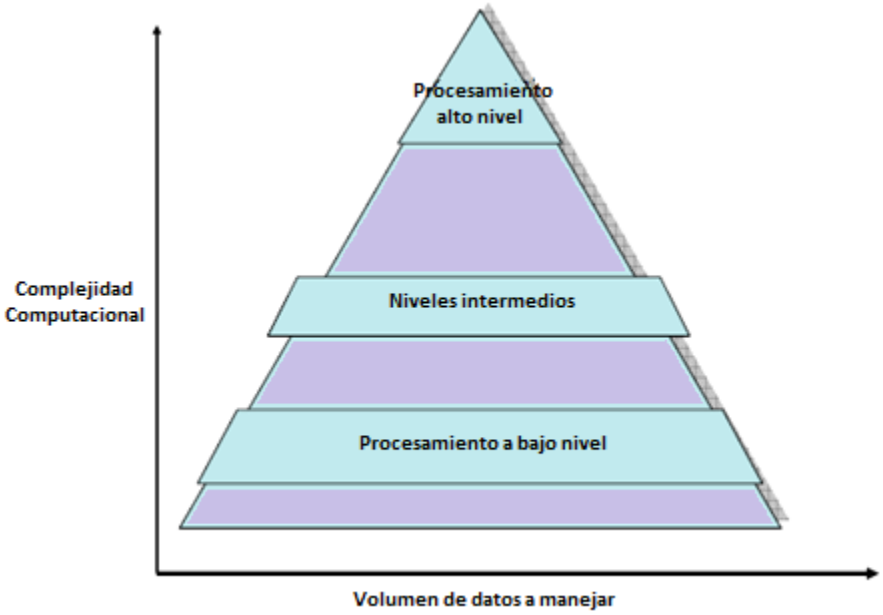



**Figura 1.1**.Pirámide ilustrativa del volumen de datos vs. Complejidad computacional del procesamiento de imágenes **(Hamid, 1994)**.

**Tabla 1.1.**Características y ejemplos de los diferentes niveles de complejidad computacional

| Nivel | Características | Ejemplos |
|---|---|---|
| **Bajo** | Accesos a píxeles vecinos cercanos, operaciones simples, manejo de gran cantidad de datos | Detección de bordes, filtrado, operaciones morfológicas simples |
| **Medio** | Acceso a bloques de píxeles, operaciones más complejas | Transformada de Hough, conectividad. |
| **Alto** | Acceso aleatorio, operaciones complejas | Matching |

El procesado digital de imágenes requiere habitualmente el manejo de un alto número de operaciones a nivel de bit que se desean ejecutar en el menor tiempo posible. Debido la arquitectura secuencial de los ordenadores convencionales, no se pueden ejecutar concurrentemente muchas operaciones. Por ello, cuando el número de datos a manejar es grande, el rendimiento del sistema es muy bajo. Sin embargo, una plataforma hardware de propósito específico (diseñada expresamente para una aplicación), puede dar excelentes resultados. Así, operaciones relativamente sencillas como filtros, descompresión de imágenes, etc., se realizan con un elevado rendimiento. Por contra, si se desea implementar algoritmos más complejos, éstos hay que reformularlos para explotar las características de la plataforma sobre la que se ejecutará.

## 1.2 DEFINICIÓN DEL PROBLEMA

Existen diversas aplicaciones hechas por computadora que aplican la visión artificial, un ejemplo cercano, son los robots autónomos o sistemas de control inteligentes. Estas aplicaciones por lo general realizan sus tareas con mucha latencia, debido a que los algoritmos que utiliza la visión artificial son muy complejos y procesados en computadoras secuenciales. El proceso de captura, envió de la imagen y procesado de imagen en una computadora secuencial, produce mucho tiempo para que el sistema tome la decisión final. Los algoritmos de visión requieren muchos recursos de arquitectura para su procesamiento, por lo que, una computadora ordinaria no lo es suficiente. El principal problema de las aplicaciones de visión artificial es el tiempo de ejecución de los algoritmos y el coste de los equipos. La mayor demanda de prestaciones de los algoritmos de procesamiento de imágenes, unido a la mayor resolución espacial, hace que cada vez sean mayores las demandas de carga computacional. Si además se desea trabajar en tiempo real, las exigencias de los tiempos de ejecución de los algoritmos son aún más críticas.

En este trabajo se utiliza una arquitectura especializada que cumpla con las demandas de la visión artificial. Una de las arquitecturas muy potentes en la actualidad son las FPGAs. Una FPGA es un dispositivo programable que consiste en arreglos múltiples de celdas lógicas. Dada la gran densidad de compuertas y registros con las que cuenta un FPGA, es posible diseñar circuitos que realicen funciones matemáticas complejas de manera rápida o veloz, ya que todos sus procesos los ejecuta en paralelismo. Entre las aplicaciones que nos interesa esta el procesamiento o tratamiento de imágenes digitales en tiempos relativamente cortos.

## 1.3 ESTADO DEL ARTE

### 1.3.1 Procesamiento de imágenes en FPGA

Desde la aparición a finales de la década de los ochenta de las FPGAs, su uso en el procesado de imágenes ha ido incrementándose continuamente. Inicialmente, los primeros trabajos de procesamiento con FPGAs usaban como plataforma una máquina genérica (*custom machine*) (Arnold, Buell, & Davis, 1992) (Ziegler, Byoungro, Hall, & Diniz, 2002) (Piacentino, Van der Wal, & Hansen, 1999). De manera simultánea, otros trabajos como los expuestos en (Hamid, 1994) (Wiatr, 1998), comenzaban a explotar las características de estos dispositivos en algoritmos sencillos de preprocesamiento de imágenes.

Algoritmos simples, como filtrados o convoluciones, eran aplicaciones habituales de estos sistemas. Algunos ejemplos de este tipo de operaciones se muestran en (Hamid, 1994), donde se realizan con FPGAs operaciones de medias aritméticas sobre imágenes de resolución espacial de 256 x 256. Otro ejemplo, es el mostrado en (Wiatr, 1998). En este caso, la tarjeta con FPGAs se introducía como un elemento de pre-procesado (media, convolución o filtrado) dentro de un sistema completo de adquisición y procesamiento de imágenes.

En (Jamro & Wiatr, 2001) se muestra el diseño de un hardware multiplicador implementado en una FPGA para realizar operaciones de convoluciones sobre imágenes. Dentro de esta línea existen numerosos trabajos como los mostrados en (Dawood, Visser, & Willianms, 2002) (Draper, Beveridge, Willem Böhm, Ross, & Chawathe, 2003) (Kessal, Abel, & Demigny, 2003), donde se realizan aplicaciones de máscaras de visión (normalmente de 3x3), sobre imágenes monocromáticas de resolución espacial 256x256.

(Quintero & Vallejo, 2006) Realizaron un trabajo que tiene por objeto, mostrar las metodologías empleadas en una FPGA para realizar el procesamiento de imágenes y el reconocimiento de objetos dentro de dichas imágenes. El resultado final es un procesador de imágenes cuya principal limitante son los algo ritmos iterativos de etiquetado e identificación de objetos. En donde realizaron un filtrado binario utilizando una maquina de estado, que analiza la vecindad de un píxel en una ventana de 3x3. La máquina de estados tiene como entrada un *bit*, correspondiente al píxel de la ventana, y de salida 8 bits correspondientes al valor del píxel después del filtrado. Si el numero de píxeles blancos es mayor o igual a 6 el píxel hace parte del objeto y debe ser blanco (255d) sino hace parte del fondo y debe ser negro (0d), el algoritmo empleado que tuvo un desempeño de 110 *fps*.

El trabajo desarrollador por (Tanvir & Mohd, 2007) presenta una arquitectura basada en FPGA para la detección de bordes Sobel. Esta arquitectura diseñada tiene como entrada un decodificador de pixel, una entrada de umbral y la señal de reloj. El decodificador recibe los datos que son enviados por la PC, la salida de este están conectados a 4 registros. La salida del primer con el segundo registro se le saca un promedio, al igual con los otros dos. El resultado de las dos operaciones, se le vuelve a sacar un promedio, el resultado de este se compara con la entrada del umbral, obteniendo así una imagen con bordes blanco y fondo negro.

### 1.3.2 Seguimiento de trayectoria

A partir de años recientes se empezó a crear robots seguidores de trayectoria, con la finalidad de trasportar cargas en algunas zonas industriales. Maquinas inteligentes con la capacidad de realizar cualquier tarea dentro de un espacio a través de una trayectoria conocida.

En el trabajo desarrollado por (Payá, Reinoso, Gil, Pedrero, & Juliá, 2007) presentan un método basado en apariencia aplicado al seguimiento de rutas en sistemas multi-robot, usando la información capturada por una cámara convencional y sin calibrar. En la etapa de aprendizaje, la información más relevante a lo largo de la ruta es almacenada en una base de datos creada mediante técnicas PCA incremental. Gracias a esta técnica, el robot que sigue la ruta puede comenzar la misma mientras el líder aun la está grabando. Durante la navegación, el robot seguidor lleva a cabo en primer lugar, un proceso de autolocalización, comparando la vista actual con la información almacenada en la base de datos y usando técnicas probabilísticas. Tras ello, se realiza la etapa de control, en la que utilizando un regulador difuso, se calcula su velocidad lineal y angular para seguir la ruta grabada. En todo momento se trabaja con la apariencia global de las imágenes, sin necesidad de extraer puntos característicos.

En (Cervera & Chiem, 2004) presenta un método sencillo para el despliegue de múltiples robots autónomos, estos robots se encargan de seguir a un robot líder a través de identificación de color. Cada robot seguidor toma de las estimaciones de la posición y orientación de su líder y construye una curva de Bézier que describe la trayectoria entre su posición actual, y de la posición del robot líder. Las trayectorias generadas son lo suficientemente exactos como para ampliar el enfoque a los robots seguidores, manteniendo al mismo tiempo una formación en línea.

## 1.4 JUSTIFICACIÓN

El proyecto se origina a través de la necesidad de crear robots autónomos inteligentes capaces de distinguir trayectorias, objetos o algo que impida el paso o el fin de este. Para llegar a crear un robot móvil inteligente se hace uso de la visión artificial utilizando una cámara que emulan la visión humana. Al emplear esta metodología el proyecto se somete a problemas de respuesta en el tiempo, debido a que los algoritmos son muy complejos y utiliza mucho recurso para su uso.

En general el tratamiento de imágenes digitales con aplicación de visión artificial se realiza con el uso de la computadora PC (Computadora Personal) bajo un sistema operativo de interfaz grafica.

Para la adquisición de imágenes se realiza a través de una cámara conectada vía USB, estas imágenes son almacenadas para ser leídas desde software. La CPU se encarga de procesar secuencialmente todas las tareas indicadas por el sistema operativo a través del software. En este caso, procesar las imágenes provenientes de la cámara aplicando visión artificial.

Se puede entender que en cada etapa la computadora pierde un tiempo determinado, desde la adquisición de imágenes hasta la visualización de imágenes sobre el monitor o pantalla. Si se toma en cuenta el tiempo de latencia, que existe cuando se ocupa todo el ancho de banda USB de los datos que son capturados por el sensor CCD de la cámara o si fuera más se le suma a ello cuando algunos deciden utilizar transmisiones de imágenes inalámbricamente, en el uso de la robótica. El tiempo en que el microprocesador se lleva en operar secuencialmente todas las tareas que le ordena el sistema operativo, desde las múltiples variables que existen del usuario al estar manipulando y trabajando sobre el sistema operativo bajo software; hasta la tarea de generar imágenes o video con respecto a estas variables. Ahora si a eso le sumamos el tiempo en que tarda de procesar todos los algoritmos matemáticos que utiliza la visión artificial para manipular las imágenes e interpretarlas, se logra a imaginar que todo este proceso se lleva un tiempo relativamente grande.

Debido a este problema, se llega a pensar que el empleo de la visión artificial no es para computadora personales (PC), pues, para su estudio se necesita de una arquitectura especializada. El error más grande que existe en este problema es la adquisición de imágenes, el sistema operativo y la CPU. Para la aplicación de sistema de visión artificial, no se necesita un sistema operativo o software para trabajar, esto va demás, todo estos algoritmos debe estar implementados desde el hardware y no desde software. Otro punto importante es el CPU. Para no ejecutar tareas secuenciales, este se remplaza por un FPGA, debido a que es una arquitectura que realiza tareas o procesos en paralelismo. Además tiene la ventaja diseñar cualquier hardware, en lugar de realizar un diseño de circuito electrónico tradicional. Logrando con el FPGA un ahorro económico considerable, además del tiempo de desarrollo e implementación.

Por último para la adquisición de imágenes se realiza por comunicación paralela, al igual que para el envió de la imagen a la pantalla LCD TFT. Logrando así agilizar todas las etapas y procesamiento de manera rápida y veloz.

## 1.5 OBJETIVOS

### 1.5.1 Objetivo general

Implementar algoritmos de visión por computadora en una arquitectura FPGA para diseñar sistemas autónomos de seguimiento de trayectorias cuya velocidad de respuesta sea en tiempo real.

### 1.5.2 Objetivo específicos

- Recaudar la información necesaria acerca del tema de visión por computadora.
- Estudiar el funcionamiento básico de la TV.
- Aprender el lenguaje de programación HDL Verilog.
- Controlar la cámara que utiliza la tarjeta DE2 de Altera.
- Almacenar las imágenes captadas por la cámara en una memoria SDRAM.
- Controlar la pantalla LCD TFT para mostrar resultados.
- Utilizar algoritmos de visión artificial a través de la imagen almacenada en memoria.
- Desarrollar un algoritmo capaz de reconocer una trayectoria definida a través de la imagen.
- Diseñar un robot móvil para la comprobación de los algoritmos desarrollados.
- Realizar pruebas de los algoritmos implementados.
- Corregimientos de fallas o errores cometidos.

## 1.6 DELIMITACIÓN DEL PROBLEMA

En el actual trabajo se pretende desarrollar un sistema autónomo inteligente utilizando como medio de percepción la visión artificial. El principal componente de estudio y de análisis será la cámara, ya que por medio de ella a través de imágenes que serán procesados en FPGA, se entenderá el universo que lo rodea. El sistema autónomo inteligente se enfocara a realizar tareas de seguimiento de diferentes trayectorias definidas. Estas trayectorias serán analizadas aplicando técnicas de filtros en la imagen como la binarización y la de Sobel. De tal manera que al aplicar estos filtros se eliminen en la imagen las partes que no son de interés (ej. El fondo), si no simplemente el objeto de estudio, en este caso será la línea a seguir. A través de la

implementación de filtros, se calcula la centroide del objeto (línea) con la finalidad de que el sistema trate de recuperar su centro sobre la línea, así el sistema consiga seguir una determinada trayectoria. Además a ello se hará un cálculo de la curvatura o la pendiente de la línea, empleando técnicas de triangulación, con el objetivo de regular la velocidad del sistema según el ángulo de la curvatura.

En general el trabajo se basa en el procesamiento de imagen en FPGA, la toma de decisiones y por último la visualización de resultados en una pantalla TFT-LCD. Para la comprobación del sistema se desarrolla un robot móvil con ruedas. Para la implementación del sistema y la incrementación de la velocidad del procesamiento de imagen, se utiliza la tarjeta DE2 Cyclone II de Altera. Este kit cuenta con una cámara y pantalla TFT-LCD, que hacen accesible para trabajar con sistema de adquisición de imágenes de manera fácil y desplegar resultados sobre la pantalla, sin la necesidad de utilizar la computadora (PC) como medio, si no tener un microcomputadora que sea especializada a la visión artificial.

## 1.7 ALCANCES Y LIMITACIONES

### 1.7.1 Alcances

Esta investigación solo pretende implementar algoritmos de visión artificial a un FPGA debido a que esta línea de investigación es nueva. Posteriormente se dará una aplicación a la cual se eligió el estudio y análisis de la línea o trayectoria para optimizar el funcionamiento de los sistemas autónomos con la opción de poder acelerar o parar si existe algún cambio en la línea. Además en un futuro se pretende que los sistemas autónomos aprendan durante la experiencia o situaciones a la que se exponen constantemente, si necesidad de tener la misma respuesta al mismo caso presentado.

### 1.7.2 Limitaciones

Las limitaciones de este sistema, al trabajar con sistemas de visión artificial, se someten a problemas de luminosidad sobre el espacio donde se realiza. Debido que los valores RGB de la escena cambian con respecto a las condiciones de luz. Por consecuencia los algoritmos no funcionan de manera correcta, provocando errores en el sistema. Para su aplicación, la luminosidad debe estar en condiciones totalmente controlada. En la implementación y comprobación del sistema, debe trabajar en un área totalmente limpia y la línea debe ser continua.

## 1.8 CARACTERIZACIÓN DEL ÁREA EN QUE SE PARTICIPO

El desarrollo del presente trabajo de residencia profesional fue desarrollado en el Área de Posgrado del Instituto Tecnológico de Tuxtla Gutiérrez (Figura 1.2).

El Instituto Tecnológico de Tuxtla Gutiérrez es una institución pública dependiente de la Secretaría de Educación Pública. Imparte 9 licenciaturas y 2 programas de posgrado.


**Figura 1.2.**Instituto Tecnológico de Tuxtla Gutiérrez

### 1.8.1 Misión, visión y valores de la institución

*Misión*: Formar de manera integral profesionistas de excelencia en el campo de la ciencia y la tecnología con actitud emprendedora, respeto al medio ambiente y apego a los valores éticos.

*Visión:* Ser una Institución de excelencia en la educación superior tecnológica del Sureste, comprometida con el desarrollo socioeconómico sustentable de la región.

*Valores:* El ser humano, el espíritu de servicio, el liderazgo, el trabajo en equipo, la calidad y el alto desempeño.

## 1.8.2 Ubicación geográfica del área

### 1.8.2.1 Macrolocalización

El Instituto Tecnológico de Tuxtla Gutiérrez se encuentra localizado en el país Estados Unidos Mexicanos, específicamente en la Ciudad llamada Tuxtla Gutiérrez capital política del estado de Chiapas al sureste del país antes mencionado. La Figura 1.3 muestra la ubicación de Tuxtla Gutiérrez.

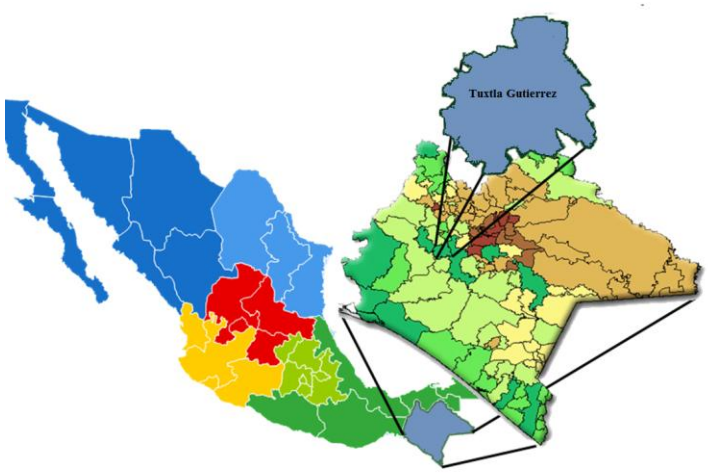



**Figura 1.3.**Tuxtla Gutiérrez, Chiapas

### 1.8.2.2 Macrolocalización

Las instalaciones del Instituto Tecnológico de Tuxtla Gutiérrez se ubican dentro de la ciudad de Tuxtla Gutiérrez, Chiapas. La dirección de ésta institución educativa es Carretera Panamericana Km. 1080. Tuxtla Gutiérrez, Chiapas, México. C. P. 29000, Apartado Postal 599 (Ver Figura 1.4).


**Figura 1.4.** Instituto Tecnológico De Tuxtla Gutiérrez

## 1.9 INFRAESTRUCTURA DE EQUIPO DE CÓMPUTO

El presente proyecto de residencia profesional fue desarrollado con el hardware y software que se menciona a continuación.

**Hardware:**
Computadora portátil son las siguientes características: Procesador Turión x 2 (1.66 GHz), Memoria de 2,048 MB DDR II, Disco Duro de 160GB, Pantalla de 15.1", DVD±RW, Red Inalámbrica 802.11bg.
Tarjeta DE2 FPGA Cyclone II de Altera
.
**Software:** El software utilizado para el desarrollo fue Quartus II 9.1 Edición Web bajo el Sistema Operativo Windows XP Service Pack 3.

# 2 Fundamentos Teóricos

## 2.1 VISIÓN ARTIFICIAL

La visión es uno de los mecanismos sensoriales de percepción más importantes en el ser humano aunque evidentemente no es exclusivo ya que una incapacidad visual no impide en absoluto el desarrollo de ciertas actividades mentales.

El interés de los métodos de procesamiento de imágenes digitales se fundamenta en dos áreas principales de aplicación: a) mejora de la calidad para la interpretación humana; b) procesamiento de los datos de la escena para la percepción de las máquinas de forma autónoma. En el intento por dotar a las máquinas de un sistema de visión aparece el concepto de Visión Artificial.

Desde una perspectiva general, la visión artificial por computador es la capacidad de la máquina para ver el mundo que le rodea, más precisamente para deducir la estructura y las propiedades del mundo tridimensional a partir de una o más imágenes bidimensionales.

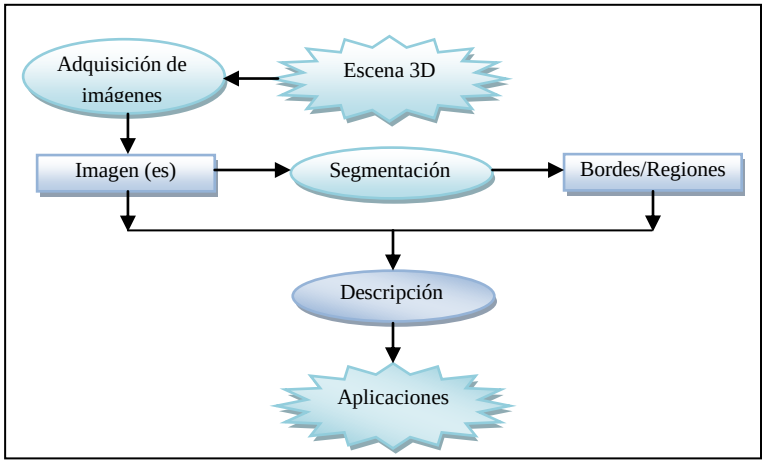



**Figura 2.1.**Diagrama de bloques mostrando el proceso completo de la Visión Artificial

El proceso general se puede sintetizar en la figura 2.1 en la que las cajas representan datos y las burbujas procesos. Se parte de la escena tridimensional y se termina con la aplicación de interés.

En la visión por computador, la escena tridimensional es vista por una, dos o más cámaras para producir imágenes monocromáticas o en color. Las imágenes adquiridas pueden ser segmentadas para obtener de ellas características de interés tales como bordes o regiones. Posteriormente, de

las características se obtienen las propiedades subyacentes mediante el correspondiente proceso de descripción. Tras lo cual se consigue la estructura de la escena tridimensional requerida por la aplicación de interés (Pajares & Manuel, 2002).

### 2.1.1 Representación digital de imágenes

El termino imagen *monocromática* o simplemente *imagen* se refiere a un función bidimensional de intensidad de luz $(x, y)$, donde $x$ e $y$ representa las coordenadas espaciales y el valor de $f$ en un punto cualquiera $(x, y)$ es proporcional al brillo (o nivel de gris) de la imagen en ese punto. La figura 2.2 ilustra el convenio de ejes utilizado.

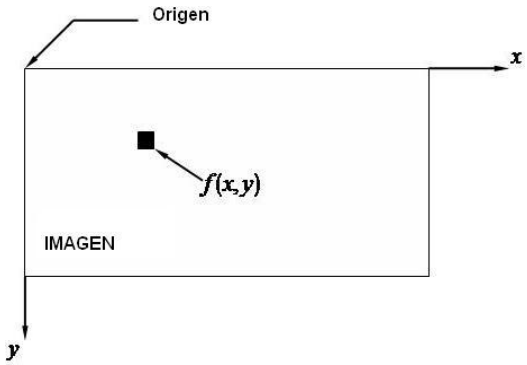



**Figura 2.2.** Convección de ejes usados

Una imagen digital es una imagen $f(x, y)$ que se ha discretizado tanto en las coordenadas espaciales como el brillo. Una imagen digital puede considerarse como una matriz cuyos índices de fila y columna identifican un punto de la imagen y el valor del correspondiente elemento de la matriz indica el nivel de gris en ese punto. Los elementos de una distribución digital de este tipo se denominan elementos de la imagen, o más comúnmente pixels o pels, abreviaturas de su denominación inglesa *pictures elements* (Gonzáles & Woods, 1996). Para trabajar con número en la computadora, el nivel de brillo, o valor de cada píxel, es cuantiado a códigos binarios enteros positivos (el brillo no puede ser negativo). Cada píxel ofrece cierta información sobre una región elemental de la imagen. En imágenes en niveles de gris esta información es el brillo. En imágenes en color, la información corresponde a la intensidad de cada una de las componentes de una base de color (por ejemplo RGB). [Pratt 1991].

### 2.1.2 Procesamiento digital de imágenes

En visión artificial se suelen distinguir tres procesos, que en ocasión se solapan, a saber *Procesamiento, Análisis y Aplicación*. El proceso implica la manipulación de las imágenes vistas como señales digitales, para extraer la información más elemental subyacente. El análisis se

encamina a determinar ciertas estructuras elementales como bordes o regiones así como las relaciones entre ellas, y finalmente las aplicaciones tratan de dar solución a los problemas relacionados con ciertas situaciones del mundo real (Pajares & Manuel, 2002).

Los pasos fundamentales que se siguen en el procesado de imágenes como se representa en la figura 2.3.

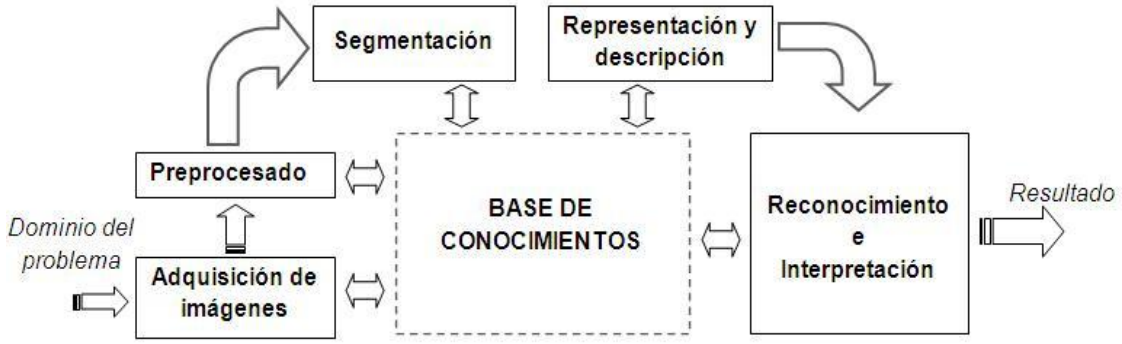



**Figura 2.3**. Etapas fundamentales para el procesamiento de imágenes

Las estepas que componen el procesamiento digital de imágenes se explican a continuación de forma detallada.

1. A**dquisición de la imagen**
   La primera etapa del proceso es la adquisición de la imagen, es decir, la adquisición de la imagen digital. Para ello se necesita de un sensor de imágenes y la posibilidad de digitalizar la señal producida por el sensor. El sensor puede ser una cámara de televisión, monocroma o de color, que produce una imagen del dominio del problema cada 1/30 de segundo. El sensor de imágenes puede ser también una cámara de barrido de línea que produzca una imagen bidimensional. Si la salida de la cámara o de otro sensor de imágenes no está en forma digital, puede emplearse un convertidor analógico-digital para su digitalización.

2. **Preprocesado**
   Con el fin de mejorar la calidad de la imagen obtenida se emplean ciertos filtros digitales que eliminan el ruido en la imagen o bien aumentan el contraste. En este ejemplo, el preprocesamiento trata típicamente las técnicas de mejorar el contraste, eliminar ruido y aislar regiones cuya textura indica la probabilidad de información alfanumérica.

3. **Segmentación**

   Definida de una forma general, la segmentación consiste partir una imagen de entrada en sus partes constituyentes u objetos. En general, la segmentación autónoma es una de las labores más difícil del tratamiento digital de imágenes. Por una parte, un procedimiento de segmentación demasiado tosco dilata la solución satisfactoria de un problema de procesamiento de imágenes. Por otra, un algoritmo de segmentación débil o errática casi siempre garantiza que tarde o temprano habrá un fallo (Gonzáles & Woods, 1996).

4. **Descripción ( Extracción de características)**

   Se realiza una medición objetiva de ciertos atributos de intereses del objeto de estudio para ser posteriormente tratados por computadora.

5. **Interpretación (reconocimiento)**

   De acuerdo a los valores obtenidos en las mediciones se lleva a cabo una interpretación del objeto.

El modelo de la figura 2.4 puede usarse por ejemplo para la determinación del grado de maduración de la fruta. En este caso se mide el color y a partir de esta información se determina si la fruta cumple con las normas para salir al mercado.

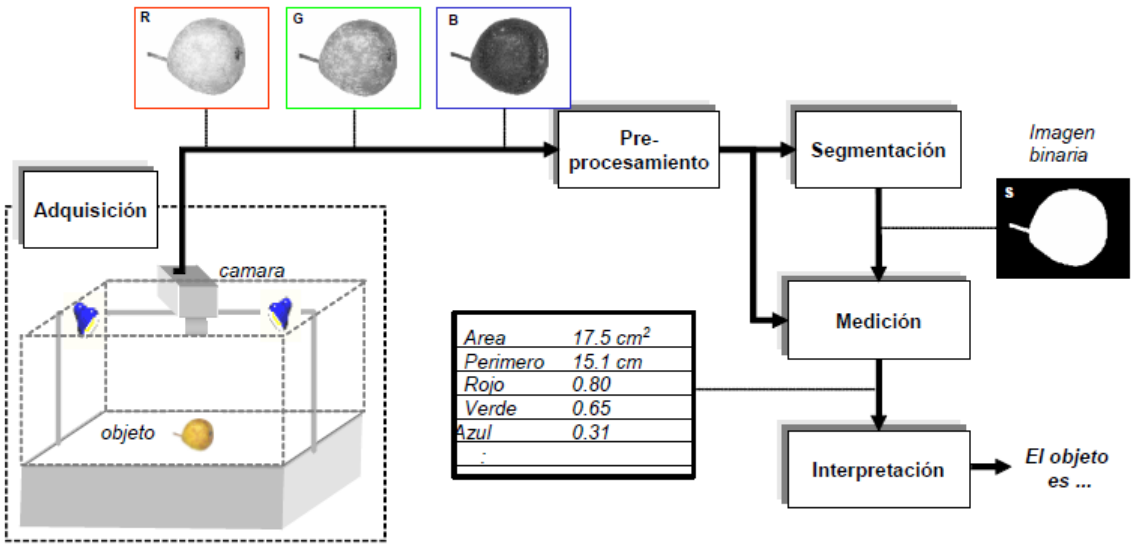



**Figura 2.4.**Esquema de un proceso de análisis de imágenes en una fruta

Hasta ahora no se ha dicho nada sobre la necesidad del conocimiento previo o sobre la interpretación entre la base de conocimiento y los módulos de procesamiento. El conocimiento

sobre un dominio del problema esta codificado en un sistema de procesamiento de imágenes como una base de datos de conocimiento. Este conocimiento pude ser tan simple como detallar las regiones de una imagen donde se sabe que se ubica información de interés, limitando así la búsqueda que ha de realizarse para hallar la información (Gonzáles & Woods, 1996).

### 2.1.3 Filtro espacial

Filtrado espacial es la operación que se aplica a imágenes ráster para mejorar o suprimir detalles espaciales con el fin de mejorar la interpretación visual. Ejemplos comunes incluyen aplicar filtros para mejorar los detalles de bordes en imágenes, o para reducir o eliminar patrones de ruido. Filtrado espacial es una operación "local" en procesamiento de imagen en el sentido de que modifica el valor de cada píxel de acuerdo con los valores de los píxeles que lo rodean; se trata de transformar los ND originales de tal forma que se parezcan o diferencien más de los correspondientes a los píxeles cercanos.

