UNIVERSIDAD TECNOLÓGICA DE LA MIXTECA

DIVISIÓN DE ESTUDIOS DE POSGRADO

Diseño e implementación de una herramienta HW/SW enfocada al procesamiento digital de video sobre lógica reconfigurable

TESIS

PARA OBTENER EL GRADO DE:

Maestro en Electrónica, Opción: Sistemas Inteligentes Aplicados

PRESENTA:

Ing. Carlos González Rojas

Director de tesis: Dr. Enrique Guzmán Ramírez

Co-Director de tesis: Dr. Iván Antonio García Pacheco

Huajuapan de León, Oaxaca, Febrero de 2015.

Tesis presentada en Febrero de 2015 ante los sinodales:

Dr. Felipe de Jesús Trujillo Romero

Dr. José Anibal Arias Aguilar

Dr. Rosebet Miranda Luna

Director de tesis:

Dr. Enrique Guzmán Ramírez

Co-Director de tesis:

Dr. Iván Antonio García Pacheco

Resumen

El procesamiento digital de video es un área en constante evolución y de gran interés debido al creciente número de aplicaciones donde es requerido. En particular en el área de la visión artificial el procesamiento digital de video toma un papel importante, debido a que el objetivo de la visión artificial es el comprender la información visual proveniente del mundo real, con el fin de producir decisiones en forma de información simple, numérica o simbólica. Como disciplina tecnológica la visión artificial busca aplicar sus teorías y modelos para la construcción de sistemas que cumplan con estos requisitos. Algunos ejemplos de aplicaciones de la visión artificial incluyen sistemas de visión estéreo, la detección de eventos para la vigilancia visual, la navegación de vehículos autónomos o robots móviles, el modelado de objetos o entornos, la inspección automática, la interacción con entornos, etc., donde en el mayor de los casos se requiere de sistemas autónomos capaces de efectuar procesamiento digital de video en tiempo real. En este sentido, la tecnología basada en FPGAs tiene gran potencial en estas áreas, esto debido a que explota el procesamiento paralelo teniendo un rendimiento próximo al de un circuito de aplicación específica con la capacidad de reconfiguración de un microprocesador. Sin embargo, para el desarrollo de aplicaciones de procesamiento de video en tiempo real basadas en FPGAs, se requiere de sólidos conocimientos en estas áreas. Por tal motivo, en este trabajo se presenta una herramienta HW/SW que permite la implementación y evaluación de algoritmos de procesamiento de video en un FPGA, cuyo objetivo es apoyar el proceso de enseñanza/aprendizaje en estas áreas.

Dedicatoria

Dedico la presente tesis:

A mis padres y hermanos quienes con su cariño, apoyo y comprensión incondicional han estado siempre a lo largo de mi vida.

También quiero dedicar esta tesis a mi novia y amiga Ivonne, por ser alguien muy especial en mi vida y por demostrarme que en todo momento cuento contigo. Gracias por existir, te amo peque.

Carlos González Rojas.

Agradecimientos

Quiero expresar mi más sincero agradecimiento, reconocimiento y cariño a mis padres Luis y Angela por todo el esfuerzo que hicieron para darme una profesión y hacer de mi la persona que soy, gracias por los sacrificios y la paciencia que demostraron todos estos años; gracias a ustedes he llegado a donde estoy.

Gracias a mis hermanos Kary, Luis Angel y Elsita quienes han sido mis amigos sinceros, en los que puedo confiar y apoyarme para seguir adelante.

Agradezco de manera especial a mi director y amigo Dr. Enrique Guzmán Ramírez (Kike) quién con sus conocimientos, apoyo y amistad supo guiar el desarrollo de la presente tesis desde el inicio hasta su culminación.

De igual forma agradezco al Dr. Iván Antonio García Pacheco por su amistad y por compartir sus conocimientos en mi formación académica y en el enriquecimiento de este trabajo de tesis.

A mis sinodales: Dr. Rosebet Miranda Luna, Dr. Felipe de Jesús Trujillo Romero y Dr. José Anibal Arias Aguilar, por sus valiosas sugerencias. Gracias por todo su tiempo invertido en la revisión de esta tesis.

Finalmente, gracias a todas aquellas personas que de una u otra forma me ayudaron a crecer como persona y como profesional.

Índice

RESUMEN V
DEDICATORIA VII
AGRADECIMIENTOS IX
ÍNDICE XI
ÍNDICE DE FIGURAS XV
ÍNDICE DE TABLAS XIX
1 INTRODUCCIÓN 1
1.1 INTRODUCCIÓN 1
1.2 MOTIVACIÓN 3
1.3 JUSTIFICACIÓN 4
1.4 HIPÓTESIS 6
1.5 OBJETIVOS 6
1.6 PUBLICACIONES GENERADAS 6
1.7 ESTRUCTURA DE LA TESIS 7
2 PROCESAMIENTO DIGITAL DE VIDEO 9
2.1 VIDEO DIGITAL 9
2.1.1 Video progresivo y entrelazado 10
2.1.2 Profundidad de color 11
2.1.3 Espacios de color 11
2.1.4 Espacio RGB 11
2.1.5 Espacio YCbCr 12
2.1.6 Redundancia en el video 13

2.1.6.1 Redundancia espacial 13
2.1.6.2 Redundancia temporal 13
2.2 INTRODUCCIÓN AL PROCESAMIENTO DIGITAL DE VIDEO 14
2.2.1 Mejora y restauración de video 15
2.2.1.1 Operaciones puntuales 16
2.2.1.1.1 El negativo 16
2.2.1.1.2 Mejora de contraste 16
2.2.1.2 Operaciones de Filtrado 16
2.2.1.2.1 Filtros espaciales 17
2.2.1.2.2 Filtrado en el dominio de la frecuencia 17
2.2.1.2.3 Filtros espacio temporales 20
2.2.1.2.3.1 Filtrado temporal 20
2.2.1.2.3.2 Filtrado espacio temporal 21
2.2.2 Estimación y detección de movimiento 22
2.2.2.1 Flujo óptico 22
2.2.2.2 Análisis del movimiento basado en la correspondencia 23
2.2.2.3 Métodos diferenciales 24
2.2.3 Segmentación de video 24
2.2.3.1 Métodos de Detección de Bordes 25
2.2.3.2 Crecimiento de Regiones 26
2.2.3.3 Segmentación basada en el modelo de fondo 26
2.2.3.4 Segmentación basada en flujo óptico 27
2.2.4 Compresión de video 28
2.2.4.1 Codificador 28
2.2.4.1.1 Estructura de los macro bloques 30
2.2.4.1.2 Estimación y compensación de movimiento 31
2.2.4.1.3 La Transformada Discreta del Coseno 31
2.2.4.1.4 Cuantificación 32
2.2.4.1.5 Codificación de entropía 32
2.2.4.2 Decodificador 33
2.3 PROCESAMIENTO DE VIDEO EN TIEMPO REAL 33
2.3.1 Paralelismo en las operaciones de procesamiento de video 35
2.3.2 Sistemas de procesamiento de video basados en FPGAs 36
2.3.3 Herramientas para la implementación de algoritmos de procesamiento de video basados en FPGAs 38

3 MÉTODOS UTILIZADOS 45

3.1 ARREGLO DE COMPUERTAS PROGRAMABLES EN CAMPO (FPGAS) 45
3.1.1 Arquitectura de las FPGAs 46
3.1.1.1 Bloques lógicos configurables 47
3.1.1.2 Líneas de interconexión 47
3.1.1.3 Bloques configurables de entrada/salida 47
3.1.2 Programación de un FPGA 47
3.2 CONSTRUCTIVISMO 48
3.2.1 Características fundamentales para un aprendizaje efectivo 50
3.2.1.1 Compromiso activo 50
3.2.1.2 Participación en grupos 51

3.2.1.3 Interacción frecuente y realimentación 51
3.2.1.4 Contacto con el contexto del mundo real 52

4 ESTRUCTURA DE LA HERRAMIENTA PROPUESTA 55

4.1 SISTEMA DE CAPTURA Y PROCESAMIENTO DE VIDEO BASADO EN UN FPGA 55
4.1.1 Descripción de los componentes Hardware 57
4.1.1.1 Elemento central de procesamiento 58
4.1.1.2 Sistema de comunicación 59
4.1.1.3 Sistema de visualización de resultados 60
4.1.1.4 Sistema de memoria 61
4.1.1.5 Sistema de captura de video 63
4.1.2 Descripción de los componentes software 64
4.1.2.1 Interfaz gráfica de usuario 64
4.1.2.1.1 Protocolo de comunicación 67
4.1.2.2 Descripción HDL del elemento central de procesamiento 69
4.1.2.2.1 Administrador de comunicación 70
4.1.2.2.2 Administrador de reloj 74
4.1.2.2.3 Administrador de visualización de resultados 75
4.1.2.2.4 Administrador de captura de video 78
4.1.2.2.5 Administrador del sistema de memoria. 83
4.1.2.2.6 Administrador de algoritmos 86
4.2 CONSIDERACIONES PARA EL DESARROLLO DE PRÁCTICAS EXPERIMENTALES MEDIANTE EL USO DEL SCPV 90
4.2.1 Fase de planeación 90
4.2.2 Fase de creación 91
4.2.2.1 Diseño 91
4.2.2.2 Implementación 92
4.2.2.3 Documentación 93
4.2.3 Fase de aprendizaje 93

5 REPERTORIO DE ALGORITMOS 95

5.1 IMPLEMENTACIÓN DEL ALGORITMO DEL NEGATIVO 95
5.2 IMPLEMENTACIÓN DEL ALGORITMO DE LA BINARIZACIÓN POR UMBRAL. 98
5.3 ALGORITMO DE SUSTRACCIÓN DE FONDO 101
5.4 ALGORITMO DE EXTRACCIÓN DE BORDES MEDIANTE EL OPERADOR SOBEL 104

6 CONCLUSIONES Y TRABAJOS FUTUROS 109

6.1 CONCLUSIONES 109
6.2 TRABAJOS FUTUROS 111

7 REFERENCIAS 113

8 ACRÓNIMOS 121

Índice de figuras

Figura 1.1 Rol de la herramienta propuesta en el proceso de enseñanza/aprendizaje. 3
Figura 2.1 Secuencia de video digital. 10
Figura 2.2 Video progresivo y entrelazado. 11
Figura 2.3 Cubo de color RGB. 12
Figura 2.4 Redundancia espacial. 13
Figura 2.5 Redundancia temporal. 14
Figura 2.6 Ejemplo de una máscara de 3x3. 17
Figura 2.7 Proceso de filtrado en el dominio de la frecuencia. 19
Figura 2.8 Función de transferencia ideal pasa-bajas con una frecuencia de corte $D_{0,}$ a) 1D, b) 2D. 19
Figura 2.9 Función de transferencia de un filtro Butterworth paso- bajas, a) 1D, b) 2D. 20
Figura 2.10 Filtro temporal con compensación de movimiento sobre el pixel *g(n, k).* 21
Figura 2.11 Ejemplo de la aplicación d una ventana espacio temporal. 22
Figura 2.12 Operador Sobel. 25
Figura 2.13 a) Imagen original, b) Magnitud del gradiente aplicando el operador Sobel, c) Información de los bordes. 26
Figura 2.14 a) Modelo de fondo, b) Secuencia de video, c) Segmentación de objetos en movimiento. 27
Figura 2.15 Codificador híbrido de video. 29
Figura 2.16 Ejemplo de grupo de imágenes codificadas. 30
Figura 2.17 Estructura de un macro-bloque. 30
Figura 2.18 Estimación de movimiento basado en una imagen anterior. 31
Figura 2.19 Decodificador de video híbrido. 33
Figura 2.20 Arquitectura de la herramienta UltraSONIC [HAY02]. 38
Figura 2.21 Arquitectura de la herramienta propuesta en [ATI04]. 39
Figura 2.22 Arquitectura propuesta en [DES09]. 40

Figura 2.23 Sistema de procesamiento de video propuesto en [HIR10]. 40
Figura 2.24 Arquitectura para un sistema de visión estéreo [JIN10] 41
Figura 2.25 Diagrama a bloques de la arquitectura propuesta para el procesamiento de video en [GOP12]. 42
Figura 2.26 Arquitectura para una herramienta de prototipado rápido [SAI12]. 43
Figura 2.27 Resumen de la plataforma utilizada en [DES10] y [DES12]. 43
Figura 2.28 Estructura de la herramienta propuesta en [GAR13]. 44
Figura 3.1 Estructura general de un FPGA 46
Figura 4.1 Sistema de procesamiento digital de video basado en un FPGA. 57
Figura 4.2 Tarjeta de evaluación Digilent Genesys. 59
Figura 4.3 Diagrama de casos de uso de la interfaz de usuario. 65
Figura 4.4 Clase PuertoUSB. 66
Figura 4.5 Formulario de configuración del sistema de adquisición de video. 66
Figura 4.6 Formulario de configuración del sistema de visualización de resultados. 67
Figura 4.7 Estructura de la trama de comunicación. 67
Figura 4.8 Arquitectura del elemento central de procesamiento del SECPV. 70
Figura 4.9 a) Símbolo del *administrador de comunicación*, b) Estructura interna del *administrador de comunicación* 70
Figura 4.10 Diagrama de tiempos para el proceso de escritura del DLP-USB1232H. 72
Figura 4.11 Diagrama de tiempos para el proceso de lectura del DLP-USB1232H. 73
Figura 4.12 Maquina de estados del módulo *decodificador de trama.* 73
Figura 4.13 a) Símbolo del *administrador de reloj*, b) Estructura interna. 74
Figura 4.14 a) Símbolo del *administrador de visualización de resultados*, b) Estructura interna. 75
Figura 4.15 Diagrama de tiempos para el envío de video hacia el LCD AT043TN24 V.7. 77
Figura 4.16 Estructura de la señal de video digital. 78
Figura 4.17 a) Símbolo del *administrador de captura de video,* b) Estructura interna. 78
Figura 4.18 Secuencia de reinicio de la cámara Aptina MT9D112. 80
Figura 4.19 Proceso de escritura en el protocolo I^2C. 81
Figura 4.20 Proceso de lectura en el protocolo I^2C. 82
Figura 4.21 a) Símbolo del *administrador de memoria, b*) Estructura interna. 86
Figura 4.22 a) Símbolo del *administrador de algoritmos,* b) Estructura interna. 87
Figura 4.23 Diagrama de conexiones internas del e*lemento central de procesamiento.* 89
Figura 4.24 Marco de trabajo para el desarrollo de prácticas experimentales. 90
Figura 5.1 Negativo de una imagen. 96
Figura 5.2 a) Símbolo del algoritmo del negativo, b) Estructura interna del símbolo. 97
Figura 5.3 Circuito implementado para la obtención del negativo. 97
Figura 5.4 Formulario de configuración del algoritmo del negativo. 98
Figura 5.5 Resultado al aplicar el algoritmo del negativo. 98
Figura 5.6 Resultado de aplicar la transformación de binarización. 99
Figura 5.7 a) Símbolo del algoritmo de binarización, b) Estructura interna. 100
Figura 5.8 Circuito que implementa la binarización por umbral. 100
Figura 5.9 Interfaz de usuario para la configuración del algoritmo de binarización. 101
Figura 5.10 Resultado del procesamiento de binarización en la herramienta. 101
Figura 5.11 a) Símbolo del módulo que implementa el algoritmo de sustracción de fondo, b) Estructura interna. 103
Figura 5.12 Diagrama funcional del módulo *sustracción.* 103

Figura 5.13 Resultados del esquema de sustracción de fondo.....104
Figura 5.14 Interfaz de usuario para la configuración del algoritmo de sustracción de fondo..104
Figura 5.15 Operación de convolución.....105
Figura 5.16 Circuito de convolución con una máscara de 3×3.....106
Figura 5.17 Circuito implementado para la obtención de bordes mediante el operador Sobel. 106
Figura 5.18 a) Símbolo del módulo de extracción de bordes, b) Estructura interna del símbolo. 107
Figura 5.19 Resultados de la implementación del extractor de bordes en la herramienta.....107
Figura 6.1 Tiempo estimado para la realización de un sistema de procesamiento de video. 110
Figura 6.2 Tiempo estimado para la realización de un sistema de procesamiento de video basado en la herramienta propuesta. 110

Índice de tablas

Tabla 1 Descripción de los pines del DLP-USB1232H.....60
Tabla 2 Descripción de las señales que intervienen en el control del LCD. 61
Tabla 3 Descripción de los puertos de la memoria DDR2. 62
Tabla 4 Conexiones que provee el módulo Digilent VmodCam. 64
Tabla 5 Tipo de tramas soportadas por el protocolo de comunicación. 68
Tabla 6 Descripción de las entradas y salidas del módulo administrador de comunicación. 71
Tabla 7 Tiempos para los procesos de lectura y escritura del DLP-USB1232H. 72
Tabla 8 Descripción de las señales de entrada y salida del *administrador de reloj*. 74
Tabla 9 Descripción de las señales de entrada y salida del *administrador de visualización de resultados*. 75
Tabla 10 Parámetros para el envío de la señal de video. 77
Tabla 11 Descripción de las señales de entrada y salida del administrador de captura de video. 79
Tabla 12 Descripción de las señales de entrada y salida del administrador de memoria. 84
Tabla 13 Descripción de las señales de entrada/salida del *administrador de algoritmos.* 87
Tabla 14 Resumen de recursos consumidos por el *elemento central de procesamiento* en el FPGA. 88

Capítulo 1

Introducción

1.1 Introducción

Con el desarrollo tecnológico ha crecido la necesidad de facilitar el aprendizaje y la aplicación de los conocimientos, para ello la enseñanza universitaria en las áreas de Ciencias de la Computación y Electrónica requiere de herramientas especializadas que faciliten el proceso de enseñanza/aprendizaje a través de un enfoque práctico. En este sentido, de acuerdo con Aziz [AZI11], Tsai [TSA11] y Shyr [SHY12], la enseñanza universitaria en ingeniería ha sufrido diversos cambios en los últimos años debido a la integración de las Tecnologías de la Información (TI) en las aulas de clases, proporcionando un entorno innovador de enseñanza/aprendizaje a los estudiantes. Así, el crecimiento de áreas como la Automatización y la Robótica exige sistemas autónomos dotados de sofisticados sistemas de percepción como la visión artificial, en dónde las técnicas de procesamiento digital de imágenes y de video son elementos claves para su desarrollo. Sin embargo, la educación a nivel ingeniería no está proporcionando buenos resultados en este sentido, dado que los estudiantes no participan en la creación de sus propios conocimientos, son solamente observadores pasivos.

En este sentido, el procesamiento digital de video es la manipulación matemática, mediante la ejecución de un algoritmo, de una secuencia de imágenes relacionadas entre sí con la finalidad de analizar, modificar y/o mejorar la información contenida en ella y aplicarla a un fin específico. En el área de procesamiento digital de video se encuentran las siguientes categorías de algoritmos: adquisición, procesamiento, análisis y comprensión de la secuencia de video. En cada una de estas categorías un gran número de teorías y modelos han sido propuestos, documentados y aplicados a la vida cotidiana. Por otro lado, el desarrollo de aplicaciones basadas en sistemas embebidos es un área tecnológica de gran crecimiento en las últimas

décadas. A diferencia de lo que ocurre con una computadora personal (PC), que está diseñada para cubrir un amplio rango de necesidades, las aplicaciones basadas en sistemas embebidos se diseñan para cubrir necesidades específicas. En el desarrollo de aplicaciones que emplean un sistema embebido para el procesamiento digital de video, destacan plataformas basadas en procesadores digitales de señales (DSP, *Digital Signal Processor*) [ILL00], arreglos de compuertas programables (FPGA, *Field Programmable Gate Array*) [PRI06], e híbridas (DSP-FPGA) [FRE02][CHR04].

La creciente demanda de este tipo de aplicaciones ha originado el desarrollo de herramientas, tanto en el ámbito educativo como industrial, que faciliten su proceso de enseñanza y desarrollo. Dentro de este contexto varias herramientas han sido diseñadas, sin embargo la presente tesis se concentra únicamente en aquellas que tienen como elemento central de procesamiento a un FPGA. Algunos ejemplos de estas investigaciones son brevemente descritos a continuación:

En [ATI04], Atitallah *et al*. proponen una arquitectura para una plataforma HW que facilita el desarrollo de aplicaciones de video embebidas en un FPGA. El sistema de procesamiento de video propuesto se compone de los módulos de adquisición, procesamiento y visualización. El módulo de adquisición incluye una interfaz para una cámara de 8 bits de color y el módulo de visualización se compone de una interfaz VGA de 24 bits, ambos módulos se desarrollaron en base al IP DMA (*Intellectual Property Direct Memory Access*) lo que permite la transferencia directa de los datos entre la cámara y la memoria SDRAM, y de la SDRAM al monitor VGA. En cuanto al sistema de control, la arquitectura propuesta integra el microprocesador *Niossoftcore* de 32 bits de arquitectura RISC interconectado con los otros módulos mediante el bus *Avalon*.

De igual forma Said y Saidani proponen en [SAI12] una herramienta de prototipado rápido para aplicaciones de procesamiento de video en HW. Esta herramienta proporciona un entorno de desarrollo que permite a los diseñadores empezar rápidamente a experimentar con el procesamiento de video utilizando la tarjeta VSK Spartan-3A DSP desarrollada por Xilinx.

Por otra parte, Hiraiwa y Amano proponen una arquitectura de procesamiento de video basada en FPGA para sistemas embebidos de visión en tiempo real [HIR13]. La arquitectura de procesamiento de video implementada es segmentada, lo que ofrece la flexibilidad de intercambiar los módulos de procesamiento.

Como se puede observar, las herramientas descritas anteriormente se enfocan a utilizar a un FPGA como elemento principal de procesamiento, sin embargo ninguno de ellos proporciona un método para introducir la herramienta propuesta en el contexto de cursos relacionados con la ingeniería (o al menos no se muestra evidencia de esto), además estos trabajos no presentan información relacionada con su incorporación bajo algún enfoque educativo que apoye el proceso de aprendizaje. En este sentido, el paradigma constructivista y el enfoque PBL (*Project-based learning*) han sido explorados ampliamente por varios investigadores como un soporte eficiente en la educación en el área de las Ciencias de la Computación [HUR06][ALO09][APP10][SAM10], puesto que los estudiantes toman un papel activo en su aprendizaje dado que no solamente retienen la información, sino que también la relacionan con el conocimiento previamente adquirido en las aulas y lo mejoran con la construcción de su nuevo conocimiento a través de herramientas tecnológicas.

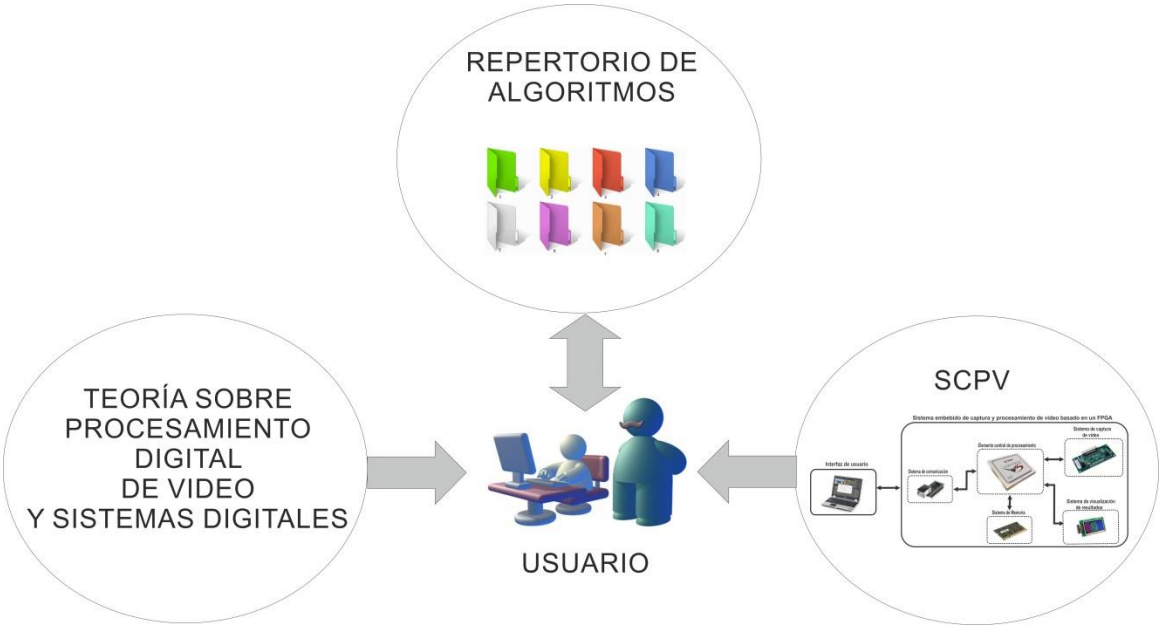

Figura 1.1 Rol de la herramienta propuesta en el proceso de enseñanza/aprendizaje.

Por lo tanto, el presente trabajo de tesis propone una herramienta HW/SW enfocada al desarrollo y evaluación de algoritmos de procesamiento de video en un FPGA, que sirva de soporte a cursos relacionados con las Ciencias de la Computación a nivel ingeniería y de posgrado (ver Figura 1.1). Con esta propuesta, el conjunto de componentes básicos (interfaz de usuario, sistema de adquisición, control y almacenado de secuencias de video) forman parte de la herramienta SCPV (Sistema de Captura y Procesamiento de Video), lo que permite que el usuario centre su atención únicamente en el diseño y modelado del algoritmo en estudio, concentrándose en aprender a través de su experiencia. Además, con base en características tomadas del enfoque constructivista, se incorpora un repertorio de algoritmos que pueden ser adaptados al tema que es objeto de estudio. Este repertorio de algoritmos puede ser fácilmente ampliado y modificado, para que sean los estudiantes quienes generen su propio conocimiento con la práctica.

1.2 Motivación

El procesamiento digital de video es un área en constante evolución y de gran interés debido al creciente número de aplicaciones donde es requerido, entre las cuales se pueden mencionar: la visión artificial, compresión de video, videoconferencias, servidores de análisis de video, TV de alta definición (HDTV), interpretación de secuencias de video en seguridad civil y militar, etc.

Por otro lado, cuando se trata de implementar aplicaciones de procesamiento de video, los sistemas embebidos son vistos como la alternativa idónea a las aplicaciones basadas en PC, esto debido a factores como la disminución del costo del producto final, el gran avance tecnológico en esta área, la existencia de una amplia gama de dispositivos de procesamiento, la evolución de las herramientas para la automatización del diseño electrónico (EDA, *Electronic Design Automation*), la ventaja de tratar con un sistema dedicado a una aplicación específica, etc. Estos factores y la progresiva demanda de este tipo de aplicaciones, han contribuido a que los cursos especializados (a nivel ingeniería, maestría y doctorado), en los que conviven el procesamiento digital de video y los sistemas embebidos sean mejorados de tal forma que los estudiantes

comprendan cómo funcionan los algoritmos antes de implementarlos. Debido a esto, existe una creciente necesidad de contar con herramientas especializadas que permitan al estudiante analizar el comportamiento y desempeño de los algoritmos de procesamiento y evaluar la eficiencia de la implementación de modelos de procesamiento de video, variantes de éstos o de nuevos modelos, a través de sistemas embebidos. Además, es de vital importancia que los diseñadores de dichas herramientas consideren para su diseño las características de un enfoque educativo que permita integrar estas ideas con la TI en el contexto de un programa educativo, con la finalidad de facilitar el proceso enseñanza/aprendizaje.