#### 2.1.3.1 Filtro detector de borde

Realizan otro tipo de operaciones con los datos, pero siempre con el resultado de enfatizar los bordes que rodean a un objeto en una imagen, para hacerlo más fácil de analizar. Estos filtros típicamente crean una imagen con fondo gris y líneas blancas y negras rodeando los bordes de los objetos y características de la imagen. La filosofía básica de muchos algoritmos de detección de borde es el cómputo de operadores derivada locales (primera o segunda).

**Concepto de derivada en la extracción de bordes**

En la figura 2.5 se puede observar que los bordes (transición de oscuro a claro o viceversa) se modelan como una rampa en lugar de hacerlo como un cabio brusco de intensidad, debido a que la imagen original suelen estar desdibujados como resultado del muestreo.

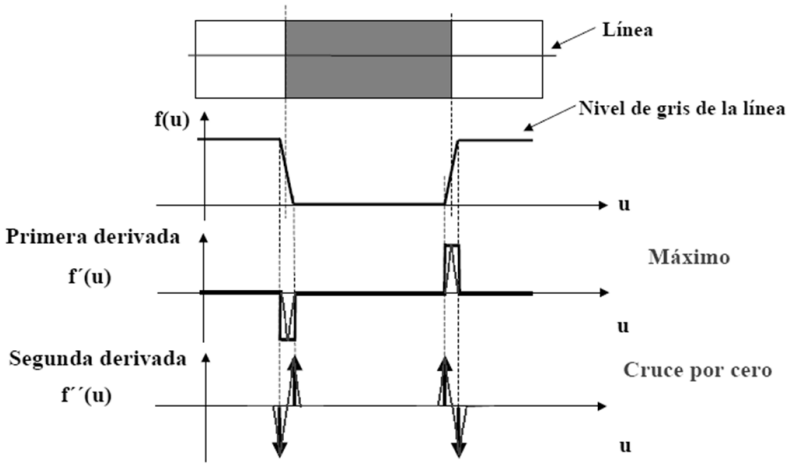



**Figura 2.5.**Concepto de primera derivada y segunda derivada para la extracción de bordes

La primera derivada es cero en todas las regiones de intensidad constante y tiene un valor constante en toda la transición de intensidad. La segunda derivada, en cambio, es cero en todos los puntos, excepto en el comienzo y el final de una transición de intensidad. Por tanto, un cambio de intensidad se manifiesta como un cambio brusco en la primera derivada y presenta un paso por cero, es decir se produce un cambio de signo en su valor, en la segunda derivada.

Basándose en estas observaciones y en los conceptos ilustrados en la figura 2.2, es evidente que el valor de la primera derivada puede utilizarse para detectar la presencia de un borde así como el signo de la segunda derivada. Para la extracción de los bordes se utiliza operadores de gradiente basado en la primera derivada. Dentro de los operadores más utilizados para la extracción de bordes se encuentra la de Sobel, Prewitt, Roberts, Kirsch, Robinson, Frei-Chen, de los cuales solo se analizara el operador de Sobel.

**Gradiente de una imagen (primera derivada)**

El gradiente de una imagen $f(x,y)$ en un punto $(x,y)$ se define como un vector bidimensional dado por la ecuación (2.1), siendo un vector perpendicular al borde

$$\boldsymbol{G}[f(x,y)] = \begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} \dfrac{\partial f(x,y)}{\partial x} \\ \dfrac{\partial f(x,y)}{\partial y} \end{bmatrix} \qquad (Ecuación\ 2.1)$$

Donde el vector $\boldsymbol{G}$ apunta en la dirección de variación máxima de $f$ en el punto $(x\,,y)$ por unidad de distancia con la magnitud y dirección dada por

$$|\boldsymbol{G}| = \sqrt{{G_x}^2 + {G_y}^2}; \qquad \emptyset(x,y) = tan^{-1}\left(\frac{G_y}{G_x}\right) \qquad (Ecuación\ 2.2)$$

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

$$|\boldsymbol{G}| = |G_x| + |G_y| \quad (Ecuación\ 2.3)$$

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 la magnitud del gradiente supere  o no un determinado umbral, pues bien solo es de ajustar el valor del umbral

para que el resultado de la extracción de bordes sea el mismo tanto si se calcula la magnitud del gradiente (Ecuación 2.2) como si se hace mediante de la Ecuación 2.3 y sin embargo, esta ultima ecuación resulta mucho más fácil de implementar, particularmente cuando se realiza en hardware.

Para calcular la derivada en (2.1) se puede utilizar las diferencias de primer orden entre dos pixeles adyacentes; esto es,

$$G_x = \frac{f(x + \Delta x) - f(x - \Delta x)}{2\Delta x} \qquad G_y = \frac{f(y + \Delta y) - f(y - \Delta y)}{2\Delta y} \qquad (Ecuación\ 2.4)$$

Esta es la forma más elemental de obtener el gradiente en un punto. La magnitud del gradiente puede tomar cualquier valor real y el ángulo también cualquier valor real entre $0^o$ y $360^o$. La implementación de la derivada en un punto considerando una vecindad de dimensión 3x3 entorno al punto, puede ser utilizando operadores entre los que se encuentra Sobel, Prewitt o Roberts, Kirsch, Robinson o Frei-Chen. De los cuales solo hablaremos del operador de Sobel.

En cualquier caso, y siguiendo con el concepto de derivada definido en la ecuación 2.4, la figura 2.6 muestra la imagen gradiente de la misma original. En (b) se muestra la magnitud de la imagen gradiente, es decir $|\boldsymbol{G}|$; en (b) se muestra una imagen binaria utilizando la siguiente relación,

$$g(x,y) = \begin{cases} 1\ si\ \ G|f(x,y)| \geq T \\ 0\ si\ G|f(x,y)| \leq T \end{cases} \qquad (Ecuación\ 2.5)$$

Donde T es un valor de umbral no negativo (en este caso $T = 80$). Solo los pixeles de borde cuyo gradiente excedan del valor $T$ se consideran importantes. Así la ecuación anterior se puede ver como un procedimiento que extrae solo aquellos pixeles caracterizados por transiciones de intensidad significativas (dependiendo de T). Finalmente en la figura 2.6 (d) se muestra el ángulo de la imagen $\emptyset$ .

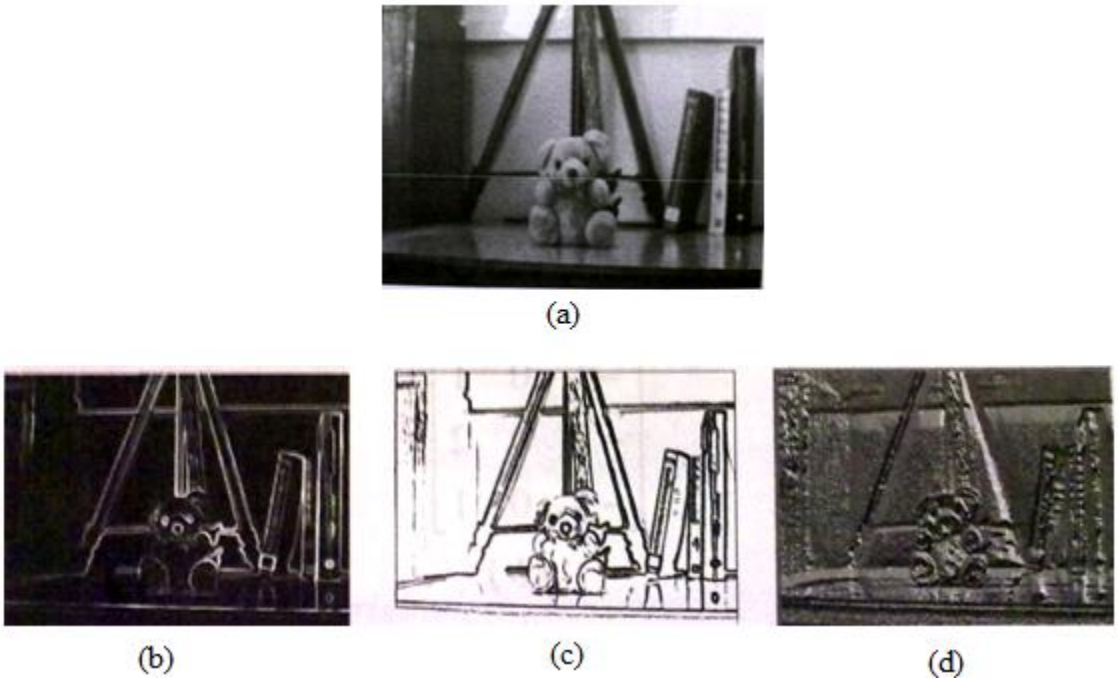



**Figura 2.6.** (a) Imagen original; (b) Imagen de la magnitud del gradiente; (c) imagen binarizada considerando un umbral T de 30 para la magnitud del gradiente; (d) imagen del ángulo del gradiente

### 2.1.3.2 Operadores de Sobel

Como hemos indicado anteriormente, los valores de $G_x$ y $G_y$ de la ecuación 2.1 pueden implementarse por convulción de la imagen con la máscara 3x3 dadas en la figura 2.6 (c) y (d), conocidas como operadores de Sobel.

Los operadores gradiente en general tienen el efecto de magnificar el ruido subyacente en la imagen, tantos los operadores de Sobel como el resto de operadores de vecindad tienen la propiedad añadida de suavizar la imagen, eliminando parte del ruido y por consiguiente, minimiza la aparición de falsos bordes debido al efecto de magnificación del ruido por parte de los operadores derivada (Pajares & Manuel, 2002).
A partir de la figura 2.4, las derivadas basadas en los operadores de Sobel son:

$$G_x = (z_3 + 2z_6 + z_7) - (z_1 + 2z_4 + z_7) \qquad (Ecuación\ 2.6)$$
$$G_y = (z_7 + 2z_8 + z_9) - (z_1 + 2z_2 + z_3) \qquad (Ecuación\ 2.7)$$

Donde los distintos valores de $z$ en la región de la figura 2.7 (a)

$$\begin{bmatrix} z_1 & z_2 & z_3 \\ z_4 & z_5 & z_6 \\ z_7 & z_8 & z_9 \end{bmatrix} \quad \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \quad \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

(a) (b) (c)

**Figura 2.7**. (a) Región de la imagen de dimensión 3x3; (b) Mascara usada para obtener $\boldsymbol{Gx}$ en el punto central de la región 3x3: (c) Máscara usada para obtener $\boldsymbol{Gy}$ en el mismo punto. Estas mascaras se denominan operadores de Sobel.

Las matrices de la figura 2.7 (b) y (c) se conocen *ventanas de Sobel*, que fue quien las propuso. Mediante ellas se calcula el gradiente en las direcciones horizontal y vertical. En la figura 2.8 se ve como el resultado de aplicar la matriz (b) de la figura 2.8 sobre la imagen Lena produce una imagen en la que aparecen los contornos horizontales de la figura de la imagen original. Este resultado se obtiene utilizando un factor de divisiones de 4 y presentando el valor absoluto de la convolución, utilizando niveles de grises en escala desde 0 como blanco hasta 255 como negro.

$$\begin{pmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{pmatrix}$$

Original Sobel

**Figura 2.8.**Filtro de Sobel en la dirección x en valor absoluto

Una alternativa muy común al uso de valor absoluto para evitar los valores fuera de rango consiste en el uso de un factor de suma que se aplica tras la convolución y la división. Por ejemplo, en el caso del filtro de Sobel un factor de división de 8 y un factor de suma de 128 evitarían los valores fuera de rango (Serrano et al, 2003).

### 2.1.4 Binarización de imágenes

La Binarización es un caso particular de la segmentación que consiste en transformar los pixeles de la imagen en 0 ó 1, dependiendo de su nivel de gris, así como se muestra en la siguiente figura.


**Figura 2.9.** (a) Imagen original de Lena; (b) Imagen Binarizada

Obtener una imagen binaria $f_B$ a partir de una imagen en niveles de gris $f_A$ consiste en:

$$\boldsymbol{f_B(x,y)} = \begin{cases} 1 \; si \; f_A(x,y) \in Z \\ 0 \; en \; otro \; caso \end{cases} \qquad (Ecuación\ 2.8)$$

Donde $Z$ en un subconjunto de todos los niveles de gris posibles en $f_A$ Pero la forma de binarizar más utilizada consiste en utilizar un umbral $T$ que determina a partir de qué nivel de gris se va a considerar primer plano y el resto fondo.

$$f_B(x,y) = \begin{cases} 1 \; si \; f_A(x,y) > T \\ 0 \; en \; otro \; caso \end{cases} \qquad (Ecuación\ 2.9)$$

Para determinar este umbral existen métodos de búsqueda automática de umbrales, como los que se han visto en teoría. Algunos de estos métodos utilizan el histograma de la imagen para calcular el umbral más apropiado (Pajares & Manuel, 2002).

## 2.2 INTRODUCCION A LA SEÑAL DE VIDEO

La señal de TV surge a partir del 1939, cuando se realizo la primera transmisión en la feria mundial Nueva York, con una característica de 340 líneas por segundo. Como no había un estándar definido, creó problemas de incompatibilidad con los demás sistemas que diseñaban otros fabricantes, generando así grandes desastres en las TV. Por estos motivos en 1953 la FCC funda el Comité de Sistema de Televisión Nacional (NTSC) en ingles, que fue la comisión

encargada de definir los parámetros del sistema de televisión compatible en color. Es un sistema de codificación y transmisión de Televisión en color analógica que se emplea en la actualidad en la mayor parte de América y Japón, entre otros países. Un derivado de NTSC es el sistema PAL que se emplea en Europa y países de Sudamérica.

El vídeo compuesto es una señal de vídeo analógica que se utiliza en la producción de televisión y en los equipos audiovisuales domésticos. Esta señal eléctrica es una señal compleja en la que se codifica la imagen en sus diferentes componentes de luz y color añadiendo los sincronismos necesarios para su posterior reconstrucción.

## 2.2.1 Sincronismos para la señal de video

### 2.2.1.1 Frecuencia Vertical

El comité de sistema de televisión nacional obtuvo sus cálculos basados en el principio del funcionamiento del cine, que tenía una frecuencia de 24 fotogramas por segundo, y en la frecuencia de la red eléctrica (60 hertzios en América y 50 horst en Europa).

Para obtener una señal de TV semejante al del cine, decidieron dividir la frecuencia de la red eléctrica entre dos (60 Hz /2), para obtener 30 cuadros por segundo, pero analizaron que esta frecuencia no era suficiente para obtener una imagen estable, por lo que existía un problema de parpadeo en la imagen, algo que en el cine no existía a una frecuencia de 24 fotogramas por segundo. En base a esto descubrieron que el cine utilizaba una técnica de sobre muestreo, que consistía en mostrar 2 veces el mismo fotograma. Para resolver el problema del parpadeo en la imagen se realiza una especie de *entrelazado*, pero además de eso el cine utiliza imágenes bidimensionales (frame completo), por lo tanto cada frame lo despliega completo al espectador, en cambio la tv, la imagen debe ser nuevamente muestreada a fin de obtener una señal unidimensional, de modo que pueda ser trasmitido por una medio normal. Para lograr esto, la imagen se muestra espacialmente en líneas horizontales, pixel por pixel, con un número de línea predeterminado hasta formar una imagen completa.

### 2.2.1.2 Número de líneas

El número de líneas en que se descompone cada imagen influye directamente en dos parámetros básicos de la señal de televisión: la calidad y grado de detalle de la imagen en el eje vertical y el ancho de banda de la señal. Es evidente que para tener una buena percepción de la imagen, el número de líneas deberá ser suficientemente elevado como para que el sistema visual no sea capaz de distinguir entre la imagen original y la imagen muestreada. La situación representada en

la figura 2.10, corresponde a un claro ejemplo en el que, siempre que la imagen se observe a una distancia de lectura normal, el número de líneas parece ser insuficiente, produciéndose una considerable pérdida de calidad respecto a la imagen original.

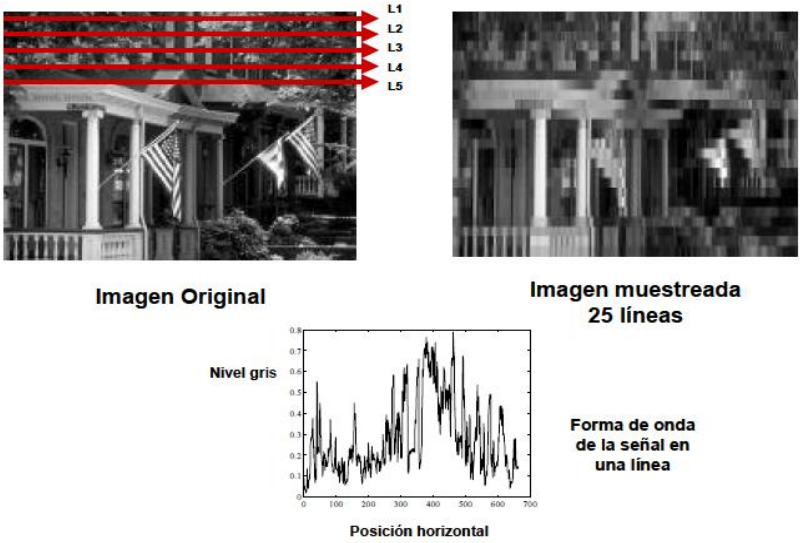



**Figura 2.10.** Muestreo de una imagen por fila

En la elección del número de líneas de un sistema de televisión intervienen diversos factores. Probablemente, el factor esencial es la propia capacidad del sistema visual humano para discernir los detalles en una imagen bajo determinadas condiciones de iluminación. Es evidente, por tanto, que no tiene sentido aumentar el número de líneas de una imagen más allá de lo que el ojo es capaz de discernir, ya que en este caso, un incremento de información en la imagen no supone ninguna mejora subjetiva en su calidad.

La medida de la capacidad de resolución del ojo se realiza mediante retículas de estas características bajo distintas condiciones de iluminación. El *poder de resolución* del sistema visual humano se define como el ángulo subtendido por dos líneas negras en el ojo cuando el patrón reticular está situado a la distancia límite en la que aún se aprecia el detalle de las líneas que lo forman. El inverso de este ángulo expresado en minutos de grado se define como la *agudeza visual* del ojo (Figura 2.11).

El número de líneas de la imagen viene determinado, tal y como se indica en la figura 2.12, por el ángulo vertical con el que se observará la altura de la pantalla de televisión. Teniendo en cuenta que la resolución espacial media del sistema visual humano en las condiciones de iluminación de una señal de televisión es del orden de un minuto de grado, el número de líneas necesario para una correcta calidad de la imagen coincidirá con el ángulo vertical de observación de la pantalla

expresado en minutos. No tendrá sentido superar este número de líneas, por cuanto supondría un aumento del ancho de banda de la señal sin ninguna mejora apreciable en la calidad de la imagen.

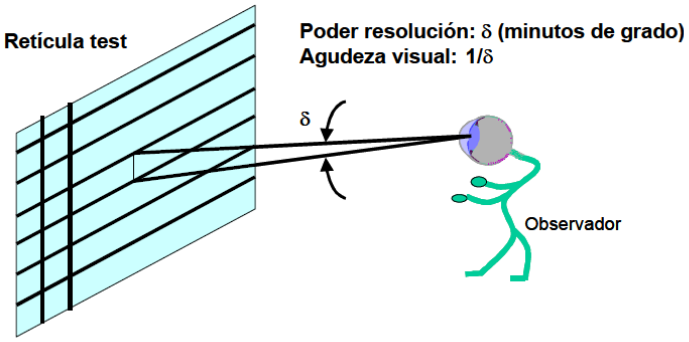



**Figura 2.11.** Poder de resolución y agudeza visual

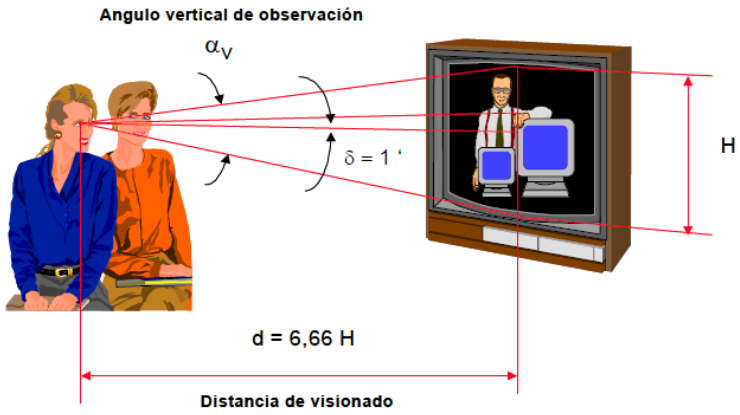



**Figura 2.12**.Ángulo vertical de observación y distancia de visionado

La decisión que finalmente se tomó en los primeros sistemas de televisión fue diseñarlos para que fueran observados desde una distancia que estuviera entre 6 y 7 veces la altura de la pantalla. Dado que, para mantener compatibilidad con el cine, la relación de aspecto de la pantalla estaba fijada en 4:3; la relación entre la distancia de visionado y la altura de la pantalla será:

$$d \approx 4.\sqrt{H^2 + \left(\frac{4H}{3}\right)^2} = 6{,}66 \cdot H \qquad (Ecuación\ 3.0)$$

Por lo tanto el ángulo vertical de observación de la pantalla vendrá dado, de acuerdo con la figura 2.13, por:

$$\alpha_V = 2\, arctg\left(\frac{H/2}{6{,}66H}\right) = 8{,}57^o \qquad (Ecuación\ 3.1)$$

Finalmente, teniendo en cuenta que la resolución espacial del sistema visual es del orden de 1 minuto de grado, y que, por tanto, dos líneas deberán situarse, como máximo, a una distancia angular de 1 minuto, obtenemos el número mínimo de líneas necesario para el muestreo espacial de las imágenes:

$$NL = \frac{8{,}57^o\, x\ 60'}{1'/linea} = 514{,}69\ lineas \quad (Ecuación\ 3.2)$$

Este número de líneas debe tomarse como un valor aproximado que nos indica aproximadamente cuál es el valor aproximado que debe tomar este parámetro. El número de líneas visibles en el sistema PAL es algo superior al obtenido en la ecuación anterior, pero en el sistema NTSC se mantiene ligeramente por debajo.

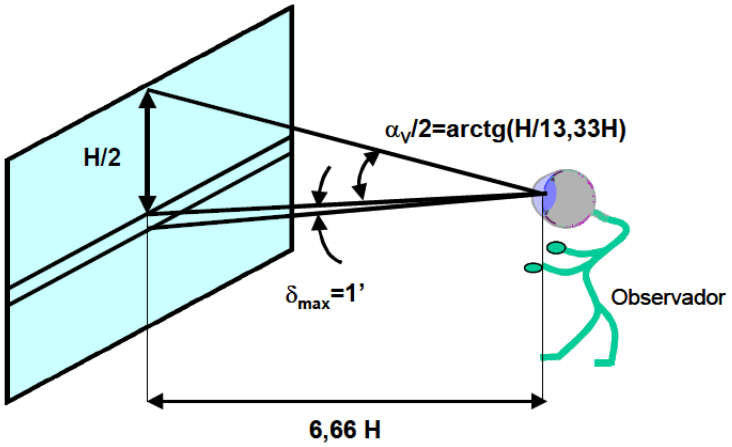



**Figura 2.13.** Cálculo del ángulo vertical de visionado

### 2.2.1.3 Selección del número de líneas

En la selección definitiva del número de líneas de la señal de televisión intervienen, además de la agudeza visual discutida en los apartados anteriores, dos factores relacionados con la tecnología electrónica existente en la época en la que se definieron los primeros sistemas y que por conveniencia, simplicidad y compatibilidad se han mantenido posteriormente. El primero es que para conseguir un perfecto entrelazado entre las dos subimágenes es necesario, como analizaremos posteriormente, que el número de líneas sea impar. El segundo factor es que para facilitar los circuitos electrónicos es conveniente que pueda establecerse una relación simple entre el número de líneas y el número de imágenes por segundo. Esta relación simple se traduce en que puedan obtenerse las distintas frecuencias que intervienen en el proceso de exploración de

la imagen a partir de un único oscilador global, cuya frecuencia se divide en relaciones enteras simples para obtener las distintas señales de barrido de línea, campo, imagen, etc. Para ello, es útil que el número de líneas de la imagen pueda expresarse como un producto de números primos cuyos valores absolutos sean relativamente bajos.

Con ello se simplifica notablemente el proceso de sincronizar las señales para la exploración vertical y horizontal de la imagen, pues todas las señales se obtienen de una misma referencia. El interés de que el número de líneas tenga una descomposición en números primos relativamente bajos radica en que es mucho más simple y fiable implementar divisores o multiplicadores de frecuencia cuando los factores de incremento o reducción son enteros bajos. En la práctica sólo se utilizan divisores de frecuencia por 2, 3, 5, 7 y raramente por 11 o por 13. En los sistemas de televisión americanos y japoneses se utilizan 525 líneas que también tienen una descomposición en números primos con características similares ($525=3 \times 5 \times 5 \times 7$). En Francia se utilizó durante algún tiempo un sistema de 819 líneas ($819=13 \times 7 \times 3 \times 3$). Este sistema ya no se usa, habiendo sido sustituido por un sistema convencional de 625 líneas.

#### 2.2.1.4 Entrelazado de las imágenes

Para evitar el parpadeo de la pantalla es necesario aumentar la frecuencia de presentación de imágenes en el receptor. La solución adoptada consiste en realizar una doble exploración entrelazada de las líneas de cada imagen tal y como se representa en la figura 2.18. La imagen se divide en dos subimágenes o campos de manera que se presenta ya no 30, si no 60 imágenes por segundo al ojo humano. Esto se hace dividiendo las líneas barridas en dos grupos, uno con líneas con número impar y el otro con las líneas de número par. Cada grupo de líneas pares o impares se llaman campo así como se muestra en la figura 2.14.

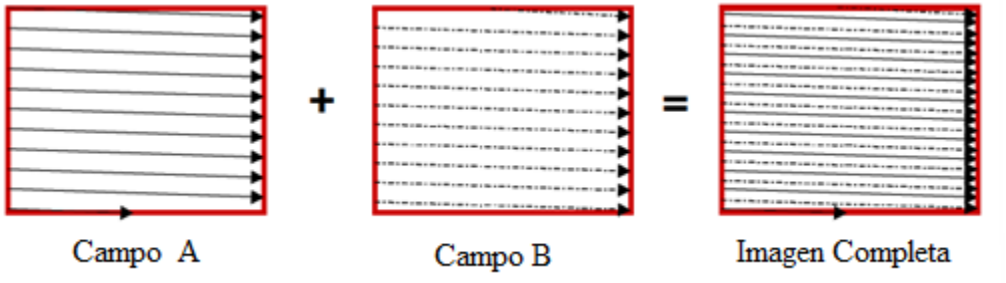



**Figura 2.14.** Barrido Entrelazado

La proximidad entre líneas consecutivas hace que el espectador integre las dos subimágenes y obtenga la sensación de que éstas se están renovando a una frecuencia doble de la real. Con ello

se consigue mantener un caudal de información reducido, suficiente para interpolar correctamente el movimiento sin que aparezca el fenómeno de parpadeo.

En cada campo se barren 262.5 líneas horizontales. Primero se barre el campo A con las líneas impares y luego se barre el campo B de las líneas pares, hasta obtener una imagen completa con 525 líneas. De esta manera se repiten 60 campos por segundos, teniendo dos campos barridos cada 1/30 segundos, que es la velocidad de barrido de un frame o cuadro. En este caso el periodo asignado a cada línea será:

$$T_{línea} = \frac{\frac{1}{30} s/imag}{525\ lin/imag} = \frac{1}{15750} = 63.49\ \mu s \qquad (Ecuación\ 3.3)$$

En la tabla 2.1 se proporcionan los valores de frecuencia de cuadro, campo y línea para los sistemas europeo y americano. Estos valores son parámetros fundamentales de la señal de TV que conviene recordar en todo momento.

**Tabla 2.1**.Características básicas de los sistemas 525/60 y 625/50

| | IMAG/S | CAMP/S | LINEAS/IMAG | $T_{línea}$ | $F_{línea}$ | $T_{imagen}$ | $F_{imagen}$ | $T_{campo}$ | $F_{campo}$ |
|---|---|---|---|---|---|---|---|---|---|
| NTSC | 30 | 60 | 525 | $63.49\ \mu s$ | $15750\ Hz$ | $33{,}33\ ms$ | 30 Hz | $16{,}66 ms$ | $60 hz$ |
| PAL | 30 | 50 | 625 | $64\ \mu s$ | $15625\ Hz$ | $40\ ms$ | $25\ Hz$ | $20\ ms$ | $50\ Hz$ |

## 2.2.2 Señales de color en el sistema NTSC

Las siglas del sistema NTSC corresponden a la *National Television System Comittee,* que fue la comisión encargada de definir los parámetros del sistema de televisión en color compatible que se adoptó en los Estados Unidos en 1953. Se trata de un sistema de 525 líneas con una frecuencia de 60 campos por segundo (30 imágenes por segundo) en el que se superponen la señal de luminancia y la de croma de acuerdo con la siguiente expresión de la ecuación 3.4 y representada en la figura 2.15:

$$X_{NTCS}(t) = Y'(t) + I'(t) \cdot cos(2\pi f_o t + 33^o) + Q'(t) \cdot sen(2\pi f_o t + 33^o) \quad (Ecuación\ 3.4)$$

Donde $Y'(t)$ corresponde a la información de luminancia (imagen blanco y negro) e $I'(t)$ y $Q'(t)$ representan una combinación lineal de las señales diferencia de color $(R-Y)'$ y $(B-Y)'$ que proporciona al receptor la información de color. En la figura 2.17 se puede observa las etapas para la obtención de las señales de video, desde el origen de una cámara (Tarrés, 2000).

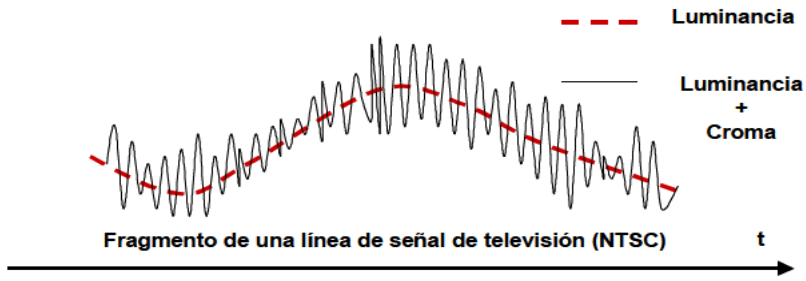



**Figura 2.15.**Superposición de la luminancia y la croma en el sistema NTSC

En resumen la señal de color o de vídeo compuesto consta de las siguientes componentes: *crominancia*, *luminancia* y *sincronismos* que indican las características del barrido efectuado en la captación de la imagen, tal y como se representa en la siguiente figura 2.16 en cual se encarga de codificar la señal RGB a YIQ.

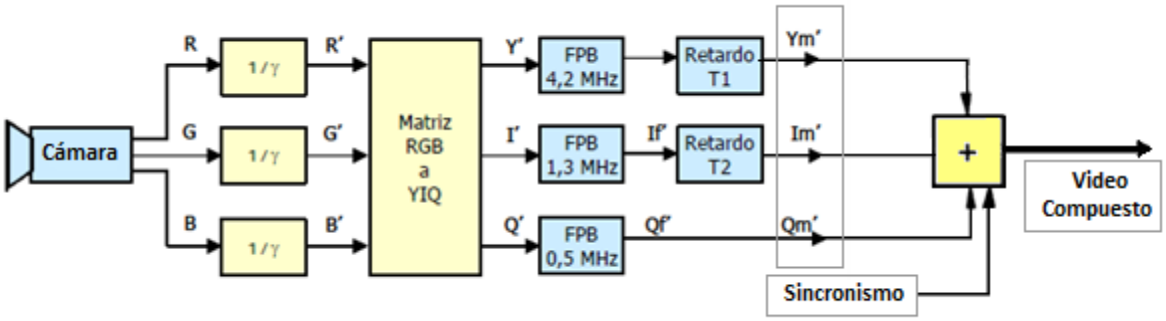



**Figura 2.16.**Obtencion de la señal de luminancia y componentes I, Q para la obtención de la señal de video compuesto en el NTSC.