De acuerdo con Gillet *et al*. [GIL03], *"en el espíritu del aprendizaje flexible, los estudiantes tienen la posibilidad de realizar un experimento en cualquier momento y desde cualquier lugar, beneficiándose a través de una experiencia cognitiva más eficiente"*. Sin embargo, la educación universitaria ha sufrido mucho tiempo por el marcado énfasis en la adquisición de conocimientos y habilidades a través de entornos académicos que no contemplan la experiencia individual del estudiante, mientras que la sociedad y la industria piden habilidades que a menudo sólo puede lograrse en un contexto de experimentación individual donde no solamente se implemente un algoritmo, si no que se entienda cómo funciona [KIR04]. Así, la naturaleza de las áreas relacionadas con las Ciencias de la Computación implica el razonamiento y aprendizaje individual como un imperativo lógico. En este sentido, Kreijns *et al*. [KRE07] afirman que el trabajo en un entorno de aprendizaje asistido por HW/SW tiene consecuencias positivas tanto en términos de resultados de aprendizaje (dimensión educativa) así como de desempeño social (dimensión psicológica/social), además de la expectativa de mejoras en la satisfacción de los estudiantes. En concreto, se intenta rebasar el paradigma tradicional donde el alumno aprende a implementar algoritmos de procesamiento de video a través de la programación pura y dura o bien a través de la simulación con SW especializado, construyendo un enfoque diferente que le permitirá probar y analizar algoritmos almacenados en un repositorio existente, pero que además le da la posibilidad de crear sus propios algoritmos y probarlos antes de implementarlos en cualquier sistema embebido.

1.3 Justificación

Para dar solución al problema planteado, el presente trabajo propone el diseño e implementación de una herramienta enfocada al desarrollo y evaluación de algoritmos de procesamiento de video sobre un sistema embebido que tiene como elemento de procesamiento a un FPGA, cuyo objetivo es apoyar al estudiante durante el proceso enseñanza/aprendizaje utilizando dicha herramienta como principal soporte de su formación. Por lo anterior, se considera necesario que se justifiquen los dos elementos clave que permitirán su desarrollo. En primer lugar, las razones que evidencian la necesidad de un FPGA como elemento central de procesamiento de la herramienta propuesta se basan principalmente en que el procesamiento de video es una tarea altamente demandante y está fuertemente vinculada con el procesamiento en tiempo real. La velocidad de procesamiento requerida para que un sistema de esta naturaleza opere en tiempo real depende del estándar de video utilizado. Por ejemplo, para un estándar NTSC, donde existen 60 cuadros por segundo, el tamaño de un cuadro de video es de 720×480 y un pixel es representado en 16 bits, por lo tanto el sistema debe ser capaz de procesar 75 Mbps. Para los microprocesadores utilizados en las PC convencionales, la implementación de algoritmos relacionados con el procesamiento de video aún resulta una tarea con mucha demanda de cómputo, esto sin mencionar la carga que el sistema operativo y otras aplicaciones representan para el microprocesador. Es verdad que el procesamiento en paralelo

es una solución alternativa para este problema, pero esto implica el uso de varias computadoras o en su caso grandes computadoras, lo cual repercute en un elevado costo para realizar dicha tarea. Además, el uso de una PC para albergar un algoritmo de procesamiento de video es poco viable cuando éste será utilizado por un sistema autónomo. Por otro lado, los dispositivos de procesamiento utilizados en los sistemas embebidos han incrementado sus capacidades en diferentes parámetros tales como velocidad, memoria, componentes de arquitectura fija, etc., a la vez que se han disminuido su costo y tamaño. Debido a esto, y a los factores mencionados en el apartado anterior, el desarrollo de aplicaciones de procesamiento digital de video mediante un sistema embebido es cada vez más frecuente. En este sentido, el elemento central de procesamiento de un sistema embebido, para este tipo de aplicaciones, puede ser un micro controlador, un DSP o un FPGA. Un DSP es un sistema basado en un procesador que posee recursos optimizados para aplicaciones que requieran operaciones numéricas a muy alta velocidad. Por su parte, el FPGA posee una estructura, formada por una gran densidad de elementos lógicos configurables comunicados entre sí a través de una red de interconexiones programables, que permite implementar técnicas avanzadas de procesamiento y conseguir altas velocidades de procesamiento. Por sus características, ambos dispositivos son especialmente útiles para el procesamiento y representación de señales en tiempo real. En relación al procesamiento de la información (ejecución del algoritmo), un DPS usa un enfoque algorítmico, es decir ejecuta un conjunto de instrucciones secuencialmente; mientras que un FPGA es capaz de evaluar varias hipótesis concurrentemente, esta característica lo hace un dispositivo idóneo para implementar técnicas de paralelismo, obtener altas velocidades de procesamiento, mejorar el desempeño y, con respecto a la propuesta presentada en este proyecto, implementar algoritmos de procesamiento de video avanzados.

En segundo lugar, el uso correcto y eficaz de la TI en la educación requiere que la aplicación sea compatible con teorías pedagógicas probadas en diversas áreas de la ingeniería. Por tal motivo, se justifica el estudio de la teoría constructivista con la finalidad de extraer especificaciones para el diseño de la herramienta, que permita fortalecer el proceso de enseñanza/aprendizaje en el procesamiento digital de video. En base a estas especificaciones se plantea dar más importancia a los contextos de aprendizaje como una alternativa a la tradicional memorización de conceptos. Esto permite la construcción de conocimientos dado que los estudiantes realizan actividades más cercanas al mundo real y por lo general éstas promueven la discusión de los resultados. De acuerdo con los autores constructivistas, los contextos significativos para cualquier área de la ingeniería están estrechamente relacionados con situaciones del mundo real que ayudan al estudiante a poner en práctica la experiencia. En este sentido, los entornos de aprendizaje deben ser flexibles y se caracterizan por el hecho de que el mismo conocimiento puede ser representado de diferentes maneras. Así, los estudiantes aprenden a través de la variedad de propuestas. En cuanto al papel de la TI en los entornos constructivistas, este proyecto se sustenta en la idea de que no se debe utilizar solamente para expresar el conocimiento. Por lo contrario, debe ser una herramienta de soporte para la experimentación y la construcción de conocimiento [MOR07]. En el contexto de este trabajo de tesis, existen diversas propuestas que han incorporado exitosamente un enfoque constructivista a la educación universitaria en diferentes áreas de las Ciencias de la Computación a través de siete principios de los cuales se pueden retomar características para ser ajustadas a la problemática planteada en este documento mediante especificaciones de diseño, estos principios son:

- Promover el contacto entre los estudiantes y profesores.

- Desarrollar la reciprocidad y cooperación entre los estudiantes.
- Motivar el aprendizaje activo.
- Proporcionar realimentación inmediata.
- Enfatizar el tiempo de una tarea.
- Comunicar altas expectativas.
- Respetar la diversidad de talentos y las formas de aprendizaje.

1.4 Hipótesis

Una herramienta enfocada al diseño y evaluación de algoritmos de procesamiento digital de video basada en un sistema embebido, que tiene como elemento de procesamiento a un dispositivo FPGA, puede ayudar al proceso aprendizaje/enseñanza dentro del ámbito educativo y agilizar el proceso diseño-modelado-implementación de dichos algoritmos en una aplicación final.

1.5 Objetivos

El objetivo principal del presente trabajo de tesis es diseñar e implementar una herramienta basada en un sistema embebido, con un FPGA como elemento de procesamiento, que permita diseñar y evaluar algoritmos de procesamiento digital de video en cursos de licenciatura y posgrado.

Del objetivo principal se desprenden los siguientes objetivos secundarios:

1. Desarrollo de una herramienta HW/SW que permita la aplicación y evaluación de algoritmos de procesamiento de video en tiempo real.
2. Incorporar a la herramienta características del enfoque constructivista para facilitar su incorporación en el ámbito educativo y facilite el proceso enseñanza/aprendizaje en temas relacionados con el procesamiento de video en dispositivos digitales.
3. Incorporar un repertorio de algoritmos que motive al usuario a proponer nuevas aplicaciones a través de los ejemplos ya existentes o versiones completamente nuevas que enriquezcan este repertorio.

1.6 Publicaciones generadas

Guzmán-Ramíre, E**.**, Arroyo-Fernández I. and Gonzalez-Rojas C., Linares-Flores J. and Oleksiy P. “FPGA-based architecture for Extended Associative Memories and its Application in Image Recognition”, *Advances in Artificial Intelligence, Lecture Notes in Computer Science LNAI,* Vol. 7629, pp. 194-204, ISSN: 0302-9743, 2013.

García I., Guzmán-Ramírez E. And Gonzalez-Rojas C. "A constructivist approach for implementing and evaluating algorithms in a machine vision course at the undergraduate level", *Procedia Social and Behavioral Science Journal*, Vol. 93, pp. 1461-1466, ISSN: 1877-0428, 2013.

1.7 Estructura de la tesis

El documento de tesis que se presenta está organizado en seis capítulos. En este primer capítulo se aborda la motivación que da origen a este trabajo y su respectiva justificación, la hipótesis planteada y los objetivos que se desean alcanzar. En el Capítulo 2 se encuentran las bases teóricas que dan un panorama general acerca del procesamiento digital de video. El Capítulo 3 describe la teoría de los FPGAs y el enfoque constructivista, los cuales son elementos claves para el desarrollo de este trabajo. La descripción detallada del sistema de captura y procesamiento de video, es mostrada en el capítulo 4, mientras que en el capítulo 5 se presentan los algoritmos que forman parte del repertorio inicial de algoritmos con el que se entrega la herramienta. Por ultimo en el capítulo 6 se presentan las conclusiones generadas del desarrollo de la herramienta y trabajos futuros que le darán continuidad.

Capítulo 2

Procesamiento digital de video

El presente trabajo ofrece una herramienta enfocada al procesamiento de video. Buscando sentar las bases para una mejor compresión de la propuesta, este capítulo examina el marco teórico del tema en estudio, incluyendo un breve repaso acerca del video digital y vertiendo los principales conceptos acerca del mismo.

2.1 Video digital

El video se puede definir como un conjunto o secuencia de imágenes con alto grado de relación entre sí, por tanto el video digital se puede considerar como una secuencia de imágenes digitales fijas [BOV09]. Esencialmente, el video es de carácter dinámico, es decir, su contenido visual evoluciona con respecto el tiempo y generalmente contiene cambios en los objetos y/o movimiento [MUR95]. Particularmente, el video digital está representado por una matriz de tres dimensiones, dos en el espacio y una en el tiempo, como se muestra en la Figura 2.1. En esta figura se aprecia como el video digital es una sucesión de imágenes bidimensionales que ocurre en instantes de tiempo discreto. Cada punto es una muestra espacial que representa un elemento de imagen o pixel (abreviación en ingles de "*Picture Elements*") y cada imagen individual o muestra temporal especifica un cuadro o imagen (comúnmente llamado "*frame*").

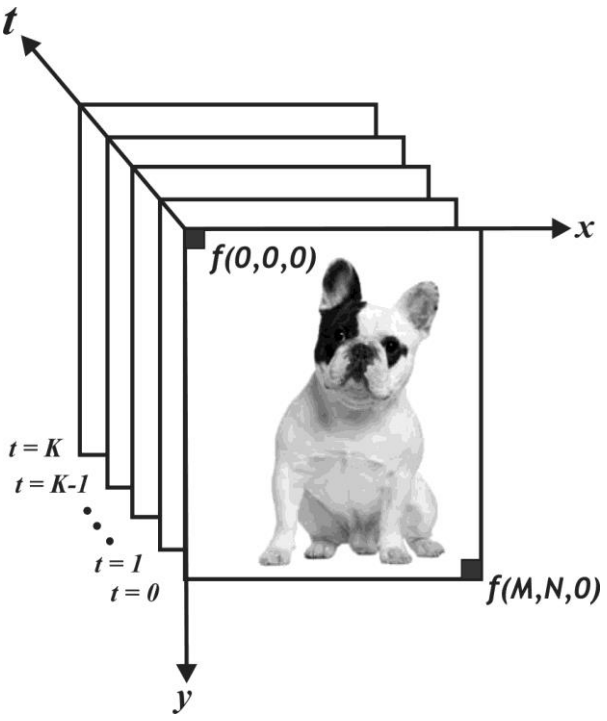

Figura 2.1 Secuencia de video digital.

El video digital puede ser obtenido a partir del muestreo de una señal analógica de video $V(t)$ o bien a partir del muestreo y cuantificación espacio temporal de la distribución de intensidad de luz incidente en un sensor de imagen [MUR95]. En cualquiera de los casos el resultado que se obtiene es una secuencia de arreglos bidimensionales de información acerca de la distribución de intensidad en un instante de tiempo, esta secuencia es representada mediante la función discreta $f(x, y, t)$, donde el valor de f representa el nivel de gris del pixel en las coordenadas (x, y) de la imagen y t indica la imagen en la secuencia a la cual pertenece el pixel. La notación de coordenadas utilizada en la mayoría de libros es la mostrada en la Figura 2.1 en donde el video tiene una resolución de M columnas por N renglones de pixeles y está compuesto por una secuencia de K imágenes [MUR95][BOV09][WOO12].

2.1.1 Video progresivo y entrelazado

Una señal de video puede ser muestreada como una secuencia de cuadros completos (video progresivo) o como una secuencia de campos entrelazados (video entrelazado), como se observa en la Figura 2.2. La técnica de "entrelazado" es utilizada para reducir la cantidad de información de cada cuadro. Esta consiste en el muestreo de la mitad de los datos de un cuadro en cada intervalo de muestreo temporal; el campo superior se forma con las líneas impares y el inferior con las líneas pares de un cuadro [BOV09].

Una secuencia de video entrelazado consiste de una serie de campos superior e inferior entrelazados, cada uno representa la mitad de la información del cuadro de video. Otra ventaja del muestreo entrelazado es que es posible transmitir y exhibir el doble de campos por segundo, con respecto a la transferencia de cuadros completos, para la misma tasa de datos, dando la apariencia de un movimiento más suave.

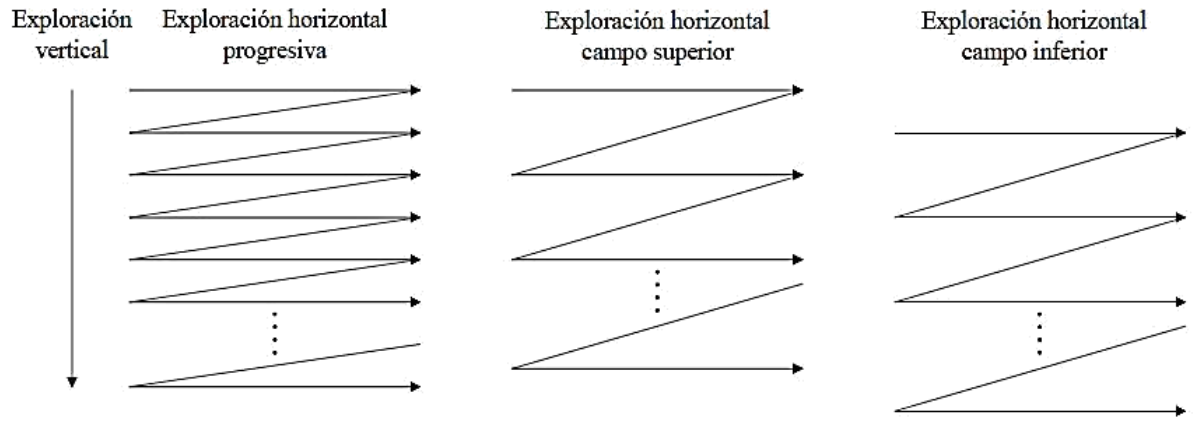

Figura 2.2 Video progresivo y entrelazado.

2.1.2 Profundidad de color

La profundidad de color, o también conocida como profundidad de bits, es el número de bits utilizados para la cuantificación de un pixel. Se trata de un concepto importante porque a mayor profundidad de bits más información contiene la imagen y por consiguiente se puede tener un mayor número de colores. Por ejemplo si la profundidad es de un solo bit, sólo se tendrán dos tonalidades en la imagen y por lo tanto será muy pobre la información.

El número de colores o niveles que podrá contener una imagen está denotado por la relación 2^L siendo L la profundidad de color. Comúnmente una imagen monocromática se representa con una profundidad $L = 8$ bits teniendo como resultado 256 niveles de gris, por otra parte en secuencias de imágenes a color RGB se usa una profundidad de $L = 24$ bits lo que genera 16,777,216 colores posibles para su representación.

2.1.3 Espacios de color

Los sistemas de video digital utilizan modelos matemáticos para representar la información de color, llamados espacios de color, siendo los más populares el RGB (*Red*, *Green*, *Blue*) y el YCbCr (*Luma*, *Croma blue*, *Croma red*) [JAC05]. Las imágenes monocromáticas requieren solo un número para indicar la brillantez o luminancia de cada pixel, mientras que las imágenes a color requieren un mínimo de tres números por cada posición del pixel para representar el color adecuadamente.

2.1.4 Espacio RGB

En el espacio RGB cada color aparece en sus componentes espectrales primarias: rojo, verde y azul. Este espacio está basado en el sistema de coordenadas cartesianas. El sub-espacio de color de interés es el tetraedro mostrado en la Figura 2.3, en el cual los valores RGB están en tres vértices, el negro corresponde al origen y el blanco se sitúa en el vértice más alejado del origen. En este modelo, la escala de grises se extiende desde el negro al blanco a lo largo de la diagonal que une esos dos puntos, y los colores son puntos dentro del cubo definidos por los vectores desde el origen.

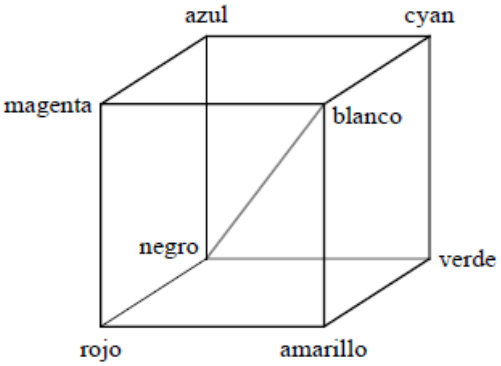

Figura 2.3 Cubo de color RGB.

Por conveniencia, se asume que todos los vectores han sido normalizados, de modo que el tetraedro de la figura es el tetraedro unitario, es decir, todos los valores de R, G y B están en el rango [0,1]. Las imágenes en este modelo se forman por la combinación en diferentes proporciones de cada uno de los colores primarios RGB [MOE12]. Las imágenes del modelo de color RGB consisten en tres planos de imagen independientes, uno por cada color primario.

2.1.5 Espacio YCbCr

El espacio de Color YCbCr se trata de una codificación no lineal del espacio RGB. La luminancia (*luma*), cuyo nivel es proporcional a la intensidad de luz presente en la escena original, se puede obtener a partir del modelo RGB considerando el peso que tiene cada una de sus componentes en la sensación de brillo de nuestro sistema visual de acuerdo con la ecuación [MOE12]:

$$Y = 0.299R + 0.587G + 0.114B \tag{2.1}$$

Las componentes Cb y Cr del modelo YCbCr se obtienen a partir de las señales de diferencia de color *B-Y* y *R-Y* como se observa en la ecuación (2.2)

$$\begin{aligned} Cb &= 0.564(B - Y) \\ Cr &= 0.713(R - Y) \end{aligned} \tag{2.2}$$

Este espacio de colores es generalmente usado en las técnicas de compresión debido a que modela los cambios de brillo de manera más significativa que los que se presentan en los colores. Esta representación de colores permite diferentes procesos de muestreo que están relacionados estrechamente con la calidad de las imágenes, en contraste al modelo RGB que requiere siempre el mismo tipo de muestreo 4:4:4 [JAC05].

El muestreo 4:4:4 es la forma original de representación del espacio de colores YCbCr compuesta por cuatro muestras de cada componente Y, Cr y Cb. Este formato no presenta perdida de información. Por otra parte el muestreo 4:2:2 consiste de cuatro componentes Y y dos componentes Cr y dos Cb. Esto representa la mitad de la información necesaria respecto al muestreo 4:4:4. Finalmente en el muestreo 4:2:0 las crominancias Cr y Cb se submuestrean de forma que su resolución es una cuarta parte que la de la luminancia Y.

2.1.6 Redundancia en el video

Analizando estadísticamente una secuencia de video se observa una correlación, tanto entre los pixeles próximos entre sí de una imagen (redundancia espacial), como entre los pixeles que ocupan la misma posición o posiciones cercanas en imágenes consecutivas (redundancia temporal) [SHA14]. Además de algunas limitaciones de visión que posee el ojo humano (redundancia de percepción).

2.1.6.1 Redundancia espacial

La redundancia espacial es la consecuencia de la correlación entre los valores de pixeles vecinos en las dimensiones espaciales x, y, dentro de la misma imagen o cuadro de video (también conocida como correlación intra). Los pixeles vecinos en un cuadro de video a menudo son muy similares entre sí, especialmente cuando la trama se divide en componentes luma y croma. Como la correlación suele ser alta dentro de un bloque, un cuadro puede ser dividido en bloques más pequeños de pixeles para aprovechar esas correlaciones entre pixeles, En otras palabras, dentro de una pequeña área de la imagen, la tasa de cambio en una dimensión espacial suele ser baja. Esto implica que, en una representación en el dominio de la frecuencia de la imagen de video, la mayor parte de la energía se concentra a menudo en la región de baja frecuencia [SHA14]. En la Figura 2.4 se muestra un ejemplo de redundancia espacial presente en una imagen de video.

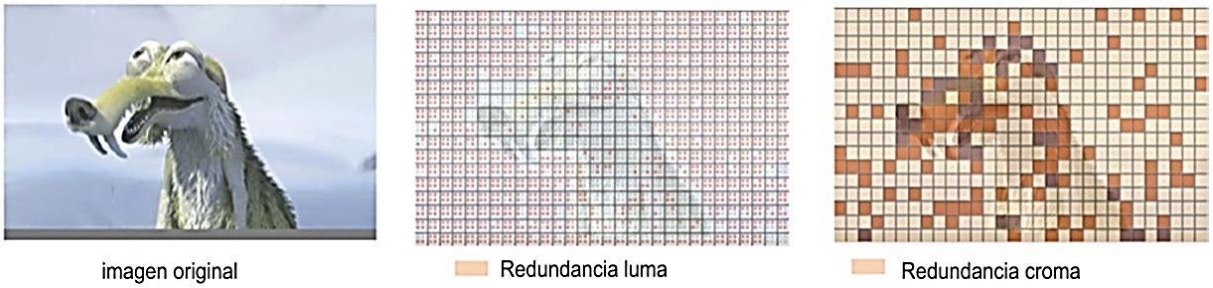

Figura 2.4 Redundancia espacial.

2.1.6.2 Redundancia temporal

La redundancia temporal es debido a la correlación entre las diferentes imágenes o cuadros de video (también conocido como correlación inter). Para que un observador humano pueda percibir un movimiento suave y continuo en un video, éste debe ser muestreado con una frecuencia de más de 15 cuadros por segundo, esto requiere que los cuadros vecinos sean muy similares entre sí, derivado de esto los videos digitales tienen una gran cantidad significativa de redundancia temporal [SHA14]. Por ejemplo en la Figura 2.5 se puede apreciar como son muy similares dos imágenes consecutivas en la secuencia de video.

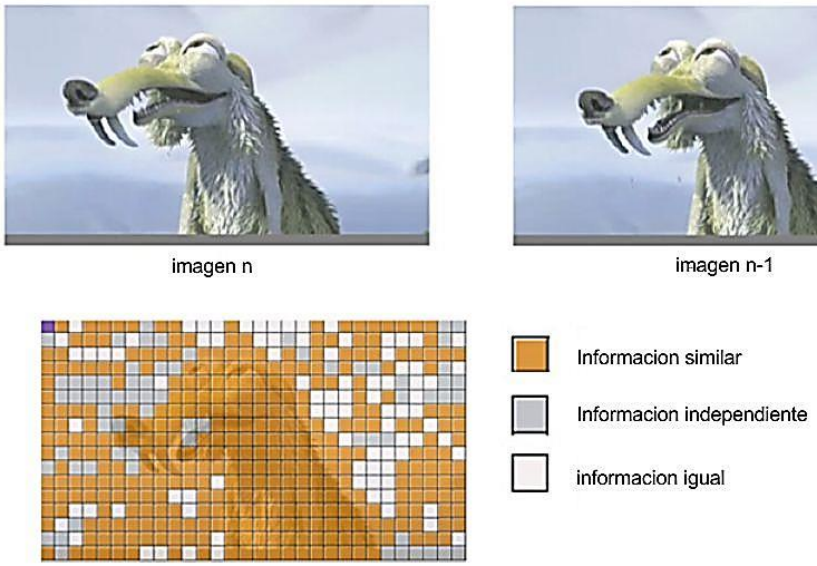

Figura 2.5 Redundancia temporal.

2.2 Introducción al procesamiento digital de video

El área del procesamiento digital de video ha experimentado un enorme crecimiento en la última década, en pocos años los sistemas de información visual emigraron de la tecnología analógica a la secuencia de imágenes y video digital, gracias al incremento de la capacidad de procesamiento de la tecnología digital, a las facilidades de la representación resultante de ser convenientemente procesada, almacenada y transmitida, y que esto se logra utilizando sistemas de cómputo de bajo costo. Las investigaciones de nuevas tecnologías, herramientas de diseño, algoritmos y arquitecturas de procesamiento, junto con el desarrollo de estándares, han dado como resultado aplicaciones cada vez más sofisticadas, que abarcan campos de acción no considerados anteriormente, lo que da soporte a mayor variedad de contenido e inter-conectividad, con altos niveles de calidad, así lo demuestra la gran cantidad de artículos relacionados con el tema que han aparecido en diversas revistas, actas de congresos y libros. En la actualidad el campo de acción del procesamiento de video se ha ampliado debido al avance tecnológico de productos de consumo multimedia basados en la electrónica, como cámaras digitales, cámaras de teléfonos celulares, sistemas de video para vigilancia inteligente, televisión de alta definición (HDTV), etc. [ZIL01][BRO05][WON05].

De acuerdo a Bovik, el procesamiento digital de video consiste en el estudio de algoritmos, métodos y técnicas, ejecutados en sistemas HW/SW, para el procesamiento de secuencias de imágenes en movimiento representadas en formato digital, con la finalidad de analizar, modificar y/o mejorar la información contenida en ella y aplicarla a un fin específico [BOV09]. Algunos ejemplos de aplicaciones en donde se hace uso del procesamiento digital de video son: la navegación autónoma, vigilancia y supervisión de tráfico vehicular, sistemas de visión nocturna, tomografía computarizada, compresión de video, mejora y restauración de video, etc.

Lo que hace diferente al procesamiento digital de video del procesamiento de una sola imagen fija, es que la secuencia de imágenes que constituyen al video contiene una cantidad significativa de correlación temporal entre sí, comúnmente conocida como redundancia temporal [MUR95]. Se pueden procesar imágenes de video como una secuencia de imágenes

fijas, en donde cada cuadro se procesa independientemente, sin embargo, diferentes procesos como el filtrado y predicción con compensación de movimiento resultan más eficientes si se aplican técnicas de procesamiento en donde se considera el uso de múltiples cuadros para aprovechar la redundancia temporal existente. Además, algunas tareas, tales como la estimación de movimiento, el análisis de una escena variable en el tiempo o la compresión de video digital, requieren considerar este tipo de redundancia.

Existe una gran variedad de algoritmos de procesamiento de video, cada uno con características propias que dependen del uso final de dicho procesamiento. Algoritmos que van desde el procesamiento de bajo nivel, donde las operaciones se realizan de manera uniforme en una imagen o secuencia, a los procedimientos de alto nivel, tales como el seguimiento de objetos e identificación [BOV00]. En la bibliografía especializada en procesamiento de video se puede encontrar una gran variedad de formas para categorizar o agrupar las técnicas pertenecientes a esta área. En este contexto es posible agrupar las técnicas en cuatro grupos:

- Mejora y restauración de video.
- Estimación y detección de movimiento.
- Segmentación de video.
- Compresión de video.

2.2.1 Mejora y restauración de video

La mejora y restauración de una secuencia de imágenes es un proceso cuyo objetivo principal es destacar detalles de interés, corregir la degradación sufrida y/o mejorar la calidad del mismo con el objetivo de que el aspecto resultante sea más adecuado para una aplicación específica [BOV09]. Este conjunto de técnicas siempre han sido de gran importancia no sólo para mejorar la calidad visual, sino también para aumentar el rendimiento de las tareas posteriores tales como el análisis y la interpretación.

Durante las dos últimas décadas, una enorme cantidad de investigaciones se han enfocado en el problema de mejorar y restaurar imágenes en dos dimensiones. Claramente, los métodos espaciales resultantes también son aplicables a las secuencias de imágenes, pero tal enfoque supone implícitamente que las imágenes individuales de la secuencia son temporalmente independientes [BOV00][MOE12]. Una diferencia importante entre la mejora y restauración de imágenes en dos dimensiones y la de secuencia de imágenes, es la gran cantidad de datos a procesar y el aprovechamiento de la correlación temporal existente entre las imágenes de la secuencia (redundancia temporal) [MUR95][BOV09].

Todos aquellos algoritmos de procesamiento destinados a resaltar, agudizar y/o contrastar determinados aspectos y también aquellos que ayudan a eliminar efectos no deseados, como toda clase de ruido y/o degradación en imágenes fijas e imágenes en movimiento, se pueden considerar como de mejora y restauración de video. Este conjunto de algoritmos lo podemos dividir en dos grandes grupos: los que realizan operaciones puntuales y los que realizan operaciones de filtrado [MUR95][BOV00][BOV09].

2.2.1.1 Operaciones puntuales

Una operación puntual hace una transformación T a nivel de pixel de una secuencia de imágenes de entrada, esto quiere decir que cada pixel de la secuencia de salida sólo depende del correspondiente pixel de entrada. Esta transformación la podemos representar matemáticamente como [MOE12]:

$$S(x, y, t) = T[f(x, y, t)] \tag{2.3}$$

Donde f es el nivel de gris del pixel de entrada y S es el nivel de gris del pixel de salida ya transformado. Dos ejemplos representativos de estas transformaciones son: el negativo y la mejora de contraste.

2.2.1.1.1 El negativo

El negativo es una transformación donde el valor del pixel de salida se obtiene a partir de restar cada uno de los valores de los pixeles de entrada a un valor máximo $(L - 1)$ de nivel de gris permitido en la imagen [BOV00].

$$S(x, y, t) = (L - 1) - f(x, y, t) \tag{2.4}$$

Este procesamiento resulta útil cuando se quieren resaltar detalles blancos o grises que se encuentran en regiones oscuras de una imagen, especialmente cuando las áreas negras son dominantes en tamaño. Además, este procesamiento es análogo al negativo fotográfico que suele ser usado en áreas como la medicina o la industria.

2.2.1.1.2 Mejora de contraste

Utilizando el valor de intensidad mínimo y máximo en una imagen, se puede aumentar su contraste. La idea básica es llevar el valor mínimo (min) a cero y el máximo (max) a 255, pensando en imágenes monocromáticas $(0 - 255)$. Esta transformación genera que las intensidades se espacien de acuerdo a cierto factor o pendiente; de acuerdo a Bovik el factor para este aumento lineal de contraste es [BOV00]:

$$S(x, y, t) = \left(\frac{f(x, y, t) - min}{max - min}\right) * 255 \tag{2.5}$$

2.2.1.2 Operaciones de Filtrado

La secuencia de imágenes $g(n, k)$ corrompida por ruido $w(n, k)$ viene dada como:

$$g(n, k) = f(n, k) + w(n, k) \tag{2.6}$$

Donde $n = (x, y)$ se refiere a las coordenadas espaciales y $k = t$ se refiere al número del cuadro en la secuencia de imágenes (se hizo el cambio para indicar que k es una variable discreta). El objetivo de la reducción de ruido es hacer una estimación $F(n, k)$ de la secuencia de imágenes original a partir de la secuencia con ruido $g(n, k)$ aplicando una transformación T [MUR95][BOV09]; así esta representación es definida por

$$F(n, k) = T[g(n, k)] \tag{2.7}$$

Los algoritmos para el filtrado de ruido en secuencias de imágenes se pueden clasificar como: filtros espaciales (intra-cuadros), filtros espacio temporales (inter-cuadros) y filtros en el dominio de la frecuencia.

2.2.1.2.1 Filtros espaciales

Las técnicas o filtros en el dominio espacial operan directamente sobre los pixeles de la imagen k que se esté procesando. Éstos operan en la vecindad de los pixeles, generalmente mediante una máscara cuadrada o rectangular de tamaño $m \times n$. Una máscara es una pequeña matriz bidimensional que consiste de una serie de valores w predeterminados para cada posición (ver Figura 2.6). La máscara se centra sobre el pixel de interés de forma que el nuevo valor del pixel depende de los pixeles que cubre la máscara [BOV00][WOO12][MOE12].

$w_{(-1,-1)}$	$w_{(0,-1)}$	$w_{(1,-1)}$
$w_{(-1,0)}$	$w_{(0,0)}$	$w_{(1,0)}$
$w_{(-1,1)}$	$w_{(0,1)}$	$w_{(1,1)}$

Figura 2.6 Ejemplo de una máscara de 3x3.

Una interpretación del filtrado espacial es convolucionar la imagen original con la respuesta impulsional del filtro almacenada en la máscara de forma que el nuevo valor del pixel es la sumatoria del producto de los pixeles vecinos con el coeficiente correspondiente de la máscara como se muestra en la siguiente ecuación [MAR11]:

$$F(x,y,k) = \sum_{j=-\frac{n-1}{2}}^{\frac{n-1}{2}} \sum_{i=-\frac{m-1}{2}}^{\frac{m-1}{2}} g(x+i, y+j, k) w(i,j) \tag{2.8}$$

El valor de los coeficientes y el tamaño de la máscara determinarán el tipo de filtro y la calidad de los resultados obtenidos al aplicarlo. Por ejemplo, se tienen los filtros de suavizado (o pasa-bajo) cuyo objetivo es eliminar ruido o detalles pequeños que no son de interés (ejemplos de éstos son: el promedio y el gaussiano) y los filtros de realce (o pasa alto) cuyo su objetivo es intensificar los detalles y cambios bruscos de intensidad mientras atenúa las bajas frecuencias.

2.2.1.2.2 Filtrado en el dominio de la frecuencia

En el caso del filtrado en el dominio de la frecuencia se hace uso de la transformada discreta de Fourier, la cual establece que cualquier función puede ser expresada como la suma de senos y/o cosenos de diferentes frecuencias, cada uno multiplicado por coeficientes diferentes (Series de Fourier) [FOU22].

Una función no periódica finita, también puede ser llevada al dominio de Fourier, expresándola como una integral de senos y/o cosenos. A esto se le conoce como la transformada de Fourier en tiempo discreto de dicha función, que al igual que las series de Fourier en tiempo discreto, recuperan a la función sin pérdida de información al aplicar el proceso inverso (Transformada de Fourier Inversa en tiempo discreto).

En el caso de una función de dos dimensiones, $f(x,y)$ como es el caso de una imagen, la transformada discreta de Fourier se expresa como:

$$F(u,v) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y)e^{-j2\pi(ux+vy)}dxdy \tag{2.9}$$

Y la transformada inversa se expresa de manera similar:

$$f(x,y) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} F(u,v)e^{j2\pi(ux+vy)}dudv \tag{2.10}$$

Sin embargo, debido a que las imágenes a tratar son de tipo discreto, es de interés trabajar con la Transformada Discreta de Fourier (DFT, *Discrete Fourier Transform*). La DFT de una imagen de tamaño $M \times N$ está dada por la siguiente ecuación:

$$F(u,v) = \frac{1}{MN}\sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f(x,y)e^{-j2\pi\left(\frac{ux}{M}+\frac{vy}{N}\right)} \tag{2.11}$$

Donde $u = 0,1,2 \ldots, M-1$ y $v = 0,1,2,\ldots,N-1$ son las variables en frecuencia. Y la transformada discreta inversa de Fourier (IDFT, *Inverse Discrete Fourier Transform*) está definida como:

$$f(x,y) = \sum_{u=0}^{M-1}\sum_{v=0}^{N-1} F(u,v)e^{j2\pi\left(\frac{ux}{M}+\frac{vy}{N}\right)} \tag{2.12}$$

El esquema general para el filtrado en el dominio de la frecuencia se muestra en la Figura 2.7 [CAS96][MAR11], donde los pasos a seguir son:

- Primero se transforma la imagen al dominio de la frecuencia mediante la Transformada Discreta de Fourier.
- Después se multiplica por un filtro con una respuesta en frecuencia $H(u,v)$.
- Finalmente se calcula la IDTF, volviendo, así, al dominio del espacio.

Existen muchas clases de filtros que se pueden aplicar en el dominio de la frecuencia. Dos de los filtros más comunes son el llamado filtro ideal y el filtro Butterworth. Ambos tipos de filtros pueden ser pasa-altos y pasa-bajos.

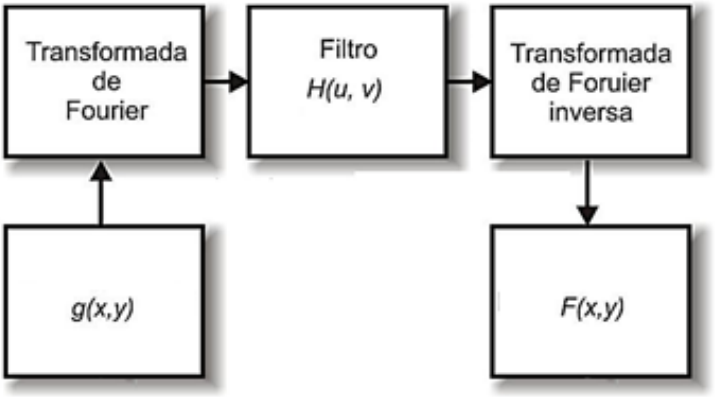

Figura 2.7 Proceso de filtrado en el dominio de la frecuencia.

El filtro ideal pasa-bajos tiene una función de transferencia $H(u,v) = 1$ para todas las frecuencias menores a cierto valor (D_0) y cero para las demás frecuencias. Un filtro ideal pasa-altos tiene la función de transferencia opuesta, es decir es cero para todas las frecuencias menores a cierto valor y uno para las demás frecuencias. En la Figura 2.8 se muestra la función de transferencia de un filtro ideal paso-bajos.

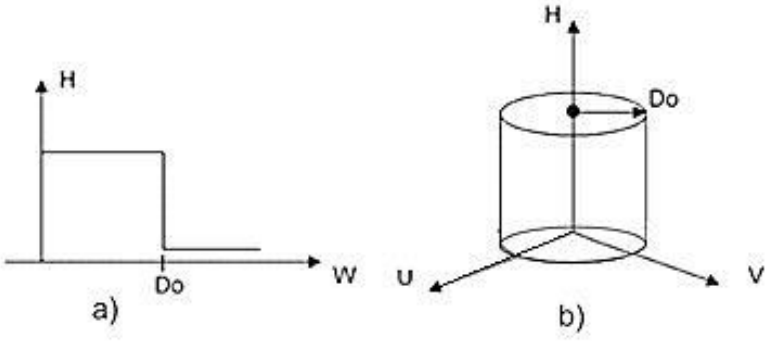

Figura 2.8 Función de transferencia ideal pasa-bajas con una frecuencia de corte D_0, a) 1D, b) 2D.

El filtro Butterworth, por su parte, tiene una función de transferencia más suave es decir; no tiene una discontinuidad evidente que establezca una frecuencia de corte bien determinada. La función de transferencia de un filtro Butterworth pasa-bajo de orden n y frecuencia de corte D_0 se define como:

$$H(u,v) = \frac{1}{1 + \left(\frac{\sqrt{u^2 + v^2}}{D_0}\right)^{2n}} \tag{2.13}$$

Esta función de transferencia se ilustra gráficamente en la Figura 2.9.

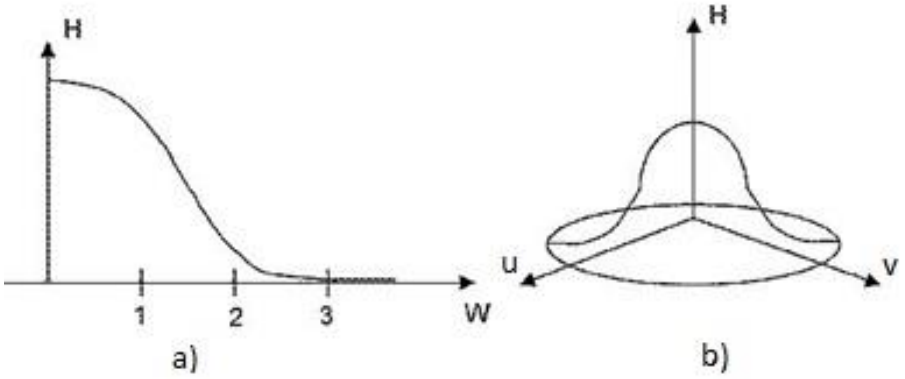

Figura 2.9 Función de transferencia de un filtro Butterworth paso- bajas, a) 1D, b) 2D.

2.2.1.2.3 Filtros espacio temporales

Los filtros espacio temporales son filtros en tres dimensiones, dos en las coordenadas espaciales y una en la coordenada temporal. Por tanto se utiliza la correlación espacial y la correlación temporal entre las imágenes de la secuencia [MUR95][BOV09][WOO12]. En este sentido, es posible agrupar estas técnicas de reducción de ruido en técnicas de filtrado temporal donde solo se explota la redundancia temporal y técnicas de filtrado espacio temporal donde se hace uso de las dos redundancias, estos a su vez pueden subdividirse en filtros con movimiento adaptativo y filtros con compensación de movimiento [MUR95]. Los filtros con movimiento adaptativo utilizan un esquema de detección de movimiento pero no requieren la estimación explicita del vector de movimiento. Por otra parte los filtros con compensación de movimiento operan solo en las trayectorias de movimiento, tal que se requiere el conocimiento exacto de la trayectoria de cada pixel [DUB84].

2.2.1.2.3.1 Filtrado temporal

Este tipo de filtrado lleva a cabo un promedio ponderado de los cuadros sucesivos en la dirección temporal. Es decir, la secuencia de imagen restaurada se obtiene por la ecuación [SEZ93][MUR95][KOK98][BOV09]:

$$F(n,k) = \sum_{l=-K}^{K} h(l)g(n,k-l) \tag{2.14}$$

Donde $h(l)$ son los coeficientes del filtro temporal y tiene una dimensión temporal de $2K+1$ cuadros consecutivos. En caso que la información de los cuadros vecinos se considere de igual importancia (como en el caso del filtro promedio), el valor de cada coeficiente está dado por:

$$h(l) = \frac{1}{2K+1} \tag{2.15}$$

Cuanto mayor sea el tamaño de la ventana, mayor es la reducción del ruido. Sin embargo, los efectos que resultan por el movimiento de objetos en la escena es el desenfoque de los mismos debido al promedio de la información del objeto y el fondo. Los efectos de movimiento se pueden reducir en gran medida por el funcionamiento del filtro (2.14) a lo largo de los elementos de imagen que se encuentran en la misma trayectoria de movimiento

[DUB84][BOV09]. La ecuación (2.14) se convierte entonces en un filtro temporal con compensación de movimiento (véase la Figura 2.10) denotado por:

$$F(n,k) = \sum_{l=-K}^{K} h(l)g(n1 - dx(n1,n2;k,l), n2 - dy(n1,n2;k,l), k-l) \quad (2.16)$$

Donde $d(n;k,l) = (\, dx(n1,n2;k,l)\,,\ dy(n1,n2;k,l))$ es el vector de movimiento estimado entre las imágenes k y l.

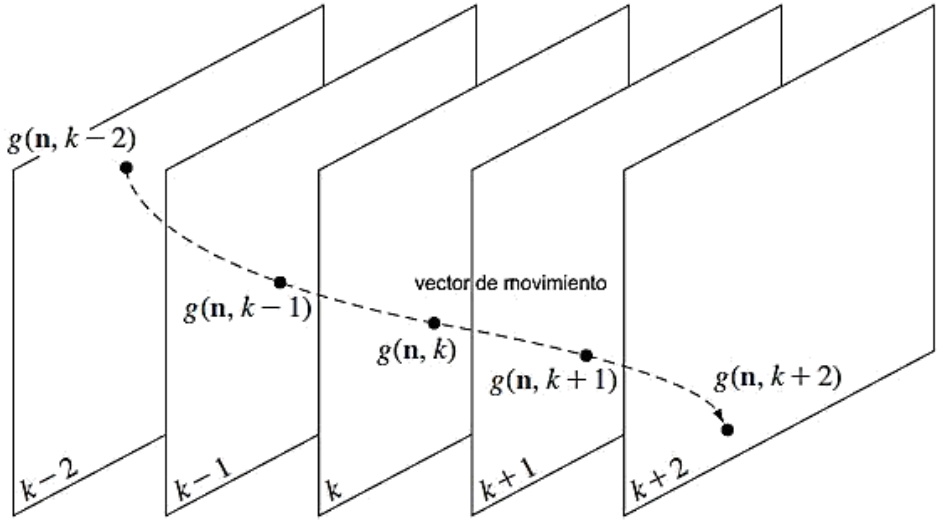

Figura 2.10 Filtro temporal con compensación de movimiento sobre el pixel *g(n, k)*.

2.2.1.2.3.2 Filtrado espacio temporal

Es fácil ver que el filtrado temporal se puede ampliar con una parte de filtrado espacial. Existen muchas variaciones de este concepto, básicamente tantas como son las técnicas de filtrado espacial para la reducción de ruido. La representación general del filtro en tres dimensiones partiendo de la ecuación del filtro temporal se expresa como [MUR95][BOV09]:

$$F(x,y,k) = \sum_{l=-K}^{l=K} \sum_{j=y-\frac{n-1}{2}}^{y+\frac{n-1}{2}} \sum_{i=x-\frac{m-1}{2}}^{x+\frac{m-1}{2}} g(i,j,k-l)h(i,j,k-l) \quad (2.17)$$

Donde los índices i, j y l son el soporte espacio temporal de la ventana tridimensional de coeficientes h que constituyen el filtro, los cuales son aplicados a un segmento tridimensional de la secuencia para obtener el valor del pixel $F(x,y,k)$ a partir de su vecindad (ver Figura 2.11).

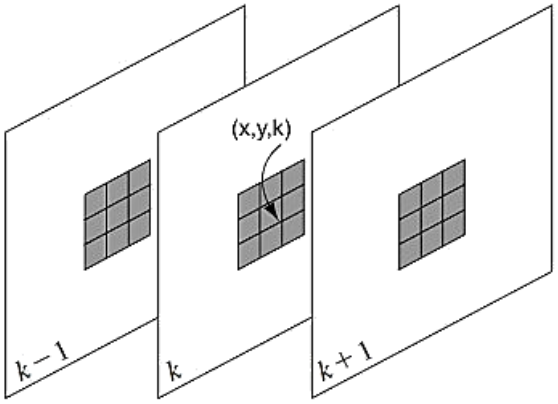

Figura 2.11 Ejemplo de la aplicación d una ventana espacio temporal.

2.2.2 Estimación y detección de movimiento

La información del movimiento tridimensional (3D) de los objetos y el movimiento de la cámara tienen una gran importancia en el procesamiento de imágenes y video. Desde el punto de vista del procesamiento de señales, las trayectorias de movimiento representan la dirección espacio temporal de los cambios de intensidad en la secuencia de imágenes. Desde el punto de vista de las mediciones, el movimiento bidimensional (2D) en la secuencia de imágenes y video es una observación que incorpora información sobre la geometría y movimiento 3D en el mundo real [REE04].

Con frecuencia el análisis de escenas dinámicas de una secuencia de imágenes se denomina análisis del movimiento o análisis dinámico de imágenes. Comúnmente los sistemas de análisis del movimiento se basan en algún algoritmo y emplean dos o más imágenes. Este hecho puede ser considerado como un proceso de análisis entre imágenes estáticas. La mayoría de los algoritmos para la estimación del movimiento 2D pueden clasificarse en tres grandes bloques: los métodos basados en la obtención del flujo óptico, los métodos basados en la correspondencia y los métodos diferenciales.

2.2.2.1 Flujo óptico

Murat define al flujo óptico como el campo vectorial bidimensional de las velocidades aparentes en una imagen, el cual se obtiene a partir de una secuencia de imágenes, mediante el análisis de los cambios que sufre la luminancia en los puntos de las mismas (restricción de brillo) [MUR95]. La aproximación más directa, cuando se analiza una secuencia de imágenes, consiste en tratar de evaluar la velocidad de los objetos a través del desplazamiento que se produce en los pixeles de la imagen.

El movimiento no se produce de forma caótica y puntual, sino con cierta homogeneidad, por lo que se puede asumir una variación suave en el campo de velocidades (flujo óptico). El modelo de Horn y Schunk [HOR81] plantea asumir el movimiento como un simple desplazamiento, de modo que cada pixel de una imagen encontrará su equivalente en la imagen siguiente, esta expresión se puede denotar como:

$$f(x, y, t) = f\,(x + \delta x\,, \mathrm{y} + \delta \mathrm{y}\,, t + \delta t) \qquad (2.18)$$

Desarrollando la expresión en series de Taylor y despreciando los términos de mayor orden se obtiene:

$$\frac{\partial f(x,y,t)}{\partial x}u + \frac{\partial f(x,y,t)}{\partial y}v + \frac{\partial f(x,y,t)}{\partial t} = 0 \tag{2.19}$$

$$\binom{u}{v} = \begin{pmatrix} \frac{\partial x}{\partial t} \\ \frac{\partial y}{\partial t} \end{pmatrix} \tag{2.20}$$

Los valores de u y v son las componentes del flujo óptico. Debido a que la restricción de brillo es una única ecuación con dos incógnitas (u, v), son necesarias otras restricciones para poder calcular de forma unívoca el valor del flujo óptico en cada pixel de las imágenes de entrada. Para solventar este problema, se propone el uso de una segunda restricción, conocida como restricción de uniformidad, que, junto con la restricción de brillo definida anteriormente permite el cálculo del flujo óptico. Dicho cálculo se lleva a cabo mediante la minimización de la función de error mostrada en la ecuación (2.21), donde α es una constante, $||\ ||$ representa la norma euclidiana y ∇ es el operador gradiente.

$$\varepsilon^2 = \iint \left(\frac{\partial f}{\partial x}u + \frac{\partial f}{\partial y}v + \frac{\partial f}{\partial t}\right)^2 dxdy + \alpha^2 \iint \left(\left|\left|\nabla u\right|\right|^2 + \left|\left|\nabla u\right|\right|^2\right) dxdy \tag{2.21}$$

Por otro lado, dado que la restricción de brillo definida en la ecuación (2.19) ha sido obtenida considerando que las variaciones *dx, dy* y *dt* son muy pequeñas, esta restricción sólo es válida únicamente si el desplazamiento entre imágenes consecutivas de la secuencia es pequeño.

2.2.2.2 Análisis del movimiento basado en la correspondencia

La detección del movimiento basado en la correspondencia o en la correlación funciona muy bien aún para intervalos de muestreo relativamente altos o para grandes desplazamientos entre dos imágenes, algo que no sucede con el método de flujo óptico. La razón es que al emplear el método del flujo óptico existe la restricción de que solo se puede utilizar si la diferencia entre las dos imágenes consecutivas es pequeña.

Estos métodos se basan en la búsqueda de los vectores de desplazamiento entre la imagen de referencia y la imagen actual, bajo un criterio de similitud de un objeto móvil [TAB98]. Un criterio comúnmente utilizado para encontrar el vector desplazamiento, consiste en minimizar la función objetivo $C(x, y)$ que calcula el valor absoluto de la diferencia entre una ventana de la imagen de referencia f_{t-1} con una ventana de la imagen actual f_t, y está representada por la ecuación:

$$C(x,y) = \sum_{x,y \in R} |f(x,y,t) - f(x,y,t-1)|^p \tag{2.22}$$

Donde R es una región de la imagen f_t que es comparada con una imagen anterior en la secuencia f_{t-1} y p es la potencia del valor absoluto, cuando $p = 1$ se considera el criterio del error absoluto medio (MAE) y cuando $p = 2$ el del error cuadrático medio (MSE). Idealmente la región de búsqueda seria toda la imagen pero esto no es práctico, tanto por el tiempo necesario para realizar la correlación en toda la imagen como por la frecuencia de muestreo

utilizada. Por tal razón la región se restringe a un área de búsqueda alrededor de la localización del pixel, característica o región de encaje sobre la imagen actual.

2.2.2.3 Métodos diferenciales

Esta técnica es de las primeras en aparecer por su bajo costo computacional [YAL81]. El principio básico de esta técnica es que considera que cualquier movimiento perceptible en la escena se traduce en cambios en la secuencia de imágenes tomadas de dicha escena, por lo cual, si tales cambios son detectados, se pueden analizar las características de este movimiento. Para identificar un conjunto de pixeles que han cambiado con respecto al tiempo (imagen diferencia FD), se hace uso de la diferencia entre la imagen actual y una imagen desplazada temporalmente en la secuencia.

$$FD(x, y, t) = f(x, y, t-1) - f(x, y, t) \tag{2.23}$$

Este mecanismo para obtener la imagen diferencia permite detectar la zona de la imagen donde se están produciendo cambios, de esta manera se concentra el esfuerzo computacional en el área detectada. Por otra parte, mediante el uso de una imagen de diferencias acumuladas, es posible recopilar la historia de movimientos a lo largo de un determinado intervalo de tiempo, pudiendo describir el movimiento [JAI79].

2.2.3 Segmentación de video

La segmentación se refiere a la identificación de regiones que son homogéneas en algún sentido en una secuencia de video. Diferentes características y los criterios de homogeneidad, generalmente, conducen a diferentes segmentaciones de los mismos datos, por ejemplo, la segmentación de color, la segmentación de textura, y la segmentación de movimiento suelen dar lugar a diferentes mapas de segmentación. Además, no existe garantía de que cualquiera de las segmentaciones resultantes sea semánticamente significativa, ya que una región semánticamente significativa puede tener múltiples colores, múltiples texturas, o múltiples movimientos [BOV09].

De acuerdo con Shalkoff, para la segmentación de secuencias de imágenes básicamente se puede optar por dos direcciones: la estática y la dinámica [SHA89]. En la primera sólo se considera cada imagen de la secuencia como independiente, de modo que sólo se pueden aplicar consideraciones espaciales, en particular la evaluación de gradientes direccionales. En la segunda se emplea en aquellos entornos donde el movimiento es importante, y por tanto se puede acudir a información de las imágenes anteriores y posteriores en la secuencia para segmentar la actual.