Debido a que el filtro para la señal Q introduce un retardo del orden de 0,2ms, es necesario retrasar la señal I, mediante un circuito de retardo. Lo mismo ocurre con la señal de luminancia, cuyo retardo es, en este caso, de aproximadamente 0.4 a 0.5 ms. La forma de las señales de entrada y salida de la matriz RGB a YIQ se muestra en la figura 2.17 (a) y la señal de de la suma de la crominancia y luminancia está compuesta por una barra de colores como se ilustra en la figura 2.17 (b). Los colores de las barras, de izquierda a derecha son: blanco, amarillo, cian, verde, magenta, rojo y azul (Pérez & Zamanillo, 2003).

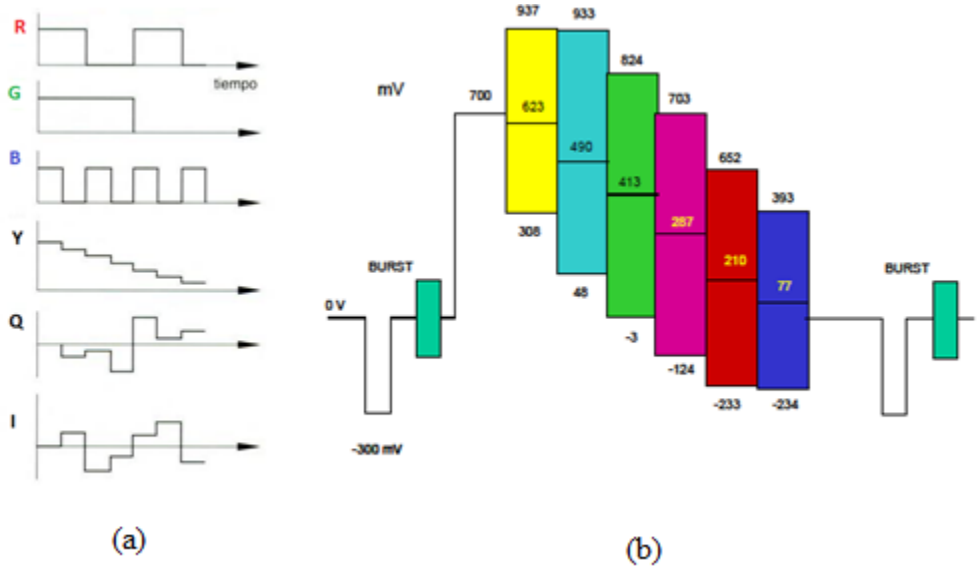



**Figura 2.17.** (a) Señal de entrada y salida de la matriz del codificador; (b) Señal de barra de color

#### 2.2.2.1 Señal de Crominancia

La crominancia incluye toda la información de color sin brillo. La crominancia y el brillo juntos especifican la información completa de imagen completamente. La crominancia se denomina también *croma*. La señal de crominancia es una combinación de las señales de color $I$ y $Q$. La señal I o señal de color en fase se genera combinando el 60% de la señal de video en rojo (R), 28% de la señal de video en verde (G) invertida y 32% de la señal de video en azul (B) invertida, y se expresa como:

$$I = 0.60R - 0.28G - 0.32B \qquad (Ecuación\ 3.5)$$

La señal Q o señal de color en cuadratura se genera combinando el 21% de la señal de video en rojo (R), 52% de la señal de video en verde (G) invertido y 31% de la señal de video en azul (B), y su expresión es:

$$Q = 0.21R - 0.52G + 0.31B \qquad (Ecuación\ 3.6)$$

Las señales I y Q se combinan para producir la señal C y debido a que las señales I y Q están en cuadratura, la señal C o crominancia es la suma vectorial de estas, y su expresión es (Grob, 1990):

$$C = \sqrt{I^2 + Q^2} \qquad (Ecuación\ 3.7)$$

**Determinación de la Frecuencia de la subportadora de color**

El valor de la frecuencia portadora de la señal de croma debe ser suficientemente elevado para que el patrón de interferencias que se produce en la pantalla de un receptor monocromo sea lo menos visible posible. Si la frecuencia es elevada podemos intuir, que la imagen representada en el receptor tendrá cambios de luminancia muy rápidos que, siempre que produzcan un patrón poco definido, serán integrados por el espectador, resultando poco visibles. Sin embargo, debemos notar que la frecuencia portadora no puede aumentarse excesivamente, ya que tendremos que mantener las señales de croma dentro del ancho de banda asignado a la componente de vídeo en un canal de televisión, que en el sistema NTSC es de aproximadamente 4,2 MHz Téngase en cuenta que este ancho de banda es una restricción impuesta por los sistemas en blanco y negro que ya estaban operativos en el momento de definir el NTSC. Los sistemas en color compatibles deben siempre acomodarse en los canales previamente utilizados por los sistemas en blanco y negro, puesto que si no fuera así, podrían introducirse interferencias en sistemas ya operativos. Además, debe tenerse presente que la portadora de sonido está situada en 4,5 MHz y debe mantenerse en esta posición si pretendemos que el sistema sea compatible.

La frecuencia que finalmente se eligió para el sistema NTSC es:

$$f_{NTSC} = \left(\frac{455}{2}\right) f_{Linea} = 227{,}5 \cdot f_{Linea} \quad (Ecuación\ 3.8)$$

Si sustituimos la señal de línea por su valor numérico obtenemos

$$f_{NTSC} = 227{,}5 \times (525 \times 30\ Hz) = 3.583125\ MHz$$

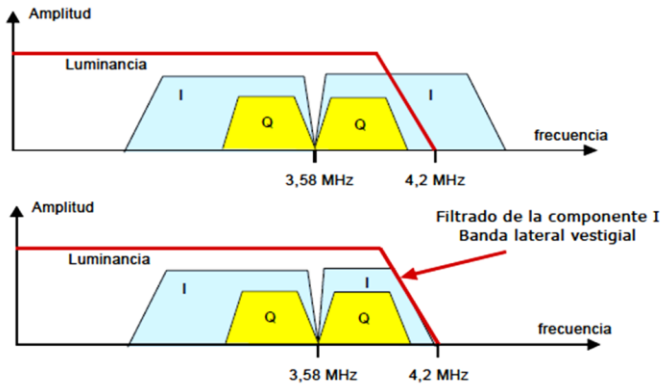



**Figura 2.18.**Ubicaciones espectrales de las componentes de luminancia y croma **(Tarrés, 2000)**

#### 2.2.2.2 Señal de Luminancia

El brillo es un atributo subjetivo del color que nos permite identificar la luminosidad aparente de los objetos. Decimos que dos objetos tienen el mismo brillo cuando, independientemente de la tonalidad de sus colores, producen en el observador la misma sensación de luminosidad. Este concepto está íntimamente ligado a los sistemas de representación de imágenes en blanco y negro, ya que en éstas se intenta reproducir la misma sensación de brillo que en la escena original utilizando distintas graduaciones de gris.

En los sistemas de televisión en blanco y negro se utiliza una señal denominada luminancia cuyo nivel es proporcional a la sensación de brillo. Esta señal se obtiene utilizando sensores de imagen, filtros ópticos y sistemas de corrección de nivel de señal cuya respuesta en frecuencia conjunta tiene una forma parecida a la curva de sensibilidad del ojo para visión diurna. La señal de luminancia o, simplemente, la luminancia, es la versión cuantitativa de la sensación de brillo. La señal de luminancia (Y) que capta una persona de una imagen real viene determinada por la relación de los colores primarios que la componen. Esta relación es de la forma:

$$Y = 0.3R + 0.59G + 0.11B \quad (Ecuación\ 3.9)$$

Obsérvese que los coeficientes obtenidos indican el grado de participación de cada uno de los primarios en la sensación de brillo y que la normalización de estos coeficientes supone que la luminancia asociada al blanco de referencia es la unidad.

Esta relación que es conocida como ley de Grassmann permite identificar la correspondencia entre la información de color de la imagen (R, G y B) y su contenido de luminancia (Grob, 1990).

## 2.3 FIELD PROGRAMMABLE GATE ARRAY (FPGA)

El diseño digital ha evolucionado desde sus comienzos debido fundamentalmente a los avances en las tecnologías de fabricación de circuitos integrados. Hasta la aparición en los años ‘80 de los primeros dispositivos programables, las opciones para diseñar sistemas digitales eran el uso de circuitos integrados estándar de baja o media escala de integración o el diseño de circuitos integrados a medida, conocidos como ASICs (Application Specific Integrated Circuits). La primera opción siempre tuvo asociado los problemas de la falta de flexibilidad de los diseños en cuanto a actualizaciones de los mismos, alto consumo de potencia, bajas velocidades de operación, grandes aéreas de los circuitos, etc.; mientras que la segunda, si bien no tenía los

problemas de consumo de energía, baja velocidad de operación, etc., siempre tuvo el inconveniente de la falta de flexibilidad además del alto costo de fabricación. Por lo tanto, la aparición de los dispositivos lógicos programables ha revolucionado el diseño digital, sobre todo en el desarrollo de sistemas digitales de mediana o baja escala de producción, donde han demostrado alcanzar velocidades de operación aceptables (hoy en día del orden de los cientos de megahertz) y gran flexibilidad para adaptar los diseños a futuros cambios. Esto ha hecho que el mercado donde se aplican los FPGA ha aumentado drásticamente en los últimos años. Actualmente son utilizados para el desarrollo de sistemas en un solo chip (SoPC, System on Programmable Chip), sistemas embebidos, aplicaciones de procesamiento digital de señales, etc.

Por estos motivos, los FPGAs son muy utilizados por fabricantes que producen tecnología a baja escala, como por ejemplo diseñadores de equipos de propósito específico, los cuales no pueden justificar la producción de ASICs por los bajos volúmenes de equipos que producen. Los FPGAs tienen una funcionalidad similar, a costos menores y con una velocidad de operación ligeramente menor.

Actualmente, son los FPGAs los dispositivos lógicos programables que integran la mayor cantidad de componentes lógicos y por lo tanto son con los que se puede obtener la mayor versatilidad de diseño. Las velocidades de operación de los mismos pueden ser cercanas a las de un ASIC (Application Specific Integrated Circuit). Los principales recursos que poseen convierten a los FPGA en adecuados para la mayor parte de las aplicaciones.

Analizando la estructura interna de un FPGA, donde cada bloque combinacional (LUT) esta seguido por un flip-flop, es conveniente realizar la implementación de funciones lógicas de muchas variables por medio del particionamiento de la lógica combinacional en distintas etapas separadas entre sí por registros (estructura de pipeline), lo cual permite mantener acotados los tiempos de propagación de las señales. Se sabe que en una estructura de pipeline la frecuencia máxima de operación está determinada por el tiempo de propagación de la etapa más lenta, por lo tanto, el dividir en etapas el cálculo de una función lógica incrementa la frecuencia de operación del circuito.

### 2.3.1 Arquitectura general de los FPGA

Las FPGA’s (Field Programmable Gate Array-Arreglos de Compuertas Programables en Campo) son dispositivos lógicos de propósito general programable por los usuarios, cuyas características pueden ser modificadas, manipuladas o almacenada mediante programación. La arquitectura de un FPGA consiste en arreglos múltiples de  celdas lógicas las cuales se comunican una tras otras

mediante canales de conexiones verticales y horizontales, tal como muestra la figura 2.19. En general, un FPGA posee una estructura altamente regular. Cada celda lógica contiene compuertas lógicas AND y OR, así como números definidos de registros y multiplexores.

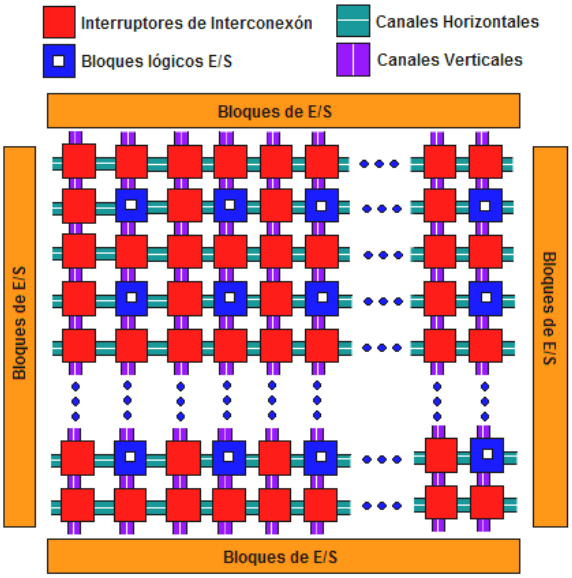



**Figura 2.19**. Arquitectura básica de un FPGA

Es un circuito integrado que contienen celdas lógicas idénticas (64 hasta 8'000.000) que se puede ver como componentes estándar. Las celdas lógicas se interconectan por medio de una matriz de cables y switches programables.

El tamaño, estructura, número de bloques y la cantidad y conectividad de las conexiones varían en las distintas arquitecturas o familias de dispositivo entre los fabricantes más importante esta Altera, Xilinx, Atmel, Actel, Cypress Semiconductor, entre otros, cada una combina entradas binarias a una o dos salidas. En este trabajo solo se abordara el dispositivo Cyclone II de Altera.

### 2.3.2 Dispositivo Cyclone II de Altera

Para el desarrollo y la implementación de los algoritmo de visión en esta residencia, se utilizo el dispositivo FPGA Cyclone II de altera.

Los dispositivos FPGA de altera se basan en lo que se conoce como arreglos de compuertas, los cuales consisten en la parte de la arquitectura. Los dispositivo de altera contienen elementos configurables como: Arreglos de bloques lógicos configurables (LABs), bloques de entrada y salida (IOEs), canales de comunicación (vertical y horizontal), bloques de memoria embebida (M4K), multiplicadores embebidos y PLLs. En la Figura 2.20 se muestra una arquitectura del FPGA Cyclone II de Altera.

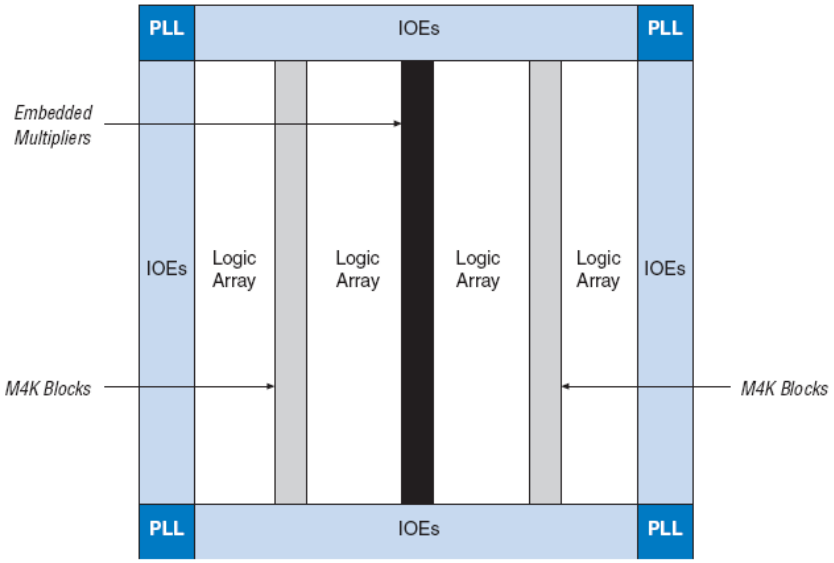



**Figura 2.20**.Arquitectura del FPGA Cyclone II de Altera

### 2.3.3 Descripción funcional de la arquitectura FPGA Cyclone II de Altera

Por dentro, un FPGA está formado por arreglos de bloques lógicos (LABs), que se comunican entre ellos y con los bloques de entrada y salidas (IOE) por medio de alambrados llamados canales de comunicación. Cada FPGA contiene una matriz de bloques lógicos idénticos, por lo general de forma cuadrada, conectados por medio de líneas metálicas que corren vertical y horizontalmente entre cada bloque.

Los dispositivos Cyclone ® II contienen una fila de dos dimensiones y arquitectura basada en columnas para la implementación de la lógica básica. La columna y fila se interconecta a los relojes (PLLs) que proveen señales a los bloques lógicos (LABs), bloques de memoria y multiplicadores embebidos.

#### 2.3.3.1 Arreglos de Bloques Lógicos (LAB)

El arreglo lógico consiste en una conjunto de LAB, con 16 elementos lógicos (LEs) por cada LAB. Un LE es una pequeña unidad lógica con implementación eficientes de funciones lógicas. Los bloque de arreglo lógicos (LABs) son agrupadas en filas y columnas a través del dispositivo. El dispositivo Cyclone II tiene un rango en densidad de 4,608 a 68416 LEs (Figura 2.21).

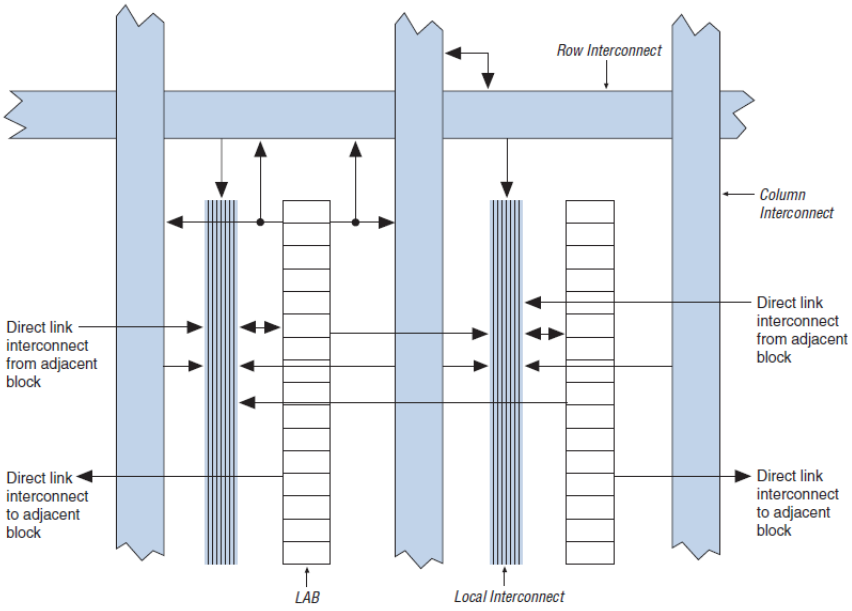



**Figura 2.21.**Arreglos de bloques lógicos (LAB)

### 2.3.3.2 Elemento lógico (LE)

Un LE es la unidad más pequeña de la arquitectura Cyclone II, el LE, es compacta y ofrece funciones avanzadas con la más eficiente utilización de la lógica. Opera en modo normal y modo aritmético. Tiene la capacidad para controlar cualquier tipo de conexión: local, fila, columna, cadena de registro y enlace directo interconexiones (Figura 2.22).

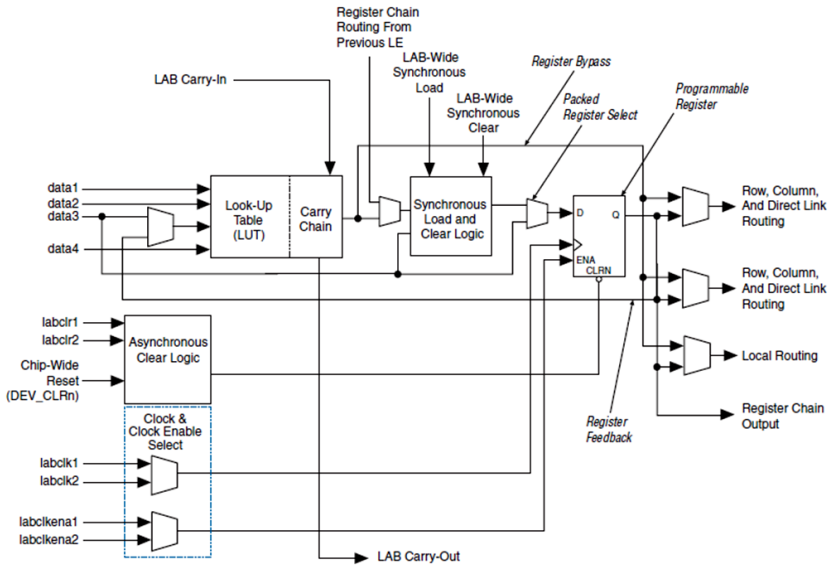



**Figura 2.22.** Elemento lógico (LE)

El diseño lógico se implementa mediante bloques conocidos como generadores de funciones o LUT (Look Up Table: tabla de búsqueda), los cuales permiten almacenar la lógica requerida, ya que cuentan con una pequeña memoria interna por lo general de 16 bits. Cuando se aplica alguna combinación en las entradas de la LUT, el circuito la traduce en una dirección de memoria y envía fuera del bloque el dato almacenado en esa dirección.

### 2.3.3.3 Red global de reloj

El dispositivo Cyclone II proporciona una red global de reloj y hasta cuatro bucles de enganche de fase (*Phase Locked-Loop*: PLL). La red global de reloj se compone de 16 líneas que controla totalmente el dispositivo, puede proporcionar señal de reloj para todo el recurso necesario dentro del dispositivo, como entradas/salidas (IOEs), LEs, multiplexores y memorias de bloques embebidas, etc. Las líneas de reloj global también pueden ser utilizadas para otras señales de fan-out altas. Cyclone II PLLs provee de manera general, sincronización con la síntesis de reloj y corrimiento de fase, así como también, salidas externas de alta velocidad con soporte diferencial de E/S (Figura 2.23).

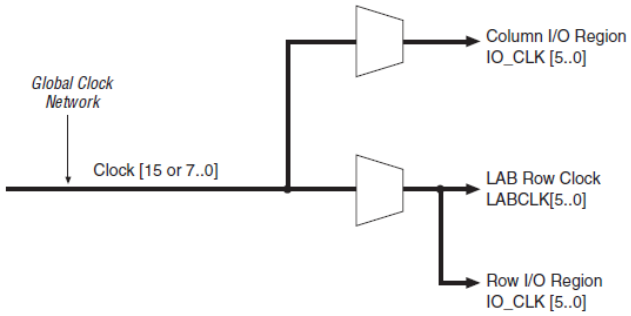



**Figura 2.23.** Conexión global de reloj

### 2.3.3.4 Bloques de memoria embebida (MK4)

Los bloques de memoria MK4 son bloques de memorias de doble puerto real, con 4K bits de memoria, mas la paridad (4,608 bits). Estos bloques dedicados proveen doble puerto real, doble puerto simple, o puerto singular de memoria hasta 36-bits de ancho y hasta 260Mhz. Estos bloques son organizados en columna a través del dispositivo y están ubicados entre cierto LABs. El dispositivo Cyclone II ofrece entre 119 a 1,152 Kbits de memoria embebida (Figura 2.24).

Muchas aplicaciones requieren el uso de memoria, de forma tal que una gran cantidad de dispositivos FPGA incluyen bloques de memoria RAM además de la RAM distribuida en las Lats. Dependiendo de la arquitectura del dispositivo, estos bloques de memoria pueden estar

ubicados en la periferia del dispositivo, dispersos sobre la superficie del mismo o ubicados en columnas.

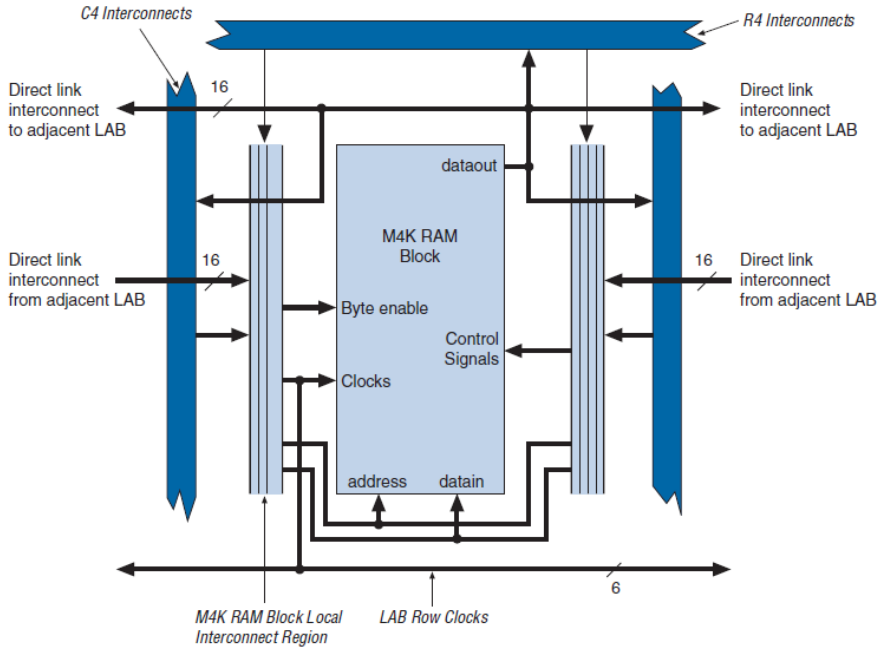



**Figura 2.24.** Bloques de memoria embebida (MK4)

Dependiendo de la aplicación, estos bloques de memoria pueden ser configurados de diversas maneras, formando memorias de un solo puerto (single-port RAMs), memorias de doble puerto (dual-port RAMs), memorias FIFO (first-in first-out), etc. Poseen además un sistema de enrutamiento dedicado que provee una eficiente comunicación entre estos bloques de memoria y los bloques lógicos.

### 2.3.3.5 Multiplicador embebido

Cada bloque de multiplicador embebido puede implementar ya sea dos multiplicador 9x9-bits, y uno multiplicador 18x18-bits con hasta 250Mhz de función. Los multiplicadores embebidos son ordenados en columnas a través del dispositivo (Figura 2.25).

Algunas funciones lógicas tales como suma y multiplicación son inherentemente lentas si se realizan interconectando un gran número de bloques lógicos. Para aplicaciones de procesamiento digital de señales (DSP, Digital Signal Processing), donde los algoritmos generalmente se basan en sumas y multiplicaciones (denominadas operación MAC, Multiply and Accumulate), el hecho de que los FPGAs cuenten con una cantidad de multiplicadores embebidos los hace ideales para ser utilizados. Generalmente, los multiplicadores se encuentran cercanos a los bloques de

memoria RAM dentro del dispositivo, ya que los mismos son utilizados para procesar datos alojados en dichos bloques.

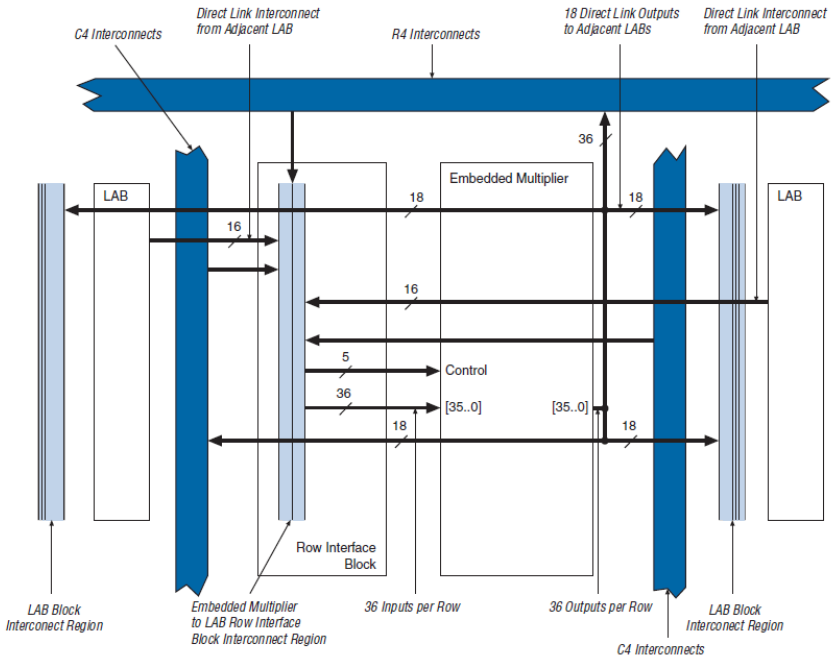



**Figura 2.25.** Multiplicador Embebido

El numero de bloques de memoria MK4, bloques de multiplicador embebido, PLLs, columnas y fila varia por dispositivo.

### 2.3.3.6 Bloques entrada/salida

La función de un bloque de entrada/salida es permitir el paso de una señal hacia dentro o hacia el exterior del dispositivo. Por este motivo, dependiendo de la interfaz eléctrica, el mismo debe contar con recursos tales como:

- Salidas configurables como Tri-State u Open-Collector.
- Entradas con posibilidad de pull-up o pull-down programables.
- Registros de salida.
- Registros de entrada.

Además, debido a que hoy en día los FPGA funcionan a altas velocidades de operación y que los mismos tienen desde cientos hasta miles de terminales de conexión, los bloques de entrada/salida deben ser capaces de proveer una adecuada terminación en cuanto a términos de impedancia se refiere, de forma tal de evitar reflexiones de las señales. Esto que en el pasado se lograba

conectando un resistor cercano al terminal del dispositivo, en la actualidad debe ser implementado dentro del propio FPGA debido a la gran cantidad de terminales que estos dispositivos poseen.

## 2.3.4 Diseño digital en FPGAs

Para describir sistemas digitales, tales como procesadores, memorias o un simple Flip-Flop en FPGA se utilizan lenguaje que describan hardware en FPGAs.

### 2.3.4.1 Lenguaje de descripción de hardware (HDL)

Los lenguajes de descripción de hardware son lenguajes que describen al hardware de los sistemas digitales en forma textual. Se parecen a los lenguajes de programación, pero están orientados específicamente a la descripción de hardware. Sirven para representar diagramas lógicos, expresiones booleanas y otros circuitos digitales más complejos. El contenido en HDL se puede almacenar, recuperar y procesar fácil y eficazmente con software de computadora. Hay dos aplicaciones del procesamiento de HDL: *simulación* y *síntesis*.

La *simulación lógica* es la representación de la estructura y el comportamiento de un sistema lógico digital empleando una computadora. El simulador interpreta la descripción en HDL y produce una salida compresible, digamos un diagrama de temporización, que predice la forma en que se comportará el hardware antes de que se fabrique físicamente. La simulación permite detectar errores funcionales de un diseño sin tener que crear el circuito físico. En el caso de esta residencia se utiliza el software de Quartus II para el diseño y simulación de hardware.

La *síntesis lógica* es el proceso de deducir una lista de componentes y sus interconexiones (lo que se conoce como *lista de circuitos*) a partir del modelo de un sistema digital descrito en HDL. La lista del circuito en el nivel de compuertas sirve para fabricar un circuito integrado o para diagramar una tarjeta de circuitos impreso. La síntesis lógica es similar a la compilación de un programa en un lenguaje de alto nivel convencional. La diferencia es que, en lugar de producir código objeto, la síntesis lógica produce una base de datos con instrucciones para fabricar un circuito digital físico que implementa los enunciados descritos por código HDL (Morris, 2003).

Existen multitud de lenguajes HDL en el mercado (de hecho inicialmente cada fabricante disponía de su propio lenguaje), sin embargo la necesidad de unificación ha hecho que en la actualidad sólo existan dos grandes lenguajes: VHDL y Verilog. Ambos están acogidos a estándares IEEE (VHDL en 1987 y Verilog en 1995). Existen defensores y detractores de cada uno de ellos. Con carácter general se dice que es más fácil aprender Verilog al ser un lenguaje

más compacto[1]. Verilog nació en 1985 como un lenguaje propietario de una compañía (Cadence Design System), pero en 1990 se formó **OVI** (*Open Verilog International*) haciendo dicho lenguaje de dominio público, permitiendo a otras empresas que pudieran emplear Verilog como lenguaje, con objeto de aumentar la difusión de dicho lenguaje.