La mayoría de los algoritmos de segmentación aplicados a imágenes estáticas suelen estar basados en las propiedades de discontinuidad y de similitud [MAR11]. Aquellos que se basan en la discontinuidad dividen la imagen a partir de los cambios abruptos en la intensidad, en esta categoría se encuentran los detectores de bordes. Por otra parte, aquellos algoritmos cuyo principio de operación se basan en la propiedad de similitud, logran su objetivo partiendo la imagen en regiones donde los elementos cumplen con criterios previamente establecidos, un ejemplo se encuentra en los algoritmos basados en crecimiento de regiones. Entre los métodos dinámicos destacan los basados en el modelo de fondo y en el flujo óptico.

2.2.3.1 Métodos de Detección de Bordes

Los ejes o bordes, se encuentran en zonas de una imagen donde el nivel de intensidad fluctúa bruscamente, cuanto más rápido se produce el cambio de intensidad, el eje o borde es más fuerte [BOV00]. Un buen proceso de detección de bordes facilita la elaboración de las fronteras de objetos con lo que, el proceso de reconocimiento de objetos se simplifica.

Para poder detectar los bordes de los objetos, se deben detectar aquellos puntos borde que los forman. Así, un punto de borde puede ser visto como un punto en una imagen donde se produce una discontinuidad en el gradiente. El gradiente es un vector, donde sus componentes miden la rapidez en que los valores de los pixeles cambian en la distancia y en las direcciones espaciales. El gradiente de una función bidimensional $f(x, y)$ es definido como el siguiente vector:

$$\nabla f(x,y) = \begin{pmatrix} Gx \\ Gy \end{pmatrix} = \begin{pmatrix} \frac{\partial f(x,y)}{\partial x} \\ \frac{\partial f(x,y)}{\partial y} \end{pmatrix} \tag{2.24}$$

$$M = \sqrt{Gx^2 + Gy^2} \approx |Gx| + |Gy| \tag{2.25}$$

M representa la magnitud del gradiente y se puede aproximar con la suma de los valores absolutos del gradiente en la dirección x y el gradiente en la dirección y como se muestra en la ecuación (2.25).

Para la implementación y el cálculo del gradiente se utilizan máscaras o filtros que representan o equivalen a dichas ecuaciones en forma digital. Muchas técnicas basadas en la utilización de máscaras para la detección de bordes utilizan máscaras de tamaño 3×3 o incluso más grandes. La ventaja de utilizar máscaras grandes es que los errores producidos por efectos del ruido son reducidos mediante medias locales tomadas en los puntos en donde se superpone la máscara. Por otro lado, las máscaras normalmente tienen tamaños impares, de forma que los operadores se encuentran centrados sobre los puntos en donde se calculan los gradientes. Uno de los operadores gradiente más común es el operador Sobel [MAR11]. El cual está formado por dos mascaras; una para la detección de los bordes horizontales y la otra para los bordes verticales como se muestra en la Figura 2.12.

$Gx =$

-1	-2	-1
0	0	0
1	2	1

$Gy =$

-1	0	1
-2	0	2
-1	0	1

Figura 2.12 Operador Sobel.

Para definir si un pixel es parte o no del contorno, se hace pasar la magnitud del gradiente en ese pixel por un umbral (proceso de binarizacion), obteniendo así una imagen con la información de los bordes. En la Figura 2.13 se muestran los resultados al aplicar el operador Sobel a una imagen monocromática.

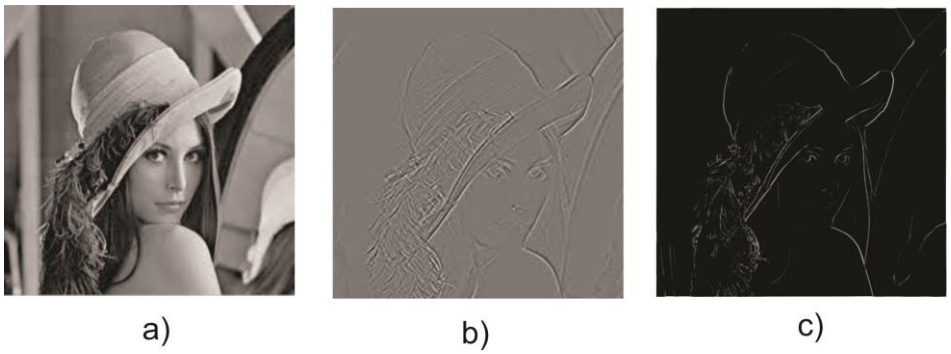

Figura 2.13 a) Imagen original, b) Magnitud del gradiente aplicando el operador Sobel, c) Información de los bordes.

2.2.3.2 Crecimiento de Regiones

Es un método de segmentación en donde las regiones crecen mediante agregación de pixeles similares en valor respecto a la propiedad P que se utilice para realizar la segmentación [PAV88]. Este tipo de algoritmos necesita que el usuario seleccione un conjunto de puntos semilla en la imagen. Estos puntos semilla servirán como puntos de comienzo del proceso de crecimiento de las regiones, con lo cual, el número final de regiones ha de ser como máximo igual al número de semillas sembradas por el usuario (puede ser menor, pues en algún paso del algoritmo se puede decidir unir dos regiones para formar una sola).

Para poder realizar la agregación de pixeles similares será necesario definir el concepto de similitud, el cual no tiene que ser el mismo para todo tipo de aplicaciones. Posibles criterios ya utilizados en algoritmos desarrollados pueden ser la diferencia entre el valor del pixel a agregar y el valor de la semilla o que el valor medio de la región ya formada sea menor que un cierto umbral predeterminado.

2.2.3.3 Segmentación basada en el modelo de fondo

La segmentación de objetos en movimiento basada en un modelo de fondo, o discriminación entre frente (*foreground*) y fondo (*background*), es una etapa clave en la mayoría de los sistemas de análisis de video. Esta técnica clásica de segmentación a nivel de pixel busca modelar el fondo mediante una capa o imagen del fondo que tiene que ser actualizada con cada nuevo cuadro del video. El frente se detecta como alteraciones sobre este modelo, esta técnica comúnmente es conocida como sustracción del fondo. Los resultados obtenidos por ésta técnica son aceptables en entornos donde el fondo es simple y estático, la calidad del video es buena y la complejidad de los objetos baja.

El modelado de fondo, también denominado actualización de fondo, es la etapa fundamental en los algoritmos de segmentación de objetos en movimiento. Su función es la inicialización, actualización y representación de un modelo de fondo robusto de la secuencia de video analizada. El fondo se describe mediante un modelo matemático, para cada pixel de la imagen en cada instante de tiempo.

Asumiendo que la cámara está fija, existen diversos métodos para llevar a cabo la sustracción del fondo; el más sencillo consiste en hallar la diferencia entre la imagen actual y el modelo de fondo de referencia (donde el modelo de fondo es estático), la diferencia $FD(x, y, t)$ entre el cuadro f y el de fondo R se define como [MOE12]:

$$FD(x,y,t) = R(x,y,t) - f(x,y,t) \tag{2.26}$$

La información del objeto en movimiento se obtiene a partir de la diferencia pixel a pixel entre las dos imágenes. Para distinguir las diferencias debidas a un cambio en la escena, se puede lograr una segmentación umbralizando la imagen de la diferencia como:

$$Z(x,y,t) = \begin{cases} 1 & FD(x,y,t) > T \\ 0 & FD(x,y,t) \leq T \end{cases} \tag{2.27}$$

$Z(x,y,t)$ se conoce como el campo de etiquetas de segmentación y es igual a 1 para regiones que cambian entre el cuadro entrante y el modelo del fondo, y es igual a 0 en otro caso (ver Figura 2.14). Ahora bien, para discernir entre lo que es ruido en la imagen y lo que realmente se está moviendo, se requieren métodos que identifiquen qué pixeles de la imagen son objeto en movimiento y cuáles pertenecen al fondo. Para ello, existen modelos de fondo en los que a cada pixel se asocia un rango de valores probables de intensidad, por ejemplo a través de una distribución Gaussiana. De este modo, se pueden desechar factores como cambios de iluminación progresivos en la escena, ruido inherente en el sistema de captación de video, etc. No obstante, pueden aparecer otro tipo de variaciones del fondo, como cambios bruscos de iluminación o movimientos de las ramas de los árboles, arbustos, etc. De hecho, existe un sinfín de elementos en movimiento en las escenas de video que no pueden modelarse a través de un único parámetro.

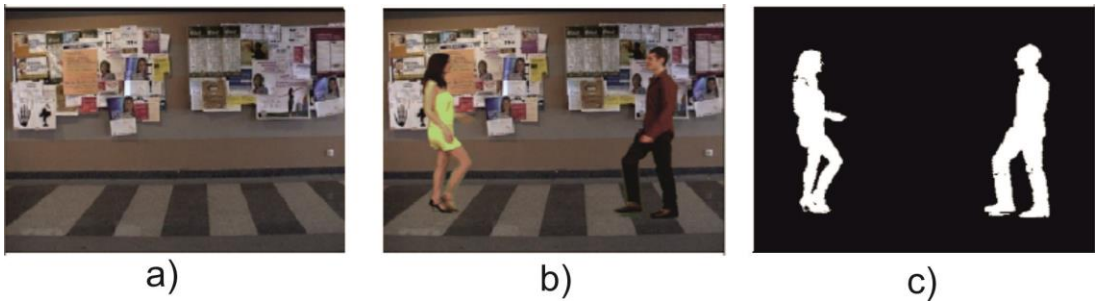

Figura 2.14 a) Modelo de fondo, b) Secuencia de video, c) Segmentación de objetos en movimiento.

2.2.3.4 Segmentación basada en flujo óptico

En este caso se trata de la segmentación de un campo de flujo dado, usando parámetros del modelo de flujo como características. Se asume que existen K objetos en movimiento independientes y cada vector de flujo corresponde a la proyección de un movimiento tridimensional rígido de un solo objeto opaco [MUR95]. Por lo tanto cada movimiento distinto puede ser acertadamente descrito por un conjunto de parámetros de mapeo.

Los métodos de segmentación basados en el flujo óptico se pueden resumir de la siguiente forma: Primero se supone que se tienen K conjuntos de vectores de parámetro, donde cada conjunto define una correspondencia o un vector de flujo en cada pixel. Los vectores de flujo definidos por los parámetros de mapeo (*mapping*) se llaman vectores basados en modelo o vectores de flujo sintetizados. Por lo tanto se tienen K vectores de flujo sintetizados en cada pixel. El procedimiento de segmentación asigna la etiqueta de cada vector sintetizado al que está más cercano al vector de flujo calculado en cada sitio. Sin embargo, existe un pequeño problema con este esquema simple, tanto el número de clases, K, como los parámetros de mapeo para cada clase no se conocen a priori. Considerando un valor particular para K, los parámetros de mapeo para cada clase se pueden calcular en el sentido de que los vectores de

flujo óptico asociados con las clases respectivas se conocen. Por lo tanto se requiere conocer los parámetros de mapeo. Esto sugiere un procedimiento iterativo similar al algoritmo de agrupamiento de *K*-medias, donde tanto las etiquetas de segmentación como las clases no se conocen.

2.2.4 Compresión de video

Las técnicas de codificación comúnmente denominadas algoritmos de compresión, tienen como objetivo, transformar un flujo de datos en un flujo de palabras código, si la transformación es efectiva las palabras código ocuparán menos bits que los datos originales, mientras que la calidad del video reconstruido o recuperado satisfaga los objetivos de la aplicación [SHI08]. La calidad requerida de la información reconstruida depende de la aplicación. Por ejemplo, en diagnósticos médicos y algunas mediciones científicas, las imágenes y videos reconstruidos necesitan ser idénticos al original, por lo que sólo se permiten algoritmos reversibles, sin pérdidas, que preserven la información, a diferencia de aplicaciones como la televisión, donde se permite una pérdida de cierta cantidad de información. A pesar de que existen muchas técnicas de compresión, según Navabi, sólo se pueden caer en dos categorías: compresión de datos sin pérdidas y con pérdidas [NAV03].

El proceso de compresión consiste en compactar la información en un número menor de bits, de acuerdo al tamaño original. La codificación de video digital es el proceso de compactar o condensar una secuencia de video dentro de un número menor de bits. El video no codificado generalmente requiere una alta tasa de transferencia que vuelve impráctico el almacenamiento o transmisión del video digital [WAN02]. Particularmente la compresión de video se basa en aprovechar las propiedades de las imágenes y las características perceptuales del sistema visual humano para reducir la mayor cantidad posible de la redundancia espacial, temporal y de percepción sin afectar el contenido de la secuencia de imágenes.

En los estándares de codificación dos organismos internacionales de estandarización la ITU (*International Telecommunication Union*) y la ISO (*International Organization for Standardization*) han elaborado sucesivamente estándares basados en el codificador híbrido que tienen como objetivo reducir el régimen binario de salida a partir de reducir la redundancia estadística (espacial y temporal) y de percepción, manteniendo la calidad de la imagen [SHA14]. Se trata de un esquema básico al que los diferentes estándares que han ido apareciendo han añadido variantes que permiten incrementar su capacidad de compresión.

Los sistemas de compresión de video basados en el codificador híbrido están compuestos por dos componentes básicos para comprimir (*encoder* o codificador) y descomprimir (*decoder* o decodificador) el video digital [WAN02][BOV09][SHI08]. El codificador provee al sistema los mecanismos necesarios para convertir la información original a un medio óptimo, reduciendo el número de bits empleados, para su almacenamiento y transmisión; mientras que, el decodificador tiene la tarea de transformar la información comprimida a una representación del video original, tal representación no constituye el video original, pues durante el proceso de compresión se presenta perdida de información.

2.2.4.1 Codificador

El esquema de codificación híbrido (o codificador híbrido, ver Figura 2.15) permite reducir la redundancia espacial, temporal y de percepción. En este esquema, los bloques de pixeles en los que se encuentra dividida cada imagen de una secuencia de video se transforman al dominio de

la frecuencia obteniéndose una serie de coeficientes que representan las componentes de las diferentes frecuencias que contienen los pixeles del bloque.

Los coeficientes obtenidos son cuantificados con un doble objetivo, por un lado reducir su rango de variación para que puedan ser codificados con un menor número de bits y, por otro, eliminar los coeficientes que contienen la información de alta frecuencia que el ojo humano no es capaz de apreciar. Finalmente, los coeficientes cuantificados son procesados por un codificador de entropía que aprovecha las propiedades estadísticas de éstos y asigna códigos de menor longitud a los elementos de la trama que son más probables, reduciendo de este modo el régimen binario de salida [SHA14]. Este proceso lleva asociadas pérdidas debidas a la cuantificación, ya que se elimina información de la señal transformada que posteriormente no puede ser recuperada.

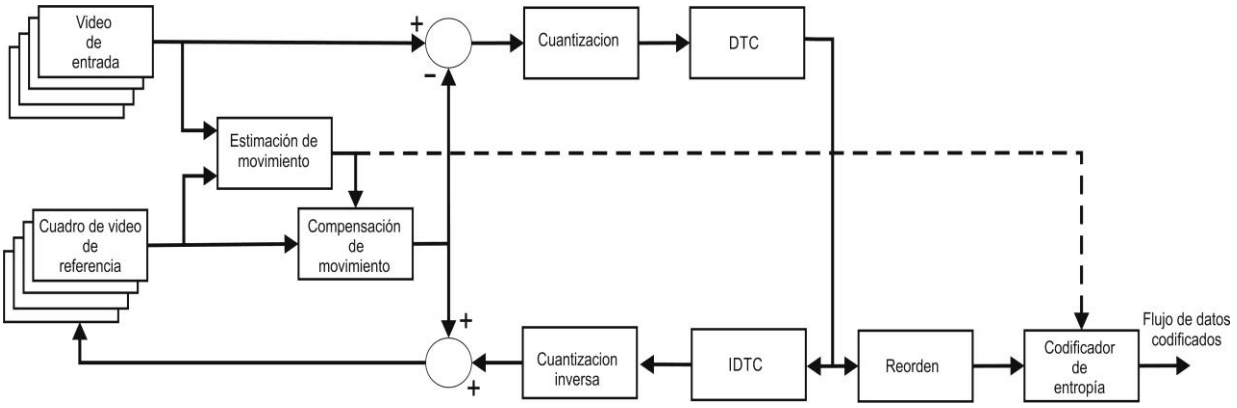

Figura 2.15 Codificador híbrido de video.

El compensador de movimiento permite reducir la redundancia temporal. Para ello se almacenan imágenes previamente codificadas para buscar en ellas macro bloques lo más parecidos posible al de la imagen actual que se desea codificar. A partir de la calidad de la predicción, el codificador decide para cada macro bloque si emplea la predicción obtenida o si codifica los pixeles directamente. Por lo tanto, el codificador posee en su entrada un selector que permite tomar esta decisión para cada macro bloque de la imagen de entrada; así, si se ha localizado una buena predicción, se codifica la diferencia entre dicho macro bloque y su predicción. Las muestras de error tienen valores pequeños por lo que al transformarlas al dominio de la frecuencia dan lugar a una matriz de coeficientes en la que muchos de ellos son cero y el resto suelen estar concentrados en la zona de las bajas frecuencias espaciales.

Los macro bloques que se codifican de forma independiente reciben el nombre de macro bloques intra (o macro bloques tipo I) mientras que los que emplean predicción de imágenes anteriores y/o posteriores son denominados inter. Los macro bloques de tipo inter se dividen a su vez en dos tipos: los que poseen una única referencia a una imagen anterior (o macro bloques tipo P) y los que poseen dos referencias, una a una imagen anterior y otra a una posterior (o macro bloques tipo B). En función del tipo de macro bloques que incluye una imagen es posible definir tres tipos de imagen; imágenes tipo I, imágenes tipo P e imágenes tipo B [SHA14].

Para que sea posible codificar un macro bloque a partir de otros que se encuentran en imágenes posteriores (codificación bidireccional), es necesario que el orden en el que se codifican las imágenes no sea el mismo que el orden natural de las mismas. Esto implica que antes de codificar cualquier imagen de tipo B es necesario haber codificado las imágenes I y/o P de las

que depende. La Figura 2.16 muestra una secuencia que ha sido codificada empleando imágenes de los tres tipos.

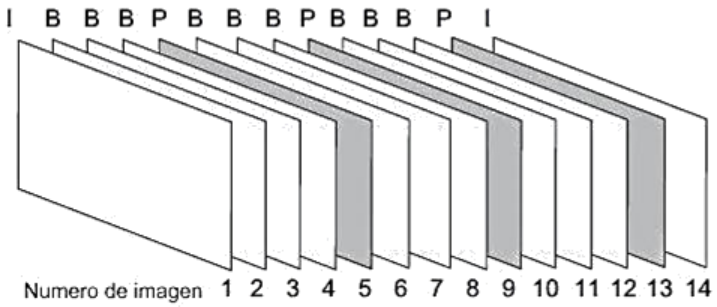

Figura 2.16 Ejemplo de grupo de imágenes codificadas.

2.2.4.1.1 Estructura de los macro bloques

El realizar un tratamiento a una secuencia de imágenes tomando como base el cambio entre pixeles presenta una carga computacional considerable que depende de la resolución de las imágenes. Es por esto que, una de las prácticas empleadas en las técnicas de compresión consiste en la segmentación de las imágenes en bloques de tamaño $m \times n$ pixeles [SHA14]. En la mayoría de los estándares, las imágenes son divididas en áreas cuadradas de tamaño 16×16 para la información de luminancia (Y) y de tamaño variable para la información de crominancia (Cb y Cr) dependiendo del esquema de muestreo que se haya empleado para la digitalización de las imágenes. Las áreas de 16×16 pixeles se denominan macro bloques (MB). A su vez, cada MB está dividido en áreas de 8×8 pixeles a las que se denomina bloques. El número de bloques de luminancia siempre es 4, mientras que el número de bloques de las crominancias puede ser 8, 4 ó 2 en función del esquema de muestreo que se utilice (4:4:4, 4:2:2 ó 4:2:0, respectivamente). La Figura 2.17 muestra la estructura de un macro bloque, según el esquema de muestreo utilizado.

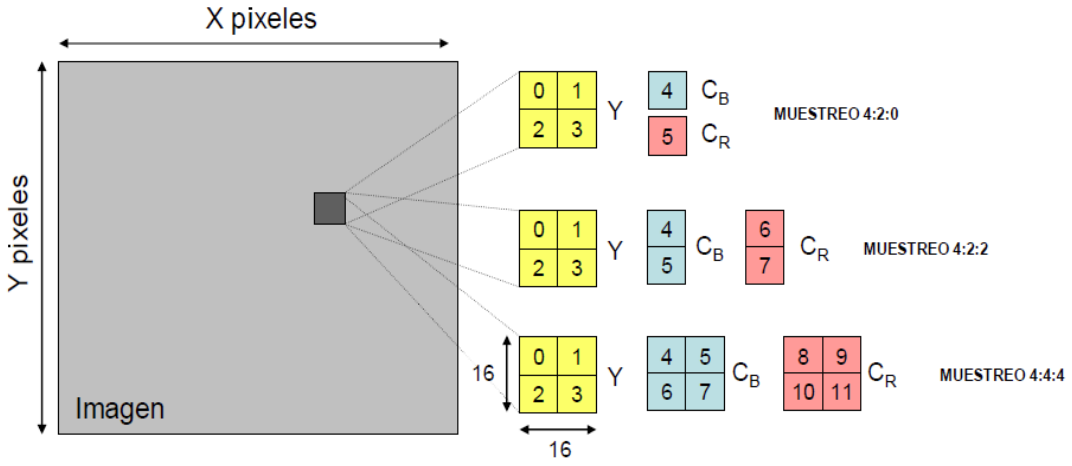

Figura 2.17 Estructura de un macro-bloque.

2.2.4.1.2 Estimación y compensación de movimiento

El objetivo de este proceso es realizar una predicción de los macro bloques de una imagen (macro bloque de referencia) a partir de imágenes anteriores y/o posteriores. Para cada macro bloque que se desea codificar se realiza una búsqueda en posiciones próximas a la posición que ocupa ese bloque en imágenes previamente codificadas, denominada área de búsqueda (ver Figura 2.18). Uno de los criterios que se suelen emplear consiste en la menor diferencia entre ambas regiones en términos de la energía residual (relación señal a ruido de pico, PNSR) [SHA14].

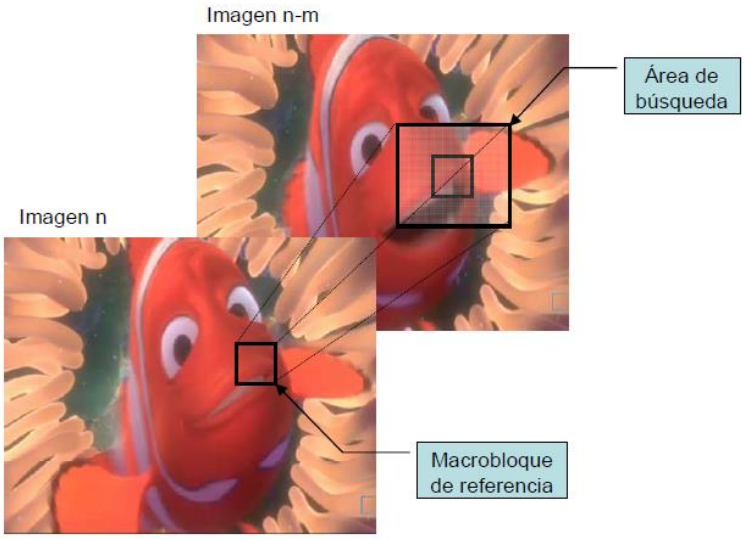

Figura 2.18 Estimación de movimiento basado en una imagen anterior.

La región seleccionada del área de búsqueda es sustraída de la imagen para ser codificada de manera conjunta con el cambio de posición de esta región. Este proceso genera vectores de movimiento los cuales describen el cambio de posición de las regiones entre cuadros consecutivos.

2.2.4.1.3 La Transformada Discreta del Coseno

Las transformaciones que se emplean en los estándares de codificación de video están basadas en la transformada discreta del coseno (DCT, *Discrete Cosine Transform*) bidimensional que descompone la señal de entrada en sumas ponderadas de cosenos de diferentes frecuencias espaciales [AHM74]. De las propiedades de la DCT en la compresión de imágenes, destacan que concentra la energía en un número reducido de coeficientes (compactación de la energía) y que minimiza la interdependencia entre los mismos (de correlación) [NAV03].

La DTC bidimensional aplicada a bloques de 8×8 pixeles es la que habitualmente se emplea en los estándares de compresión. La expresión de esta transformada normalizada para este tamaño de bloque se muestra en la Ecuación 2.28 [RAO96].

$$F(u,v) = \frac{C(u)C(v)}{4} \sum_{u=0}^{7} \sum_{v=0}^{7} f(x,y) cos\left[\pi(2x+1)\frac{u}{16}\right] cos\left[\pi(2y+1)\frac{v}{16}\right] \quad (2.28)$$

Dónde: x, y son las coordenadas en el dominio espacial. $f(x, y)$ es el valor que toma en (x, y) el pixel del bloque que se va a transformar. u, v son las coordenadas para el coeficiente F en el dominio de la frecuencia y $C(p)$ para $p = u, v$ está dado por:

$$C(p) = \begin{cases} \frac{1}{\sqrt{2}}, & p = 0 \\ 1, & p \neq 0 \end{cases} \tag{2.29}$$

El resultado de aplicar esta transformación es un bloque de 64 coeficientes; el primero (en las coordenadas 0,0) es el coeficiente de continua o coeficiente DC y el resto son coeficientes correspondientes a frecuencias espaciales crecientes (coeficientes AC).

La DCT no produce por tanto una compresión de la información, dado que genera el mismo número de coeficientes que los pixeles que tiene el bloque de entrada; es más, generalmente se produce un aumento en el número de bits con el que se codifica cada bloque debido a la mayor precisión con la que hay que representar los coeficientes en el dominio de la frecuencia. La ventaja que aporta es la concentración de la energía de cada bloque en el dominio espacial alrededor de las bajas frecuencias del dominio transformado, lo que es aprovechado por las etapas siguientes del proceso de compresión.

Finalmente la transformada inversa del coseno (IDCT), para bloques de 8x8 pixeles, se obtiene a partir de la expresión mostrada en la ecuación (2.30). Los parámetros que aparecen en la expresión de la IDCT tienen el mismo significado que los presentados para la DCT.

$$f(x, y) = \frac{1}{4} \sum_{u=0}^{7} \sum_{v=0}^{7} C(u)C(v)f(u, v)cos\left[\pi(2x + 1)\frac{u}{16}\right] cos\left[\pi(2y + 1)\frac{v}{16}\right] \tag{2.30}$$

2.2.4.1.4 Cuantificación

Una vez que se conoce la forma en que se concentra la información, es necesario determinar cuáles coeficientes de la matriz resultante deben codificarse sin que exista una pérdida significativa en la calidad visual. El proceso de cuantificación permite al codificador reducir el número de niveles con los que se representan los coeficientes transformados en el dominio de la frecuencia. Para ello se emplea una función de cuantificación que asigna un determinado código a cada dato que se encuentra en un rango de entrada. Para determinar el nivel real de los coeficientes transformados se emplean matrices de cuantificación. Estas matrices están determinadas por el tipo de componente de color y la calidad buscada. Al analizar la cuantificación, la mayoría de los coeficientes resultantes toman el valor cero, esto permite una codificación entre códigos de longitud variable y códigos de longitud de series.

2.2.4.1.5 Codificación de entropía

La codificación de entropía es un esquema de codificación sin pérdidas, donde el objetivo es codificar los bloques de pixeles con $-log_2 P_i$ bits (dado un conjunto de datos, donde la probabilidad de que un determinado bloque i ocurra está denotado por P_i, donde $i = 0, 1, 2, \cdots, L - 1$.), de modo que la tasa de bits media es igual a la entropía H del bloque de M pixeles [BOV09]:

$$H = \sum_i P_i \left(-log_2 {}^{1}/_{\mathrm{P_i}}\right) \tag{2.31}$$

Esto resulta en un código de longitud variable para cada bloque de M pixeles, con menores longitudes de código asignados a los bloques de pixeles altamente probables, reduciendo de este modo el régimen binario de salida del codificador. Shannon mostró un límite inferior para la longitud media de los códigos de muchos elementos codificados [SHA48]. Un método bien conocido para la construcción de un código que se acercó a este límite inferior es descrito por Huffman [HUF52]. En cada estándar se definen las tablas que deben emplearse para asignar los códigos de longitud variable a cada uno de los elementos sintácticos que componen la trama de bits de salida del codificador. Estas tablas pueden ser estáticas o modificarse dinámicamente durante el proceso de codificación (codificación adaptativa).

2.2.4.2 Decodificador

El proceso de decodificación tiene una menor complejidad en comparación al proceso de codificación. Este reconstruye los cuadros del video contenidos en el flujo comprimido. El diagrama a bloques mostrado en la Figura 2.19 [SHA14] muestra de manera simplificada la estructura del decodificador. Primero la secuencia de video comprimido entra al decodificador de entropía, éste extrae la información de cabecera, los coeficientes y los vectores de movimiento. Posteriormente la información es ordenada y almacenada de acuerdo a los diferentes tipos de cuadros de video (I, P, B). Para cada cuadro, sus coeficientes son re-escalados y transformados usando la IDTC para obtener el bloque de diferencia. La decodificación del vector de movimiento es usada para extraer la información de los cuadros de video previamente decodificados, esto se convierte en el bloque de predicción con compensación de movimiento para el cuadro actual. El bloque de diferencia y el bloque de predicción con compensación de movimiento son sumados uno a uno y así se obtiene el cuadro de video reconstruido, el cual es almacenado y utilizado para formar los bloques de predicción en los siguientes cuadros de video.

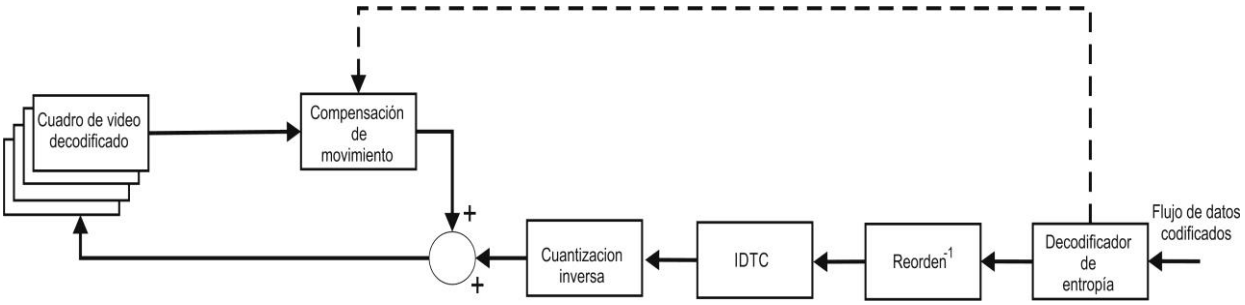

Figura 2.19 Decodificador de video híbrido.

2.3 Procesamiento de video en tiempo real

Los Sistemas de Procesamiento de Video en Tiempo Real (SPVTR), implican la manipulación de grandes cantidades de información de imágenes con el fin de extraer información útil. Esencialmente el video digital es una señal multidimensional y por lo tanto la manipulación de sus datos es una tarea intensiva que requiere una cantidad significativa de recursos de cálculo y memoria para su procesamiento [BOV09]. Por ejemplo si se considera una secuencia de video con un tamaño de imagen de $M \times N$ pixeles, con profundidad de color de P bits y una frecuencia de cambio de F imágenes por segundo, la tasa de bits a procesar por segundo seria de $M \times N \times P \times F$. En este contexto existen una gran variedad de algoritmos de procesamiento de video, con características dependientes del uso final de la secuencia de video. Algoritmos

que van desde el procesamiento de bajo nivel, donde las operaciones se realizan de manera uniforme en una imagen completa o en una secuencia, a los procedimientos de alto nivel, tales como el seguimiento de objetos y la identificación.

La necesidad de procesamiento de imágenes y video en tiempo real, puede satisfacerse mediante la explotación del paralelismo inherente en un algoritmo, por otro lado es importante discutir qué es exactamente lo que se entiende por el término "tiempo real". A partir de la literatura, se puede derivar que hay tres interpretaciones principales del concepto de "tiempo real", éstas son,

- El tiempo real en el sentido de percepción.
- El tiempo real en el sentido de la ingeniería de software.
- El tiempo real en el sentido del procesamiento de señales.

En el sentido perceptual el término tiempo real se utiliza principalmente para describir la interacción entre una computadora y un humano, es decir para describir una respuesta casi instantánea del dispositivo a una entrada por un usuario humano. Por ejemplo, Bovik [BOV00] define el concepto de tiempo real en el contexto de procesamiento de video como: "*El resultado del procesamiento aparece de forma instantánea (por lo general en un sentido de percepción) una vez que la entrada está disponible"*. También, se define el concepto de procesamiento de imágenes en tiempo real como: "*El procesamiento digital de una imagen que se produce inmediatamente; sin que se perciba el retraso de cálculo"*. Lo que se puede observar aquí es que el tiempo real consiste en la interacción entre el usuario y la computadora en la que los términos "aparece" y "perceptible", se refieren a la capacidad del ser humano para detectar retrasos. Por lo tanto el tiempo real en sentido perceptual, denota un tiempo de retardo máximo para la ejecución del procesamiento basado en la capacidad de percepción humana.

Por su parte, el tiempo real en el sentido de la ingeniería de software también se basa en un concepto de un tiempo de respuesta limitado como en el caso del sentido perceptual. Dougherty y Laplante [DOU95] señalan que *"un sistema de tiempo real es el que debe satisfacer las limitaciones de tiempo de respuesta acotadas de forma explícita, para evitar un fallo"*. En este sentido en la ingeniería de software, el concepto de tiempo real se clasifica basado en la delimitación del tiempo máximo de respuesta como: *tiempo real duro*, *tiempo real firme* y en *tiempo real suave*. En el tiempo real duro, es absolutamente imperativo que la respuesta del sistema a eventos externos ocurra dentro del tiempo especificado, en caso contrario se considera un fracaso o fallo. El tiempo real firme se refiere al caso en el que una cierta cantidad de plazos en tiempo real no se cumplen pero aun así es aceptable y no constituye una falla. Por último, el tiempo real suave se refiere al caso en que los plazos en tiempo real no cumplidos resultan en una degradación del rendimiento en lugar de un fracaso. En conclusión desde el punto de vista de la ingeniería de software, la cuestión de tiempo real es más sobre un rendimiento predecible y no sólo de un procesamiento rápido [LAP03].

Con respecto al procesamiento de señales, el concepto de tiempo real se basa en la idea de completar el procesamiento en el tiempo disponible entre las muestras de entrada sucesivas. Por ejemplo, en [KEH04] "tiempo real" se define como: *"completar el proceso dentro del tiempo permitido o disponible entre las muestras"*, y se afirma que un algoritmo en tiempo real, es aquel donde *"la cantidad total de instrucciones es menor que el número de instrucciones que se pueden ejecutar entre dos muestras consecutivas"*. Mientras que en [ACK99], el procesamiento

en tiempo real se define como: *"el cálculo del número de operaciones necesarias sobre la cantidad de datos de entrada dentro de un intervalo de tiempo específico, establecido por el período durante el cual llegan los datos".*

En conclusión, cada definición es válida según el contexto donde se esté empleando. Sin embargo la definición dada por Young [YOU82], la cual dice que; *"un sistema en tiempo real es aquel que realiza el procesamiento de información en respuesta a un estímulo externo y dentro de un periodo de tiempo específico y finito",* engloba las tres definiciones anteriores partiendo de la premisa que el tiempo para ejecutar el procesamiento cumpla con los requisitos de la aplicación, es decir que el cumplimiento en los requisitos temporales de la aplicación determinarán si el procesamiento se está ejecutando en tiempo real.

En este sentido, para efectuar un procesamiento de video en tiempo real, una de las líneas de investigación hace uso de la tecnología FPGA, en donde su naturaleza de procesamiento concurrente es idónea para efectuar operaciones con un nivel alto de paralelismo. Por tal motivo para una mejor comprensión de este trabajo, se introducen los conceptos de paralelismo en las operaciones de procesamiento de video y por otra parte se analiza el estado del arte de aplicaciones de procesamiento digital de video sobre FPGAs y las herramientas propuestas para facilitar el desarrollo de éstas aplicaciones.

2.3.1 Paralelismo en las operaciones de procesamiento de video

El concepto de procesamiento paralelo es la clave en el desarrollo de algoritmos de procesamiento de video en tiempo real. De hecho, gran parte de los sistemas eficientes de procesamiento de imagen y video, explotan diferentes formas de paralelismo que existen en un algoritmo, los cuales pueden ser el paralelismo a nivel de datos (DLP, *Data Level Parallelism*) o el paralelismo a nivel de instrucción (ILP, *Instruction Level Parallelism*) [DON01][HUN03]. El paralelismo a nivel de datos se manifiesta en la aplicación de la misma operación en diferentes conjuntos de datos, mientras que el paralelismo a nivel de instrucción se manifiesta en la ejecución simultánea de múltiples operaciones independientes, a lo que se le conoce como procesamiento *pipeline*.

En este sentido, las operaciones de procesamiento de imagen y video se pueden clasificar en tres niveles principales: bajo, intermedio y alto, donde cada nivel sucesivo se diferencia en relación de los datos de entrada y los datos de salida. Estos niveles se describen brevemente de la siguiente forma:

- Las operaciones de bajo nivel tienen una secuencia de imagen como su entrada y producen una imagen a su salida.
- Los operadores de nivel intermedio toman una imagen como su entrada y generan atributos de la misma.
- Los operadores de alto nivel toman atributos de la imagen como su entrada y generan una interpretación, comúnmente la interpretación provoca algún tipo de control basado en el conocimiento.

Esta clasificación jerárquica se puede representar como una pirámide con las operaciones intensivas de datos de pixeles en el nivel inferior, seguidas con las operaciones de extracción de

características en el nivel intermedio y las operaciones de interpretación basadas en el conocimiento en el nivel superior.

Las operaciones de bajo nivel transforman los datos de imagen en datos de imagen. Esto significa que dichos operadores tratan directamente con los datos de la matriz de imagen a nivel de pixel. Ejemplos de tales operaciones incluyen transformaciones de color, mejora de contraste, el filtrado, transformaciones de dominio de frecuencia, etc. El objetivo final de este tipo de operaciones es mejorar los datos de imagen, posiblemente para enfatizar ciertas características clave, preparándolos para su visualización por los seres humanos, o características de extracto para el procesamiento en el nivel intermedio.

Estas operaciones se pueden sub clasificar en operaciones puntuales, operaciones de vecindad local, y operaciones globales [BOV00][GON02]. Las operaciones puntuales son las operaciones más simples, dado un pixel de entrada éste se transforma en un pixel de salida, donde la transformación únicamente depende del pixel de entrada. Tales operaciones incluyen operaciones aritméticas, operaciones lógicas, las operaciones de umbral, etc.

En las operaciones de vecindad local, la transformación de un pixel de entrada a un pixel de salida depende de los pixeles que se encuentran en la vecindad del pixel de entrada. Tales operaciones incluyen las operaciones morfológicas, el filtrado espacial y espacio temporal, etc. Dado que cada pixel de salida es una función del pixel de entrada y sus vecinos, estas operaciones requieren una gran cantidad de cálculos.

Por último, las operaciones globales se basan en las operaciones de barrio en el que un sólo pixel de salida depende de cada pixel de la imagen de entrada. Un ejemplo destacado de tal operación es la transformada de Fourier discreta que depende de toda la imagen.

Las operaciones de nivel intermedio transforman los datos de imagen con el objetivo de extraer ciertos atributos o características de interés de una imagen. Esto significa que este tipo de operaciones también se ocupan de la imagen a nivel de pixel, pero una diferencia clave es que las transformaciones implicadas causan una reducción en la cantidad de los datos de entrada a la salida. Las operaciones intermedias incluyen principalmente segmentar una imagen en regiones u objetos de interés, la extracción de bordes, líneas, contornos, u otros atributos de imagen de interés, tales como características estadísticas. El objetivo mediante la realización de estas operaciones es reducir la cantidad de datos para formar un conjunto de características apropiadas para su posterior procesamiento de alto nivel.

Por último, en las operaciones de nivel alto se da una interpretación de los datos abstractos provenientes desde el nivel intermedio. Estas operaciones incluyen la clasificación y o reconocimiento de objetos o una decisión de control con base en algunas características extraídas. Por lo tanto los datos a procesar son menores pero no así las operaciones utilizadas para realizar su objetivo, tales operaciones son candidatas adecuadas para la explotación del paralelismo a nivel de instrucción.

2.3.2 Sistemas de procesamiento de video basados en FPGAs

Un dispositivo lógico reconfigurable FPGA cierra la brecha entre hardware y software, proporcionando un rendimiento más próximo al de un circuito de aplicación específico (ASIC), mientras que tiene la capacidad de reconfiguración de un microprocesador. La principal razón para el uso de FPGAs sobre otras plataformas (por ejemplo los GPUs o los DSPs), es que los FPGAs ofrecen un bajo costo de desarrollo, tienen alto rendimiento y se explota el

procesamiento paralelo, lo que lo vuelve adecuado para la transición de casi cualquier tipo de algoritmo de procesamiento de imagen y video desde un entorno de desarrollo para una verdadera aplicación en tiempo real. En la literatura se encuentra una gran cantidad de aplicaciones donde se hace uso de los FPGAs para el procesamiento de secuencias de imagen y video, es por ello que en este apartado se da un breve panorama al respecto.

Se inicia con el trabajo presentado por Gupta y Sinha [GUP04], donde se presenta la aplicación de filtros morfológicos difusos para el procesamiento de secuencias de video. La implementación fue optimizada haciendo uso de la tarjeta FPGA Xilinx Virtex XCV300, en la cual se obtuvo un rendimiento de 179 imágenes o cuadros por segundo (FPS, *frames per second*) con un tamaño de imagen de 512×512 pixeles.

Otro ejemplo donde se muestra la potencia de un FPGA para acelerar tareas de procesamiento se encuentra en [VEN05], donde se aborda el problema de la implementación del filtrado mediante el uso de la mediana y por convolución para el procesamiento de imágenes médicas en tercera dimensión. La implementación de las operaciones de filtrado se llevaron a cabo utilizando multiplicadores rápidos. El dispositivo utilizado fue el FPGA Xilinx Virtex II Pro 2VP125FF1696-6, y se logró un rendimiento de 95 FPS para imágenes de tamaño $128 \times 128 \times 128$ y de 12 FPS para tamaño de $256 \times 256 \times 256$.

En el trabajo mostrado en [DIA06], Díaz *et al*. describen una arquitectura *pipeline* para el procesamiento del flujo óptico basado en la técnica de Lucas-Kana. Su sistema hace uso de un sólo FPGA y procesa 30 cuadros por segundo con un tamaño de imagen de 640×480 pixeles.

Por su parte Lai *et al*. [LAI07] diseñaron una arquitectura hardware para la detección de rostros mediante el algoritmo AdaBoost, la aplicación puede procesar video VGA con una frecuencia de imágenes de 143 FPS utilizando una sola ventana de exploración funcionando a 126 MHz.

Otro trabajo de interés es el presentado por Irick *et al*. [IRI07], donde proponen la implementación de una red neural para la detección y clasificación de rostros. La arquitectura propuesta se implementó en el FPGA Virtex-4 de Xilinx y es posible procesar hasta 175 FPS con una resolución de 320×240 pixeles.

En este mismo contexto, Cho *et al*. [CHO09] también propusieron un hardware detector de rostros basado en AdaBoost con tres clasificadores paralelos utilizando un FPGA Virtex 5. La tasa de detección reportada fue de 7 cuadros por segundo para imágenes VGA y mejorando casi 19 veces el rendimiento de la implementación en software.

En [COP10] se presenta un enfoque sistemático para la comparación entre el GPU nVidia GeForce 7900 GTX y el FPGA Xilinx Virtex-4, usando una variedad de algoritmos de procesamiento de imágenes: corrección de color, la convolución 2D, el cambio de tamaño de video, la ecualización del histograma, y la estimación del vector de movimiento. El FPGA muestra ser superior en los algoritmos que requieren grandes cantidades de accesos a memoria, mientras que el GPU es superior para los algoritmos con reutilización de datos variables. En otro estudio realizado en [CHA08], se aplica el algoritmo de flujo óptico en el cual se obtuvo un rendimiento similar entre el FPGA y el GPU, pero la aplicación en el FPGA requiere un tiempo de diseño 12 veces mayor al diseño en el GPU.

La implementación de un algoritmo de segmentación utilizando el modelo de mezclas gausianas (GMM, *Gaussian mixture model*) para la segmentación del fondo fue propuesto por Genovese y Napoli [GEN11]. El esquema propuesto utiliza las operaciones morfológicas de

erosión, dilatación, apertura y cierre para la eliminación de ruido y es implementado en un dispositivo FPGA Virtex 5 xc5vlx50. El sistema es capaz de procesar video de alta definición HD de 1920×1080 a 24 FPS.

En [CAV11] se presenta una aplicación en donde el FPGA, es configurado específicamente para probar toda la electrónica que debe integrarse en una cápsula endoscópica, como una cámara, un motor de compresión de imágenes, un sistema de telemetría de alta velocidad, iluminación y sensores inerciales. Gracias a su gran flexibilidad, varias características fueron probadas y evaluadas, lo que permitió encontrar la configuración óptima, en términos de consumo de energía, actuaciones y tamaño, para el diseño de una cápsula. Como resultado final, con el FPGA se obtiene una velocidad promedio de 19 FPS a través de un canal de transmisión de 1,5 Mbit/s.

Finalmente se menciona el trabajo presentado por Atitallah *et al*. [ATI11], en el cual se describe una arquitectura FPGA del codificador H.264 / AVC que realiza la codificación en tiempo real. Para reducir la longitud del camino crítico y para aumentar el rendimiento, el codificador utiliza una arquitectura paralela y todos los módulos se han optimizado con respecto al costo de área. El diseño se describió en VHDL y se implementó en el FPGA Altera Stratix III. El rendimiento de la arquitectura alcanza una tasa de procesamiento mayor que 177 millones de pixeles por segundo a 130 MHz, lo que permite su uso en la norma H.264 / AVC dirigida a televisión de alta definición HDTV.

2.3.3 Herramientas para la implementación de algoritmos de procesamiento de video basados en FPGAs

La creciente demanda de este tipo de aplicaciones ha originado el desarrollo de herramientas, que faciliten el proceso de desarrollo. Dentro de este contexto varias han sido diseñadas.

La herramienta UltraSONIC (ver Figura 2.20) desarrollada por Haynes [HAY02], permite el desarrollo rápido de aplicaciones HW/SW de procesamiento de video en tiempo real, tales como la encriptación de video, la captura y reproducción de video comprimido proveniente desde una PC, y la compresión/descompresión MPEG, esto se logra a través de su arquitectura reconfigurable basada en elementos de procesamiento independientes que pueden ser conectados al bus principal; además cada uno de éstos puede ser un dispositivo FPGA individual que realiza una función específica de procesamiento sobre el flujo de los datos, esto ofrece la capacidad de procesamiento en paralelo.

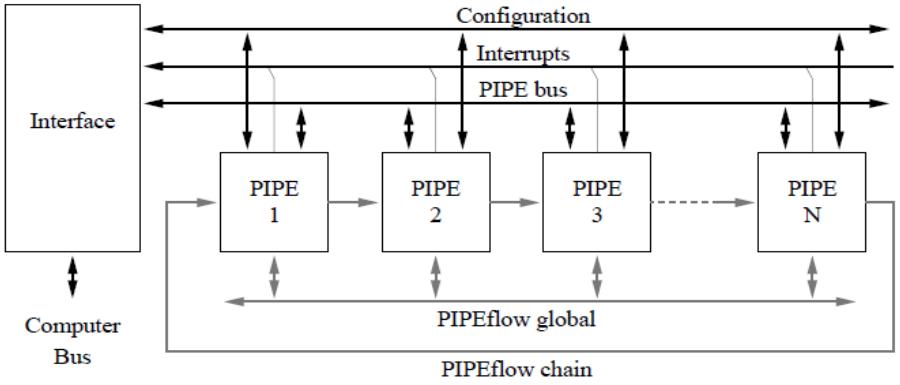

Figura 2.20 Arquitectura de la herramienta UltraSONIC [HAY02].

Por otra parte Atitallah *et al.* [ATI04], propone una arquitectura para una herramienta HW que facilita el desarrollo de aplicaciones de procesamiento de video en un FPGA(ver Figura 2.21). El sistema de procesamiento de video propuesto se compone de los módulos de adquisición, procesamiento y visualización. El módulo de adquisición incluye una interfaz para una cámara de 8 bits de color y el módulo de visualización se compone de una interfaz VGA de 24 bits, ambos módulos se desarrollaron en base al IP DMA (*Intellectual Property Direct Memory Access*) lo que permite la transferencia directa de los datos entre la cámara y la memoria SDRAM, y de la SDRAM al monitor VGA. En cuanto al sistema de control, la arquitectura propuesta integra el microprocesador *Nios softcore* de 32 bits de arquitectura RISC interconectado con los otros módulos mediante el bus *Avalon*.

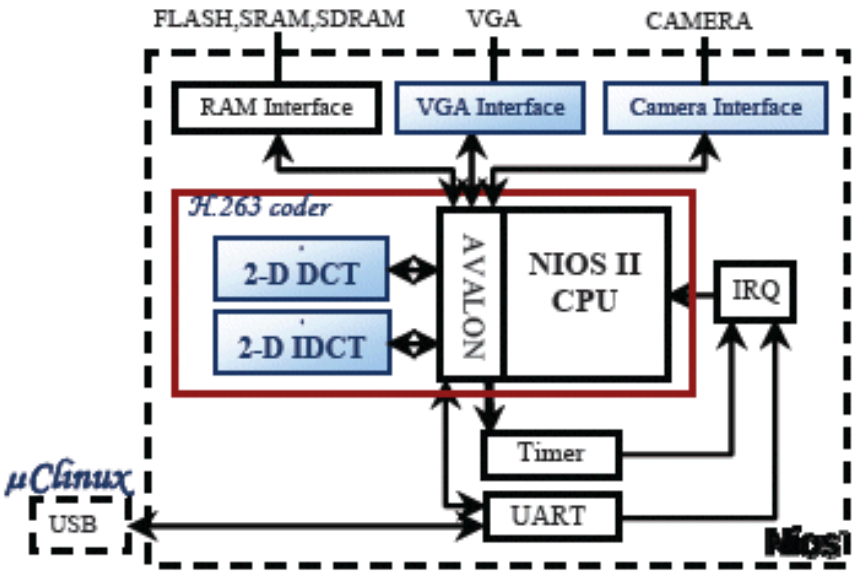

Figura 2.21 Arquitectura de la herramienta propuesta en [ATI04].

La plataforma de procesamiento de video VPP (*Video Processing Platform*), cuya arquitectura es mostrada en la Figura 2.22 fue propuesta por Desmouliers *et al*. [DES09]. Se trata de una herramienta de co-diseño HW/SW construida en base al FPGA Xilinx Virtex II pro, esta herramienta ofrece gran flexibilidad al usuario para un rápido desarrollo y validación de algoritmos de procesamiento de imagen y video en un enfoque de hardware reconfigurable, permitiendo el procesamiento y visualización de los resultados en tiempo real de las secuencias de video en niveles de gris y en formato RGB. Como prueba de viabilidad de la herramienta propuesta se implementó un algoritmo de detección de movimiento en tiempo real.

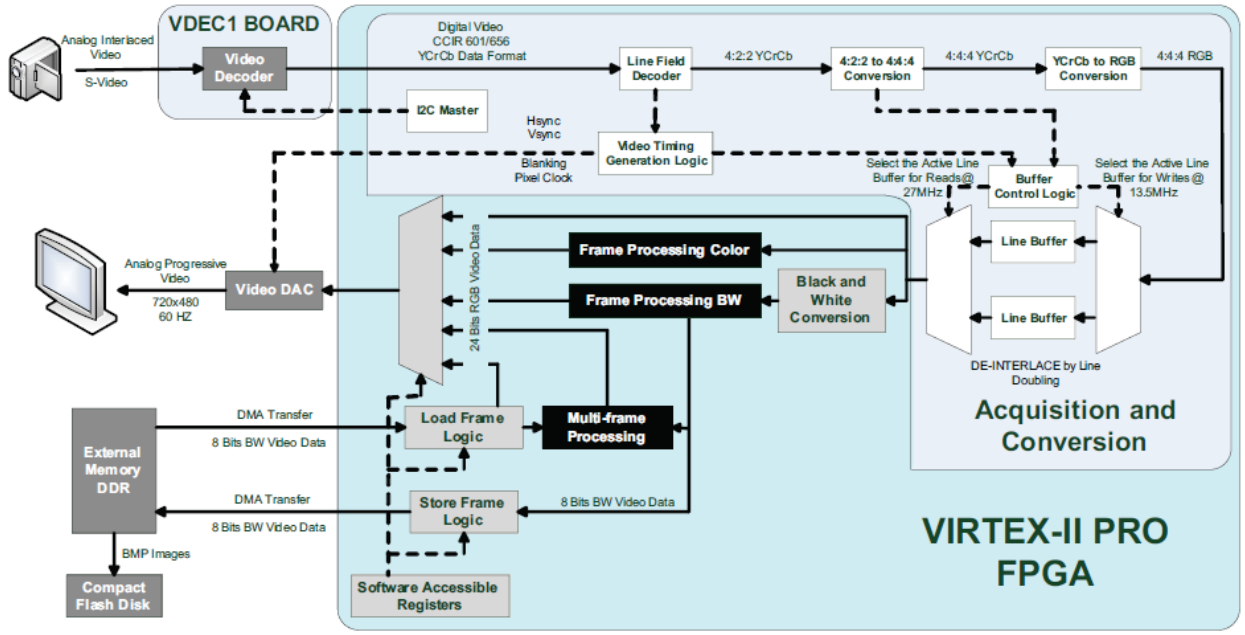

Figura 2.22 Arquitectura propuesta en [DES09].

Una de las aplicaciones frecuentemente utilizada en el área de visión artificial es la de seguimiento de objetos en movimiento, por esta razón en [HIR10] se propone el diseño e implementación de un sistema flexible de procesamiento de video basado en el FPGA Virtex-4 (Figura 2.23), para la segmentación en tiempo real de objetos en movimiento usando técnicas de sustracción de fondo y operaciones morfológicas. La herramienta propuesta ofrece la flexibilidad de modificar o cambiar los módulos de procesamiento, lo que permite la aplicación de diferentes algoritmos sin la necesidad de cambiar toda la arquitectura.