#### 2.3.4.2 El Lenguaje

La sintaxis de Verilog describe con precisión las construcciones válidas que se puede usar en el lenguaje. En particular, Verilog utiliza cerca de 100 palabras clave: identificadores predefinidos, en minúscula, que define la construcciones del lenguaje. Como por ejemplo de palabras clave podemos citar module, endmodule, input, output, wire, and, or, not, etcétera. Todo texto comprendido entre dos diagonales (//) y el fin de la línea se interpreta como un comentario. Se hace caso omiso de los espacios en blanco y se hace distinción entre mayúscula y minúsculas. El bloque de construcción en Verilog es módulo. Declaramos un módulo con la palabra clave **module** y marcamos su final con la palabra clave **endmodule**. A continuación se muestra su estructura general

1. module <nombre del modulo> ( <señales> );
2. <declaraciones de señales>
3. <funcionalidad del módulo>
4. endmodule

En el ejemplo HDL 2-1 se hace una descripción en HDL del circuito de la figura 2-3. La línea que inicia con dos diagonales es un comentario que explica la función del circuito. La segunda línea declara el modulo, e incluye un nombre y entre paréntesis la señales de E/S. El nombre (circuito_smpl en este caso) es un identificador que sirve para referirse al módulo. Los identificadores son nombres que se dan a las variables para poder referirse a ellas en el diseño. Consta de caracteres alfanuméricos y el carácter de subraya (_), y hay distinción entre mayúscula y minúscula. Los identificadores deben iniciar con un carácter alfabético o subraya; no pueden iniciar con un número. La lista de puertos es la interfaz a través de la cual el módulo se comunica con su entorno. Los argumentos del módulo (que le permiten comunicarse con el exterior) pueden ser de tres tipos:

- **input**: entradas del módulo. Son variables de tipo wire

[1] Un mismo diseño que ocupa menos líneas de código

- **output**: salidas del módulo. Según el tipo de asignación que las genere serán: wire si proceden de una asignación continua o reg si proceden de una asignación procedural.
- **inout**: entrada y salida del módulo. Son variables de tipo wire.

En este ejemplo, los puertos son las entradas y salidas del circuito. La lista de puertos se encierran entre paréntesis y se usan comas para separar los elementos de la lista. El enunciado termina con un signo de punto y coma (;). Todas las palabras clave (que deben estar en minúscula) se han imprimido aquí en negritas por claridad, pero el lenguaje no requiere de esto. A continuación, las declaraciones **input** y **output** definen cuáles puertos son de entradas y salidas. Las conexiones internas se declaran como alambres. El circuito tiene una conexión interna en la terminal *e*, la cual se declara con la palabra **wire**. La estructura del circuito se especifica empleando las compuertas primitivas predefinidas como palabras clave. Cada declaración de compuerta consiste en un nombre opcional (como g1, g2, etc.) seguido de la salida y de las entradas de la compuerta, separada por comas y encerradas entre paréntesis. La salida siempre va primero, seguida de las entradas. Por ejemplo, la compuerta OR se llama *g3*, su salida es $x$ y tiene como entrada $e$ y $y$. la descripción del módulo termina con la palabra clave **endmodule**. Observe que cada enunciado termina con un signo de punto y coma, pero no hay un punto y coma después de **endmodule**.

**Ejemplo HDL 2-1**

```
// Descripción del circuito simple de la fig. 2.26
module circuito_smpl ( A, B, C, x, y);
        input A, B, C;
        output x, y;
        wire e;
        and g1(e, A, B);
        not  g2(y, C);
        or   g3(x, e, y);
endmodule
```

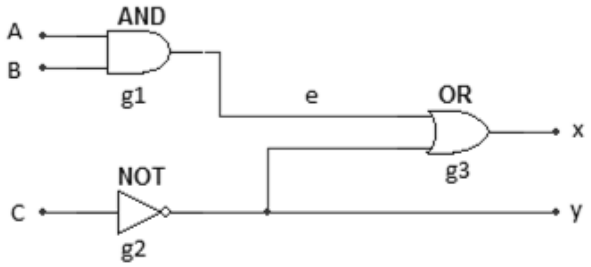



**Figura 2.26.** Circuito para ilustrar HDL

#### 2.3.4.3 Niveles de Modelados en Verilog

En Verilog es posible describir un módulo con cualquiera de las siguientes técnicas de modelados (o mediante una combinación de ellas).

- Modelado en el nivel de compuertas creando ejemplares de compuertas primitivas y módulos definidos por usuario.
- Modelado de flujo de datos empleando enunciado de asignación continua con la palabra clave **assign**.
- Modelado de comportamiento (Behavioral level) utilizando enunciados de asignación procedimentales con la palabra clave **always**.

El modelado a nivel de compuertas describe el circuito especificando las compuertas y cómo se conectan entre sí. El modelado de flujo sirve para describir circuitos combinacionales. El modelo de comportamiento sirve para describir sistemas digitales en un nivel de abstracción más alto.

## 2.4 SISTEMAS DIGITALES

En el mundo actual, el termino digital se ha convertido en parte de nuestros vocabulario común, debido a la dramática forma en que los circuitos y las técnicas digitales se han vuelto tan utilizados en casi todas las áreas de la vida: computadoras, automatización, robots, ciencia médica y tecnología, transporte, telecomunicaciones, entretenimiento, exploración en el espacio, etcétera. En este capítulo se hablaran de los principales componentes que se utilizaron para el desarrollo del proyecto en cuanto al procesamiento de imágenes y para la implementación matemática de la visión artificial.

### 2.4.1 Circuitos secuenciales

Para llevar a cabo el procesamiento de imagen no es necesario utilizar solo circuitos combinacionales si no también elementos de almacenamiento en conjunto. Debido a que los circuitos combinacionales sus salidas dependen exclusivamente de las entradas actuales. Cualquier condición anterior en relación con los niveles de entrada no tiene efecto alguno sobre las salidas actuales, ya que los circuitos lógicos combinacionales no tienen memoria (Tocci, Widmer, & Moss, 2007).

En la Figura 2.27 se presenta un diagrama de bloques de un circuito secuencial. Consiste en un circuito combinacional al que se conectan elementos de almacenamiento para formar una trayectoria de retroalimentación. Lo elementos de almacenamiento son dispositivos capaces de aguardar información binaria. La información almacenada en estos elementos en cualquier momento dado define el estado del circuito secuencial en ese momento. El circuito secuencial recibe información binaria de entradas externas. Esas entradas, junto con el estado actual de los elementos de almacenamiento, determinan el valor binario de las salidas. También determinan la condición para cambiar el estado de los elementos de almacenamiento.

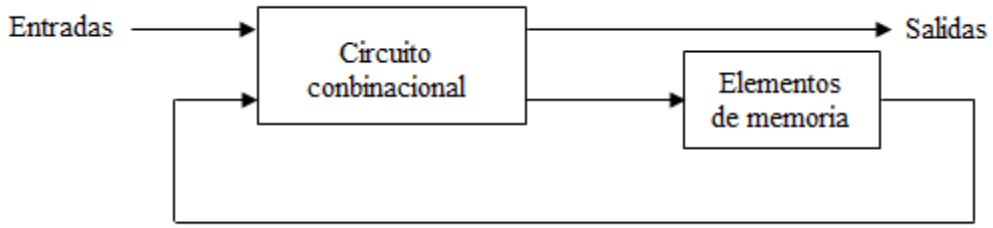



**Figura 2.27.**Diagrama a bloque de un circuito secuencial

Hay dos principales de circuitos secuenciales; síncronos y asíncronos. El comportamiento asíncrono depende de las señales de entrada en cualquier instante dado y del orden en que cambian las entradas. Así, un circuito asincrónico podría considerarse un circuito combinacional con retroalimentación. Un circuito secuencial sincrónico utiliza señales que afectan a los elementos de almacenamiento únicamente en instantes discretos. La sincronización se logra con un dispositivo de temporización llamado generador de reloj, el cual produce un tren periódico de pulsos de reloj. Los pulsos de reloj se distribuyen por todo el sistema de manera que los elementos de memoria solo se vean afectados por cada pulso.

Los elementos de almacenamiento empleados en los circuitos secuenciales con reloj se llaman *flip-flops (FFs)*. Un fli-flop es un dispositivo binario de almacenamiento que puede almacena un bit de información. Un circuito secuencial podría utilizar mucho flip-flops para almacenar tantos bits como sea necesario. En la Figura 2.28 se ilustra el diagrama de cloques de un circuito secuencial síncrono con reloj.

Las salidas puedes provenir del circuito combinacional o de los flip-flops, o de ambos. Los flip-flops reciben sus entradas del circuito combinacional y también de una señal de reloj cuyos pulsos se representan en intervalos fijos de tiempo, como se observa en el diagrama de la entrada

de reloj. El cambio de la salida del flip-flop cambia mientras se activa la señal de reloj ya sea en PGT o NGT, dependiendo de la entrada (Morris, 2003).

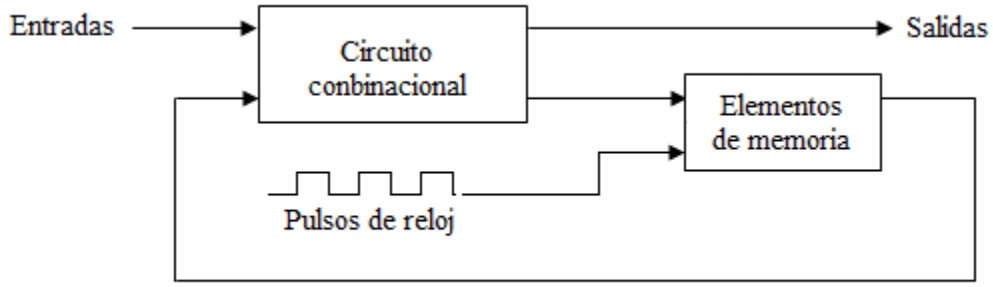



**Figura 2.28.** Diagrama de un circuito secuencial sincrónico con reloj

### 2.4.2 Almacenamiento y transferencia de datos

Hasta ahora, el uso más común de los flip-flops (FFs) es para el almacenamiento de datos o información. Los datos pueden representar valores numéricos (por ejemplo, números binarios, números decimales codificados en BCD) o cualquiera de una amplia variedad de tipos de datos que hayan sido codificados en binario. Por lo general, estos tipos de datos se almacenan en un grupo de FFs, llamados *registros*.

La operación que se realiza con más frecuencia sobre los datos que se almacenan en un FF o un registro es la *transferencia de datos*; lo que implica la transferencia de datos de un FF o registro hacia otro. La Figura 2.31 muestra cómo puede lograrse la transferencia de datos entre dos FFs mediante el uso de flip-flops sincronizados por reloj en S-R, J-K y D. En cada caso, el valor lógico que está almacenado en A del FF se transfiere a B del FF cuando ocurre la NGT del puso Transferir. Por ende, después de esta NGT la salida B será la misma que la salida A.

Las operaciones de transferencia en la Figura 2.29 son ejemplo de transferencia síncrona, ya que se utilizan las entradas de control y CLK síncronas para realizar la transferencia (Tocci, Widmer, & Moss, 2007).

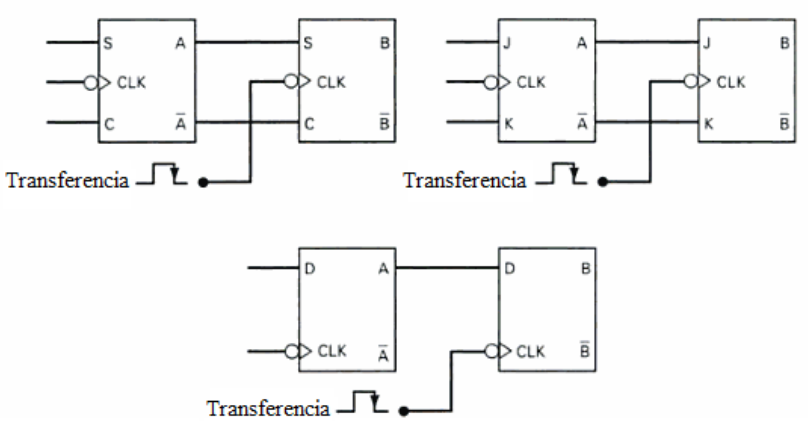



**Figura 2.29.** Operación de transferencia de datos síncrona, realizada por varios tipos de FFs sincronizados por reloj.

### 2.4.2.1 Transferencia de datos en serie: Registro de desplazamiento

Un registro capaz de desplazar su información binaria en una dirección o en la otra se llama *registro de desplazamiento*. La configuración lógica de un registro de desplazamiento consiste en una cadena de flip-flops reciben el pulsos de relojes comunes, que activan el desplazamiento de una etapa a la siguiente.

El registro de desplazamiento más sencillo posible usa sólo flip-flops, como se indica en la Figura 2.30. Cada pulso de reloj desplaza el contenido del registro una posición de bit a la derecha. La entrada en serie determina que sucede en el flip-flop de la extrema izquierda durante el desplazamiento. La salida en serie se toma de la salida del flip-flop de la extrema derecha. A veces es necesario controlar el desplazamiento de modo que sólo se efectúe con ciertos pulsos, pero no con otros. Esto se logra inhibiendo el reloj de la entrada del registro para impedir que se desplace. Decimos que un sistema digital opera en modo serie cuando la información se transfiere y manipula bit por bit. La información se transfiere bit por bit desplazando los bits del registro de origen hacia el registro de destino. Esto se contrasta con la transferencia paralela, en la que todos los bits del registro se transfieren al mismo tiempo (Morris, 2003).

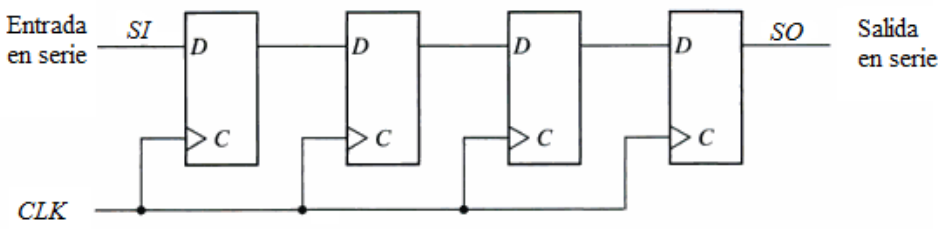
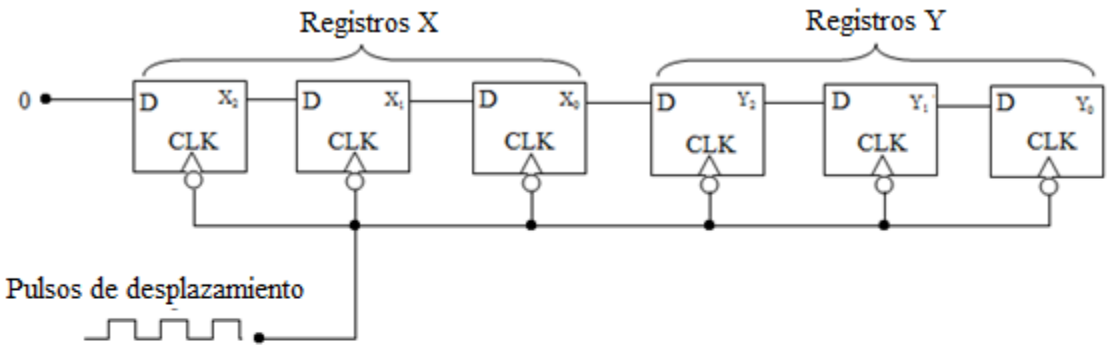



**Figura 2.30.** Registro de desplazamiento de cuatro bits

La figura 2.31 (a) muestra dos registro de desplazamiento de tres bits, conectados de forma tal que el contenido del registro X se transferirá en serie hacia el registro Y. Observe como $X_0$, el último FF del registro X, está conectado a la entrada D de $Y_2$, el primer FF del registro Y. por ende, a medida que se aplican los pulsos la transferencia de información se realiza de la siguiente manera: $X_2 \rightarrow X_1 \rightarrow X_0 \rightarrow Y_2 \rightarrow Y_1 \rightarrow Y_0$.




(a)

| $X_2$ | $X_1$ | $X_0$ | $Y_2$ | $Y_1$ | $Y_0$ | |
|---|---|---|---|---|---|---|
| 1 | 0 | 1 | 0 | 0 | 0 | ← Antes de la aplicación de pulsos |
| 0 | 1 | 0 | 1 | 0 | 0 | ← Después del primer pulso |
| 0 | 0 | 1 | 0 | 1 | 0 | ← Después del segundo pulso |
| 0 | 0 | 0 | 1 | 0 | 1 | ← Después del tercer pulso |

(b)

**Figura 2.31.** Transferencia en serie de información, del registro X al registro Y.

El FF $X_2$ cambiara a un estado determinado con base en su entrada *D*. por ahora, *D* se mantendrá en bajo, de manera que $X_2$ cambiara abajo en el primer pulso y permanecerá ahí.

Para ilustrar lo anterior supongamos que antes de aplicar cualquier pulso de desplazamiento el contenido del registro X es 101 (es decir, $X_2$=1, $X_1$=0, $X_0$=1) y que el registro Y esta en 000. Observe la tabla en la figura 2.30 (b), la cual muestra cómo cambian los estados de cada FF a medida que se aplican los pulsos de desplazamiento (Tocci, Widmer, & Moss, 2007).

### 2.4.3 Aritmética digital

Las computadoras y calculadoras digitales realizan las diversas operaciones aritméticas sobre números que se representan en forma binaria. En el procesamiento de imagen aplicando la visión

artificial se hace uso de la aritmética digital para representar toda la matemática o algoritmos complejos que se aplican para su utilización. El tema de aritmética digital pude ser muy complejo si queremos comprender todos los diversos métodos de cálculo y la teoría detrás de ello.

### 2.4.3.1 Representación de números con signos

En las computadoras digitales, los números binarios se representan mediante un conjunto de dispositivos de almacenamiento binario (por ejemplo, flip-flops). Cada dispositivo representa un bit. Por ejemplo, un registro FF de seis bits puede almacenar número binarios que varían desde 000000 hasta 111111 (de 0 a 63 en decimal). Esto representa la *magnitud* del número. Debido a que la mayoría de las computadoras y calculadoras digitales manejan números tanto negativos como positivos, se requiere de algún medio para representar el *signo* del número (+ o -). Por lo general lo que se hace es agregar otro bit al número; a este bit se le llama **bit de signo**. En general, la convención común es que un 0 en el signo representa a un número positivo y un 1 en el bit de signo representa a un número negativo. Esto se ilustra en la figura x.x. El registro A contiene los bits 0110100. El 0 en el bit mas a la izquierda ($A_6$) es el bit del signo que representa al signo +. Los otros seis bits son la magnitud del numero $110100_2$, el cual es equivalente al 52 decimal. Entonces, el número almacenado en el registro A es +52. De manera similar, el número almacenado en el registro B es -52 ya que el bit del signo es 1, que representa al -.

El bit de signo se utiliza para indicar la naturaleza positiva o negativa del número binario almacenado. Los números de la figura 2.32 consisten de un bit de signo y de seis bits de magnitud. Los bits de magnitud son el equivalente binario real del valor decimal que se está representando. A este se le conoce como sistema de **signo-magnitud** para representar números binarios con signos.

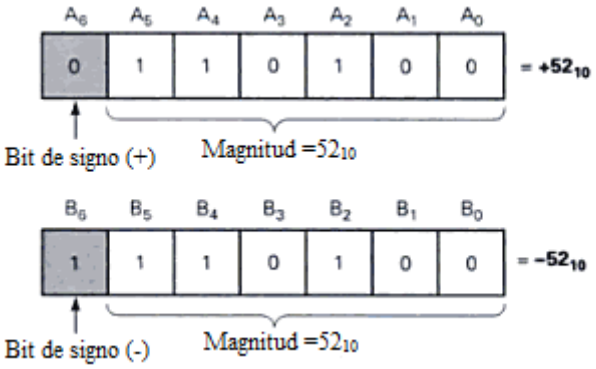



**Figura 2.32.** Representación de numero con signo en la forma signo-magnitud.

Aunque el sistema de signo-magnitud es simple, las calculadoras y computadoras no lo utilizan de manera usual porque la implementación del circuito es más compleja que en otros sistemas. El sistema más común que se utiliza para representar números binarios con signo es el **sistema de complemento a 2**.

### 2.4.3.2 Complemento a 2

El complemento a 2 de un numero binario se forma al tomar el complemento a 1 de cualquier numero binario y sumarle 1 a la posición del bit menos significativo. El proceso se ilustra a continuación para el número $101101_2 = 45_{10}$.

| | |
|---|---|
| 1 0 1 1 0 1 | Numero binario original |
| 0 1 0 0 1 0 | Complemento a 1 |
| + 1 | Se le suma 1 para formar el complemento a 2 |
| 0 1 0 0 1 1 | Complemento a 2 del número original binario original |

El sistema de complemento a 2 para representar números con signo funciona así:

- Si el numero es positivo, la magnitud se representa en su forma binaria real y se coloca el bit de signo 0 enfrente del MSB. Esto se muestra en la figura 2.35 para el número $+45_{10}$.
- Si el número es negativo, la magnitud se representa en su forma de complemento a 2 y se coloca el bit de signo 1 enfrente del MSB. Esto se muestra en la **Figura 2.35** para el número $-45_{10}$.


**Figura 2.33.** Representación de número con signo en el sistema de complemento.

El sistema de complemento a 2 se utiliza para representar números con signo ya que, permite realizar operaciones de resta a partir de una suma. Esto es importante ya que significa que en una computadora digital puede utilizar los mismos circuitos tanto para sumar como para restar, lo cual redunda en un ahorro en el hardware.

### 2.4.3.3 Unidad aritmética/lógica

Todas las operaciones aritméticas se llevan a cabo en la **unidad aritmética/lógica (ALU)** de una computadora. La figura 2.34 es un diagrama de bloques en el que se muestran los elementos principales que se incluyen en una ALU común. El propósito principal de la ALU es recibir datos binarios que se almacenan en la memoria y ejecutar operaciones aritméticas y lógicas sobre estos datos, de acuerdo con las instrucciones provenientes de la unidad de control.

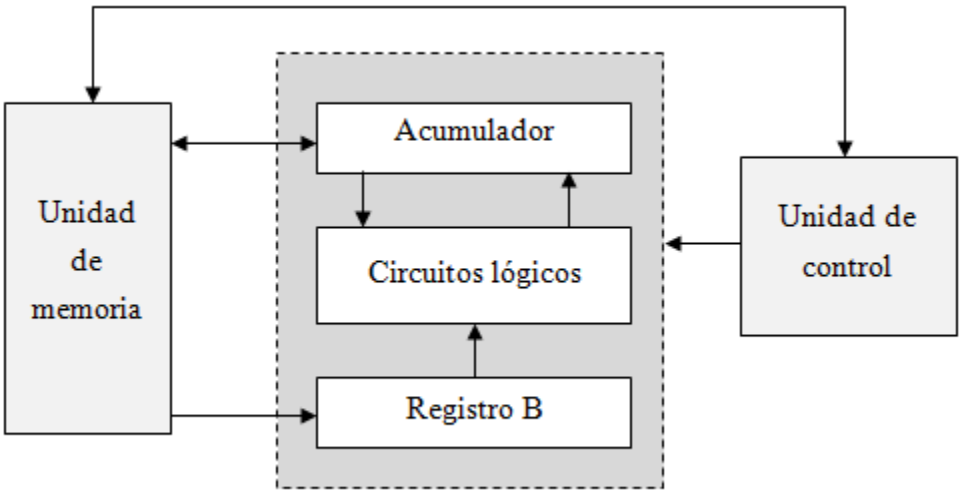



**Figura 2.34.** Unidad aritmética/lógica

La unidad aritmética/lógica contiene al menos dos registro de flip-flops: el registro B y el registro acumulador. También contiene lógica combinacional, la cual realiza las operaciones aritméticas y lógicas sobre los números binarios que se almacenan en el registro B y en el acumulador. Una secuencia común de operaciones puede ocurrir de la siguiente manera:

1. La unidad de control recibe una instrucción (de la unidad de memoria) en la que se especifica que un numero almacenado en cierta posición de memoria (dirección) se debe sumar al número que se encuentra almacenado en el registro acumulador.
2. El número que se va a sumar se transfiere de la memoria al registro B.
3. El número en el registro b y el número en el registro acumulador se suman en los circuitos lógicos (cuando lo ordena la unidad de control). La suma resultante se envía al acumulador para que se almacenada.
4. El nuevo número en el acumulador puede permanecer ahí para que se le pueda sumar otro número, o si termino ese proceso aritmético, pueda transferirse de vuelta a la memoria para su almacenamiento.

Estos pasos deben dejar claro de donde proviene el nombre del registro acumulador. Este registro "acumula" las sumas que ocurren cuando se realiza suma anterior. De hecho, para cualquier problema aritmético que contengan varios pasos, el acumulador, por lo general, contiene los resultados de los pasos intermedios a medida que se van completando, junto con el resultado final cuando se termina el problema (Tocci, Widmer, & Moss, 2007).

### 2.4.4 Dispositivos de memoria

Unos de los principales de los sistemas digitales sobre los analógicos es su habilidad para almacenar con facilidad grandes cantidades de información y datos digitales, durante periodos cortos y largos. Esta capacidad de memoria es lo que hace a los sistemas digitales tan versátiles y adaptables a muchas situaciones.

Las memorias están hechas de grandes cantidades de flip-flops o registros en un solo chip, ordenados en diversos formatos de arreglos de memoria. Estas memorias semiconductoras y MOS son los dispositivos de memorias más veloces disponibles.

Las memorias semiconductoras se utilizan como memoria principal de una computadora (Figura 2.35), en donde la operación rápida es importante. La memoria principal de una computadora (también conocida como su memoria de trabajo) está en comunicación constante con la unidad central de procesamiento (CPU) a medida que se ejecuta un programa de instrucciones. El programa y cualquier información que este utilice residen en la memoria principal mientras la computadora trabaja con ese programa. La memoria principal está conformada por memoria RAM y ROM.

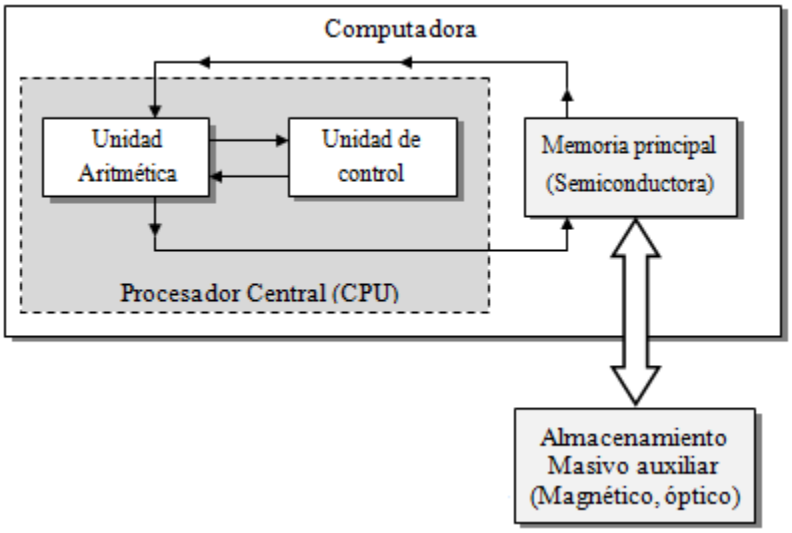



**Figura 2.35.**Un sistema computacional utiliza, por lo general, memoria principal de alta velocidad y memoria auxiliar externa más lenta.

La memoria auxiliar es otra forma de almacenamiento en la computadora (Figura 2.35); esta memoria auxiliar (también conocida como almacenamiento masivo) está separada de la memoria de trabajo principal y tiene la capacidad de almacenar cantidades masiva de información, sin necesidad de energía electrónica. La memoria auxiliar opera a una velocidad mucho más lenta que la memoria principal; almacena programas y datos que la CPU no utiliza en ese momento. Esta información se transfiere a la memoria principal cuando la computadora la necesita (Tocci, Widmer, & Moss, 2007).

#### 2.4.4.1 Operación general de la memoria

Una unidad de memoria almacena información binaria en grupo de bits llamados **palabras**. Una palabra de memoria es una entidad de bits que siempre se guardan o sacan juntos, como una unidad. Una palabra de memoria es un grupo de unos y ceros o un conjunto de bits. Un grupo de 8 bits es un **byte**. Casi todas las memorias de computadora manejan palabras cuya longitud es un múltiplo de ocho bits. Así, una palabra de 16 bits contiene dos bytes, y una de 32 bits consta de cuatro bytes. La capacidad de una unidad de memoria por lo regular se da como el número total de bytes que es capaz de guardar (Morris, 2003).

Cada tipo de memoria es distinto en su operación interna, ciertos principios de operación básicos son iguales para todos los sistemas de memoria.

Todo sistema de memoria requiere varios tipos distintos de líneas de entrada y de salida para realizar las siguientes funciones:

1. Seleccionar la dirección en memoria a la que se va a acceder para una operación de lectura y escritura.
2. Seleccionar una operación de lectura o de escritura a realizar.
3. Suministrar los datos de entrada que se va almacenar en memoria durante una operación de escritura.
4. Retener los datos de entrada que provienen de la memoria durante una operación de lectura.
5. Habilitar (o deshabilitar) la memoria, de manera que responda (o no) a las entradas de dirección y a la línea de selección de lectura/escritura.

La figura 2.35 ilustra estas funciones básicas en un diagrama simplificado de una memoria de 32 x 4, la cual almacena 32 palabras de cuatro bits. Como el tamaño de palabra es de cuatro bits, hay cuatro líneas de entrada de datos ($I_0$ a $I_3$) y cuatro líneas de salida de datos ($O_0$ a $O_3$). Durante una

operación de escritura, los datos que se van a almacenar en memoria deben aplicarse a las líneas de entrada de datos. Durante una operación de lectura, la palabra que se va a leer de memoria aparece en las líneas de salida de datos.

**Entrada de dirección**

Como esta memoria almacena 32 palabras, tiene 32 distintas ubicaciones de almacenamiento y, por lo tanto, 32 distintas direcciones binarias, las cuales varían de 00000 a 11111 (de 0 a 31 en decimal). Por ende, hay cinco entradas de dirección ($A_o$ a $A_4$).

Para acceder a una de las ubicaciones de memoria para una operación de lectura o de escritura se aplica el código de dirección de cinco bits para esa ubicación específica a las entradas de dirección. En general, se requieren $N$ entradas de dirección para una memoria con una capacidad de $2^N$ palabras.

Podemos visualizar la memoria de la figura 2.36 (a) como un arreglo de 32 registros, en donde cada registro almacena una palabra de cuatro bits, como se muestra en la figura 2.36 (b). La ubicación de cada dirección que se muestre contiene cuatro celdas de memoria que se retienen 1s y 0s, lo cuales conforman la palabra de datos que se almacena en esa ubicación. Por ejemplo, la palabra de datos 0010 se almacena en la dirección 00000, la palabra de datos 1001 se almacena en la dirección 00001, y así en lo sucesivo.

**La entrada *R/W***

Esta entrada que operación de memoria se va a realizar: lectura (R) o escritura (W). La entrada se identifica como $R/\overline{W}$; no hay barra sobre la *R*, lo cual indica que la operación de escritura se realiza cuando $R/\overline{W} = 1$. La barra sobre $W$ indica que la operación de escritura se realiza cuando $R/\overline{W} = 0$. A menudo se utilizan otras etiquetas para identificar esta entrada. Dos de las más comunes son $\overline{W}$ (escritura) y $\overline{WE}$ (habilita escritura). De nuevo, la barra indica que la operación de escritura se lleva a cabo cuando la entrada esta en BAJO. En estos últimos casos se sobreentiende que la operación de lectura ocurre cuando la entrada esta en ALTO.