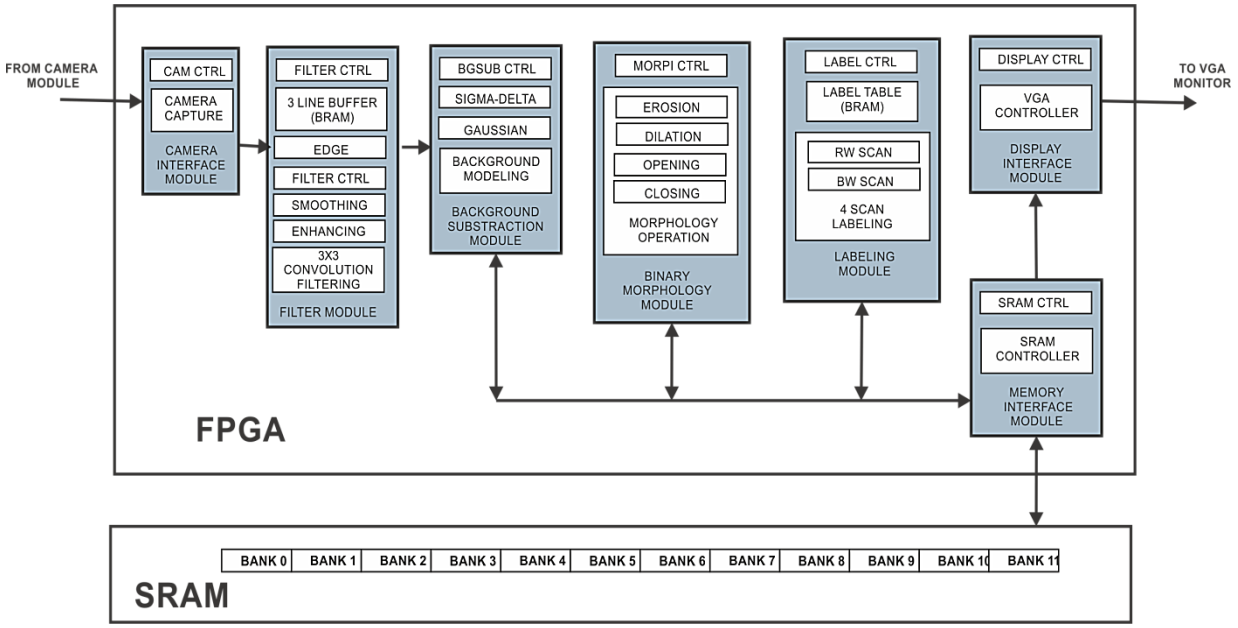

Figura 2.23 Sistema de procesamiento de video propuesto en [HIR10].

La visión estéreo es un método que asemeja el mecanismo básico de la visión humana. Sin embargo, la complejidad computacional y la gran cantidad de datos, hacen que el procesamiento en tiempo real de la visión estéreo sea un proceso altamente demandante. Con el fin de resolver este problema, en el trabajo mostrado en [JIN10], se propone un sistema de visión estéreo con una arquitectura *pipeline* (ver Figura 2.24) que proporciona una imagen de disparidad en tiempo real. Todas las etapas del proceso de la visión estéreo, tales como rectificación, coincidencia estéreo, y post-procesamiento, se realizan utilizando un único FPGA sin la necesidad de integrar otros dispositivos externos de procesamiento. El software implementado en el FPGA opera arriba de 230 veces más rápido en comparación con uno que se ejecuta en una computadora convencional.

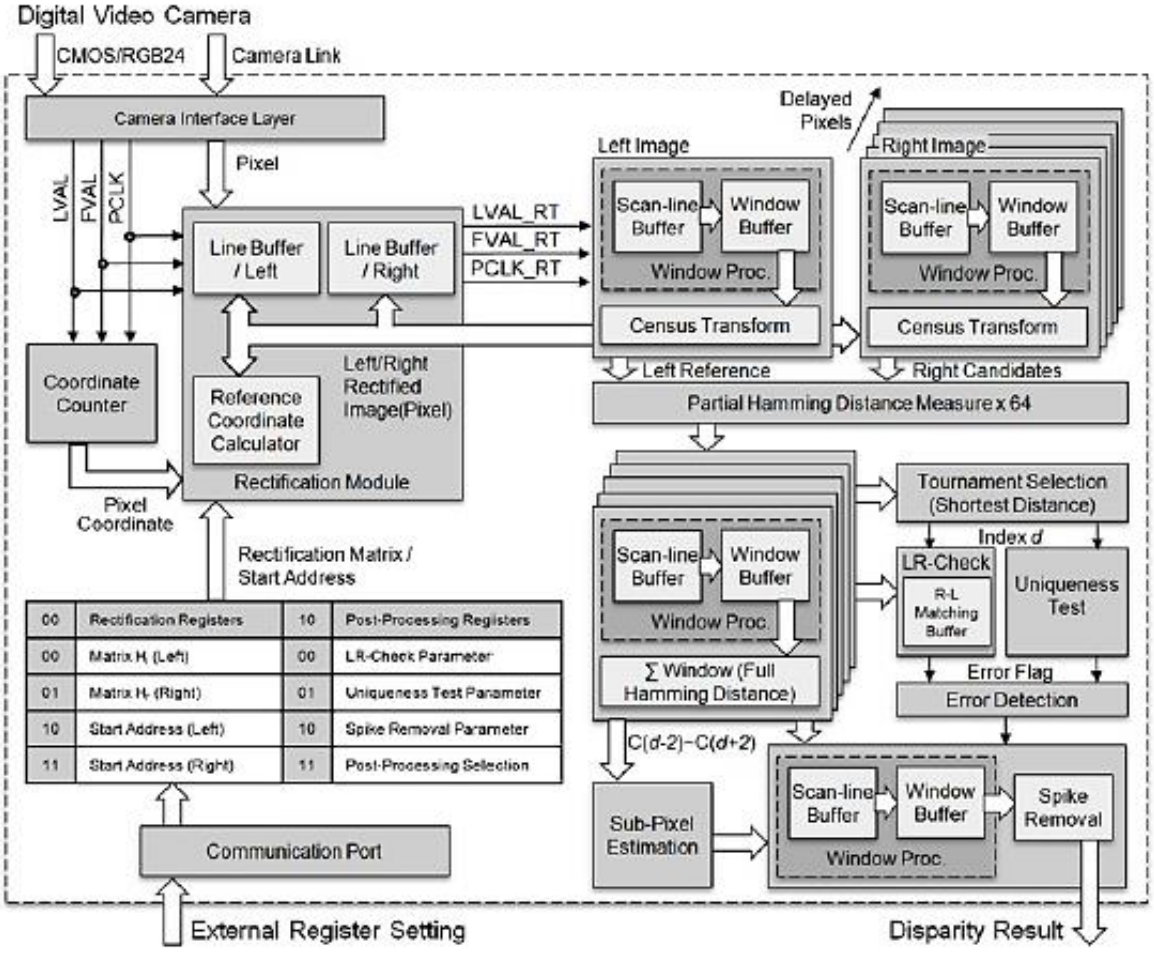

Figura 2.24 Arquitectura para un sistema de visión estéreo [JIN10]

Por su parte Gopal [GOP12], muestra el desarrollo de una herramienta HW/SW con una arquitectura escalable en un FPGA Virtex-5 y un procesador embebido PowerPC 440, para aplicaciones de procesamiento de video *streaming* en una cámara inteligente (ver Figura 2.25). La arquitectura es fácilmente configurable para diferentes resoluciones de video. Además, la arquitectura se puede ampliar y utilizar en diversas aplicaciones de procesamiento de imagen y de video sin la necesidad de modificar los componentes.

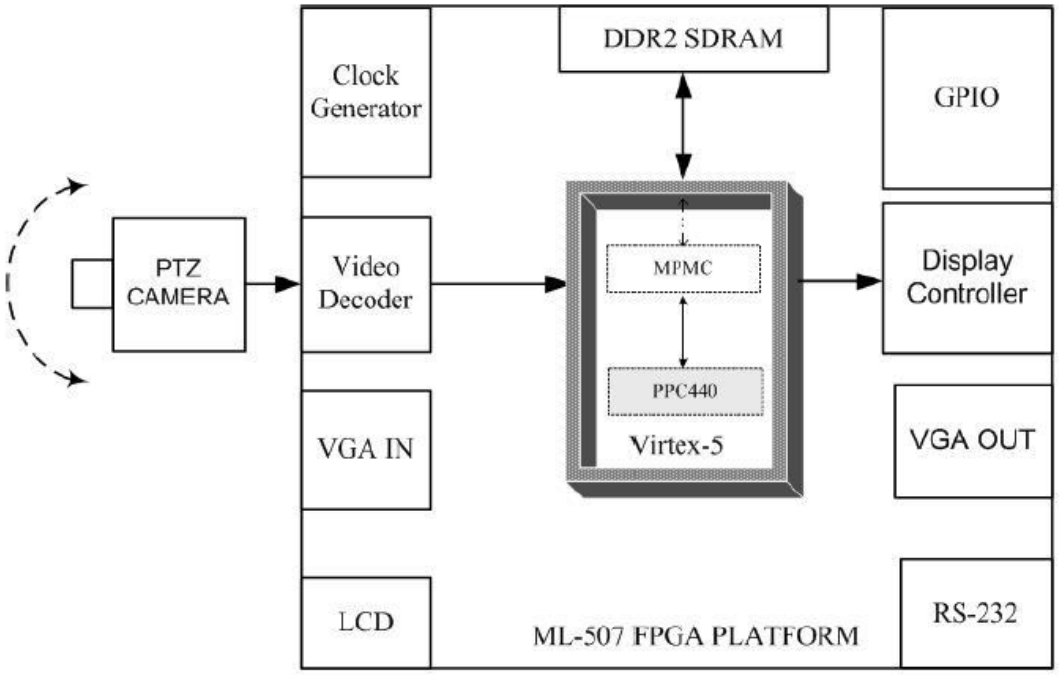

Figura 2.25 Diagrama a bloques de la arquitectura propuesta para el procesamiento de video en [GOP12].

Said y Saidani proponen en [SAI12] una herramienta de prototipado rápido para aplicaciones de procesamiento de video en HW (ver Figura 2.26). Esta herramienta proporciona un entorno de desarrollo que permite a los diseñadores empezar rápidamente a experimentar con el procesamiento de video utilizando la tarjeta VSK Spartan-3A DSP desarrollada por Xilinx. Se proporciona una entrada de video mediante una cámara CMOS y se despliegan los resultados en un monitor conectado al puerto DVI. El sistema está controlado por un procesador embebido *MicroBlaze* que inicializa los periféricos y controla el procesamiento de video mediante la lectura y escritura de registros de control en el sistema. Con la finalidad de mostrar el rendimiento y la flexibilidad de su herramienta, los autores implementaron la detección de bordes mediante el operador *Prewitt* y la transformada discreta *wavelet*.

Desmouliers *et al*. [DES10] desarrollaron la plataforma de procesamiento de imagen y video para aplicaciones en tiempo real IVPP (*Image and Video Processing Platform*). Esta herramienta de co-diseño HW/SW utiliza como elemento de procesamiento un FPGA Virtex-5 de la firma Xilinx (ver Figura 2.27). Los bloques de interfaz de video se realizaron en un nivel jerárquico de descripción RTL y la fase de inicialización se lleva a cabo utilizando un procesador *MicroBlaze*, lo que permite el apoyo de múltiples resoluciones de video. La herramienta agiliza el tiempo de desarrollo proporcionando todos los bloques lógicos necesarios para las operaciones de captura y despliegue de datos de video procesados. Además, en [DES12], los autores presentan un nuevo flujo de diseño de alto nivel de síntesis (HLS, *High Level Syntesis*) basado en el lenguaje C. esto quiere decir que el usuario puede diseñar aplicaciones de procesamiento de imagen y video en lenguaje C, después, convertirlos en HW usando la herramienta *Synphony* y luego implementarlos y probarlos fácilmente usando la IVPP, reduciendo con esto el tiempo de diseño y desarrollo.

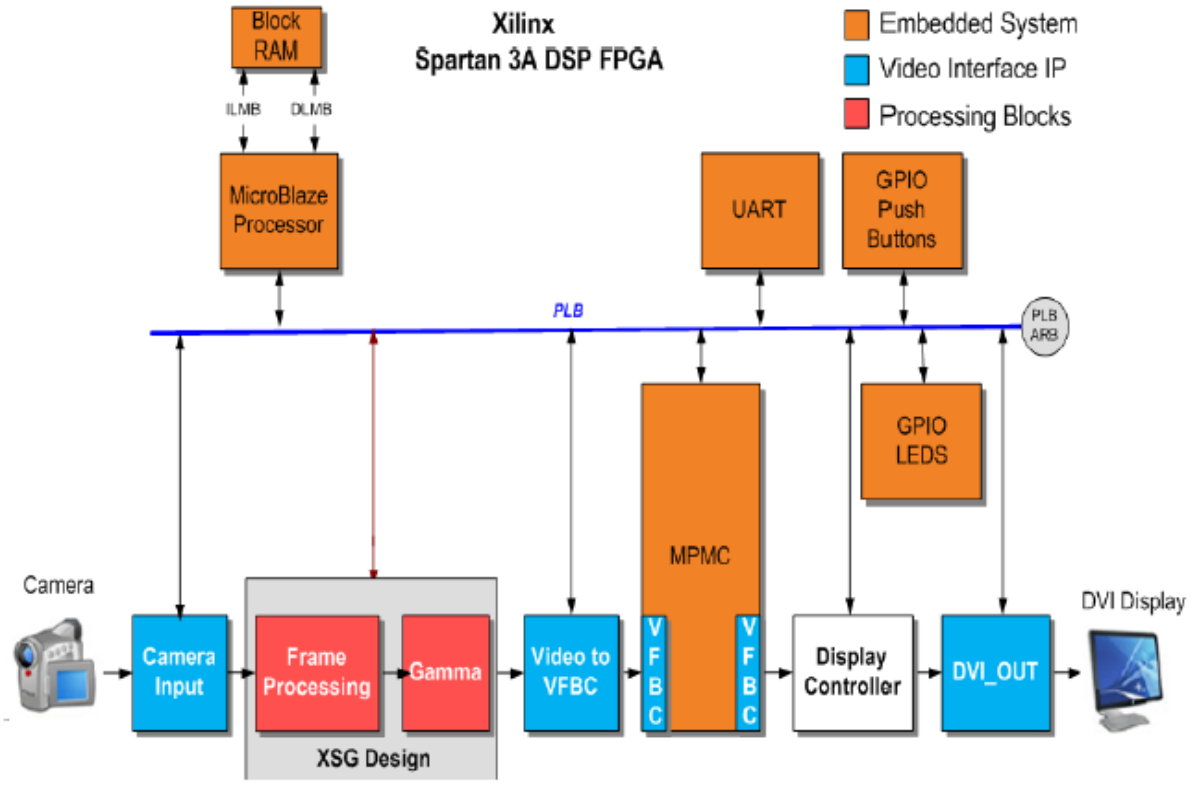

Figura 2.26 Arquitectura para una herramienta de prototipado rápido [SAI12].

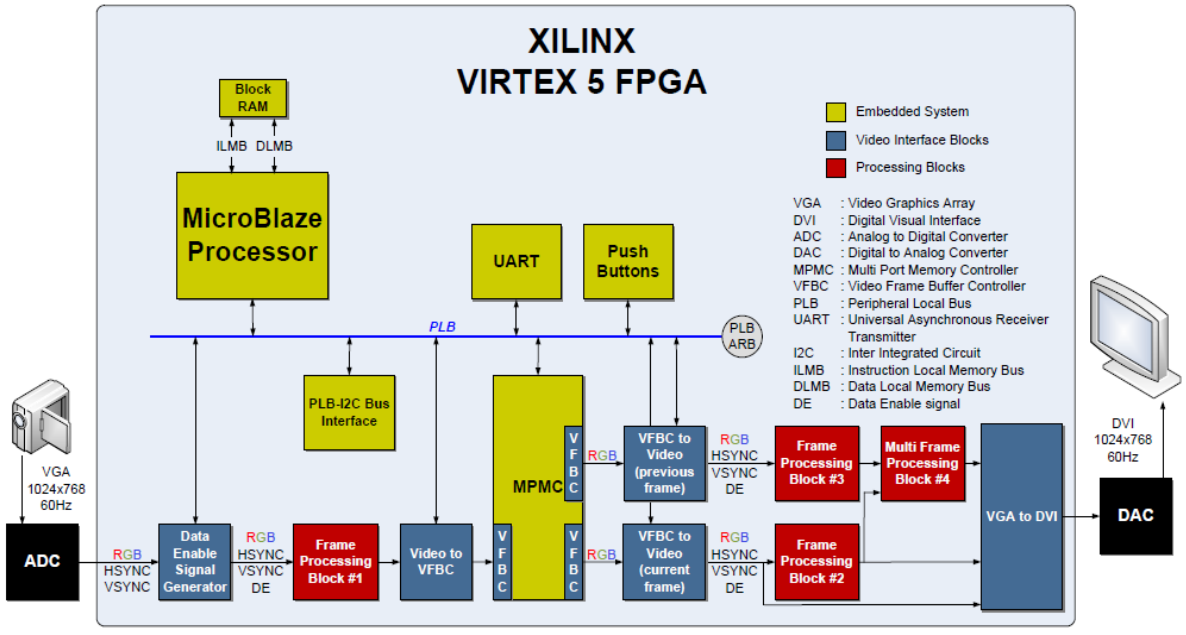

Figura 2.27 Resumen de la plataforma utilizada en [DES10] y [DES12].

De manera similar, Hiraiwa y Amano [HIR13] proponen una arquitectura de procesamiento de video basada en FPGA para sistemas embebidos de visión en tiempo real. La arquitectura de procesamiento de video implementada es segmentada, lo que ofrece la flexibilidad de

intercambiar los módulos de procesamiento. Cada módulo se implementa utilizando Verilog HDL y Vivado HLS para su evaluación.

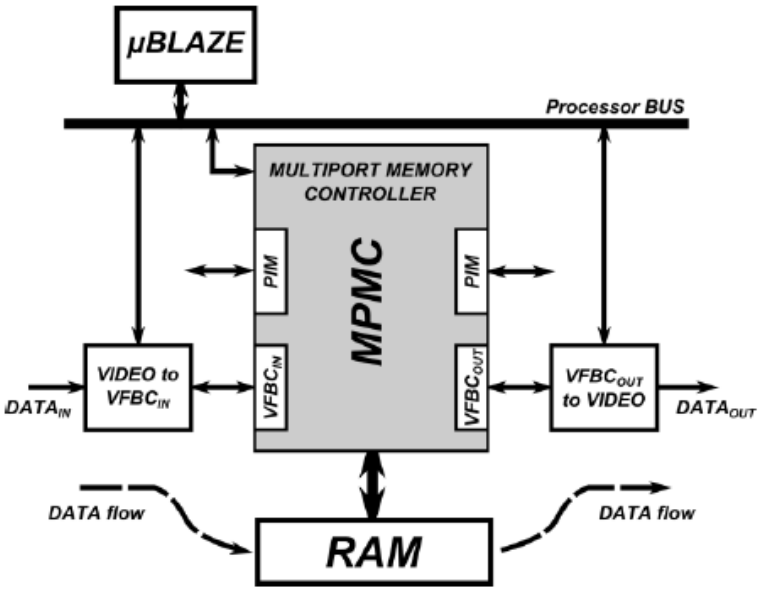

Figura 2.28 Estructura de la herramienta propuesta en [GAR13].

Por último, Garcés y su grupo de colaboradores, ofrecen una herramienta de prototipado (ver Figura 2.28) a partir de módulos HW para el desarrollo rápido de prototipos de sistemas de procesamiento de video basados en el controlador de memoria para fotogramas de video (VFBC) de Xilinx [GAR13]. Los módulos permiten el prototipo rápido de sistemas para el procesamiento de video que requieran acceder simultáneamente a información procedente de distintos fotogramas. Los resultados de esta investigación muestran la implementación de un detector de movimiento con la placa de desarrollo de Xilinx Spartan-6 SP605. La verificación funcional se llevó a cabo realizando una co-simulación HW/SW del diseño desde el entorno de trabajo de Simulink. La co-simulación HW/SW, provista por Xilinx System Generator, permite la comprobación del diseño desarrollado, direccionando el flujo de datos desde el entorno de Matlab hasta la placa y su retorno al espacio de trabajo para la comprobación del sistema y la detección de errores.

Capítulo 3

Métodos utilizados

El presente trabajo de tesis se enfoca en el diseño e implementación de una herramienta para la evaluación e implementación de algoritmos de procesamiento de video embebidos en un FPGA, por tal motivo en este capítulo se muestra una introducción sobre los dispositivos FPGAs. Además, la herramienta propuesta tiene como propósito el ser introducida como material de apoyo en el proceso de aprendizaje/enseñanza en cursos de Ingeniería, en este sentido, se estudian los conceptos de la teoría pedagógica constructivista con la finalidad de identificar características que se puedan integrar a la herramienta propuesta, también se presenta cómo ésta teoría ha sido utilizada con éxito en diferentes escenarios de enseñanza lo que justifica el por qué es importante tomar ésta teoría en cuenta al momento de establecer especificaciones de diseño de una herramienta de apoyo a la educación.

3.1 Arreglo de compuertas programables en campo (FPGAs)

Los arreglos de compuertas programables en campo o FPGAs, fueron inventados por Ross Freeman y Bernard Vonderschmitt en 1984 buscando subsanar las limitantes de los dispositivos lógicos programables o PLDs y como una evolución de los circuitos integrados de aplicación específica o ASICs. Así, en 1985 la compañía Xilinx lanzó al mercado su primer FPGA, el XC2064 [XIL85] con una capacidad de 1000 puertas lógicas equivalentes.

Estos dispositivos son chips de silicio reprogramables a nivel de hardware, cuentan con bloques de lógica configurables (CLB) y recursos de ruteo reprogramables. Cada CLB puede implementar funciones lógicas simples, mientras que la función de los recursos de ruteo es permitir interactuar a los CLBs entre sí y con el exterior. Al reconfigurar las conexiones

internas del chip, se obtiene un hardware personalizado, el cual puede ser empleado en aplicaciones específicas.

Entre los beneficios con los que cuentan estos dispositivos se tiene: tiempo más rápido de respuesta de entrada/salida y funcionalidad especializada, fiabilidad de hardware determinístico dedicado y se puede actualizar en campo, eliminando el alto costo inicial, largos ciclos de desarrollo y la falta de flexibilidad de las ASICs. También, permite mejoras de diseño en campo, sin la necesidad de reemplazar el hardware existente. A diferencia de los procesadores, los FPGA llevan a cabo diferentes operaciones de manera concurrente, por lo que éstas no necesitan competir por los mismos recursos. La aplicación es implementada y ejecutada en hardware en lugar de implementarse y ejecutarse en software, y cada tarea o proceso es independiente, asignándose a una sección dedicada del chip, y puede ejecutarse de manera autónoma sin afectar a otros bloques de lógica. Esta flexibilidad de la estructura de un FPGA permite albergar sistemas de gran complejidad. Además se tiene el control completo sobre la implementación del diseño sin la necesidad de tiempos perdidos en la fabricación de circuitos integrados [MEY04].

3.1.1 Arquitectura de las FPGAs

Existe una gran variedad de FPGAs provistos por varias compañías como Xilinx, Altera, Atmel y Lattice. Cada fabricante provee a su FPGA con una arquitectura única. La Figura 3.1 muestra la estructura general de un FPGA,

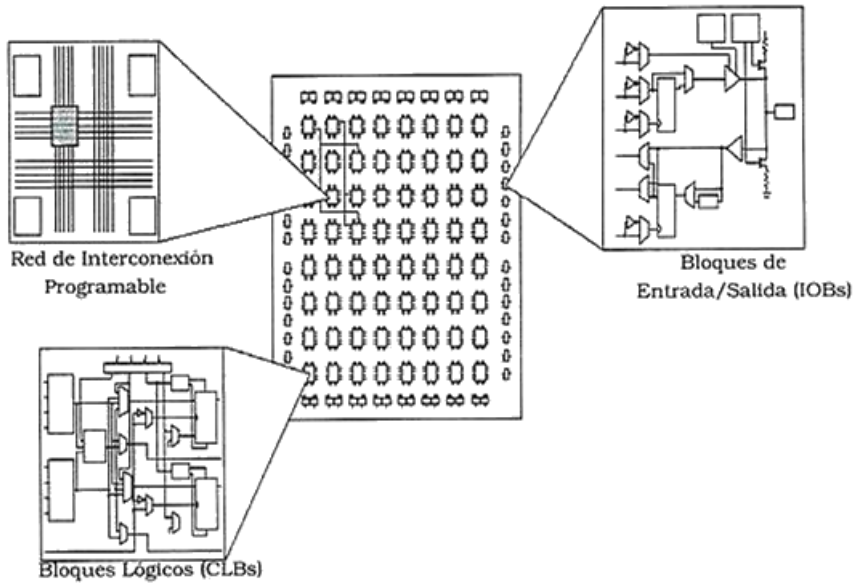

Figura 3.1 Estructura general de un FPGA

Se trata de una estructura simétrica compuesta por una matriz de CLBs, cada elemento CLB está rodeado por líneas dedicadas que conforman la red de interconexión. La matriz de CLBs está rodeada por un anillo de bloques de interfaz, denominados bloques configurables de entrada/salida.

3.1.1.1 Bloques lógicos configurables

Los Bloques Lógicos Configurables son recursos lógicos que permiten al usuario realizar diferentes funciones lógicas. Un bloque lógico está formado por *Slices* y cada *Slice* está formado por Celdas Lógicas (LC, *Logic Cells*). Una LC está formada por tablas de búsqueda (LUT, *Look-Up Table*) y elementos de almacenamiento (*flip-flops*). Las LUTs son usadas para implementar funciones lógicas combinacionales normalmente de cuatro entradas y una salida. Los elementos de almacenamiento pueden ser usados como *flip-flops* o *latches*. Además de los elementos mencionados, los bloques lógicos contienen recursos especiales para manejo de operaciones aritméticas, multiplexores extra para enlazar varios bloques lógicos en mejores condiciones y líneas de interconexión rápidas para el manejo de señales de reloj, de acarreo o de *reset*.

3.1.1.2 Líneas de interconexión

Conjunto de líneas e interruptores programables a través de los cuales las señales se transmiten internamente entre los bloques lógicos internos y también con los bloques de entrada/salida. Existen varios tipos de líneas de conexión en los cuales se pueden encontrar líneas largas para conectar bloques distantes, líneas cortas para conectar bloques lógicos adyacentes, líneas rápidas para rutas críticas en circuitos aritméticos, líneas de *set* y *reset* para los *flip-flops* y líneas de transmisión de reloj usadas para distribuir señales de sincronización.

3.1.1.3 Bloques configurables de entrada/salida

Estos bloques están dedicados a proporcionar la interconectividad entre el FPGA y el exterior, es decir, controlan la entrada y salida de datos entre los pines de entrada/salida y la lógica interna. Cada bloque es bidireccional y soporta operaciones de tercer estado, para conseguir estas características un bloque está dotado de *biestables, latches* y *buffers* de tercer estado. Además, en ocasiones incluye resistores *pull-up* y/o *pull-down* en la salida. La polaridad de la señal de salida es programable.

3.1.2 Programación de un FPGA

La programación de un FPGA consiste básicamente en configurar las celdas lógicas y especificar los puntos de conexión entre los diferentes tipos de bloques lógicos. En los FPGA reprogramables, la información de configuración e interconexión o ruteo se almacena en RAM estática dentro del mismo dispositivo [ROT98]. La tarea de programar se resume entonces en crear una cadena de bits (*bitstream*) que contenga la información de configuración para descargarla al FPGA.

El proceso de programación comienza típicamente con la descripción del sistema, pudiendo ser hecha mediante un lenguaje descriptor de hardware (HDL) como Verilog o VHDL (*Very High Speed Integrated Circuit Hardware Description Language*). Estos lenguajes permiten el modelado de sistemas digitales en diferentes niveles, grados de abstracción y jerarquía, como pueden ser los de estructura y de comportamiento [ASH08]. En el nivel de comportamiento, se indican las funciones que realiza el sistema, aunque no se indica cómo es que se implementan dichas funciones. El nivel de estructura tiene que ver con la forma en que el sistema está compuesto y la interconexión de los subsistemas. Cada uno de estos niveles puede dividirse a su vez en distintos grados de abstracción, de ahí la clasificación jerárquica.

El paso siguiente es la síntesis, en el cual la lógica de alto nivel especificada en el modelo de estructura y el modelo de comportamiento, son convertidos a lógica de bajo nivel, como compuertas lógicas, por ejemplo. Después el proceso de mapeo separa las compuertas en grupos que mejor se adapten a los recursos lógicos (CLBs) del FPGA. Entonces, se lleva a cabo un proceso de ruteo que se encarga de interconectar los CLBs que forman parte del sistema modelado. Finalmente, se crea un archivo binario que contiene la información para configurar los bloques lógicos y para realizar el ruteo apropiadamente [ASH08].

3.2 Constructivismo

El verbo construir proviene del latín *struere,* que significa arreglar o dar estructura. El principio básico de esta teoría proviene justo de su significado. La idea central es que el aprendizaje humano se construye, elaborando nuevos conocimientos a partir de la base de conocimientos anteriores. En este sentido, el aprendizaje de los estudiantes debe ser activo, éstos deben participar en actividades en lugar de permanecer de manera pasiva observando lo que se les explica.

El constructivismo difiere con otros puntos de vista en los que el aprendizaje se forja a través del paso de información entre personas (maestro hacia alumno), en este caso construir no es lo importante, sino recibir. En el constructivismo el aprendizaje es activo, no pasivo. Una suposición básica es que las personas aprenden cuándo pueden controlar su aprendizaje y están conscientes del control que poseen. Debe aclararse que esta teoría es del aprendizaje, no una descripción de cómo enseñar, a través de la cual los alumnos construyen conocimientos por sí mismos. Cada uno individualmente construye significados a medida que va aprendiendo.

Las personas no entienden, ni utilizan de manera inmediata la información que se les proporciona. En cambio, el individuo siente la necesidad de construir su propio conocimiento. El conocimiento se construye a través de la experiencia. La experiencia conduce a la creación de esquemas. Los esquemas son modelos mentales que se almacenan en la mente. Estos esquemas van cambiando, agrandándose y volviéndose más sofisticados a través de dos procesos complementarios: la asimilación y el alojamiento [PIA52].

Se puede decir entonces que el constructivismo es una filosofía, una teoría, un modelo, una metodología para orientar el accionar pedagógico activo. Los principios básicos del constructivismo, de acuerdo con Barberá *et al*. [BAR00] y Carretero [CAR97], residen en que:

- El conocimiento no es pasivamente recibido e incorporado a la mente del alumno, sino que es activamente construido.
- Sólo el sujeto que conoce construye su aprender.
- La cognición tiene una función de adaptación y para ello sirve la organización del mundo experiencial.
- La realidad existe cuando se origina una construcción mental interna interpretativa del alumno.
- Aprender es construir y reconstruir esquemas, modelos mentales.

- Aprender es un proceso individual y colectivo de diseño y construcción/reconstrucción de esquemas mentales previos, como resultado de procesos de reflexión e interpretación.

De acuerdo a estos principios es importante que el alumno asimile cómo aprender y no solamente qué aprender. Es necesario recordar que se trata de un proceso interno activo e interpretativo, por lo cual el profesor debe facilitar el aprendizaje en la medida que el aprendiz conozca, tenga conciencia y monitoree su forma de aprender. Dentro de los modelos constructivistas se pueden distinguir: cognitivo/biológico, social y radical, entre otros, cuyos representantes más notables son Piaget, Vygotsky y Von Glasersfeld, respectivamente.

De acuerdo con Piaget, el aprendizaje es considerado un proceso interno y personal, cuya finalidad es la adaptación del individuo al entorno, mediante la relación de equilibrio que involucran los procesos de asimilación y acomodación. Cualquier nuevo concepto o idea es asimilado y genera perturbación, una disonancia, un conflicto cognitivo, que se resuelve mediante una acomodación y reacomodación de esquemas y estructuras mentales, para luego asumir un estado de equilibrio y adaptación cognitiva entre conocimientos nuevos y conocimientos previamente construidos, dando significado al nuevo concepto en su estructura mental de pensamiento. El equilibrio es la tendencia propia de las personas a modificar esquemas y estructuras mentales para dar significado al mundo [PIA70]. En este contexto, el aprendizaje es una actividad personal o cognitiva. A esta tendencia se le conoce como *constructivismo cognitivo/biológico*.

En el *constructivismo histórico social*, propuesto por Vygotsky, el conocimiento se construye a través de la interacción entre un individuo y su medio, por lo que la interacción, la colaboración y el diálogo son elementos imprescindibles para que se genere un aprendizaje en los alumnos [VYG78]. Lo social es prioritario al desarrollo cognitivo. El aprendizaje surge a partir de la interiorización de los elementos externos en relación con los aprendizajes adquiridos previamente por el individuo en su interacción con otros. De Vygotsky proviene la idea de un aprendizaje distribuido y contextualizado, y una cognición situada o pertinente.

El *constructivismo radical*, por otro lado, enfatiza que la realidad está completamente dentro del conocedor. El alumno es el único que conoce y construye su conocer. En este contexto, para Von Glasersfeld la realidad es únicamente determinada internamente por el conocedor [VON94][VON95]. Para él, el conocimiento es instrumental, de manera que lo primero es dar a los alumnos las razones de por qué ciertas formas de actuar y pensar son deseables. De acuerdo con este planteamiento, el profesor no puede decirle a los aprendices qué conceptos construir o cómo construirlos, pero con un juicioso uso del lenguaje puede prevenirlos de construir en la dirección no deseada, puede motivarlos y orientarlos. Decir a los alumnos que tienen que cambiar sus ideas debido a que no son verdaderas, puede generar un alumno obediente, pero no genera entendimiento. Así, enseñar no es una forma de exponer y dictar, sino una forma de conversar. Von Glasersfeld señala que cuando una persona ve lo que otras hacen y escucha lo que otros dicen, se ve afectado inevitablemente lo que ésta hace y dice, y se reflejará en su pensamiento. Por ello, el trabajo con otros es fundamental para resolver incompatibilidades, incongruencias y perturbaciones, de tal forma que sea posible acomodar el conocimiento, aceptando el concepto de negociación de significado y conocimiento de los constructivistas sociales.

Por otra parte, en [AUS78] se propone una explicación teórica del proceso de aprendizaje de acuerdo con el punto de vista cognoscitivo, pero tomando en cuenta además factores afectivos tales como la motivación. Para Ausubel, el aprendizaje significa la organización e integración de información en la estructura cognoscitiva del individuo. El concepto más importante de la teoría de Ausubel es el de *aprendizaje significativo*. Este aprendizaje ocurre cuando la nueva información se enlaza con las ideas pertinentes de afianzamiento (para esta información nueva) que ya existen en la estructura cognoscitiva del que aprende. Es decir, el aprendizaje significativo es un proceso a través del cual una nueva información se relaciona con un aspecto relevante de la estructura del conocimiento del individuo. Este proceso involucra una interacción entre la información nueva (por adquirir) y una estructura específica del conocimiento que posee el aprendiz, a la cual Ausubel ha llamado concepto integrador. En este sentido, Ausubel ve el almacenamiento de información en el cerebro humano como un proceso altamente organizado, en el cual se forma una jerarquía conceptual donde los elementos más específicos del conocimiento se anclan a conocimientos más generales e inclusivos (asimilación). La estructura cognoscitiva es, entonces, una estructura jerárquica de conceptos, producto de la experiencia del individuo.

3.2.1 Características fundamentales para un aprendizaje efectivo

Durante los últimos 10 años, diferentes universidades alrededor del mundo han adoptado las TI como una alternativa para implementar métodos de aprendizaje y enseñanza más efectivos. Teniendo en cuenta esta influencia y reconociendo el impacto del desarrollo tecnológico en la sociedad, las universidades están incorporando la tecnología en el proceso educativo, descubriendo así su potencial para mejorar la eficacia estratégica de la enseñanza. Una línea de investigación que ha producido resultados positivos hace uso del *método constructivista* para la enseñanza. La idea del constructivismo trajo como resultados avances importantes en el entendimiento de cómo funciona el desarrollo cognitivo en las personas. La conexión entre la tecnología y el aprendizaje no es un hecho casual. Las aulas tradicionales resultan en muchos casos pobres para el soporte de la enseñanza, en cambio las nuevas tecnologías, si son utilizadas de manera efectiva, habilitan nuevas maneras para enseñar que se adecuan mucho más con la manera en como las personas aprenden.

En la interacción de los estudiantes con las nuevas tecnologías, se pueden aplicar los resultados que han mostrado muchas de las investigaciones que se encuentran relacionadas con el desarrollo cognitivo y el constructivismo, donde la conclusión ha sido la demostración de que el aprendizaje es más efectivo cuando están presentes cuatro características fundamentales, que son: compromiso activo, participación en grupo, interacción frecuente y realimentación, y contacto con el contexto del mundo real [ROS01].

3.2.1.1 Compromiso activo

Las investigaciones del aprendizaje constructivista han demostrado que los estudiantes aprenden mejor a través de la construcción de conocimiento por medio de una combinación de experiencia, interpretación e interacciones estructuradas con los integrantes del aula escolar (compañeros de clase y profesores). Cuando los estudiantes son situados en un rol pasivo, en el cual su función básica es la de recibir información por medio de clases, que son impartidas por el profesor y a través de los textos que les son asignados, usualmente fallan en tratar de desarrollar el entendimiento suficiente para aplicar lo que han aprendido en situaciones fuera de

los textos leídos y del aula escolar. También es importante tener en cuenta el hecho de que las personas tienen estilos diferentes de aprendizaje. El uso de las nuevas tecnologías para la adquisición del conocimiento ayuda a la creación de entornos, en donde el estudiante tiene herramientas que puede utilizar con independencia y a su antojo, logrando así una experiencia que fomenta la adquisición de un proceso de aprendizaje en el que el alumno se siente involucrado en su propio proceso de enseñanza. Las aplicaciones de las nuevas tecnologías deben servir para que el estudiante desarrolle sus ganas de independencia, tomando un papel activo al solucionar problemas, comunicarse efectivamente, analizar información y diseñar soluciones.

3.2.1.2 Participación en grupos

El constructivismo se enfoca sobre la base social del aprendizaje en las personas. El contexto social da a los estudiantes la oportunidad de llevar a cabo, de una manera más exitosa, habilidades más complejas que lo que pueden realizar por sí mismos. En los individuos, el componente social es muy importante, tener amigos y compartir con ellos. Las nuevas tecnologías se enfocan en este tema, aportando las herramientas necesarias para que las personas que accedan a ellas puedan compartir con los demás sus conocimientos, intereses, ideas y gustos.

Llevar a cabo tareas entre un grupo de estudiantes les proporciona una oportunidad en la que no sólo empiezan a comprender y adoptar ideas de los demás, sino también empiezan a discutir sus actividades y hacen que sus pensamientos sean visibles. El aprendizaje está relacionado con el significado y el uso correcto de las ideas, símbolos y representaciones. A través de las conversaciones sociales, los estudiantes y profesores pueden proporcionar consejos explícitos, resolver confusiones y asegurar que sus errores sean corregidos oportunamente. Además, las necesidades sociales son normalmente una razón para conducir el aprendizaje, porque la identidad social se mejora a través de la participación en la comunidad o al convertirse en miembro de algún grupo de su interés y compartir ideas. Involucrar a los estudiantes en una actividad intelectualmente social puede ser un motivador poderoso y puede llevar a un mejor aprendizaje, que el que resulta cuando los alumnos trabajan individualmente en su escritorio.

3.2.1.3 Interacción frecuente y realimentación

En las aulas tradicionales, las personas normalmente tienen muy poco tiempo para interactuar con los materiales, sus compañeros y el profesor. Además, los estudiantes usualmente deben esperar varios días o semanas después de entregar un trabajo escolar, para poder saber el resultado y la reacción del profesor ante sus ideas. El aprendizaje continúa de una manera más rápida cuando los alumnos tienen oportunidades frecuentes para aplicar las ideas que están aprendiendo y cuando las observaciones del éxito o fracaso de una idea aparecen en un espacio de tiempo corto.

Las nuevas tecnologías apoyan este principio de aprendizaje en al menos tres formas:

- Las herramientas tecnológicas por sí mismas pueden fomentar la interacción rápida y la realimentación. Por ejemplo, en los blogs, los estudiantes pueden compartir sus ideas e inmediatamente tanto sus compañeros como el profesor tienen acceso a leerlas, comentarlas y emitir opiniones sobre el tema.

- Las herramientas tecnológicas pueden mantener ocupados a los estudiantes en un periodo extenso de tiempo, tanto si están realizando un proyecto por sí solos o dentro de un grupo pequeño: esto crea más tiempo para que el profesor pueda realizar comentarios individuales sobre el desempeño particular de los estudiantes.
- En algunas situaciones, las herramientas tecnológicas pueden ser utilizadas para analizar el rendimiento de cada alumno y para proporcionar observaciones, de parte del profesor, más personales y con una mayor dedicación de tiempo, en comparación con las que típicamente reciben los estudiantes.

3.2.1.4 Contacto con el contexto del mundo real

Uno de los inconvenientes que se encuentra en el aprendizaje de los estudiantes es la frecuencia con la que fracasan en el momento de aplicar lo aprendido en el aula a los problemas con los que se enfrentan en la vida real. Las investigaciones realizadas sobre el tema concluyen que las personas deben primero llegar a dominar los conceptos esenciales, no simplemente memorizar hechos y técnicas de solución de una manera simplificada o contextos artificiales. Las asignaciones típicas de resolución de problemas no ofrecen al estudiante la oportunidad de aprender cuándo aplicar ideas particulares, porque es usualmente obvio que las ideas correctas para emplear son aquellas que preceden inmediatamente al texto.

Las nuevas tecnologías pueden proporcionar al estudiante herramientas excelentes para la aplicación de conceptos en una variedad de contextos, por lo tanto, rompen con el aislamiento artificial escolar llevando a situaciones del mundo real. Las nuevas tecnologías traen oportunidades para la participación activa de los estudiantes en la experimentación, diseño y reflexión, con un acceso a las mismas herramientas que muchos profesionales utilizan actualmente.

Por ejemplo, Hundhausen [HUN02] inspirado en la teoría del aprendizaje del constructivismo social, realizó un estudio etnográfico que exploró el valor educativo sobre la construcción y presentación de animaciones. Aunque esta investigación no midió en términos cuantitativos el aprendizaje, sí obtuvo resultados cualitativos interesantes. Por ejemplo, el uso de herramientas típicas de animación de algoritmos, distraía a los alumnos de las cuestiones importantes a la hora de diseñar las animaciones. Sin embargo, si se usaban métodos menos técnicos, como el diseño de viñetas (o comics), su motivación mejoraba claramente, aumentando su participación y con ello las posibilidades de aprender más.

Siguiendo esta misma línea Hundhausen *et al*. llevaron a cabo un estudio mucho más general y riguroso, con resultados significativamente favorables para el uso de las animaciones [HUN02a]. Este trabajo ha supuesto un punto de inflexión en la investigación del uso de animaciones de algoritmos con fines educativos. Tras hacer una revisión de 24 experimentos sobre la eficacia pedagógica del uso de las animaciones, llegaron a la conclusión de que *"lo realmente importante es lo que el alumno hace con la visualización y no lo que ve en ella".* Por otro lado, la conclusión más importante obtenida por los autores es que el uso más eficaz de las animaciones se corresponde con los enfoques constructivistas, y la construcción de animaciones

es precisamente un uso constructivista. De hecho, el aprendizaje mejora si se compara con el enfoque típico que no hace uso de las animaciones de ninguna forma.

Por otra parte, en [ZUR04], Zurita y Nussbaum muestran que el uso de dispositivos móviles se puede combinar con el aprendizaje tradicional con el objetivo de incrementar la motivación de los estudiantes, promover la interacción entre los miembros de los grupos de trabajo, facilitar el desarrollo de ciertas habilidades y dar soporte al aprendizaje constructivista. En este sentido manifiestan que el proceso de aprendizaje bajo un enfoque constructivista, tiene mayor impacto cuando se incorporan herramientas tecnológicas que permiten el desarrollo de actividades educativas en un ambiente colaborativo. Sus conclusiones se sustentan en base a un experimento realizado con dos grupos de niños (de entre 6 y 7 años), en donde ambos grupos desarrollaron actividades de construcción de palabras a través de silabas, mediante el uso de dos distintas herramientas didácticas. El primer grupo realizó sus actividades mediante el uso de material didáctico compuesto por un conjunto de tarjetas y plantillas ilustrativas que guiaron y simplificaron la actividad de construcción de palabras a través de un trabajo en equipo. Mientras que el segundo grupo recibió como herramienta una aplicación software basada en dispositivos móviles, que permitió la realización de las actividades de construcción de palabras en un ambiente colaborativo, donde cada integrante recibió un dispositivo conectado a la red local, con el cual pudo interactuar con los demás integrantes del grupo para el desarrollo de la actividad. Como resultado de su experimento, se observó que el 95% de los niños que realizaron su actividad con el uso de los dispositivos móviles obtuvieron mejores calificaciones en las pruebas de construcción de palabras, con respecto a los niños que realizaron su actividad en papel.

En el trabajo descrito en [PAT06], Patten *et al.,* se realizó una categorización de las aplicaciones basada en aspectos funcionales y pedagógicos, de esta manera, se contó con una referencia que permitió agrupar las aplicaciones educativas móviles. Las categorías de aplicaciones propuestas son: aplicaciones de administración, aplicaciones de referencia, aplicaciones interactivas, aplicaciones de micromundo, aplicaciones colaborativas, aplicaciones de localización y aplicaciones de recolección de datos. En donde particularmente se menciona que las categorías de aplicaciones colaborativas, aplicaciones de localización y aplicaciones de recolección de datos son adecuadas para el aprendizaje con dispositivos móviles cuando se les introducen las teorías de aprendizaje colaborativo, contextual y constructivista.

Mediante los estudios realizados en [BLI10], Blikstein y Wilensky, corroboraron, que es posible extender los conceptos sobre Ingeniería de Diseño, el construccionismo, y los enfoques basados en proyectos *(Project-based learning)*, a los cursos teóricos de ingeniería como es el caso del curso de Ciencia de los Materiales. El estudio consistió en la introducción de un conjunto de modelos y actividades construidos por los autores en el entorno NetLogo (NetLogo es un descendiente directo del lenguaje Logo [WIL99]), que permite la realización de prácticas en temas relacionados con la ciencia de los materiales. A este conjunto de modelos y actividades lo nombraron MaterialSim, en el cual los estudiantes pueden crear y/o interactuar con modelos de procesos físicos fundamentales en los materiales como por ejemplo: el movimiento a nivel atómico y la reducción de la energía libre. En concreto, sus resultados sugieren que la importancia en la creación de modelos por parte de los estudiantes generó una experiencia de aprendizaje particularmente valiosa.

En particular, cuando la realización de prácticas experimentales es el elemento básico en la educación en ingeniería, la experimentación a distancia tele operada abre nuevas dimensiones

en la adquisición de conocimientos. En este sentido la plataforma PeTEX (*Platform for E-Learning and Telemetric Experimentation*), descrita en [TER11], fue desarrollada para el diseño e implementación de programas educativos y de formación en el campo de la Ingeniería en Manufactura, bajo un entorno de experimentación a distancia y de aprendizaje electrónico *(e-Learning).* La experimentación tele operada proporcionada por la plataforma, permite a los estudiantes llevar a cabo eficazmente pruebas como: la caracterización de materiales mediante el ensayo de tensión axial, la soldadura por fricción y el establecimiento de parámetros adecuados para el proceso de corte. De esta manera, el proyecto proporciona la adquisición de conocimiento individual y de grupo en el campo de la Ingeniería en Manufactura, así como el aprendizaje social e intercultural en una comunidad de aprendizaje global.

Para finalizar en [GAR11], García y Cano presentan un enfoque alternativo para el aprendizaje en el diseño de sistemas embebidos bajo la teoría del construccionismo. Su investigación se centra en la incorporación de una plataforma construccionista para el desarrollo de experimentos en base a tecnologías de comunicaciones inalámbricas de corto alcance, utilizando una metodología dedicada para el desarrollo de sistemas embebidos trabajando en conjunto con el enfoque construccionista. Entre sus principales conclusiones se menciona que la introducción del construccionismo fue una estrategia excelente para mejorar la adquisición de conocimiento y habilidades en cursos relacionados con la Ingeniería en Electrónica. Además, muestran que es importante implicar a los estudiantes en entornos de desarrollo con problemas reales y situar los proyectos en un contexto que sea conocido por ellos, donde la realimentación es una gran motivación para los estudiantes y promueve el interés para terminar correctamente sus proyectos asignados.

En resumen el uso de herramientas tecnológicas, enriquece el proceso de enseñanza/aprendizaje. En particular cuando estas herramientas son introducidas con un enfoque de enseñanza constructivista proporcionan un soporte eficiente en la educación, puesto que los estudiantes toman un papel activo en su aprendizaje a través de actividades significativas que se relacionan con los conocimientos previamente adquiridos.

Capítulo 4

Estructura de la herramienta propuesta

En este capítulo se describe a detalle la arquitectura del sistema de procesamiento de video. Este sistema está basado en un dispositivo FPGA en el cual se integran los elementos necesarios para el desarrollo de aplicaciones de procesamiento de video, de una forma flexible y funcional. Por otra parte, se describen las consideraciones necesarias a tomar en cuenta para el uso de la herramienta en el desarrollo de prácticas experimentales.

4.1 Sistema de captura y procesamiento de video basado en un FPGA

Como se ha mencionado anteriormente, este trabajo de tesis propone una herramienta HW/SW enfocada al desarrollo y evaluación de algoritmos de procesamiento de video sobre un sistema embebido que tiene como elemento central de procesamiento a un dispositivo FPGA. Además, se considera para su diseño que en un futuro ésta será incorporada a cursos relacionados con las Ciencias de la Computación a nivel ingeniería y de posgrado a través de un enfoque constructivista.

En este sentido, como principal requisito, la herramienta debe permitir la evaluación del modelado, implementación y ejecución en HDL de algoritmos de procesamiento digital de video, propiciando de esta manera la construcción activa del conocimiento, base fundamental del enfoque de enseñanza constructivista. Para lograr esto la herramienta debe proporcionar la flexibilidad para alojar distintas arquitecturas de algoritmos a implementar, sin restringir, arbitrariamente a una arquitectura específica del algoritmo en cuestión; es decir, la herramienta debe ser fácilmente adaptada a la arquitectura del algoritmo, sin que esto conlleve a cambios drásticos. Ahora bien, el propósito de extraer especificaciones de diseño en base al enfoque constructivista tiene como objetivo el proporcionar un medio de aprendizaje significativo, en

temas relacionados con el diseño y modelado de sistemas embebidos de procesamiento de video en tiempo real, y permitir integrar conocimientos previamente adquiridos en cursos anteriores (Sistemas digitales y Procesamiento digital de imagen y video).

Ahora bien, a partir del requisito principal, se estudió la teoría acerca del procesamiento digital de video, las características de los FPGAs y el enfoque constructivista, de donde se identificó el siguiente conjunto inicial de requisitos del sistema se resumen en:

- Implementar la herramienta en base a la filosofía de código abierto (*open source*), para permitir la evolución de la herramienta mediante un entorno colaborativo, donde se aporten nuevas funcionalidades, métodos y algoritmos, a partir de los ya existentes o versiones completamente nuevas, para enriquecer el entorno de desarrollo.
- La herramienta debe poder ser adaptada a distintas metodologías de desarrollo.
- Mediante el uso de la herramienta se debe fomentar la interacción rápida y la realimentación.
- Permitir enfocar el esfuerzo de los usuarios solamente en el diseño de las funcionalidades del algoritmo que es objeto de estudio.
- Reducir el tiempo de desarrollo e implementación de algoritmos de procesamiento digital de video en sistemas embebidos basados en un FPGA.
- La arquitectura del sistema debe ser flexible y modular.
- Debe incorporar un sistema de adquisición de video.
- Debe permitir la visualización de los resultados del procesamiento.
- Proporcionar un sistema de memoria adecuado para la implementación de algoritmos de video.
- Debe ser un sistema de procesamiento de alta velocidad.
- Incorporar una interfaz entre el usuario y la herramienta a través de una computadora personal. La cual debe cumplir con los siguientes requisitos:
    - Permitir el monitoreo de resultados.
    - Gestionar los parámetros de configuración de la herramienta y los algoritmos.
    - Ser una interfaz de alta velocidad.
    - Debe ser flexible a modificaciones por el usuario.

Partiendo de la definición de los requisitos, el esquema general de la herramienta propuesta consiste en una interfaz de usuario y un sistema embebido de captura y procesamiento de video basado en un FPGA (SECPV) (ver Figura 4.1). A su vez el SECPV está integrado por: el elemento central de procesamiento, sistema de comunicación, el sistema de memoria, el sistema de captura de video y el sistema de visualización de resultados.

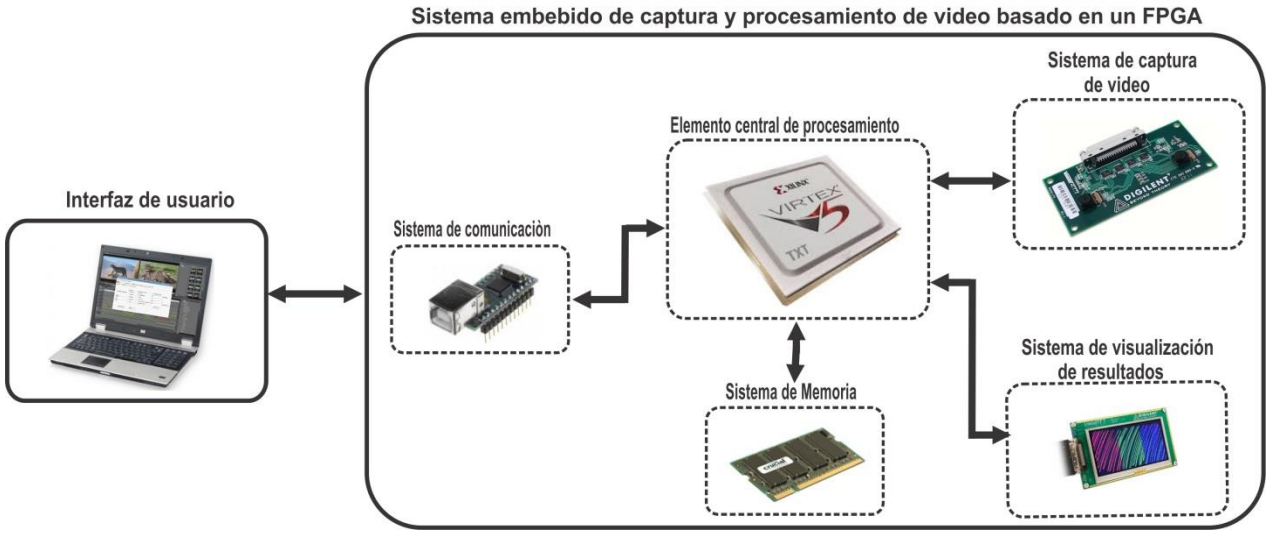

Figura 4.1 Sistema de procesamiento digital de video basado en un FPGA.