**Habilitación de memoria**

Muchos sistemas de memoria tienen ciertos medios para deshabilitar por completo toda o parte de la memoria, de manera que no responda a las entradas. Esto se presenta en la **Figura 2.38** como la entrada HABILITACION DE MEMORIA (MEMORY ENABLE), aunque puede tener distintos nombres en los diversos sistemas de memoria, como habilitación de chip (*CE*, acrónimo de CHIP ENABLE).

Todos los tipos de memoria tienen el mismo principio de funcionamiento, en los siguientes capítulos se da a conocer las diferentes memoria.

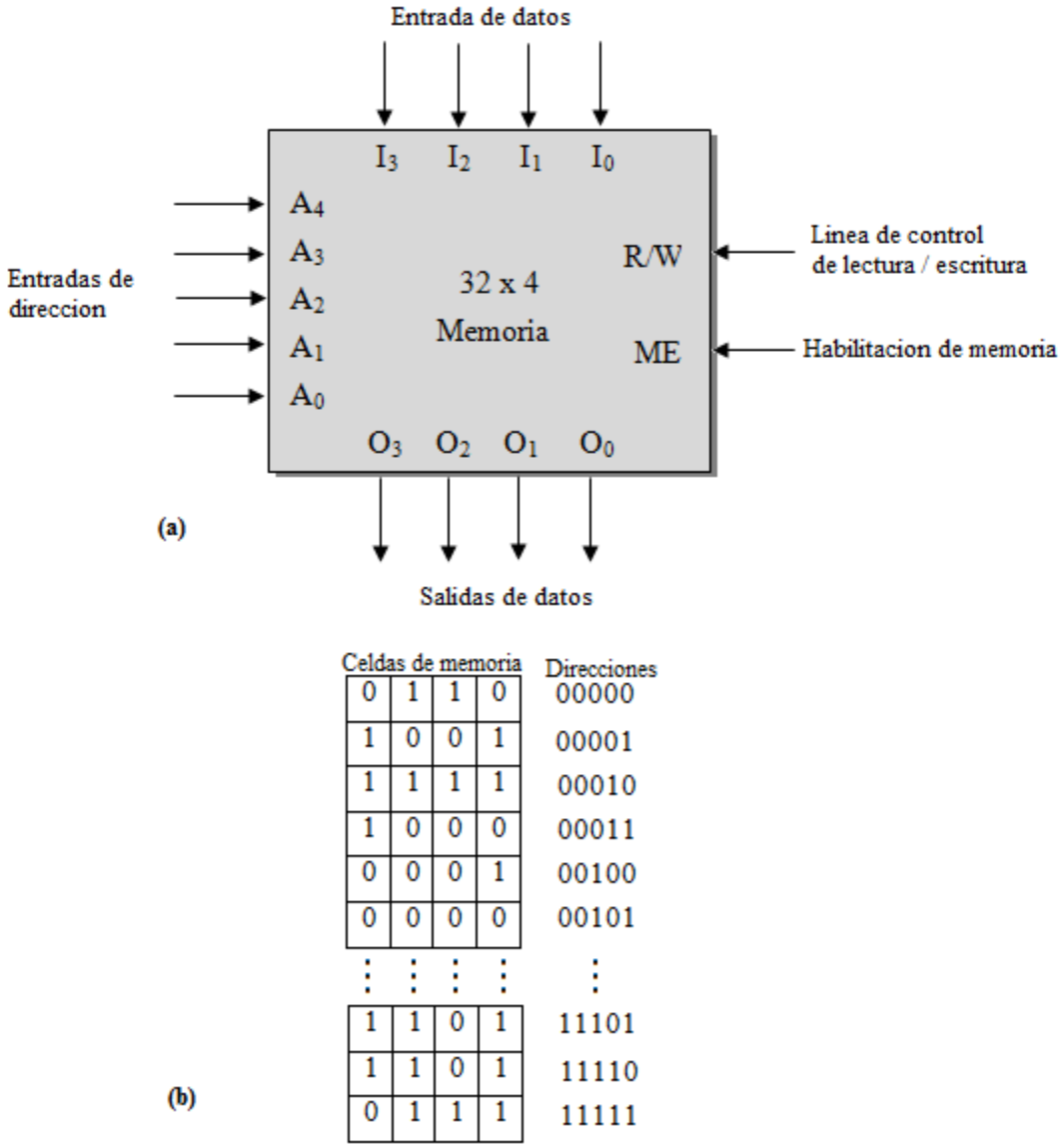



**Figura 2.36.** (a) Diagrama de una memoria de 32x4; (b) arreglo virtual de las celdas de memoria en 32 palabras de cuatro bits.

### 2.4.4.2 MEMORIA DE ACCESO ALEATORIO (RAM)

Una unidad de memoria es un conjunto de celdas de almacenamiento junto con los circuitos asociados que se requieran para transferir información al y del dispositivo. El tiempo que toma transferir información a o de cualquier posición al azar deseada siempre es el mismo, de ahí el nombre *memoria de acceso aleatorio* o RAM.

Las RAM se utilizan en las computadoras para el almacenamiento de programas y datos. Cuando la computadora ejecute un programa, se realizaran operaciones de lectura y escritura rápido para la RAM. Para ello se requiere tiempo de ciclos de lectura y de escritura rápidos para la RAM, de manera que no disminuya la velocidad de operación de la computadora.

La principal desventaja de la RAM es que es volátil y perderá toda la información almacenada si se interrumpe o se desconecta la energía. Sin embargo algunas RAM tipo CMOS utilizan cantidades tan pequeñas de energía en el modo de suspensión (sin que realice operaciones de lectura y escritura) que pueden operar mediante baterías cuando se interrumpe la energía principal. Desde luego que la principal ventaja de la RAM es que se puede escribir en ella y leer de ella con la misma rapidez y velocidad.

### 2.4.4.3 RAM ESTATICA (SRAM)

La memoria RAM almacena datos mientras se aplique una energía al chip. En esencia, las celdas de memoria de la RAM estáticas son flip-flops que pertenecerán en un momento dado (almacena un bit) de manera indefinida, siempre y cuando no se interrumpa la energía del circuito.
Las RAMs estáticas (SRAM) están disponibles en las tecnologías bipolares, MOS y BiCMOS; la mayoría de las aplicaciones utilizan RAM tipo NMOS o CMOS. Los dispositivos bipolares tienen la ventaja en velocidad (aunque CMOS está cada vez más cerca) y los dispositivos MOS tienen mucha mayor capacidad y bajo consumo de energía.

### 2.4.4.4 RAM DINAMICA (DRAM)

Las RAM dinámicas se fabrican mediante el uso de tecnología MOS y se distinguen por su alta capacidad, bajo requerimiento de energía y velocidad moderada de operación. A diferencia de las RAMs estáticas que almacena información en FFs, las RAMs dinámicas almacenan 1s y 0s en forma de cargas en un pequeño capacitor MOS (por lo general, de unos cuantos picos-Farads). Debido a que la tendencia de estas cargas de fugarse después de un periodo de tiempo, las RAMs dinámicas requieren una recarga periódica de las celdas de memoria; a esto se le conoce como generar la RAM dinámica. En los chips DRAM modernos, cada celda de memoria debe generase casa 2,4 u 8 ms, o se perderá sus datos.

La necesidad de regenerarse es una desventaja de la RAM dinámica en comparación con la RAM estática, debido a que puede requerir circuito de soporte externos. Algunos chips DRAM tienen circuito de generación integrados, los cuales no requieren hardware externo adicional, pero si una sincronización especial de las señales de control de entrada del chip.

Para aplicaciones en la que la velocidad y la reducción en complejidad son más importantes que las consideraciones de costo, espacio y energía, las RAMs estáticas siguen siendo la mejor opción. Por lo general, son más veloces que la RAMs dinámicas y no requieren operación de regeneración. Es más fácil diseñar circuitos con RAM estáticas, pero no puede competir con la mayor capacidad y el bajo consumo de energía de las RAMs dinámicas.

#### 2.4.4.5 MEMORIA SDRAM

La DRAM síncrona está diseñada para trasferir datos en ráfagas de disparo rápido de varias ubicaciones de memoria secuenciales. La primera ubicación a la que se accede es la más lenta, debido a la sobre carga (latencia) del proceso de fijar la dirección de fila y columna. Después el reloj del sistema de bus aplica pulso de reloj a los valores de datos (en vez de la línea de control $\overline{CAS}$) en ráfagas de ubicaciones de memoria dentro de la misma página. En su interior, las SDRAMs se organizan dos bancos. Esto permite leer datos a una velocidad muy rápida, ya que se accede en forma alternativa a cada unos de los dos bancos. Para proveer todas las características y la flexibilidad necesaria para que este tipo de DRAM funcione con una amplia variedad de requerimiento de sistema, los circuitos dentro de la SDRAM se han vuelto más complicados. Se necesita una secuencia de instrucciones para indicar a la SDRAM que opciones son necesarias, como la longitud de la ráfaga, los datos secuenciales o interpaginados.

# 3 Desarrollo del Proyecto

El desarrollo de esta residencia tendrá como objetivo principal, desarrollar hardware en FPGA encargado de procesar imágenes digitales para la implementación de algoritmos utilizados en la visión artificial. Una de las etapas principales del procesamiento de imágenes es la adquisición de los datos y la adecuación de los mismos al resto del diseño en FPGA. El desarrollo se divide en tres etapas: las adquisiciones de imágenes, la implementación de algoritmos de visión artificial y por último la visualización de resultados utilizando una pantalla TFT-LCD (Figura 3.1).

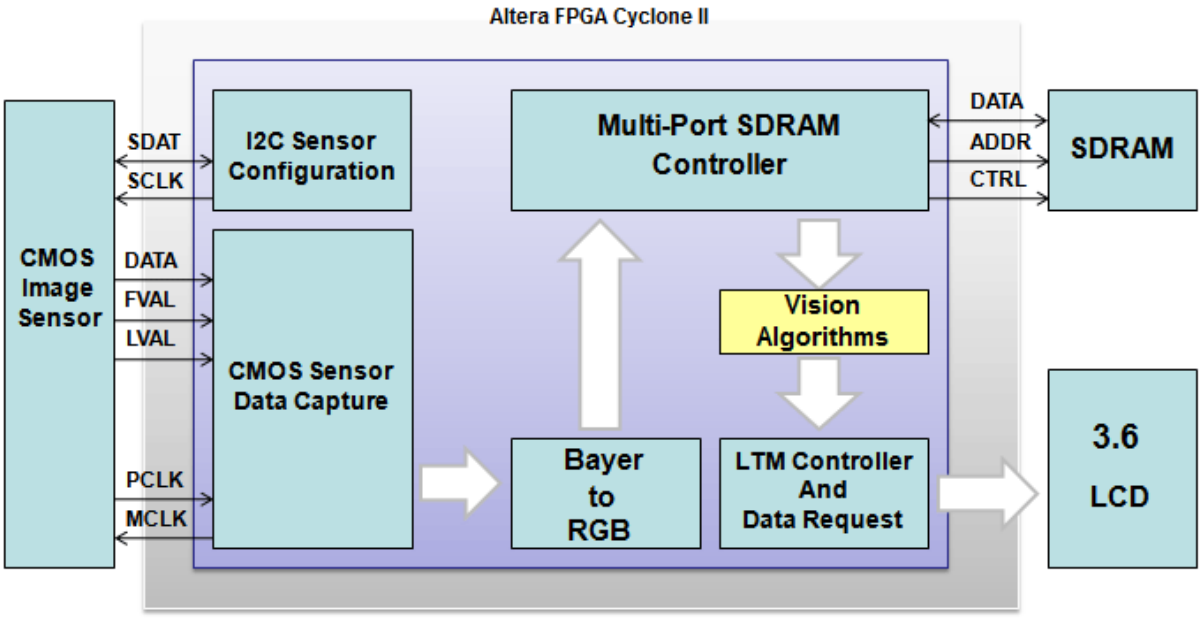



**Figura 3.1.** Diagrama general del sistema procesador de imágenes

Para el diseño de un computador encargado al procesamiento digital de imágenes, se utiliza el tarjeta de altera DE2 FPGA Cyclone II (Figura 3.2). Para la adquisición de imágenes utiliza el sensor CMOS de la cámara TRDB_CD2 de la Figura 3.2 (a) y para la visualización de video o imágenes se utiliza la pantalla 3.6¨ TRDB_LCM con tecnología TFT-LCD de matriz activa, como se ilustrar en la Figura 3.2 (b).

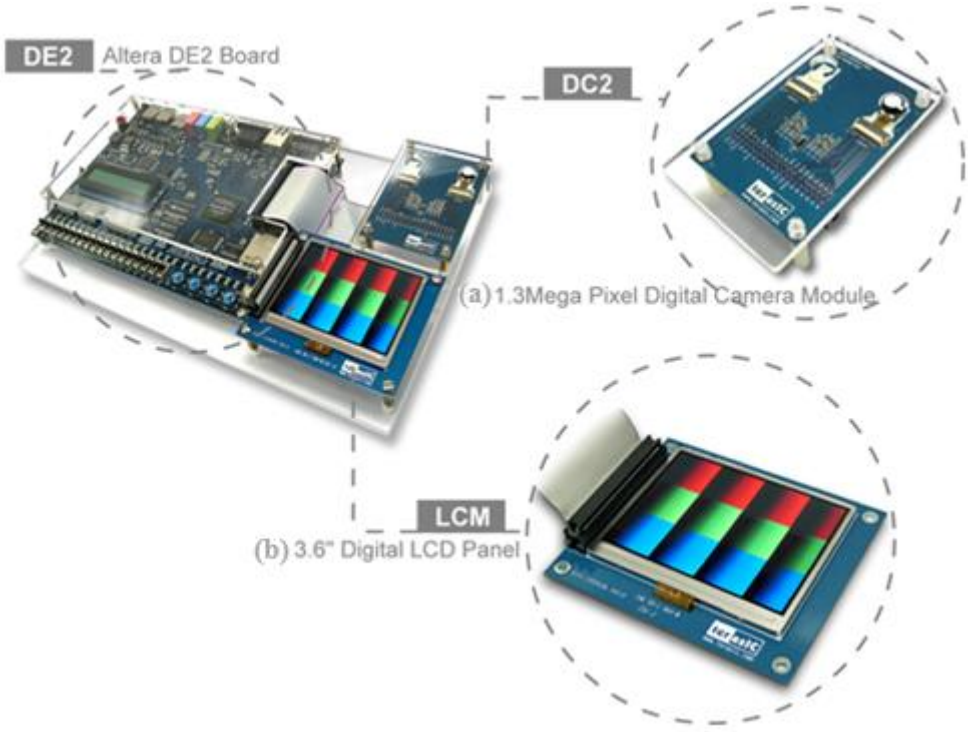



**Figura 3.2.**Tarjeta Altera DE2: (a) Cámara digital de 1.3 Mega Pixel; (b) Pantalla LCD Digital de 3.6”

## 3.1 ADQUISICIÓN DE IMÁGENES EN FPGA

En esta etapa ofrece un plan sobre la adquisición de alta velocidad y el proceso en tiempo real de datos de imágenes basado en FPGA (Field Programmable Gate Array). Este diseño se las arregla para adquirir datos de imágenes digitales y almacenarlos dentro de una memoria de tipo SDRAM. Este diseño se distingue de los tradicionales sistemas de adquisición de imágenes, y proporciona una solución universal para aplicaciones con imágenes digitales. En general la adquisición de imágenes en FPGA se basa en las siguientes etapas: Captura de imágenes, Conversión de imagen Bayer a RGB y Almacenamiento de imágenes en memoria RAM, (Figura 3.1).

La adquisición de imágenes en FPGA es una de las etapas más importante para trabajar con sistemas de visión artificial. Dentro de esta etapa juega un papel muy importante la cámara que se desea utilizar, debido que la programación estará enfocada a las señales que necesite para su configuración y captura de imágenes. La cámara que se utiliza en este proyecto contiene dos sensores CMOS MT9M011 de 1/3 inch (Figura 3.3) del cual solo uno será utilizado.

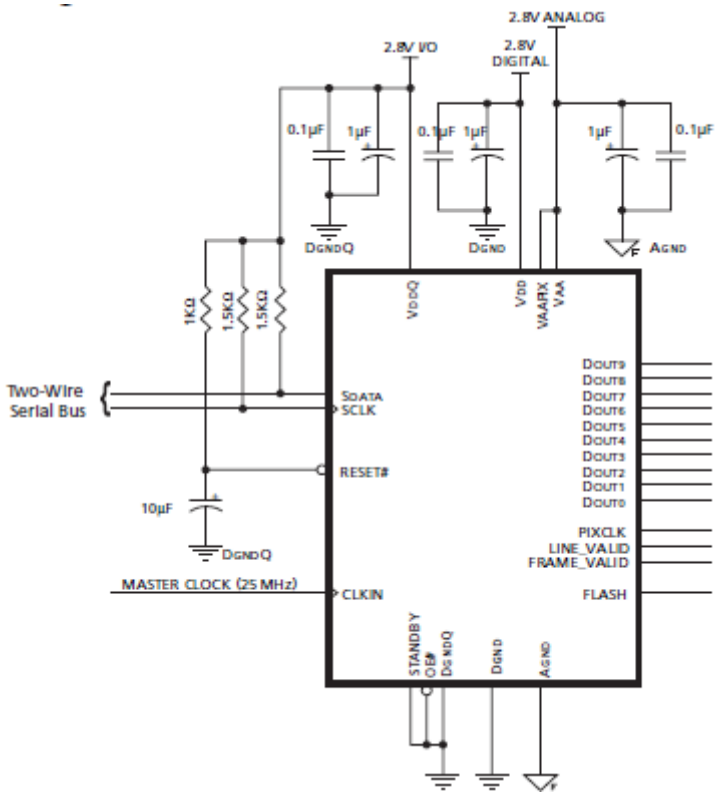



**Figura 3.3.** Sensor de imagen CMOS MT9M011

La estructura interna del sensor CMOS se muestra en la figura 3.4, en donde se observa las diferentes etapas interna que realiza el sensor para enviar y recibir señales digitales.

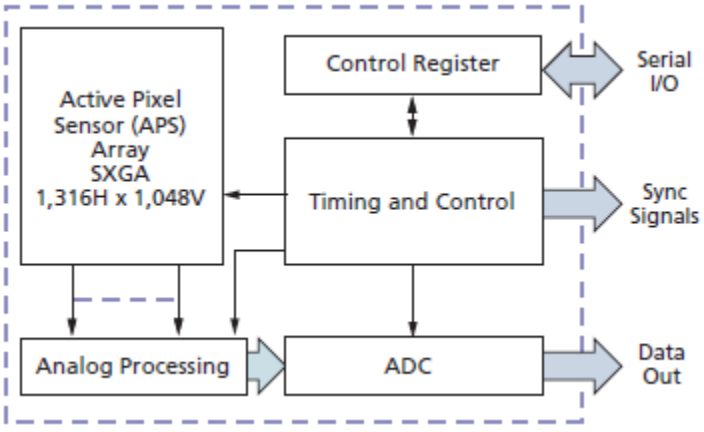



**Figura 3.4.** Diagrama a Bloque del Sensor CMOS de Imagen

El sensor puede ser operado en su modo predeterminado o programado por el usuario a través de un simple cable de dos líneas de interfaz serie I2C (SDAT, SCLK) en donde se elije el tamaño del frame o resolución de la imagen, la exposición, ajuste de ganancia, y otros parámetros.

Después de haberse programado el sensor genera las señales de datos (Dout) de 10 bits y las señales de sincronización PIXCLK, LINE_VALID y FRAME_VALID.

### 3.1.1 Captura de imagen

En esta etapa se captura una imagen completa a través de las señales DATA, FVAL, LVAL y PCLK. El sensor genera estas señales mientras la señal de reloj maestro MCLK esté funcionando.

La frecuencia de reloj maestro recomendada es de 25 MHz y se obtiene utilizando un contador asíncrono de flip-flop tipo JK con una entrada de reloj de 50MHZ, para obtener una señal de 25MHZ.

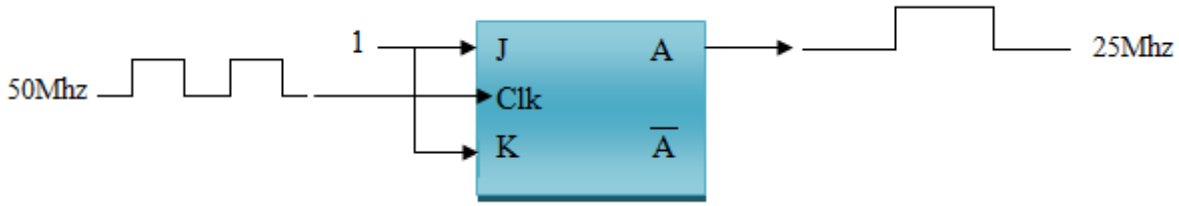



**Figura 3.5.** Obtención de una señal de 25Mhz

La señal de PIXCLK es, nominalmente, la invertida del reloj maestro, lo que permite PIXCLK para ser utilizado como un reloj para trabajar los datos. Esta señal de reloj es activado continuamente, incluso durante el período de blanqueo.

La adquisición de datos de 10 bits o lo que es un pixel en la imagen por cada periodo PIXCLK, se realiza hasta que la señal LINE_VALID este en alto. Cuando la señal LINE_VALID este en bajo, en ese momento se forma una línea en la imagen y se deja de adquirir datos, pasando a la etapa de blanqueo (Figura 3.6).

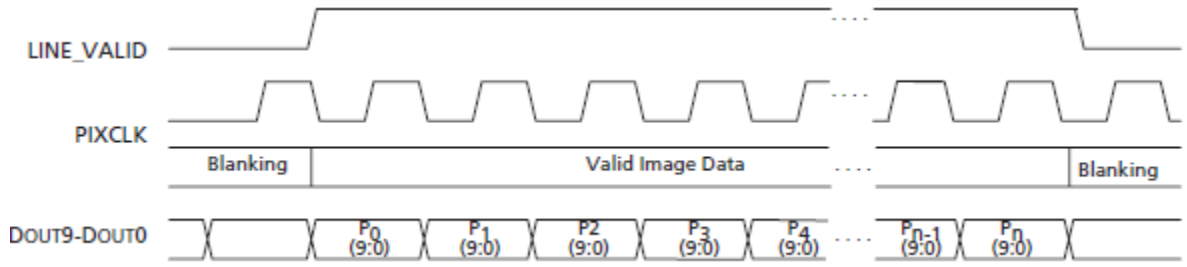



**Figura 3.6.** Sincronía Horizontal

La captura de líneas se realiza mientras FRAME_VALID este activado. Cuando FRAME_VALID este en bajo, en ese momento se forma una imagen o cuadro, por un número de

líneas capturados, que depende de la resolución con la que se haya programado el sensor MT9M01 (Figura 3.7).


**Figura 3.7**. Sincronía Vertical

La captura de una imagen o cuadro se realiza con el circuito que se observa en la figura 3.8. Donde la señal PIXCLK se pasa por el reloj principal de 10 flip-flop tipo D conectados en paralelos. En la entrada D0 [9:0] van conectados las líneas de datos de 10 bits y en la salida Q0 [9:0] la serie de datos de cada componente de cada pixel formando así una línea en la imagen.

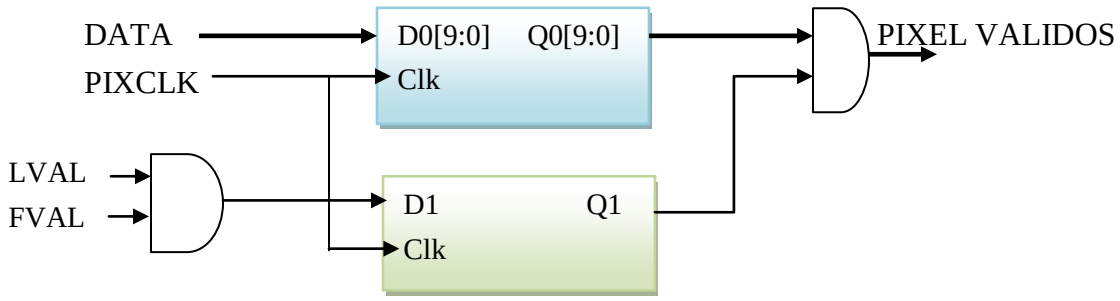



**Figura 3.8.** Adquisición de pixel validos

La señal D1 indica que cuando FRAME_VALID (FVAL) y LINE_VALID (LVAL) estén activados podemos adquirir datos. Las señales Q0 y Q1 se pasan a una compuerta AND para que los pixeles sean validos mientras Q1 este activo. En pocas palabras solo se captura pixeles validos para formar una imagen como el que se muestra en el ejemplo en la figura 3.9.

La cámara se programo mediante la comunicación I2C para una resolución de 1280x1024 como se muestra en la figura 3.9. Por lo tanto, se deja pasar 1024 líneas por cada cuadro de un ancho de 1280 pixeles de 10 bits de resolución. En la figura 3.6 y 3.7 se observa los diagramas a tiempo para formar una imagen o cuadro como se muestra en la figura 3.9.

.

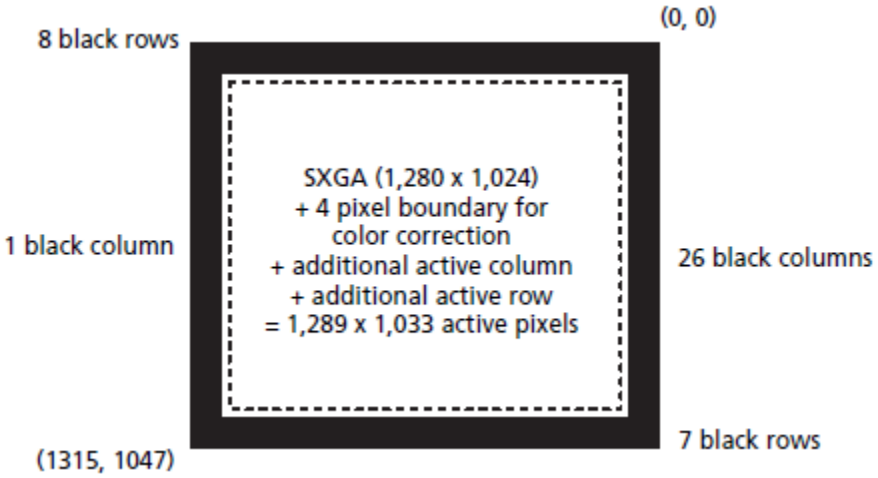



**Figura 3.9.** Resolución del sensor CMOS

Las imágenes que se generan o los datos captados por el sensor CMOS MT9M011 se adquieren en formato Bayer, así como se muestra en la figura 3.10, en donde se observa que está compuesta por RG-GB por cada cuadro de 2x2. Para trabajar con algoritmos de visión artificial es necesario que la imagen se convierta a formato RGB.

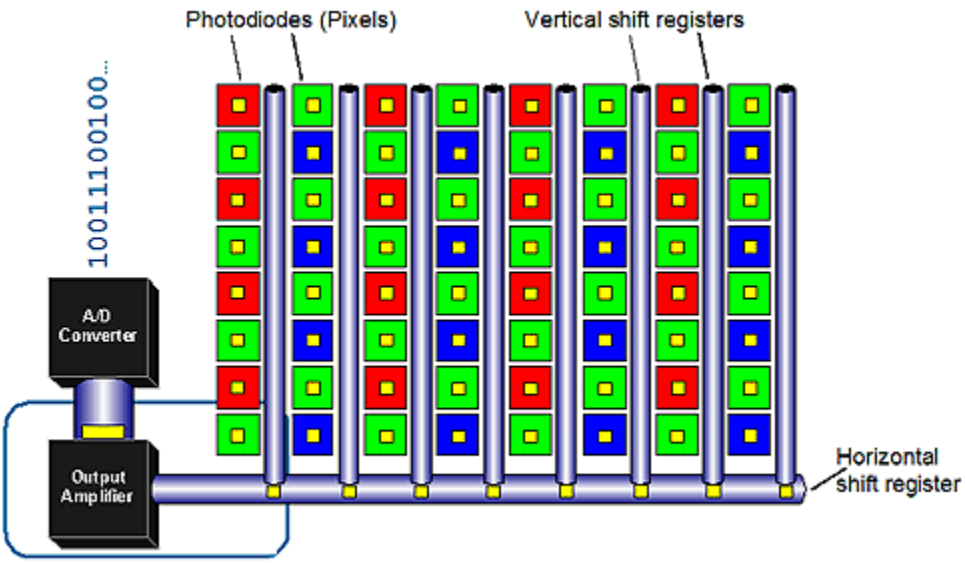



**Figura 3.10.** Captura del datos Bayer del sensor CMOS

## 3.1.2 Conversión de Bayer a formato RGB

La matriz de Filtro Bayer de colores es un formato popular para la adquisición digital de imágenes en color. El patrón de los filtros de color se muestra a continuación en la figura 3.11. La

mitad del número total de píxeles es de color verde (G), mientras que un cuarto del número total se asigna a ambas rojo (R) y azul (B).

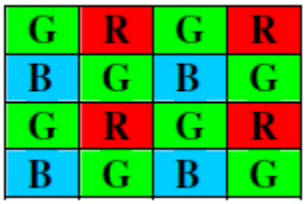



**Figura 3.11**.Patron de Formato Bayer

Para convertir una imagen de Bayer a un formato RGB, hay que interpolar los dos valores de color que falta en cada píxel. Para realizar esta tarea se implementa registro de desplazamiento en serie. Estos registros se implemento utilizando la herramienta MegaWizard Plug-In Manager de Quartus II el cual trae precargados componentes, facilitando la tarea de estar programando. Para la implementación se utiliza el componente llamado Shift Register como se muestra en la figura 3.12. Este componente se programa para entrada de datos de 10 bits y de almacenamiento 2 líneas de la imagen de tamaño de 1280 datos por línea, debido a que la imagen Bayer se repite su forma a cada dos líneas.

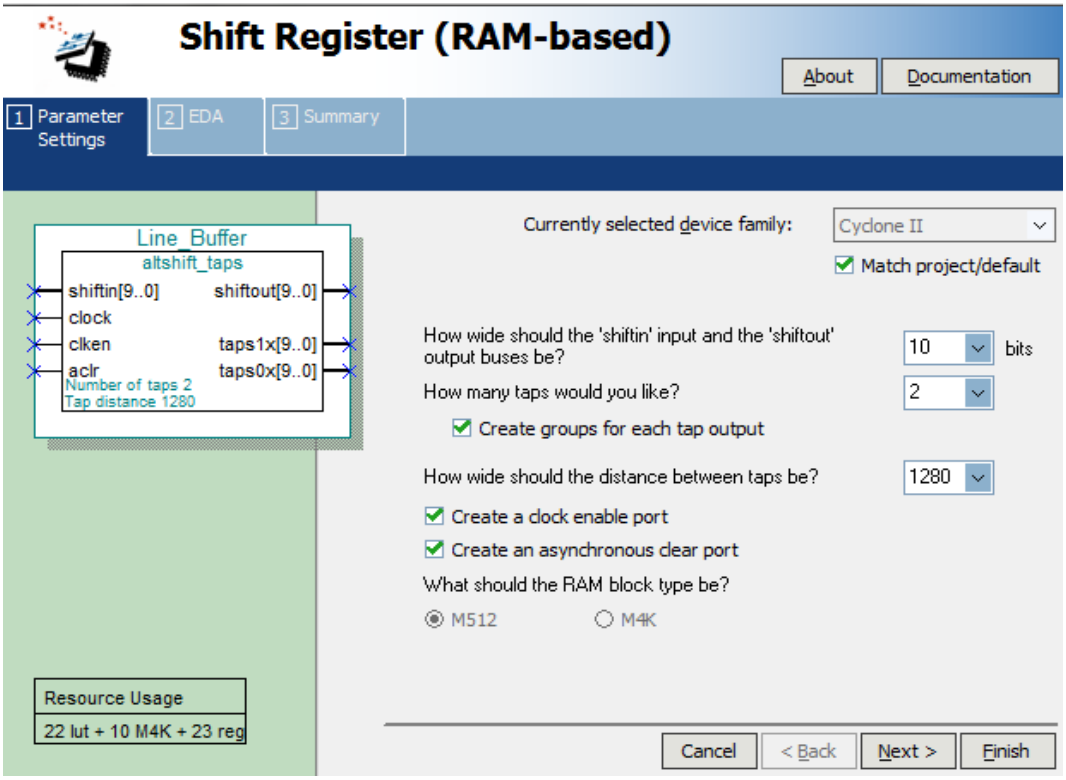



**Figura 3.12.** Registro de desplazamiento basado en memoria RAM

En la salida *taps1x* y *taps0x* se implementan un selector de datos que pertenecen a dos filas de la imagen Bayer, estos datos son seleccionados dependiendo de la ubicación en la que se encuentre. Para ello se implementan dos contadores en *X* y *Y*, que nos ubican los datos RGB sobre la imagen Bayer.

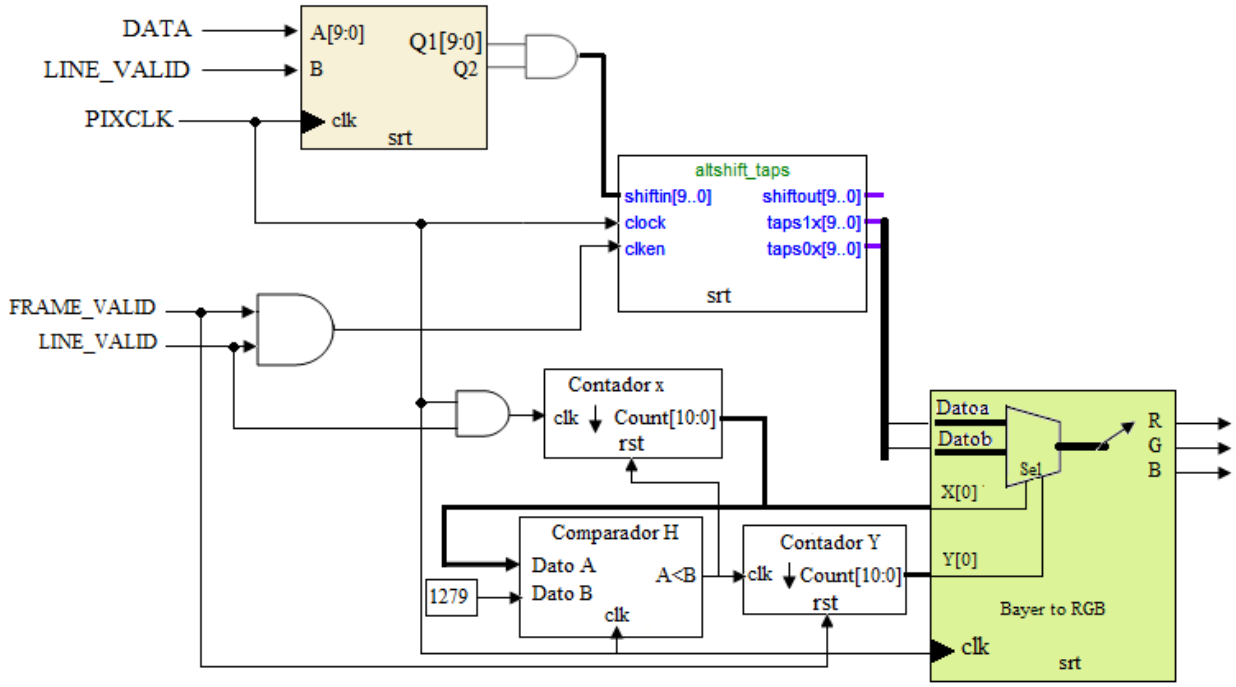



**Figura 3.13.** Conversión de Bayer a RGB

El circuito implementado para hacer la conversión de Bayer a RGB se muestra en la figura 3.13. En donde se observa en la etapa final del circuito, se implementan un multiplexor que se encarga de escoger las componentes R, G y B, que están llegando de las dos líneas de la matriz Bayer. La selección lo realiza por medio de dos contadores que nos ubican el espacio dentro de la matriz Bayer, tomando una máscara de 3x3, como se ilustra en la figura 3.14. Estos datos son enviado a un de multiplexor para tener una componente RGB en paralelo.

En la etapa de conversión de Bayer a RGB se toma una matriz de 3x3, y se van interpolando los valores R y B. Para ello se utilizara una sola matriz de las cuatro combinaciones que existen en Bayer, el cual se utiliza la que se ilustra en la figura 3.14.

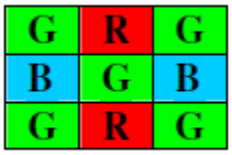



**Figura 3.14.** Matriz Bayer de 3x3

Para realizar la interpolación se va recorriendo el cuadro en 4 tiempos (Figura 3.16) dependiendo del contador X [0] y Y [0] (Figura 3.15). El Contador X [0] indica los pixel de cada línea y el contador Y [0] el numero de línea y como solo se utiliza el bit menos significativo de los contadores X y Y, solo contara dos pixel y dos líneas. En la Figura 3.15 se observa 4 combinaciones posibles 00, 10, 01 y 11 así como se muestra en la Figura 3.16.


**Figura 3.15.** Valores X [0] y Y [0] a analizar

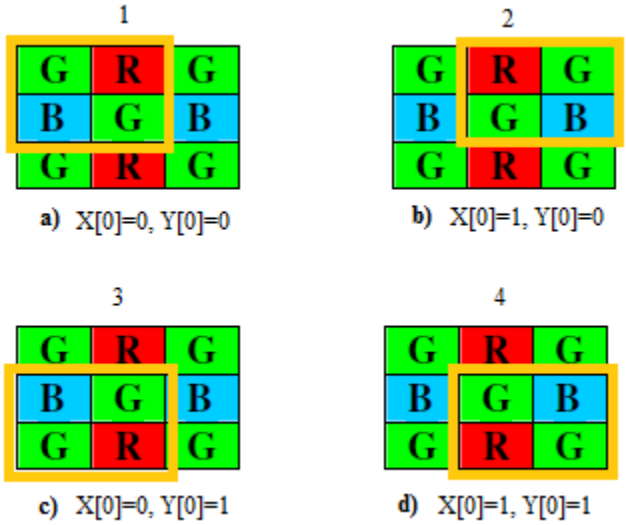



**Figura 3.16.** Análisis de interpolación en 4 tiempo según los valores de los contadores en X[0] y Y[0].

El análisis de estas 4 combinaciones se realiza mediante el empleo de 2 Flip-flops por cada fila. En donde *Data_a* es el valor de la primera línea, *Data_b* la de la segunda línea, mData_a es el valor almacenado de la primera línea y mData_b el valor almacenado de la segunda línea.

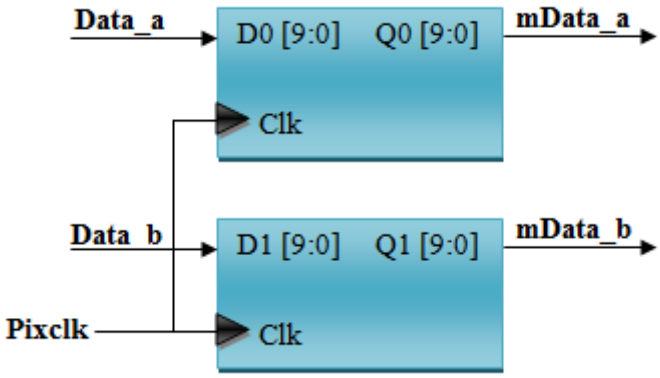


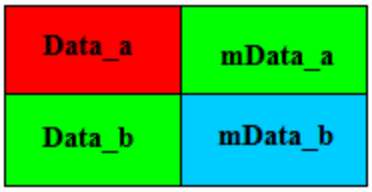

**Figura 3.17.** Generador de dos componentes por flip-flop

A continuación se analizan los valores RGB para cada caso:

**Caso 1**: Cuando X [0]= 0 y Y [0]=0, entonces

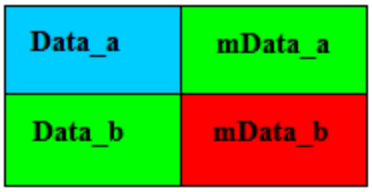

R<= mData_a
G<= Data_a + mData_b
B<= Data_b

***Caso 2:*** *Cuando X [0]= 1 y Y [0]=0, entonces*




R<= Data_a
G<= mData_a + Data_b
B<= mData_b

***Caso 3****: Cuando X [0]= 0 y Y [0]=1, entonces*




R<= mData_b
G<= mData_a + Data_b
B<= Data_a

***Caso 4****: Cuando X [0]= 1 y Y [0]=1, entonces*


R<= Data_b
G<= Data_a + mData_b
B<= mData_a

Después de  la conversión de Bayer a RGB. La imagen en RGB se pasa a una etapa donde se elimina la forma de espejo en la imagen, debido a que los datos captados por el sensor de la cámara sufren un desplazamiento en serie, esto es, el primer dato en entrar es el último en salir (Figura 3.18).

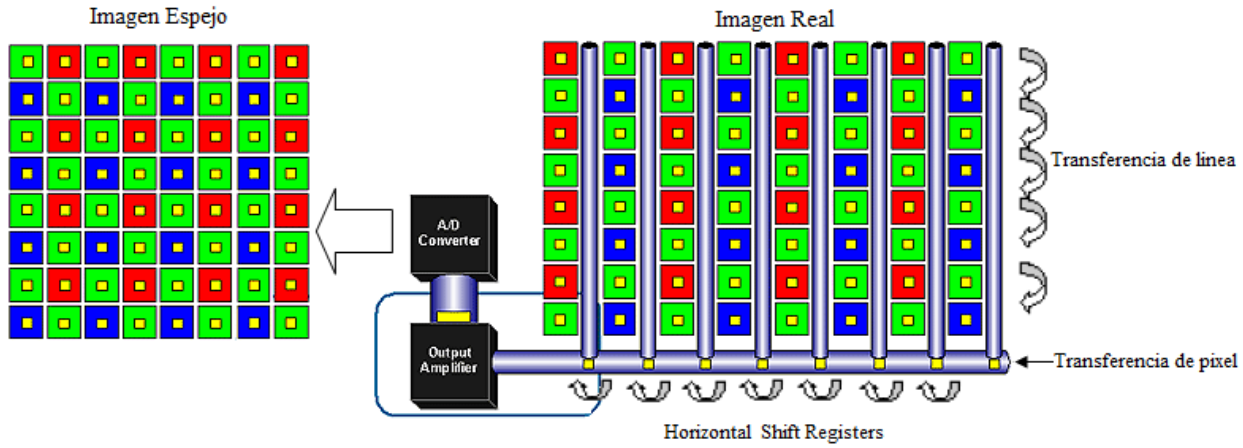



**Figura 3.18.** Transferencia de imagen Bayer a espejo

La eliminacion de espejo se logra haciendo pasar las tres componentes RGB en una memoria RAM (Stack-RAM) de 320 palabras (Figura 3.19). La implementación completa del sistema utilizando la memoria RAM, se observa en la siguiente figura 3.20 en donde se empieza a escribir desde la posición 319 a 0 de la memoria de manera que se elimine el efecto espejo.

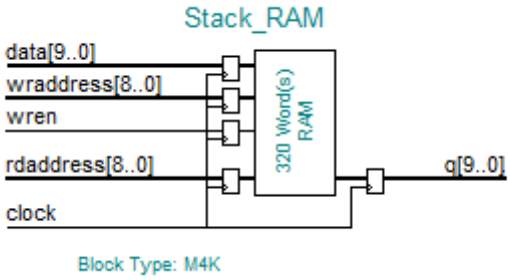



**Figura 3.19.** Stack-RAM

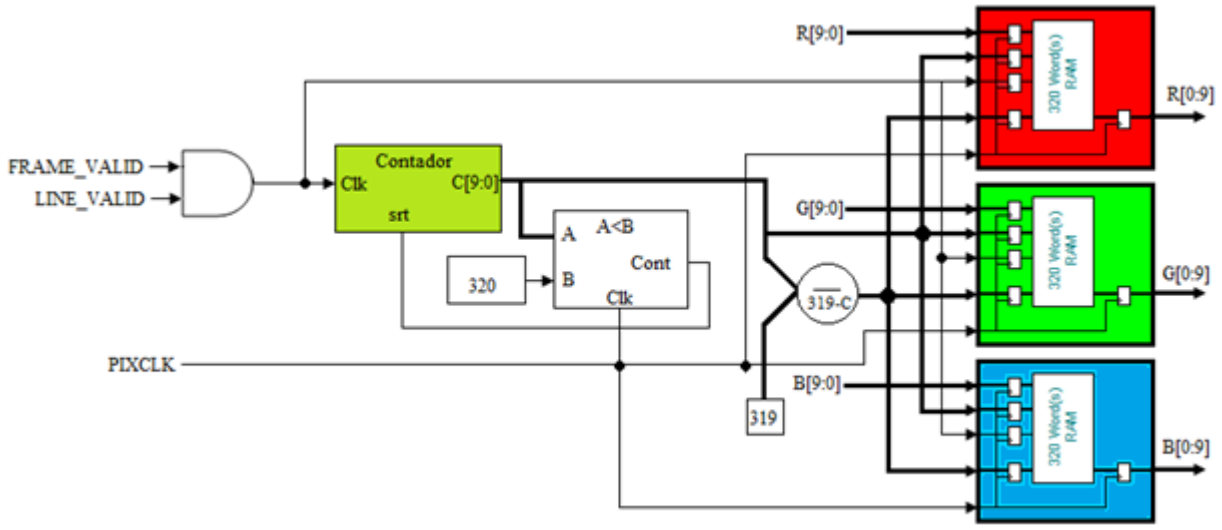



**Figura 3.20.** Eliminación de espejo en la imagen

### 3.1.3 Almacenamiento de imágenes

Después de la eliminación del efecto espejo, los datos RGB se almacenan en una memoria SDRAM. La facilidad que nos presenta almacenar datos en una memoria dinámica de acceso aleatorio es poder trabajar sobre ella, además de almacenar imágenes. Con la ayuda MegaWizard Plug-In Manager se utiliza una memoria FIFO programa en una memoria SDRAM (Figura 3.21).

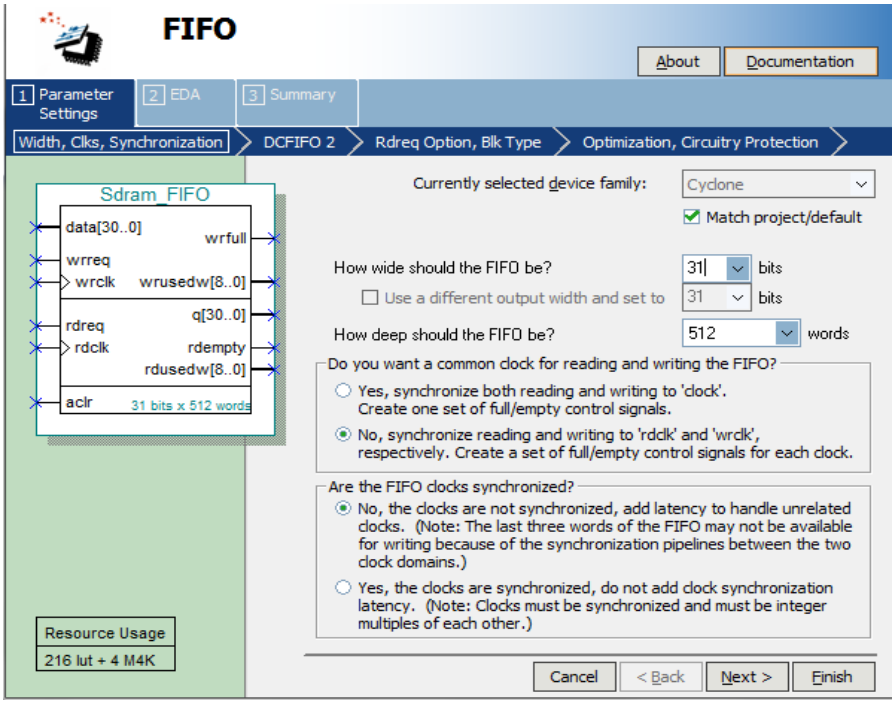



**Figura 3.21. Memoria SDRAM FIFO**

Con la utilización de esta memoria podemos almacenar las imágenes provenientes del sensor CMOS y poderlas aplicar para enviarla a una pantalla TFT-LCD. La ventaja que nos ofrece utilizar esta memoria, es que se puede leer y escribir a diferentes velocidades.

En esta residencia se utiliza una memoria externa debido a que la velocidad de captura del sensor CMOS no es igual, a la velocidad de envió de imágenes con la que la recibe la pantalla. La escritura es a una velocidad de 25Mhz y la lectura será la velocidad soportada por la pantalla TFT-LCD, en este caso 18.42 MHz.

El circuito implementado utilizando una memoria SDRAM se muestra en la figura 3.22.

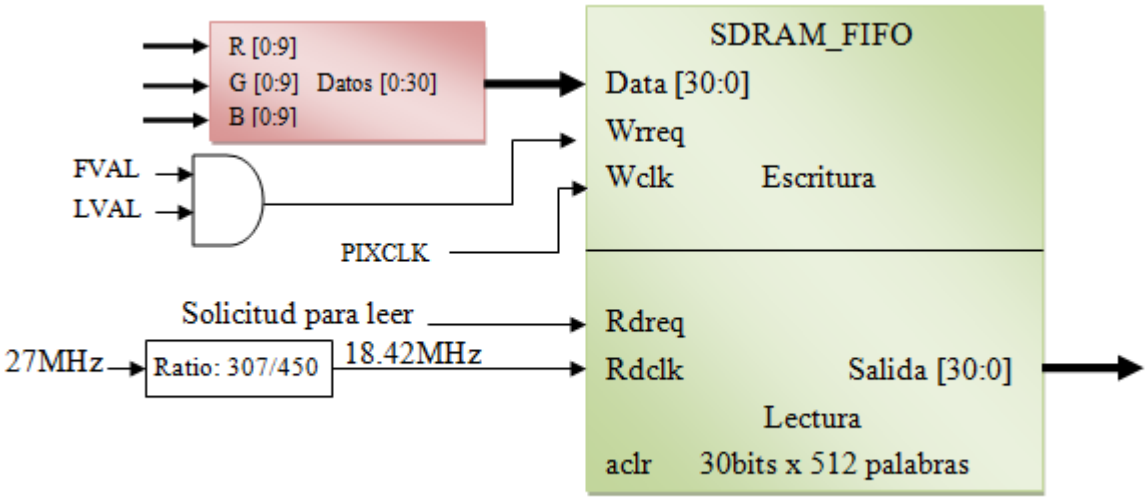



**Figura 3.22.** Circuito para almacenamiento de imagen

## 3.2 ENVIÓ DE IMAGEN A LA PANTALLA TFT-LCD (TRDB_LCM)

La imagen almacenada en la memoria SDRAM, es manipulada para ser enviada a una pantalla TFT-LCD (TRB_LCM) de 3.6 pulgadas. Para leer los datos almacenados en memoria se realiza contadores verticales y horizontales de acuerdo a los tiempos que maneja la pantalla TFT-LCD. Estos contadores sirven para obtener una señal que nos indique la habilitación de la lectura en la entrada Rdreq de la memoria y para ubicar el área activa (figura 3.23) en la pantalla, en donde será momento de enviar datos.

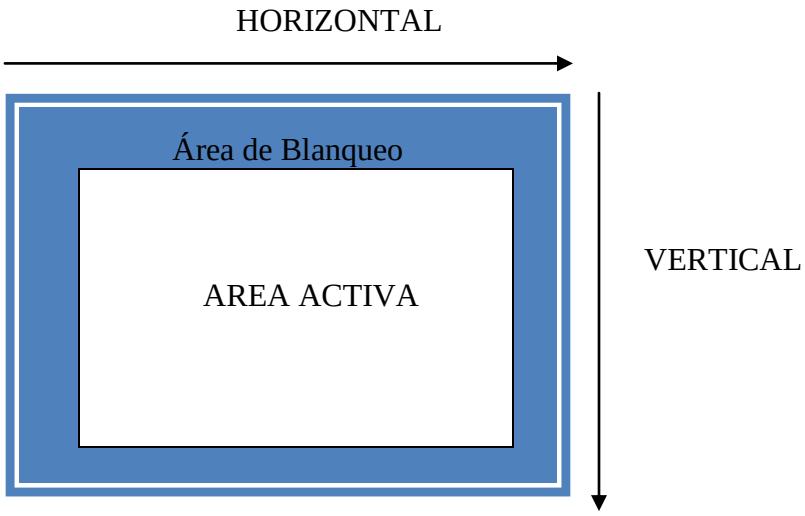



**Figura 3.23.** Área activa de la pantalla TFT-LCD

La pantalla TRDB_LCM, tiene las siguientes señales de entradas (Figura 3.24): DATOS (Entrada de pixeles); VYSCN (Sincronía Vertical); HYNC (Sincronía Horizontal); RESET GLOBAL (Reinicio de imagen) y DCLK (Señal de reloj maestro).

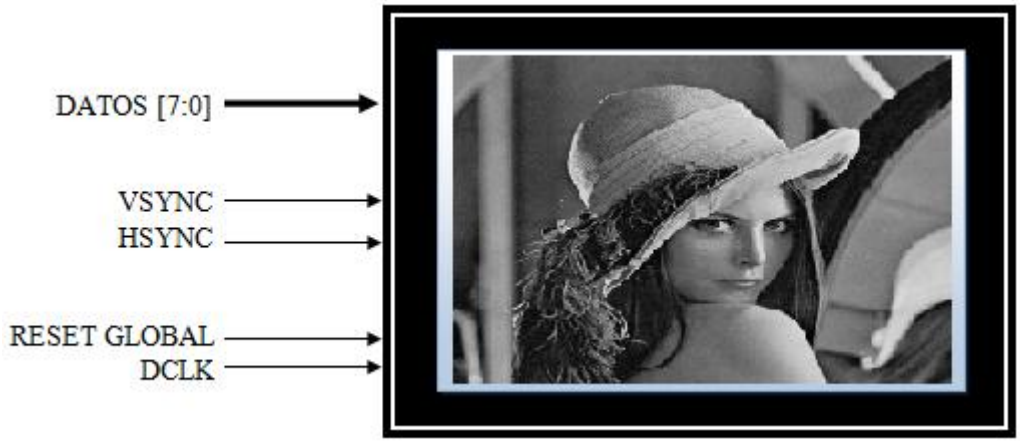



**Figura 3.24**. Señales requerida para el control de la pantalla

Las señales HSYNC y VSYNC se programan utilizando contadores y comparadores dependiendo de la configuración de tiempos que se maneje. Estos contadores tienen en la entrada de reloj principal una señal de reloj DCLK para estar sincronizados con la pantalla.

La configuración de la pantalla se realizo por medio de la comunicación I2C, en donde se eligió una imagen en formato RGB, una resolución de 960x480 e imágenes no entrelazadas para facilitar el diseño.

### 3.2.1 Sincronía horizontal

La señal de sincronía horizontal (Figura 3.24) está compuesta por los parámetros que se muestran en la tabla 3.1. De acuerdo a estos parámetros será programado el contador horizontal, de modo que nos indique en que tiempo los datos son validos por cada línea por campo. De acuerdo a la tabla 3.1 se dice que cada línea está compuesta por 1171 pulsos de reloj DCLK, dentro del cual incluye el blanqueo inicial de 152 pulsos y frontal de 59; la suma de ellos es 211 pulsos, dejando una aérea activa de 960 pulsos de reloj DCLK.

**Tabla 3.1. Parámetros de sincronía horizontal**

| Parameter | | Symbol | Panel Resolution | | | | | Unit |
|---|---|---|---|---|---|---|---|---|
| DCLK Frequency | | $F_{DCLK}$ | 10.36 | 11.63 | 12.90 | 14.18 | 18.42 | MHz |
| Horizontal valid data | | $t_{hd}$ | 492 | 558 | 640 | 720 | 960 | DCLK |
| 1 Horizontal Line | | $t_h$ | 659 | 739 | 820 | 901 | 1171 | DCLK |
| HSYNC Pulse Width | Min. | $t_{hpw}$ | 1 | | | | | DCLK |
| | Typ. | | 1 | | | | | |
| | Max. | | - | | | | | |
| Hsync blanking | | $t_{hp}$ | 102 | 113 | 117 | 122 | 152 | DCLK |
| Hsync front porch | | $t_{hfp}$ | 65 | 68 | 63 | 59 | 59 | DCLK |
| Horizontal dummy time | | $t_{hdm}$ | 6 | 9 | 4 | 0 | 0 | DCLK |

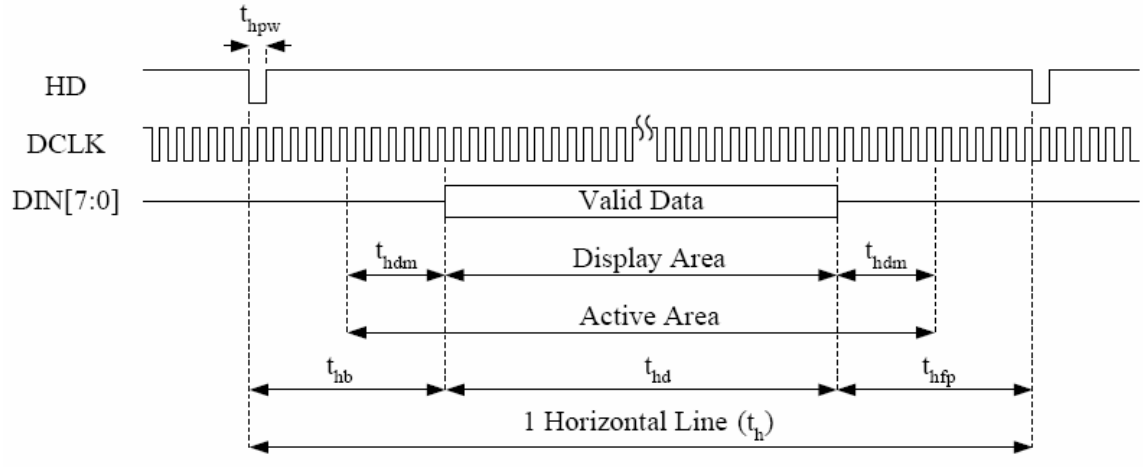



**Figura 3.25. Sincronía Horizontal**

La obtención de la sincronía horizontal, se realiza con el uso de un contador y comparadores (figura 3.31). Cuando el contador de 10 bits llega a 1171 en ese momento el comparador A manda un 0 al contador, y este se reinicia, volviendo el conteo a 0. HSYNC siempre será 0 cuando termine de contar el contador de 0-1171.

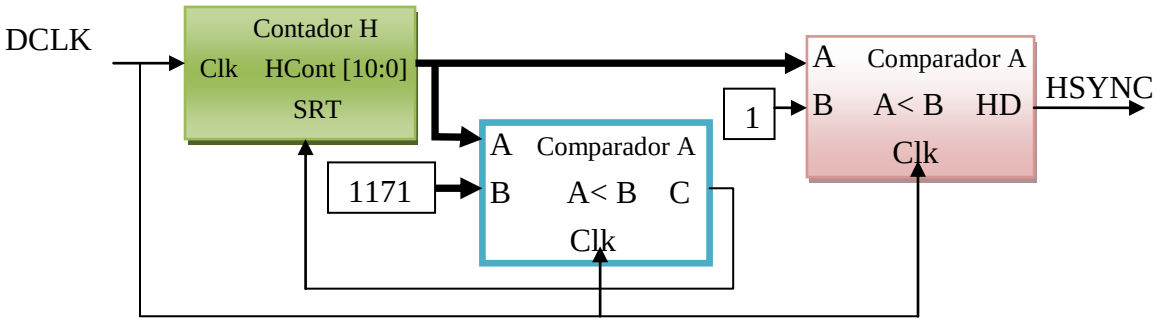



**Figura 3.26.** Circuito de Sincronía Horizontal

La señal DCLK es una señal de reloj de 18.42 MHz, esta señal de reloj solo se puede obtener haciendo uso de una componte ALTPLL de la herramienta MegaWizard Plug-In Manager, que facilita la programación de frecuencia exactas, en donde se define la proporción de cada pulso dependiendo de la señal de reloj de entrada (Figura 3.27).

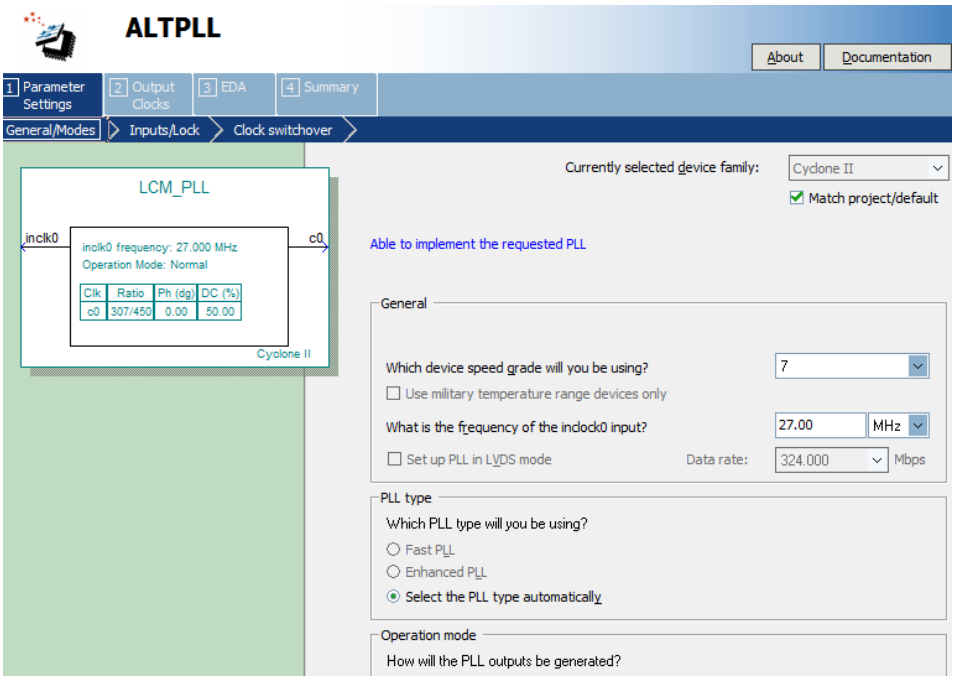



**Figura 3.27.** Divisor de frecuencia

Este componente ALTPLL con entrada de 27 MHz, se programa con un ratio de 307/450, para obtener una señal de reloj de 18.42MHZ.

### 3.2.2 Sincronía Vertical

La sincronía vertical depende muchos de la sincronía horizontal. La señal vertical está compuesta por los datos que se muestran en la tabla 3.2. La señal de sincronía vertical y la generación de un cuadro o imagen se muestran en la figura 3.28.

**Tabla 3.2. Parámetros de sincronía vertical**

| Parameter | | Symbol | Interlace | Non-interlace | Unit |
|---|---|---|---|---|---|
| Vertical valid data | | $t_{vd}$ | 240 | 240 | H |
| 1 Vertical field | | $t_v$ | 262.5 | 262 | H |
| Vsync pulse width | Min. | $t_{vpw}$ | 1 | 1 | DCLK |
| | Typ. | | 1 | 1 | DCLK |
| | Max. | | - | - | H |
| Vsync blanking | Odd field | $t_{vbo}$ | 14 | 14 | H |
| | Even field | $t_{vbe}$ | 14.5 | 14 | H |
| Vsync front porch | Odd field | $t_{vfpo}$ | 8.5 | 8 | H |
| | Even field | $t_{vfpe}$ | 8 | 8 | H |
| Vertical dummy time | | $t_{vdm}$ | 0 | 0 | H |

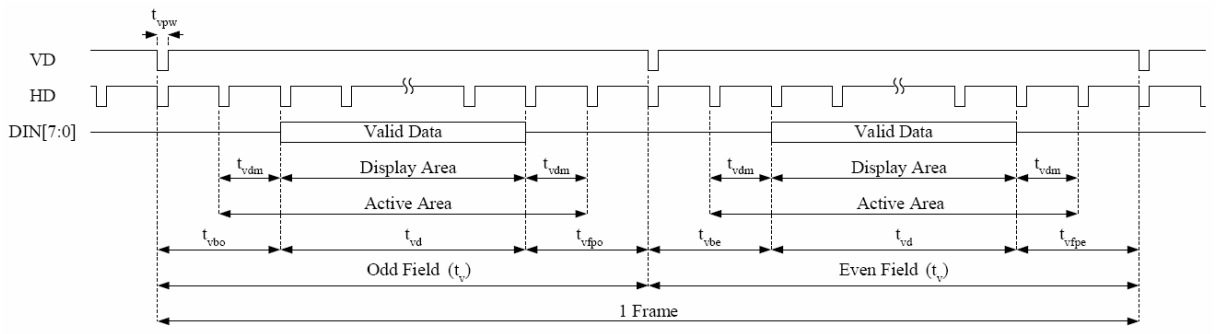



**Figura 3.28. Sincronía vertical**

En la figura 3.26 se observa que para generar una imagen o frame se desea generar dos campos, esta son par e impar. Cada campo está compuesto por 262 pulsos horizontales (H) y de área activa está compuesta por 240 pulsos horizontales, el cual sería la resolución vertical de la imagen. En cuanto a blanqueo vertical para cada campo, se tiene 14 pulsos inicial y frontal de 8 pulsos. Tomando en cuenta que cada pulso pertenece a la sincronía horizontal.

La sincronía vertical se obtiene de la misma manera que el circuito de sincronía horizontal. A diferencia que el circuito de sincronía vertical (Figura 3.29) tiene como entrada de reloj la señal de sincronía horizontal (HSYNC).

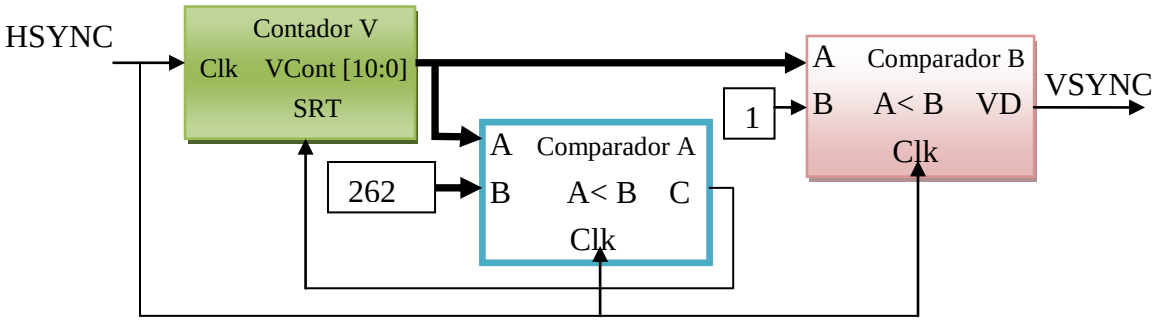



**Figura 3.29.** Circuito de sincronía vertical

Una visualización general de los tiempos de sincronización horizontales y verticales se muestra en la siguiente figura.

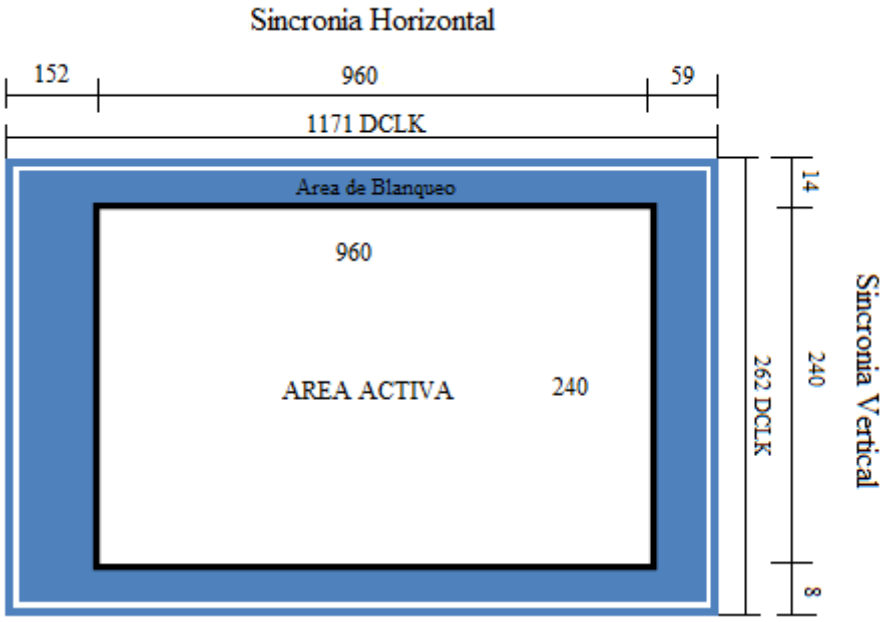



**Figura 3.30.** Espacio de sincronía vertical y horizontal

Como se utiliza una imagen no-entrelazada, los tiempos para cada campo seria el mismo (Figura 3.28), formando una imagen completa sin la necesidad de generar dos campos diferentes.

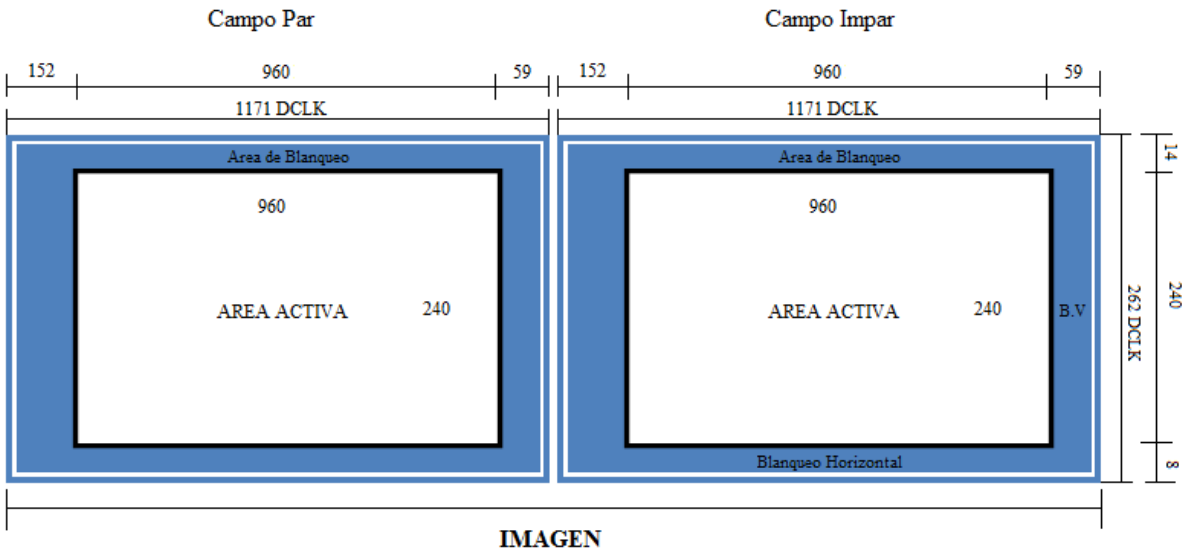



**Figura 3.31.** Creación de una imagen generando dos veces los mismo tiempo para e impar

Cada campo tiene una resolución de 320x240 debido a que cada pulso de reloj DCLK (Figura 3.32) pertenece a un pixel. La unión de estos dos campos nos forma una imagen con resolución de 640 x 480.

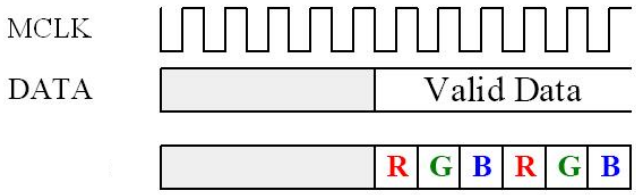



**Figura 3.32.** Pixel por pulso de reloj MCLK

### 3.2.3 Señal de activación

La señal de activación indica el momento en que se puede enviar datos a la pantalla TFT_LCD o la señal de activación para leer la memoria. La señal se basa en los tiempos del área activa que se menciona en la figura 3.30. La programación se realizo mediante la unión de los tiempos de validación horizontal y vertical. Esto se realizo utilizando dos comparadores para el blanqueo inicial y frontal de cada sincronía. Estos comparadores evaluan los tiempos de blanqueo vertical y horizontal para ubicar el área activa en la pantalla (Figura 3.33).

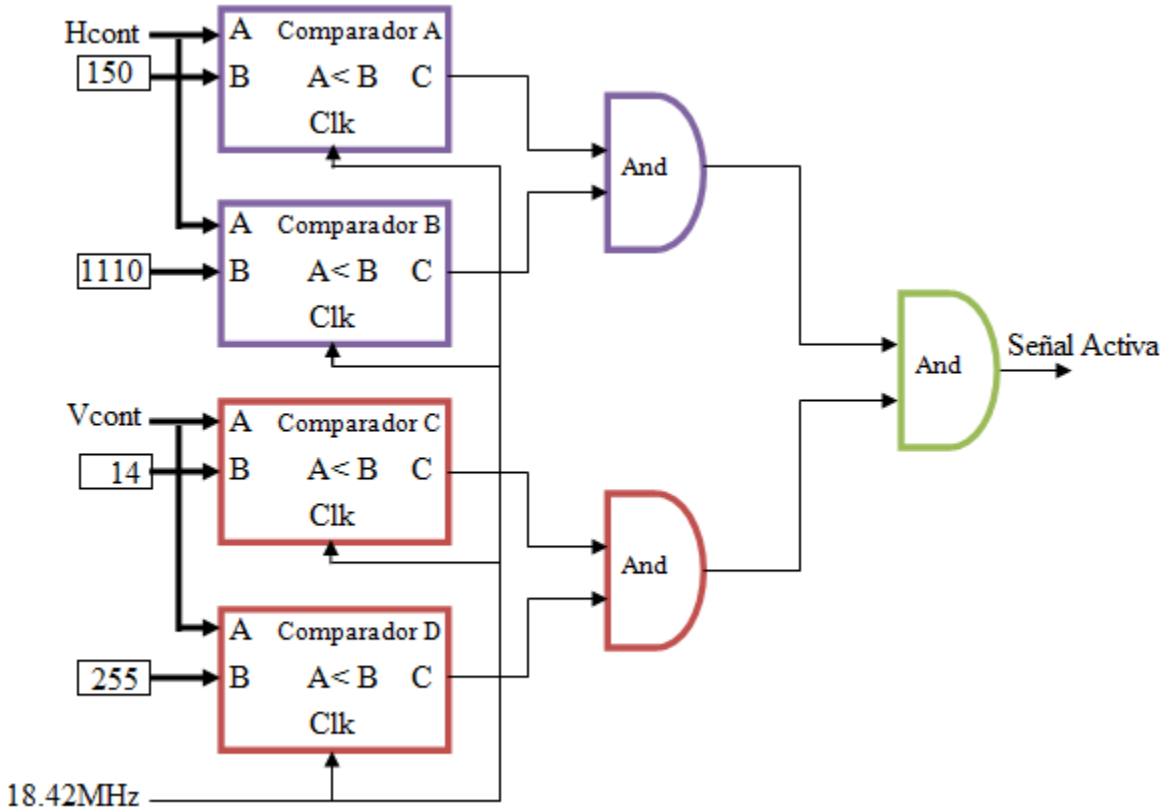



**Figura 3.33.** Circuito de señal de imagen activa

### 3.2.4 Etapa de control para la pantalla TFT-LCD

La señal resultante del circuito de señal de imagen activa se mete a una compuerta *and* junto con la señal de reloj 18.42 MHZ, el resultado de esta señal se mete a un contador, que tiene la función de generar 3 instante de tiempo, para ir mandando cada componente RGB por cada tiempo formando así un pixel en la imagen. El circuito completo se muestra en la figura 3.34.

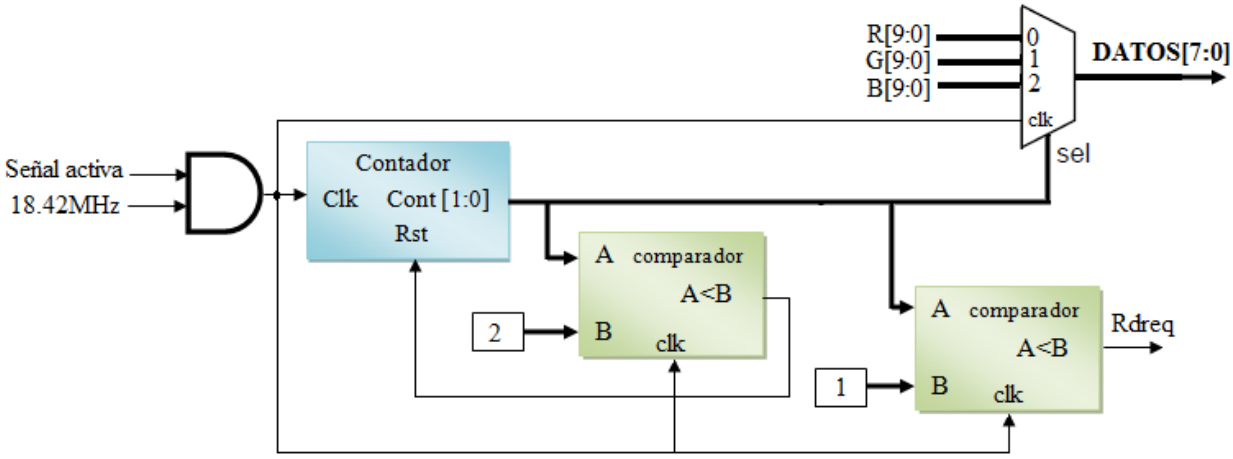



**Figura 3.34.** Circuito de control para el envió de datos a la pantalla TFT-LCD

La señal (A<B) del segundo comparador de la etapa de control va dirigida hacia la entrada Rdreq, para leer los datos que se encuentran almacenados en del la memoria SDRAM del circuito de almacenamiento de imagen. La señal DCLK de la pantalla TFT_LCD entra la señal negada del reloj de 18.42MHz y los datos que entran a la pantalla provienen de un selector de RGB que dependen del contador de 2 bits. El circuito completo para enviar imágenes a la pantalla se muestra en la figura 3.35.

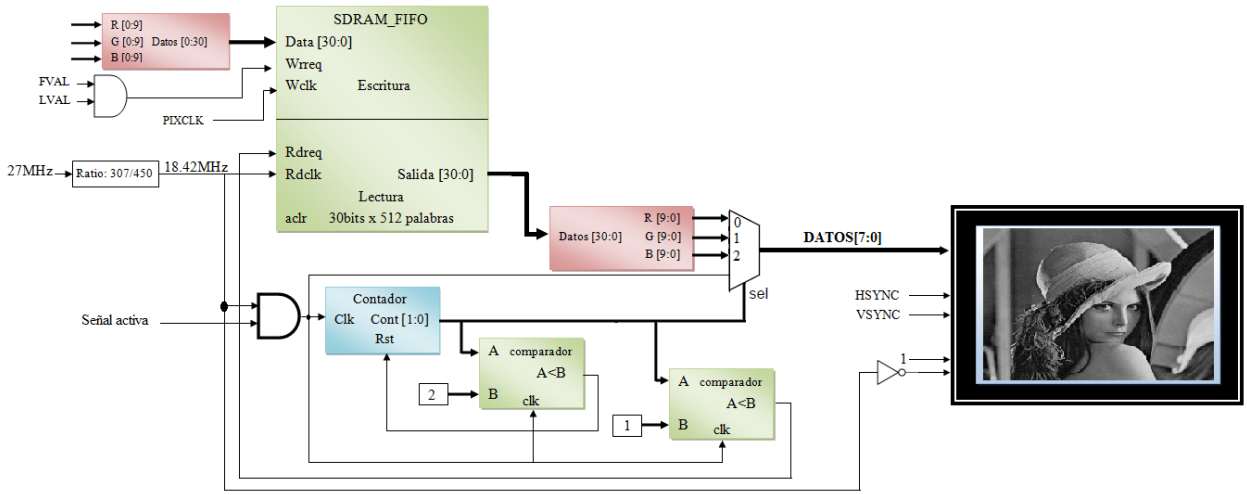



**Figura 3.35.** Etapa completa de envió de imágenes

## 3.3 VISIÓN ARTIFICIAL

Los algoritmos empleados o programados en FPGA, serán de acuerdo al estudio de la trayectoria o líneas dentro de la imagen. Los algoritmos de visión artificial se emplean en las imágenes que son capturada por la cámara y almacenadas en memoria; después de aplicar esos algoritmos, la imagen es enviada a la pantalla TF-LCD para verificación de resultados (Figura 3.36)

Para analizar con mayor facilidad y con precisión la línea o la trayectoria se hace uso de filtros. Estos filtros eliminaran todo aquello que no pertenezca al objeto de estudio. Para cuestiones de prueba, se aplicaran dos filtros (Binarización y Sobel) de los cuales solo se elegirá el adecuado para la obtención de mejores resultados del cálculo de la centroide y la curvatura de la línea.

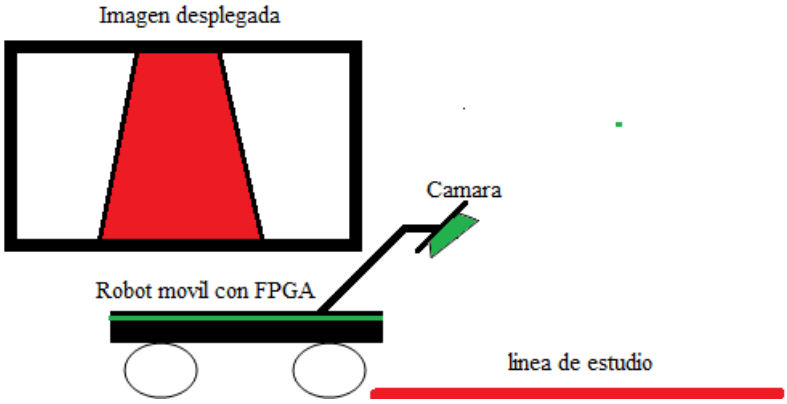



**Figura 3.36.** Sistema seguidor de trayectorias definidas

Para la implementación de filtros, es necesario que la imagen este en grises. Esto se realiza calculando la luminancia (Y) aplicando la ecuación 3.9. En donde se observa que al aplicar esta ecuación nos arrojara resultados flotantes.

En los sistemas digitales existen grandes problemas para manejar números flotantes, por lo que en esta residencia, solo se tomo en cuenta la componente G, debido que es la componente que mas observa el ojo humano o el sensor de la cámara. Por lo tanto un pixel en la imagen está compuesto por tres componentes G (ver Figura 3.34), sustituyendo los valores R y B por G. Obteniendo así una imagen en escala de grises.

### 3.3.1 Binarización de imagen

La Binarización se realiza aplicando la técnica de paso de umbral por medio de la ecuación 2.9. En donde el valor de umbral que se elije es de T=24. La Binarización se aplica utilizando un comparador para la componente G con el umbral. La señal que resulta de este comprador, es la señal de selección del multiplexor, para obtener valores de 0 y 255, dependiendo de lo valores de la componente G (Figura 3.37).

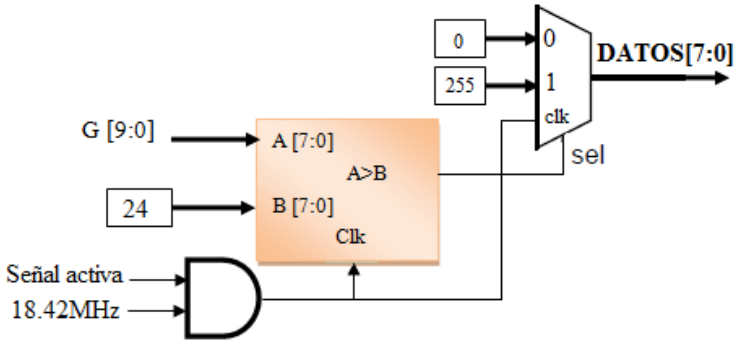



**Figura 3.37.** Circuito binarizador de imagen

Para obtener la imagen binarizada desplegada en la pantalla se le aplica este circuito a las tres entradas del multiplexor del circuito de control para el envió de datos a la pantalla TFT-LCD.

### 3.3.2 Filtro de Sobel

El filtro de Sobel está clasificado como un filtro pasa alta, detectando los cambios bruscos en la imagen, permitiendo distinguir los bordes de cualquier objeto en la imagen. La detección de bordes se obtiene analizando la gradiente en $x$ y $y$ de la imagen. Esto se realiza aplicando las ecuaciones 2.6 y 2.7. Las matrices que resultan de estas dos ecuaciones se recorren en toda la imagen de izquierda a derecha por cada tres filas y por cada pulso de reloj. Para ello esperamos a que pasen tres filas para comenzar con la operación. Para lograr almacenar estas tres líneas, se utiliza un registro de desplazamiento (Shift Register) basado en memoria RAM.

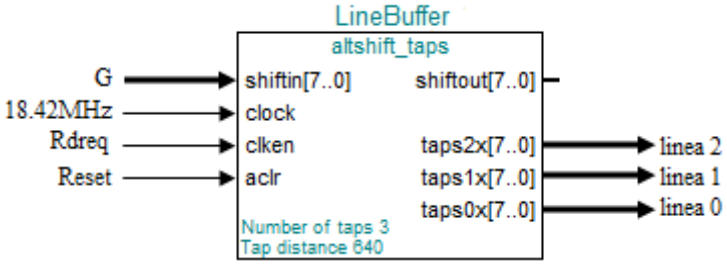



**Figura 3.38.** Componente Shift Register (RAM-based) de la herramienta MegaWizard Plug-In

El registro se programa para almacenar datos de 8 bits en una matriz de 3 x 640. En cada salida del buffer se conectan tres flip-flop para almacena un matriz de 3x3 y multiplicarla por cada matriz de la gradiente. En el siguiente circuito (Figura 3.39) se muestra la el cálculo de la gradiente $Gx$ con entrada de reloj de 18.42 Mhz.

Para calcular la gradiente $Gy$ no es necesario multiplicar la línea 2 por la segunda fila de la matriz $Gy$, debido a que el resultado siempre será 0, por lo que, se anula esa operación. El circuito para el cálculo de la gradiente $Gy$ se muestra en figura 3.40.

Una vez que se hizo el diseño de los dos circuitos para calcular las dos gradientes, se calcula el valor de la gradiente resultante por medio de la ecuación 2.3. .

El valor absoluto de cada gradiente, en hardware se obtiene comparando el último bit más significativo (MSB) del dato, si es 1 o 0. Si el MSB es igual a 1, significa que el dato es un valor negativo o un complemento a 2. Para pasar un número que está en complemento a 2 a un número real, se hace el proceso inverso. Se niegan todas las entradas y se le suma 1 y con este

procedimiento obtenemos un valor absoluto de cada gradiente, en caso contrario se trata de un valor positivo (Figura 3.41).

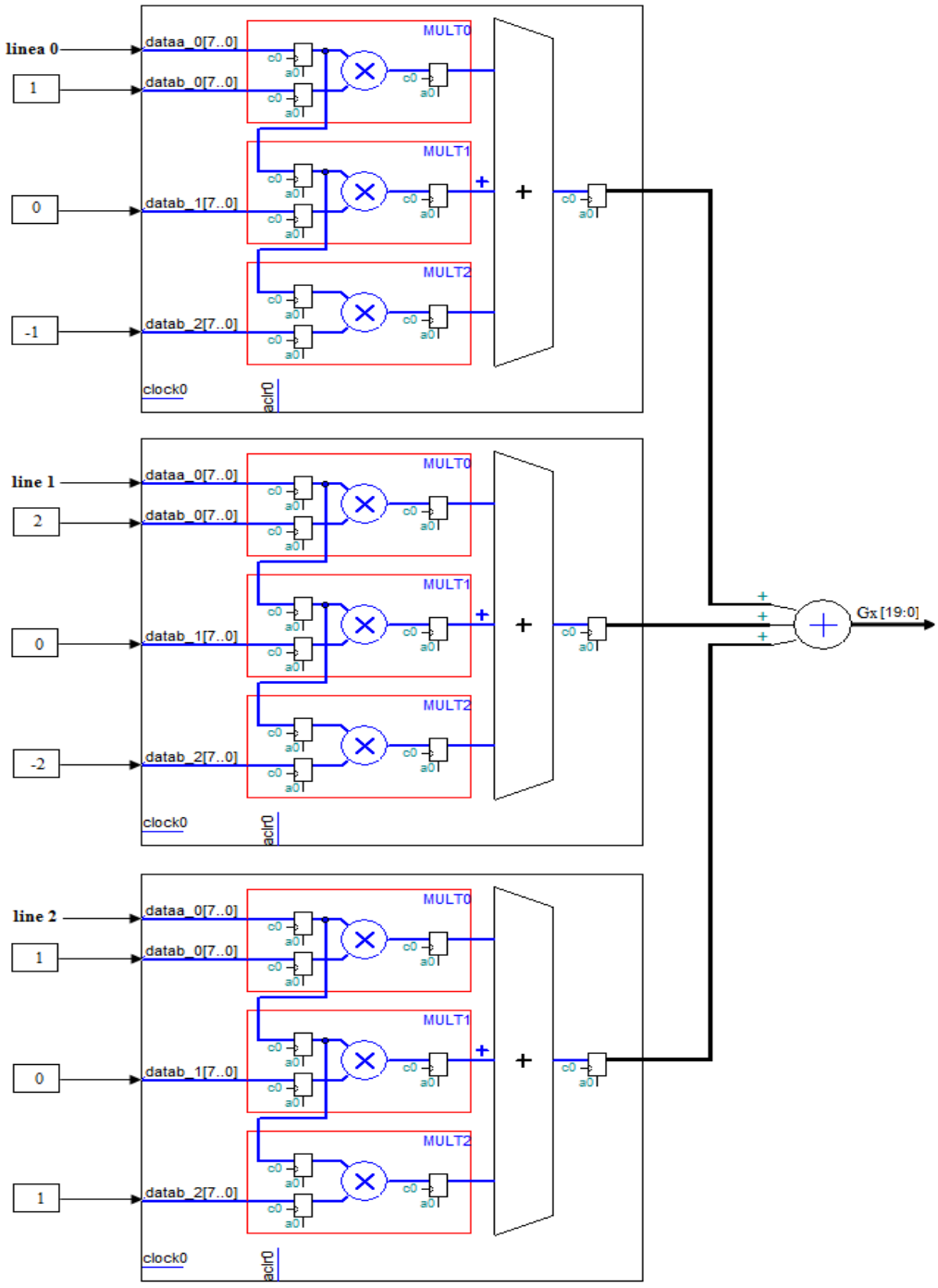



**Figura 3.39. Circuito calculador de Gx**

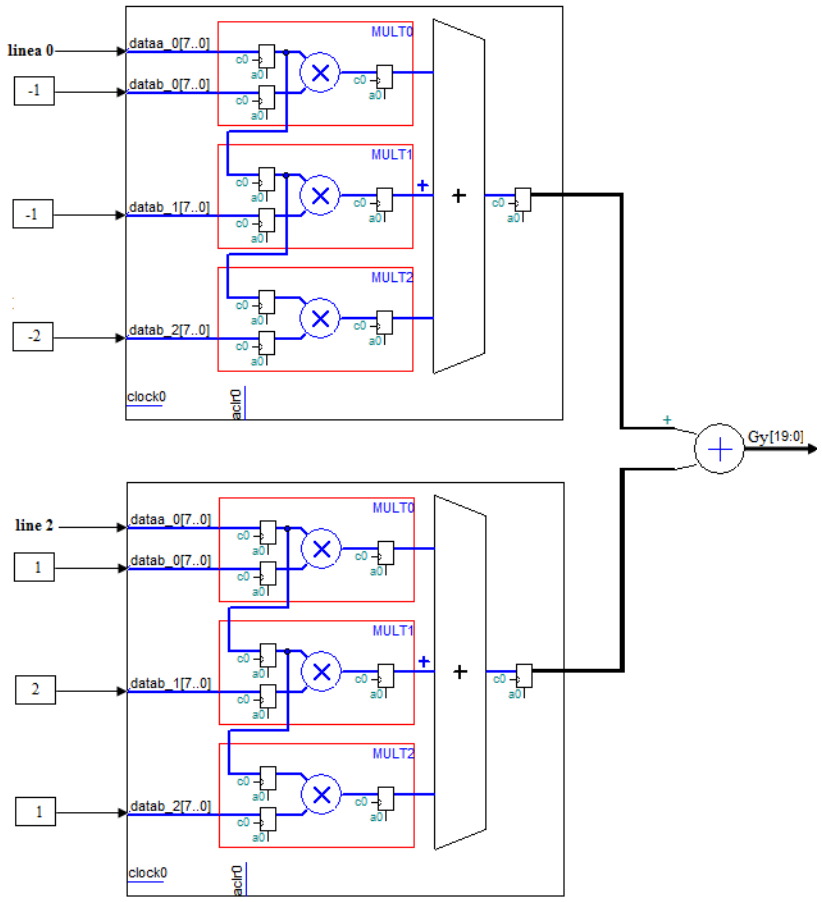



**Figura 3.40.** Circuito calculador de la gradiente Gy

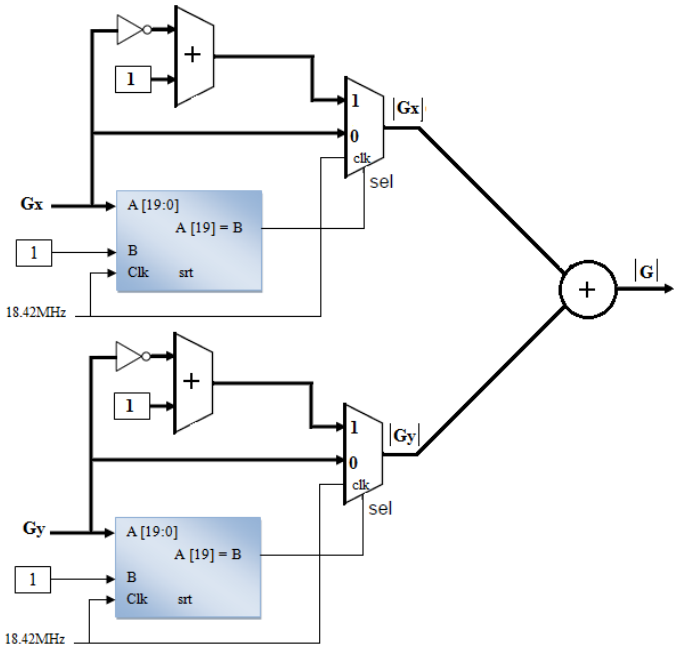



**Figura 3.41.** Circuito calculador de la gradiente resultante G

La gradiente resultante G se pasa a una etapa binarizadora utilizando el circuito binarizador, con la finalidad de distinguir los bordes en la imagen. Como borde de cualquier objeto obtendremos un color negro y de fondo un color blanco.

## 3.4 CONSTRUCCIÓN DEL ROBOT PARA SU APLICACIÓN

La robótica es una de las aplicaciones más apasionantes de la electrónica. Un robot seguidor de línea o trayectoria se clasifica en el campo de la robótica móvil un grupo de la rama de robótica. La tarea fundamental de un robot móvil es el desplazamiento en un entorno conocido o desconocido, por tanto es necesario que posea tres funciones fundamentales, la locomoción (nivel físico), la percepción (nivel sensorial) y la decisión (nivel de control). Entre las aplicaciones de robots móviles se encuentra el transporte de carga en la industria, robots desactivadores de explosivos, exploración de terrenos no aptos para el hombre entre este ultimo podemos destacar los robots Spirit y Oportunity desarrollados por la NASA.

### 3.4.1 Nivel físico

#### 3.4.1.1 Estructura

Para su construcción se utilizó una lámina de acrílico, esto proporciona apoyo para la tarjeta Altera DE2, los motores, y la rueda libre (Figura .42).

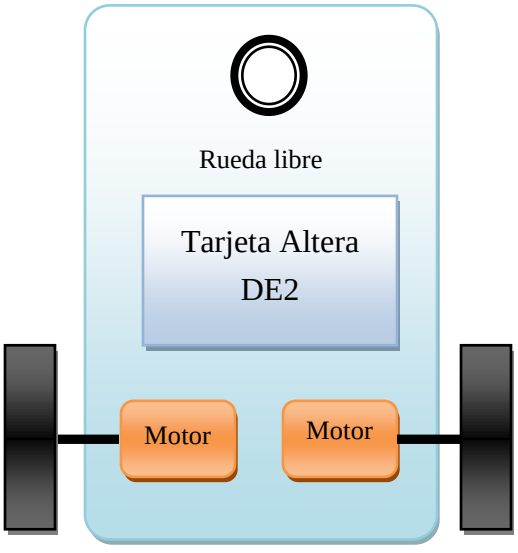



**Figura 3.42.** Modelo del robot seguidor de trayectoria

Las llantas del móvil se encuentra en configuración diferencial, debido a que la dirección que tome depende de la diferencia de velocidad entre sus dos llantas, es por eso que cada llanta es independiente de la otra.

**Figura 3.43**. Rueda libre

La rueda libre es la que aporta el apoyo en la parte posterior, esta debe exhibir la característica de rodar y pivotar sobre sí misma con un movimiento lo más suave posible para no dificultar la rotación del robot. En la figura 3.44 se muestra la manera como se ensamblo la rueda libre sobre la lámina acrílica.

**Figura 3.44.** Rueda libre ensamblada a la lámina de acrílico

### 3.4.1.2 Motores y Llantas

La mayor parte de los motores que se utilizan en un robot giran demasiado rápido y no tiene el torque suficiente, es por eso que se utiliza servomotores (figura 3.45), para que las llantas tengan

la suficiente fuerza para girar. Esta permite transformar un pequeño motor rápido, pero poco potente, en un motor más lento pero con mejor torque.

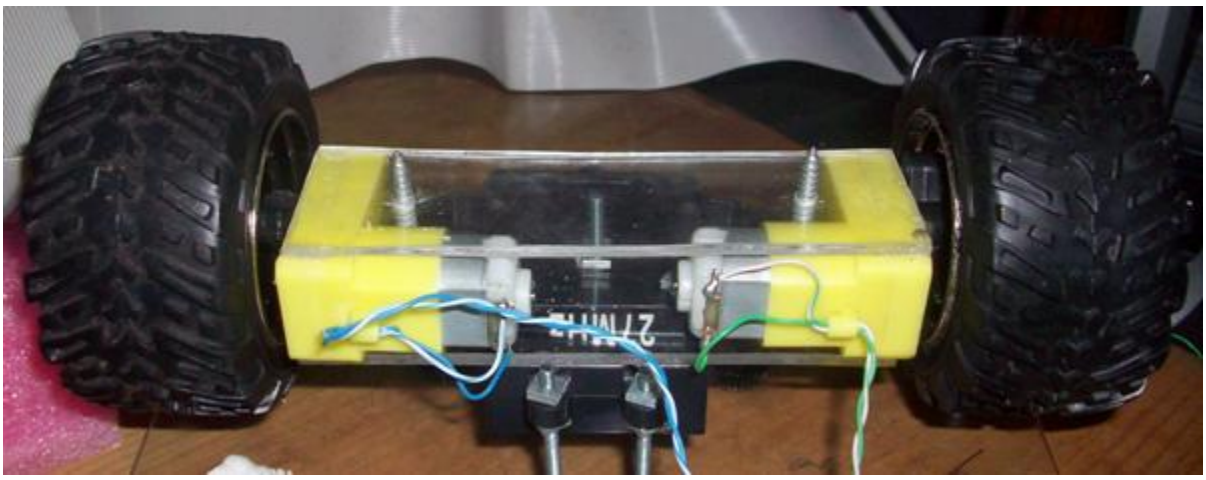


**Figura 3.45.** Motores conectados por cada llanta

Las llantas se eligieron de material de caucho o de un plástico blando para que no patinen así como se logra aprecia en la figura 3.45.

El motor del servo tiene algunos circuitos de control y un potenciómetro (una resistencia variable) esta es conectada al eje central del servo motor. En la figura 3.47 se puede observar al lado derecho del circuito. Este potenciómetro permite a la circuitería de control, supervisar el ángulo actual del servo motor. Si el eje está en el ángulo correcto, entonces el motor está apagado. Si el circuito chequea que el ángulo no es el correcto, el motor girará en la dirección adecuada hasta llegar al ángulo correcto. El eje del servo es capaz de llegar alrededor de los 180 grados. Normalmente, en algunos llega a los 210 grados, pero varía según el fabricante. Un servo normal se usa para controlar un movimiento angular de entre 0 y 180.

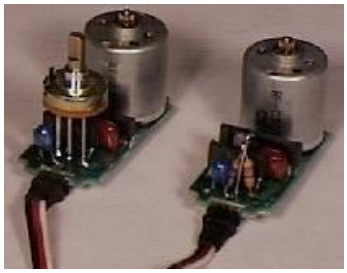


**Figura 3.46.** Circuito interno de los Servo motores

### 3.4.2 Nivel sensorial

La percepción de este robot es de tipo visual, aunque no debemos pensar que el robot va a ver. Su captación visual consiste en diferenciar entre dos colores, el color de la línea y el fondo. En este sistema puede utilizar cualquier color al igual que el fondo, ya que la cámara puede ver todos los colores que existente en el entorno, al igual que el ojo humano.

La línea es el estudio más importante de todo el sistema, ya que depende de ella los movimientos que realice el robot, como dar vuelta a la izquierda o derecha. Todo depende de cómo se comporte la línea en instante de tiempo o más bien en cada cuadro o imagen capturada.

### 3.4.3 Nivel de control

El circuito de control es el que proporciona las señales hacia los motores dependiendo de las señales obtenidas de la cámara. Este circuito está compuesto por una etapa de potencia de 9V y un driver o puente H que proporciona la dirección de las llantas. En la entrada avance y retroceso, entrara una señal PWM generado en el mismo FPGA, con el objetivo de poder acelerar dependiendo de la curvatura de la línea.

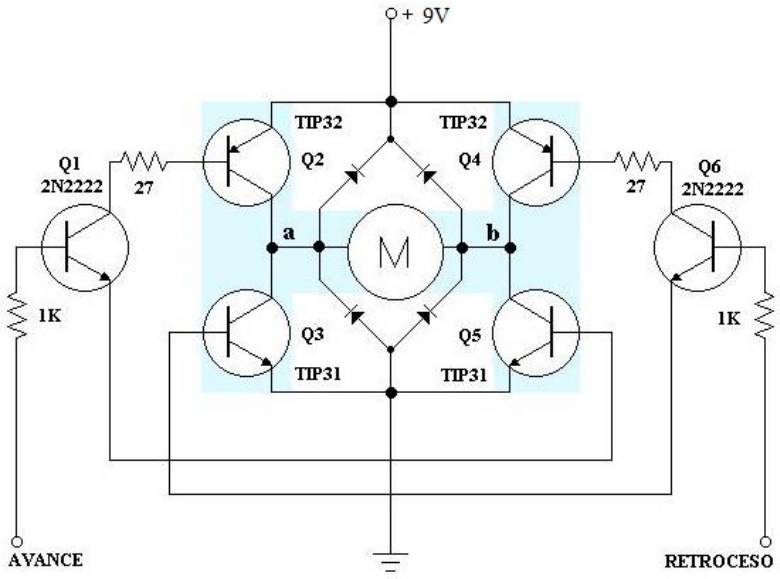



**Figura 3.47.** Circuito puente H para controlar los motores

Cada motor tiene implementado un circuito de puente H para girar en doble sentido, esto hará que avance, retroceda y gire el robot en doble sentido. Al final de toda esta tapa se obtiene el siguiente robot como el que se muestra en la figura 3.50.

**Figura 3.48.** Robot construido

# 4 Resultados

Los resultados planteados en este capítulo se basan desde la adquisición de la cámara, el tratamiento de las imágenes, la implementación de los filtros y el envió de imágenes a la pantalla. De la implementación y de la unión de las diferentes etapas que se desarrollaron para la adquisición de imágenes, utilizando los circuitos del capítulo 3, se obtuvo el siguiente resultado que se muestra en la figura 4.1.

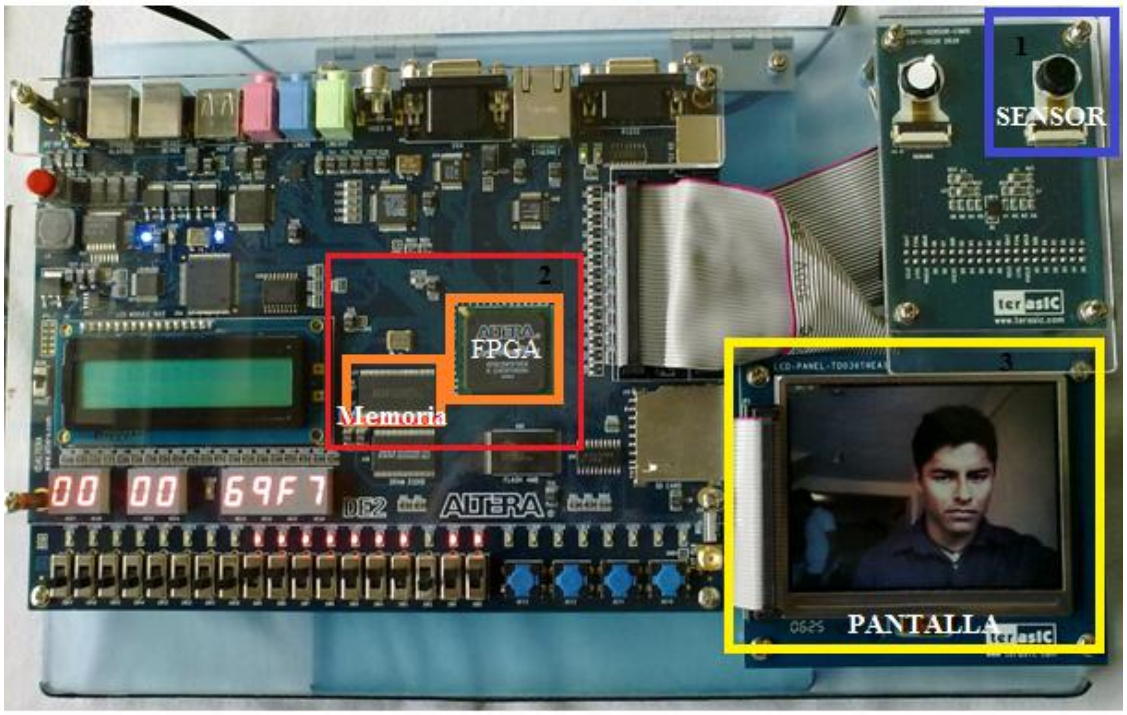



**Figura 4.1.** Resultado de la implementación de las diferentes etapas

En donde el rectángulo azul de la figura 4.1 se muestra el sensor que fue utilizado para realizar la capturar datos; luego esto datos son adquiridos por el FPGA para ser procesados y controlado con la ayuda de la memoria SDRAM, esta segunda etapa se observa en el rectángulo rojo y por último en el rectángulo amarillo se logra apreciar la imagen del rostro de una persona que fue capturada por el sensor.

En la siguiente imagen (Figura 4.2) se logra apreciar con más claridad la posición de la cámara con respecto a la persona y como se logra a ver el rostro de esta persona en la pantalla TFT-LCD.

**Figura 4.2.** Captura de imagen del rostro de una persona

Después de obtener imágenes a color, la componente verde (G) pasa a sustituir los valores R y B, para obtener una imagen a escala de grises. El resultado de hacer esta operación se muestra en la figura 4.3.


**Figura 4.3** Imagen a escala de grises; (a) imagen original, (b) imagen a grises.

La operación de pasar de una imagen de color a grises, se realiza debido a la utilización de los filtros de Binarización y de Sobel.

En la figura 4.4 se observa el filtro de Binarización que se realizo al aplicar el circuito el circuito binarizador de imagen. En donde las partes claras toman un valor de 0 y las partes oscuras del rostro toma un valor de 255, la dedición se baso por la determinación del umbral (T=18).

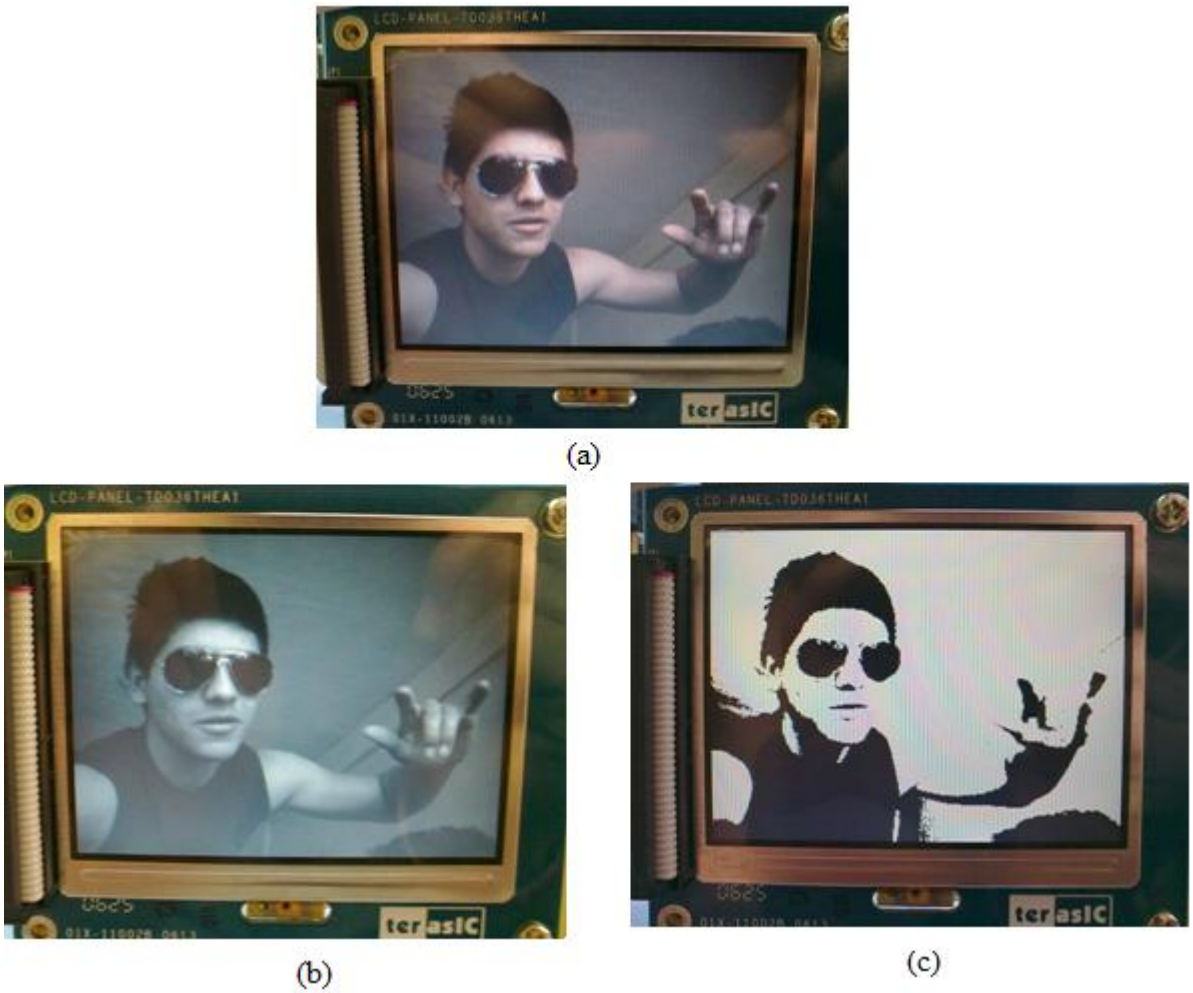



**Figura 4.4.** Binarización; (a) imagen original, (b) imagen a grises e (c) imagen binarizada

De la figura 4.4 (c) se observa una imagen de dos niveles de color que depende del valor de umbral, quien decide evaluar el rango de la intensidad de luminosidad en la imagen de la figura (b) con la finalidad de eliminar todo el ruido que se encuentra en la imagen original (figura 4.4 (a)). Después de realizar la binarización se implemento el filtro de Sobel.

La implementación del filtro de Sobel utilizando el circuito calculador de la gradiente resultante y el circuito binarizador, se obtiene como resultado las siguientes imágenes (Figura 4.5).


**Figura 4.5.** Filtro de Sobel; (a) imagen original, (b) imagen en grises y (c) imagen filtrada utilizando la mascara de sobel en la gradiente $\boldsymbol{x}$ y $\boldsymbol{y}$.

En la figura 3.41 (a) se puede observar la imagen de una silla de color blanca que resalta con respecto a lo demás objetos que se encuentran en la imagen, esto se realizo a propósito para que al aplicar el filtro de Sobel detecte esos cambio con facilidad. Obteniendo una imagen como se muestra en la figura 3.41 (c) con un umbral T=18.

El tiempo en que tarda el FPGA en realizar todo estos algoritmos mencionados en el desarrollo, lo realiza en un tiempo de operación por pixel de 0.54ns utilizando un cristal de 18.42 MHz y el tiempo total que se lleva en una imagen completa es de 16.6 ms (640 x 480 x 0.54ns). Logrando

así una ejecución concurrente que permite alcanzar ratios de velocidad de proceso entorno a 30 imágenes por segundo. El tiempo de procesamiento de imágenes en esta residencia dependió de los tiempos que establece el sistema NTCS.

La implementación de los algoritmos en FPGA Altera Cyclone II EP2C35F672C6, expuesto en el desarrollo consumieron los recursos en hardware que se expone en la Tabla 4.1. Se observa que el recurso más utilizado fue más de memoria, seguido de elementos lógicos y registros.

**Tabla 4.1.** Hardware Consumido

| **Nombre** | **Total** | **Porcentaje** |
|---|---|---|
| Elementos lógicos (LE) | 2200 | 7 % |
| Funciones combinacionales | 1800 | 6 % |
| Registros de lógica Dedicado | 1300 | 4% |
| Bits de Memoria | 79100 | 41% |
| Multiplicadores Embebidos de elementos de 9-bits | 30 | 43 % |
| PLL | 2 | 50 % |

# 5 Conclusiones

El actual trabajo se presento una metodología para procesar imágenes en FPGA e implementar algoritmos de visión artificial para ver la velocidad de respuesta a estos algoritmos, donde se concluye que el FPGA es una arquitectura viable para el uso de la visión sin necesidad de utilizar la computadora personal. Un aspecto muy importante que depende del tiempo de procesamiento, es la velocidad con la que el sensor de la cámara captura los datos. No podemos realizar un procesamiento rápido si la cámara es lenta.

Diseñar hardware en FPGA es muy viable y se diseña a la medida de cada proyecto si necesidad de gastar mucho dinero en hardware. En este proyecto el proceso de adquisición resulto muy dificultoso debido que los datos estaban en formato Bayer y se tenía que convertir a formato RGB. Con la ayuda de memoria, registros, contadores y multiplexores, que son fáciles de programarse en FPGA se logro convertir de manera rápida una imagen Bayer a RGB. Además a ello la imagen tuvo que eliminarse el efecto espejo utilizando memoria RAM. La ventaja que nos proporciona utilizar el FPGA es que se puede diseñar cual quiere hardware que uno desea para poder adquirir y procesar datos en tiempo relativamente rápidos, debido a que todos los procesos que uno programa lo va realizando de manera paralela.

Unas de las aplicaciones que se realizara en un futuro con esta metodología será el diseño de un robot seguidor de trayectoria de manera autónoma, que pueda esquivar obstáculos, acelerar dependiendo de la curvatura de la trayectoria, además que vaya aprendiendo conforme al tiempo, dependiendo de las escenas que se le presente. Agilizando su desempeño y aprendizaje.

La metodología implementada tiene muchas aplicaciones con el uso de la visión artificial, el objetivo típico incluyen reconocimiento de forma, textura, movimiento, color, detección, localización etc. Depende de cada persona la aplicación que desee desarrollar de acuerdo a sus necesidades.

# 6 Bibliografía

Arnold, J. M., Buell, D. A., & Davis, E. G. (1992). Splash 2". proceedings of the fourth annual ACM symposium on parallel algorithms and architecture. *IEEE* , 316-322.

Cervera, E., & Chiem, S. (2004). Vision-based Robot Formations with Bézier Trajectories. *Intelligent Autonomous Systems 8* (págs. 2-4). Amsterdam: IOS Press.

Dawood, A., Visser, S., & Willianms, J. (2002). Reconfigurable FPGAs for real time processing in Space. *Proceedings of the 14th International Conference on Digital Signal Processing (DSP 2002)* , 845 - 848.

Draper, B., Beveridge, J., Willem Böhm, A., Ross, C., & Chawathe, M. (2003). Accelerated Image Processing on FPGAs. *IEEE Transactions on image processing* , 1543 – 1551.

Gonzáles, R., & Woods, R. (1996). *Tratamiento digital de imagenes.* Addison-Wesley Iberoamerica, S.A. y ediciones Diaz de Santo S.A.

Grob, B. (1990). *Televisión práctica y sistemas de vídeo. Cuarta edición.* New York: McGraw-Hill.

Hamid, G. (1994). An FPGA-based coprocessor for image processing. *IEEE* , 6/1-6/4.

Ibáñez, L. (1990). *Basic television and video systems. Fifth Edition.* New York: McGraw-Hill.

Jamro, E., & Wiatr, K. (2001). Convolution Operation Implemented in FPGA Structures for Real-Time Image Processing. *Proceedings of the 2nd International Symposium on Image and Signal Processing and Analysis (ISPA 2001)* , 417-422.

Kessal, L., Abel, N., & Demigny, D. (2003). Real-time image processing with dynamically reconfigurable architecture. *Journal real-time imaging* , 297-313.

Maxinez, D., & Alcalá, J. (2002). *VHDL. El arte de programar sistemas digitales.* Mexico: CECSA.

Morris, M. (2003). *DISEÑO DIGITAL, TERCERA EDICIÓN.* California State University, Los Angeles: Pearson, Prentice Hall.

Pajares, G., & Manuel, J. (2002). *Vision por computadora: Imagenes digitales y aplicaciones* . MADRID, España.: RA-MA.

Payá, L., Reinoso, O., Gil, A., Pedrero, J. M., & Juliá, M. (2007). SEGUIMIENTO DE RUTAS MULTI-ROBOT USANDO PCA INCREMENTAL. *XXVIII Jornadas de Automática* (págs. 2-6). España: Universidad de Huelva.

Pérez, C., & Zamanillo, J. M. (2003). *Fundamentos de televisión analógica y digital.* España: Universidad de Cantabria. Publicaciones.

Piacentino, M. R., Van der Wal, G. S., & Hansen, M. W. (1999). Reconfigurable elements for a video pipeline processor. *Proceedings of the Seventh Annual IEEE Symposium on Field-Programmable Custom Computing Machines* , 82-91.

Quintero, A., & Vallejo, E. (2006). IMAGE PROCESSING ALGORITHMS USING FPGA. *Revista Colombiana de Tecnologías de Avanzada* , 6.

Ratha, N. K., & Jain, A. K. (1999). Computer Vision Algorithms on Reconfigurable Logic Arrays. *IEEE* , 29-43.

Tanvir, A. A., & Mohd, U. A. (2007). A Proposed FPGA Based Architecture for Sobel Edge Detection Operator. *J. of Active and Passive Electronic Devices, Vol 2* , 271–277.

Tarrés, F. (2000). *Sistema Audiovisuales: Televisión analógica y digital.* Barcelona: UPC (Edicion de la universitadad Politecnica de Catalunya, SL .

Tocci, R. J., Widmer, N. S., & Moss, G. (2007). *SISTEMAS DIGITALES. PRINCIPIOS Y APLICACIONES* (Español ed.). México, Mexico: Pearson Educación.

Vélez, J. F., Moreno, A. B., Sánchez, Á., & Esteban, J. L. (2003). *Visión por computador. Segunda edición.* Mexico.

Wiatr, K. (1998). Pipeline architecture of specialized Reconfigurable Processors in FPGA Structures for Real Time Image Pre-Processing. *Proceedings of the 3rd International Conference on Signal Processing,* , 131-138.

Ziegler, H., Byoungro, S., Hall, M., & Diniz, P. C. (2002). Coarse-grain pipelining on multiple FPGA architectures. *Proceedings of the 10th Annual IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM 02)* , 77 – 86.

# Anexos

## Tarjeta Altera DE2

El Ciclón FPGA II Starter Development Board (Figura 6.1) proporciona funciones integradas que permiten a los usuarios desarrollar y probar los diseños que van desde circuitos simples a varios proyectos de multimedia, todo ello sin la necesidad de aplicar complejas interfaces de programación de aplicaciones (API), el control de acogida de software, o SRAM / SDRAM / controladores de memoria flash.

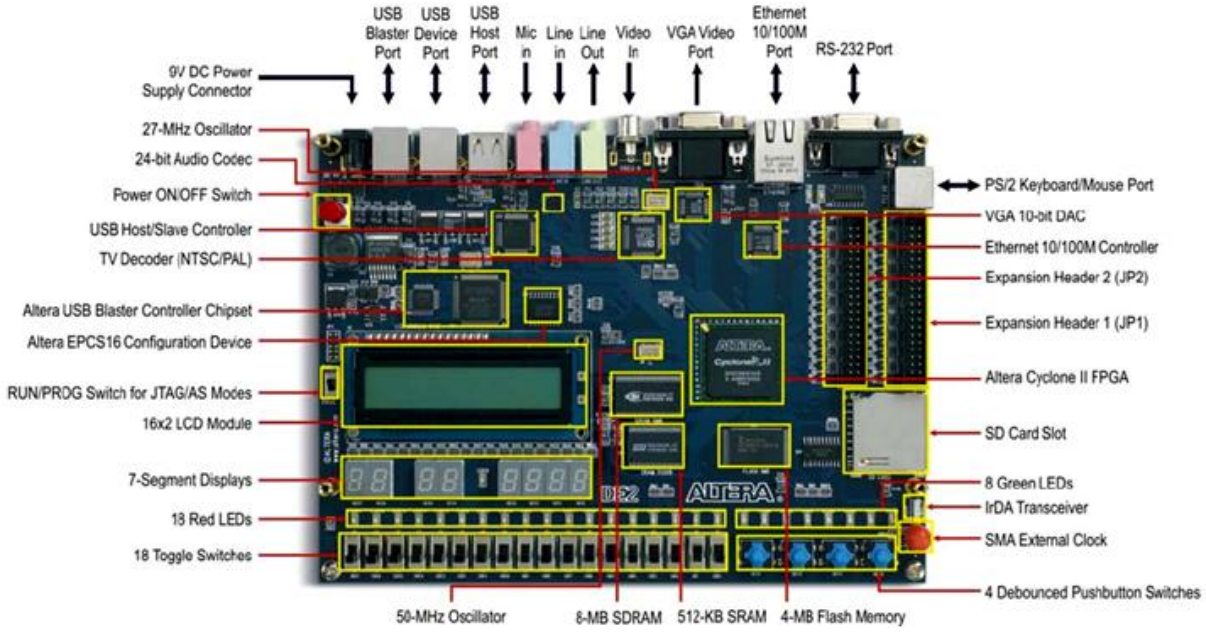



**Figura 6.1.**Tarjeta Altera DE2 Cyclone

La composición de la tarjeta DE2, se muestra en un diagrama a bloque (figura 6.2) donde se muestra las partes que la compone la tarjeta, así como componentes de entradas y salidas.

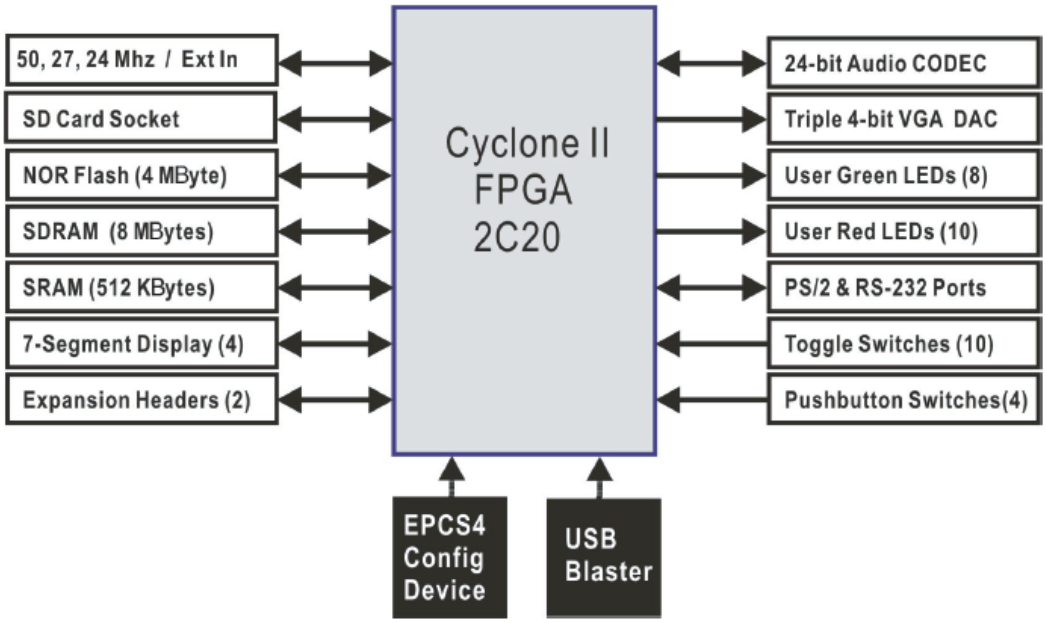



**Figura 6.2.**Diagrama a bloque de la tarjeta DE2 Cyclone II

La tarjeta DE2 cuenta con una pantalla TFT-LCD de 3.6’ de matriz activa y una minicamara de 1.3 MP el cual se utiliza para sistema de vision . Cuenta tambien con salida de video VGA, USB, Display de 7 Segmetos, swichts, botones, memorias SDRAM y FLASH. En la siguiente tabla se describe detalladamente cada elemento que la componen.

**Tabla 6.1.** Descripción de componentes de la tarjeta DE2

| Nombre | Descripción |
|---|---|
| **FPGA** | • Cyclone II EP2C35F672C6 con EPCS16 16-Mbit de configuración de dispositivos de serie |
| **Dispositivos I/O** | • Built-in cable USB Blaster para la configuración de FPG<br>• Ethernet 10/100<br>• RS232<br>• Salida de vídeo (VGA de 10-bit DAC)<br>• En Video (NTSC / PAL / Multi-formato)<br>• USB 2.0 (tipo A y tipo B)<br>• PS / 2 ratón o el teclado del puerto<br>• Línea de Entrada / Salida, Entrada de micrófono (24-bit Audio Codec)<br>• Cabeceras de Expansión (76 pines de la señal) |
| **Memoria** | • SDRAM 8-Mbytes, SRAM 512K, FLASH 4-Mbytes, Tarjeta de memoria SD |
| **Display** | • Display LCD 16X2<br>• 8 display de 7 segmentos |

| | |
|---|---|
| **Switches y LEDs** | • 18 interruptores de palanca<br>• 18 LEDs rojos<br>• 9 LEDs verdes<br>• Cuatro interruptores de pulsador debounced |
| **Relojes** | • 50 MHz para la entrada de cristal del reloj FPGA<br>• 27 de cristal MHz para aplicaciones de vídeo<br>• Entrada externa de reloj SMA |