El SECPV es un sistema basado en un FPGA, cuyo objetivo es brindar apoyo en los procesos de enseñanza e investigación, siendo esta una plataforma experimental que permite acortar los tiempos de desarrollo de algoritmos de aplicación específica, dirigidos al ámbito de procesamiento de video. A partir de su esquema general, se definen los componentes hardware y las especificaciones de los componentes software que lo integran.

Componentes Hardware:

- Elemento central de procesamiento.
- Sistema de comunicación.
- Sistema de visualización de resultados.
- Sistema de memoria.
- Sistema de captura de video.

Componentes Software:

- Interfaz gráfica de usuario.
- Descripción HDL del elemento central de procesamiento.

4.1.1 Descripción de los componentes Hardware

Considerando las especificaciones iniciales de los elementos con las que el sistema de procesamiento de video debe contar, se ha seleccionado una interfaz USB para la transferencia de datos, un sistema de memoria DDR2 con capacidad de almacenar datos de hasta 256 Mbytes, un sistema para visualizar los resultados mediante un LCD, un sistema de captura de video estéreo y, como elemento de procesamiento, un dispositivo FPGA con los recursos necesarios para albergar los módulos que conforman la herramienta y los algoritmos a evaluar.

4.1.1.1 Elemento central de procesamiento

El elemento central de procesamiento es la parte medular de este sistema, para cumplir esta función se eligió el dispositivo lógico reconfigurable FPGA Virtex5-LX50T manufacturado por la compañía Xilinx Inc. y que cuenta con las siguientes características principales:

- Posee una matriz de 120×30 (filas × columnas) de CLBs, con un total de 7200 slices (Cada slice está integrado por 4 LUTs y 4 flip-flops) y un máximo de 480 Kb de memoria distribuida entre todos los CLBs.
- En este tipo de FPGAs se encuentra una columna compuesta por 48 DSP48E de los cuales cada uno está integrado por un multiplicador de 25×18 bits, un sumador de 48 bits y un acumulador.
- Se puede hacer uso de 60 bloques RAM de 36 Kb, donde cada bloque puede ser utilizado como dos bloques independientes de 18 Kb cada uno. En total se puede contar con un total de 2,160 Kb de memoria entre todos los bloques RAM.
- Para manipular las señales de reloj cuenta con seis administradores de reloj (CMT *Clock Management Tile*), los cuales están compuestos por dos gestores de reloj digital (DCM, *Digital Clock Manager*) y un lazo de seguimiento de fase (PLL *Phase Locked Loop*).
- Para su comunicación con el exterior la Virtex 5-LX50T cuenta con 480 puertos de entrada y salida (IOB *Input Output Blocks*), agrupados en 15 bancos diferentes.
- La frecuencia interna de operación es de hasta 500Mhz.

Por otra parte, el uso de herramientas para la automatización del diseño electrónico (EDA, *Enviroment Desing Automation*) facilita el desarrollo de sistemas electrónicos, debido a que automatizan funciones como la descripción del sistema, la simulación, el prototipado, y la producción. En este sentido la herramienta EDA hardware que provee la compañía Digilent Inc., a través de la tarjeta de evaluación Genesys (ver Figura 4.2), proporciona un entorno adecuado para el desarrollo de sistemas basados en el FPGA Virtex 5 LX50T. Entre los principales recursos con los que cuenta esta tarjeta destacan los siguientes:

- Cuenta con un dispositivo FPGA Xilinx Virtex 5 LX50T con encapsulado de 1136 pines unido a la tarjeta por BGA (*Ball Grid Array*).
- 256 MByte de memoria DDR2 SODIMM con un bus de datos de 64 bits.
- Puerto 10/100/1000 Ethernet PHY para conexiones Ethernet.
- Un puerto serie RS-232.
- Múltiples puertos USB2 para la programación, transferencia de datos y hosting.
- Un puerto HDMI con una resolución de hasta 1600×1200 pixeles y profundidad de color de 24 bits.

- Un códec AC-97 con muestreo de 48 kHz para la línea de entrada y salida de micrófono y auriculares.
- Monitoreo en tiempo real del consumo de energía de las vías de alimentación.
- Provee un oscilador interno de 100MHz y un generador de reloj programable de hasta 400 MHz.
- Proporciona 112 puertos de E/S, dirigidos a los conectores de expansión (dos conectores VHDC paralelo de alta velocidad y cuatro cabeceras de 8 pines).
- Cuenta con 8 LEDs, 2 botones, un interruptor de navegación de dos ejes, 8 interruptores y un LCD de caracteres de 16×2.
- Su alimentación es mediante una fuente de alimentación externa de 20W.

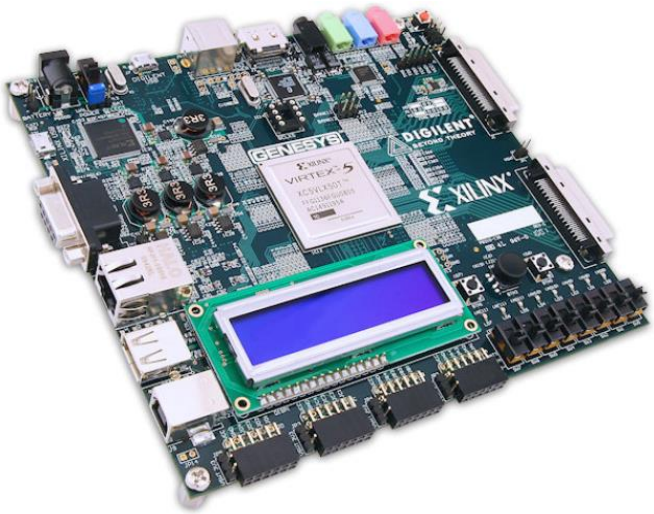

Figura 4.2 Tarjeta de evaluación Digilent Genesys**.**

4.1.1.2 Sistema de comunicación

El incorporar un sistema de comunicación en la herramienta propuesta tiene por función permitir el intercambio de información entre el SECPV y la interfaz de usuario. Para cumplir con este requisito se optó como puente de comunicación al protocolo USB mediante el uso del dispositivo DLP-USB1232H, el cual es un módulo de interfaz USB basado en el FT2232HQ de 5ª generación de la compañía FTDI. Este dispositivo tiene como principales características:

- Compatibilidad con los estándares USB 1.1 y USB 2.0.
- Una tasa de transferencia de datos de hasta 8 Mbytes por segundo usando una transferencia USB a FIFO paralelo.
- Controladores gratuitos VCP (*Virtual Com Port*) y librerías DLL (D2XX) para desarrollo de software compatible con los sistemas operativos Windows, Mac y Linux.

- No se requiere el conocimiento del protocolo USB puesto que este es manejado internamente por el dispositivo.
- Se puede configurar para hacer la conversión del protocolo USB a protocolos seriales como: UART, JTAG, SPI, I2C, o mediante un protocolo paralelo con la FIFO (en el caso de este trabajo se decidió trabajar con el protocolo paralelo).

La descripción de los pines de conexión para el modo de interfaz paralelo del DLP-USB1232H se detalla en la Tabla 1.

Tabla 1 Descripción de los pines del DLP-USB1232H.

Pin #	Descripción
1,10	GROUND
3,13,4,17,5,2,16,18	D7:D0 (Entrada y Salida) Bus de datos bidireccional.
6	PWREN (Salida). PWREN # = 0: Funcionamiento normal. PWREN # = 1: Modo de suspensión del USB o el dispositivo no se ha configurado.
7	SIWUA (Entrada) Permite enviar la señal de reactivación inmediata. Si el dispositivo está en modo de suspensión y la activación remota esta activada en la EEPROM, un pulso negativo en este pin inicializa una secuencia remota para despertar. Si el dispositivo está en funcionamiento normal, un pulso negativo en este pin carga los datos en el buffer de escritura para ser enviados a la PC en el próximo dato de entrada del USB sin tomar en cuanta cuantos bytes están en el buffer. Si este pin no va a ser utilizado se conecta directamente a VCC.
8	EXTVCC (Entrada) Usado para aplicar la alimentación (de 4.4 a 5.25 Volts) a el módulo. Debe conectarse a PORTVCC si el módulo va a ser alimentado por el puerto USB.
9	PORTVCC (Salida) Alimentación proveniente del puerto USB. Se conecta a EXTVCC si el módulo es alimentado por USB (configuración típica). La corriente máxima proporcionada es de 500mA.
11	RD# (Entrada) Cuando se encuentra en bajo, RD# lleva los 8 bits desde el estado de alta impedancia al byte actual en el buffer de la FIFO de recepción. Llevando a RD# a alto este retorna los pines de datos al estado de alta impedancia, y prepara el próximo byte.
12	WR (Entrada) Cuando pasa de un estado lógico alto a uno bajo, WR lee las líneas de 8 datos y escribe el byte en el buffer de la FIFO de transmisión.
14	TXE# (Salida) Cuando está en alto, la FIFO de transmisión está llena u ocupada guardando el último byte escrito. No intente escribir datos al buffer de transmisión cuando TXE# está en alto.
15	RXF# (Salida) Cuando está en bajo, al menos un byte está presente en el buffer de la FIFO de recepción y está listo para ser leído con RD#. RXF# está en alto cuando el buffer de recepción está vacío.

4.1.1.3 Sistema de visualización de resultados

El sistema de visualización de resultados está compuesto por el dispositivo VmodTFT proporcionado por la compañía Digilent Inc., el cual permite conectar una pantalla táctil LCD a

cualquier tarjeta de desarrollo que cuente con un puerto VHDCI. Las características que proporciona son:

- Este dispositivo está basado en el módulo LCD AT043TN24 V.7, el cual es una pantalla de cristal líquido de transistores de película fina (TFT LCD, *Thin Film Transistor-Liquid Crystal Display*) de 4.3 pulgadas con una película táctil resistiva.
- Cuenta con una resolución de 480 × 272 pixeles con una profundidad de color de 24 bits en formato RGB.
- Proporciona una frecuencia máxima de refresco de 80 cuadros por segundo.
- La luz de fondo es generada mediante un sistema de LEDs que pueden ser controlados por una señal PWM (*Pulse With Modulation*).
- La pantalla táctil resistiva tiene un controlador analógico AD7873 con una frecuencia máxima de muestreo de toque de 125 KSPS.
- Cuenta con un puerto VHDCI hembra de 68 pines.

En este trabajo sólo se considera este dispositivo para la visualización de los resultados en tiempo real, por lo tanto no se hace uso de la funcionalidad táctil que éste provee. Sin embargo, el usuario puede hacer uso de ésta, si la aplicación así lo requiere sin hacer modificaciones en el hardware. La descripción de las señales que intervienen para el control de este módulo se muestra en la Tabla 2.

Tabla 2 Descripción de las señales que intervienen en el control del LCD.

Símbolo	Descripción
TFT_R[7:0]	(Entrada) Componente Rojo del pixel en el formato de color RGB.
TFT_G[7:0]	(Entrada) Componente Azul del pixel en el formato de color RGB.
TFT_B[7:0]	(Entrada) Componente Verde del pixel en el formato de color RGB.
DISP	(Entrada) Encendido y apagado del LCD.
CLK	(Entrada) Señal de reloj que delimita la frecuencia del envío de los pixeles.
DE	(Entrada) Señal de sincronismo horizontal. Cuando está en un nivel lógico alto indica que los pixeles se encuentran en la zona visible del LCD.
TFT_EN	(Entrada) Habilita la alimentación de energía.
EN	(Entrada) Señal PWM para ajustar el brillo de fondo.

4.1.1.4 Sistema de memoria

Un elemento clave en los sistemas de procesamiento de video es el uso de sistemas de memoria de alta velocidad, el cual tiene como objetivo servir como un buffer temporal de almacenamiento donde es posible respaldar cuadros de video capturados y los resultados obtenidos del procesamiento. Para este propósito se ha elegido la memoria SDRAM DDR2 MT4HTF3264HY-667D3. Cuyas características principales se enlistan a continuación:

- Conector de 200 pines SODIMM (S*mall outline dual in line memory module*) compatible con la tarjeta de desarrollo Digilent Genesys.
- Transferencia rápida de datos de hasta 5.3 GB/s.
- Capacidad de 256 MB de almacenamiento, organizado en 32 M × 64 bits.
- Operaciones de lectura y escritura en ráfagas programable de cuatro u ocho localidades.
- Latencia de lectura programable (CL= 3, 4 o 5 ciclos de reloj).

En la Tabla 3 se muestra la descripción de las señales que intervienen en el control de la memoria.

Tabla 3 Descripción de los puertos de la memoria DDR2.

Símbolo	Descripción
A[12:0]	Address Input (Entrada) **P**rovee la dirección de la fila y la columna a la cual se quiere tener acceso. El bit A[10] determina si el proceso de lectura o escritura se realizara con precarga.
BA[1:0]	Bank address Input (Entrada) Define el banco al cual se le va a aplicar el comando de activación, de precarga, de lectura o el de escritura.
CK, CK#	Clock (Entrada) Entrada de reloj Diferencial. Todas las señales de control, comandos y direcciones son muestreadas en el cruce del flanco positivo de CK y el flanco negativo de CK#.
CKE#	Clock enable (Entrada) Permite la habilitación o des habilitación de los relojes de entrada.
DM[15:0]	Data Mask (Entrada) Mascara para escrituras.
ODT	On Die Termination (Entrada) Activación o desactivación de las resistencias terminales internas. Cuando es activada en operación normal ODT solo es aplicado a los siguientes pines: DQ, DQS, DQS#, DM y CB. La señal ODT es ignorada si es desactivada por el comando de carga de modo (LOAD MODE).
RAS,CAS,WE	Command Input (Entrada) Señales de control que definen el comando.
CS	Chip Select (Entrada) Habilita el decodificador de comandos. Activo en nivel bajo.
RESET	Reset (Entrada) Señal de reinicio de la memoria activa en un nivel lógico bajo.
CB	Check bits (Entrada/ Salida) Usada para la detección y corrección de errores.
DQ[63:0]	Data I/O (Entrada/Salida) Bus de datos bidireccionales
DQS, DQS#	Data Strobe (Entrada /Salida) Usado para detectar la presencia de nuevos datos en el bus. Es salida en las operaciones de lectura y entrada en las de escritura. Está alineada al flanco con los datos para las lecturas y alineada en el centro para las escrituras.
VDD, VDDQ	Power supply (Entrada) Alimentación de 1.8 volts.

En general las DDR2 SDRAM (*Double Data Rate Two Synchronous Dynamic Random Access Memory*) son memorias síncronas que se caracterizan porque permiten leer y escribir datos cuatro veces por cada ciclo de reloj, de modo que trabajan al cuádruple de la velocidad del bus sin necesidad de aumentar la frecuencia de reloj. Internamente están configuradas como un conjunto de bancos que a su vez están divididos en filas y columnas, de tal forma que para direccionar una posición en memoria es necesario especificar el banco, la fila y la columna. Por

otra parte los bancos son módulos DRAM, por lo que se trata de memorias dinámicas y por lo tanto es necesario realizar una actualización de su contenido cada cierto tiempo evitando así que se pierda la información.

4.1.1.5 Sistema de captura de video

La herramienta propuesta provee un sistema de adquisición de video estéreo basado en dos cámaras de imagen digital MT9D112 de 2 megapixeles, manufacturadas por la compañía Aptina. Estas cámaras se encuentran empotradas en el módulo Digilent VmodCAM, este módulo integra toda la circuitería necesaria para alimentar y configurar las cámaras proporcionando una interfaz de conexión mediante un conector VHDCI hembra de 68 pines (Las conexiones con las cámaras se describen en la

Tabla **4**). Las principales características que este módulo provee son:

- Dos cámaras de imagen digital CMOS Aptina MT9D112 de 2 mega pixeles cada una las cuales son completamente independientes entre sí.
- Una resolución máxima de 1600×1200 pixeles con una frecuencia máxima de 15 cuadros por segundo.
- 63 mm de espacio entre las cámaras (adecuado para visión estéreo).
- Cada cámara integra un procesador de flujo de imagen (IFP *Image Flow Processor*).
- Profundidad de color de 10 bits.
- Frecuencia máxima de operación de 80MHz.
- Un bus de control I^2C, independiente para cada cámara.
- Formatos de salida seleccionables: 565RGB, 444RGB, 555RGB y YCrCb.
- Exposición automática, ganancia y balance de blancos.
- Algoritmos de corrección de imagen.
- Permite el escalado de imagen.
- Memoria intermedia con salida mediante una FIFO para la transferencia de datos con una frecuencia constante.
- Bus paralelo de datos de 8 bits para transferencia de video.
- Cada cámara cuenta con un PLL integrado que permite manejar varias frecuencias de datos.
- Un conector hembra VHDCI de 68 pines.
- Las dos cámaras se pueden controlar independientemente y pueden adquirir por separado dos imágenes de forma simultánea.

Tabla 4 Conexiones que provee el módulo Digilent VmodCam.

Símbolo	Descripción
VDD_EN	(Entrada) Habilita la alimentación de las dos cámaras.
CAM1_RESET	(Entrada) señal de reinicio para la cámara 1.
CAM1_MCLK	(Entrada) señal de reloj principal para el funcionamiento de la cámara 1.
CAM1_PWDN	(Entrada) Señal que permite apagar la cámara 1, esta señal es activa en un nivel lógico alto.
CAM1_SCLK	(Entrada) Señal de reloj para la transferencia serial de registros de control de la cámara 1.
CAM1_SDA	(Entrada/Salida) Señal de datos seriales para la cámara 1.
CAM1_LV	(Salida) Esta señal indica que la salida en el bus paralelo de la cámara 1 corresponde a una línea valida de pixeles.
CAM1_FV	(Salida) Esta señal indica que la cámara 1 está transfiriendo una imagen.
CAM1_PCLK	(Salida) Señal de reloj con la cual están alineados los pixeles de salida de la cámara 1
CAM1_D[7:0]	(Salida) Bus de datos paralelo para la transferencia de pixeles por la cámara 1.
CAM2_RESET	(Entrada) Señal de reinicio para la cámara 2.
CAM2_MCLK	(Entrada) Señal de reloj principal para el funcionamiento de la cámara 2.
CAM2_PWDN	(Entrada) Señal que permite apagar la cámara 2, esta señal es activa en un nivel lógico alto
CAM2_SCLK	(Entrada) Señal de reloj para la transferencia serial de registros de control de la cámara 2.
CAM2_SDA	(Entrada/Salida) Señal de datos seriales para la cámara 2.
CAM2_LV	(Salida) Esta señal indica que la salida en el bus paralelo de la cámara 2 corresponde a una línea valida de pixeles.
CAM2_FV	(Salida) Esta señal indica que la cámara 2 está transfiriendo una imagen.
CAM2_PCLK	(Salida) Señal de reloj con la cual están alineados los pixeles de salida de la cámara 2.
CAM2_D[7:0]	(Salida) Bus de datos paralelo para la transferencia de pixeles por la cámara 2.

4.1.2 Descripción de los componentes software

Los componentes software que integran el sistema son, por un lado, la interfaz gráfica de usuario para configurar y evaluar el comportamiento de cada una de las etapas del sistema y, por otra parte, el programa en lenguaje de descripción de hardware que configura al FPGA como el elemento central de procesamiento.

4.1.2.1 Interfaz gráfica de usuario

La interfaz de usuario tiene como objetivo fungir como elemento de enlace durante el intercambio de información entre el usuario y el SECPV. Se trata de un programa realizado mediante la herramienta Borland C++ Builder versión 6.0, usando una programación orientada a objetos. La programación orientada a objetos (POO) permite que el código sea reutilizable, modular y fácil de mantener. Para utilizar este paradigma correctamente es necesario pensar las cosas de una manera distinta, e implementar los programas en términos de objetos, propiedades y métodos. En la POO, primero se definen los objetos para luego enviarles mensajes solicitándoles que realicen sus métodos. El diseño modular de la POO reduce la complejidad,

facilita los cambios y produce como resultado una implementación más sencilla, permitiendo el desarrollo paralelo de las diferentes partes del sistema.

El diagrama general utilizado para el desarrollo de la interfaz gráfica de usuario se muestra mediante el diagrama de casos de uso (ver Figura 4.3), donde los actores que intervienen son el usuario y el sistema empotrado para la captura y procesamiento de video. Las funcionalidades que proporciona la interfaz gráfica de usuario son:

- La apertura y cierre del puerto USB de la PC, para establecer la comunicación con el sistema SECPV.
- Implementa métodos de envío de información a través del puerto USB previamente abierto hacia el SECPV.
- Implementa métodos de recepción de información proveniente del SECPV.
- Forma las tramas en la cual está codificada la información en el protocolo de comunicación con SECPV.
- Permite configurar los parámetros para el funcionamiento de las cámaras, el sistema de visualización de resultados y los algoritmos.

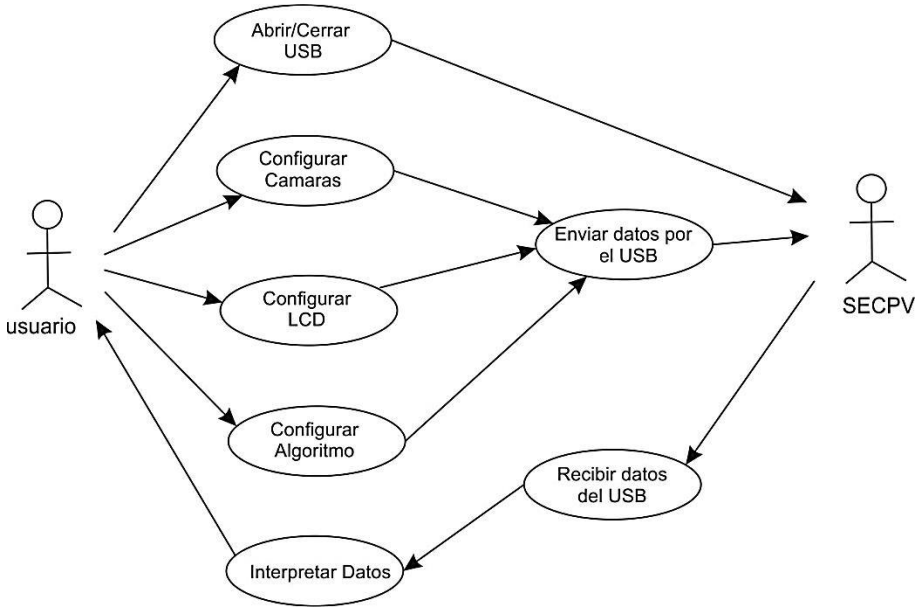

Figura 4.3 Diagrama de casos de uso de la interfaz de usuario.

Para los métodos relacionados con la gestión de la información a través del puerto USB se proporciona la clase *PuertoUSB* (ver Figura 4.4), esta clase proporciona métodos que hacen uso de las librerías DLL (D2XX) para el uso del protocolo USB mediante el DLP-USB1232H. Los métodos que proporciona son:

- *Abrir()*: Este método permite la apertura del puerto USB para establecer la comunicación con el SECPV.

- *Cerrar()*: Este método se encarga de cerrar el puerto USB con ello finaliza la comunicación con el SECPV.
- *Enviar()*: Permite el envío de datos a través del puerto USB hacia el SECPV, los datos a enviar deben estar alojados en el archivo Entrada.bin.
- *Recibir()*: Permite la recepción de datos provenientes del SECPV. Los datos recibidos son almacenados en el archivo Salida.bin.
- *Prueba()*: Ejecuta la prueba de comunicación entre el SECPV y la interfaz gráfica de usuario.

Figura 4.4 Clase PuertoUSB.

La captura de los parámetros de configuración para los sistemas de adquisición y visualización de video se realizan mediante los formularios mostrados en la Figura 4.5 y la Figura 4.6, respectivamente. En el caso de la configuración del sistema de adquisición de video, el formulario permite seleccionar entre los distintos formatos de video, que son soportados por las cámaras (Formato de color, resolución y efectos especiales de la cámara). El formulario también permite determinar con qué frecuencia se quieren recibir los datos de pixel, a través de configurar los valores con los que será configurado el controlador de reloj interno de las cámaras. Por otra parte el formulario de configuración del sistema de visualización de resultados, permite la activación o desactivación de la pantalla LCD con la que cuenta el SECPV, de igual forma, permite el envío de las características de formato con las que está cifrado el video a mostrar y la dirección del buffer donde está almacenado.

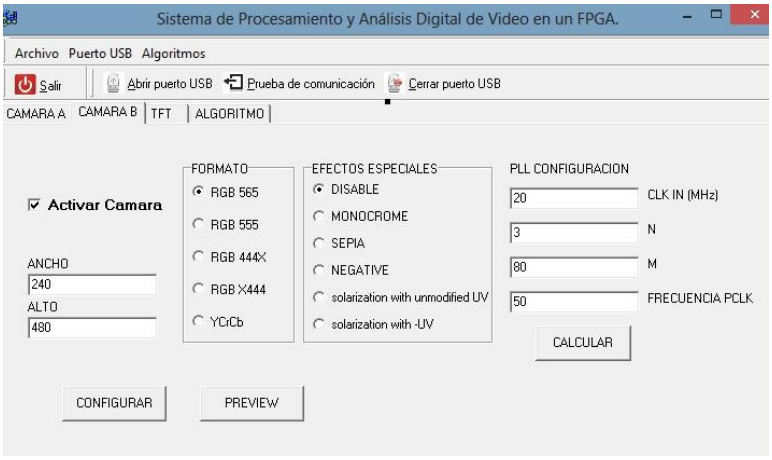

Figura 4.5 Formulario de configuración del sistema de adquisición de video.

Cabe destacar que la interfaz de usuario se entrega en un formato de código abierto (*Open source*), esto permite la modificación de la interfaz por el usuario, con la finalidad de poder ir agregando nuevas funcionalidades a ésta. En este sentido, el formulario para gestionar el algoritmo a evaluar, se deja a criterio del usuario. Para su desarrollo el usuario puede usar los métodos provistos en la clase *PuertoUSB* para enviar o recibir la información personalizada que le permita interactuar con su algoritmo siguiendo la estructura del protocolo de comunicación establecido para el SECPV.

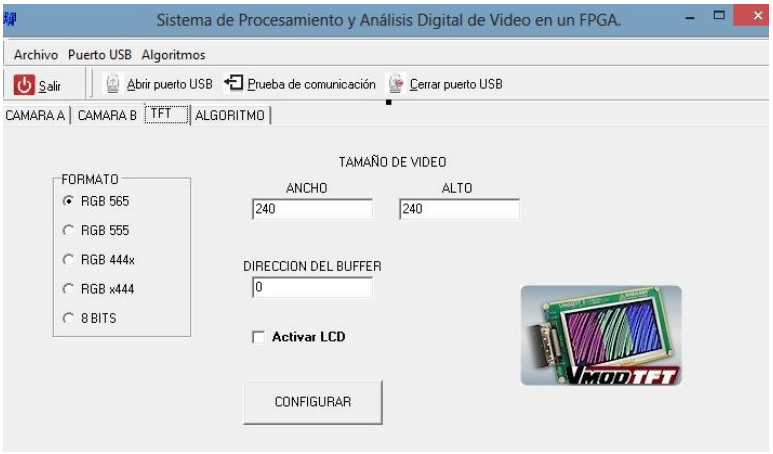

Figura 4.6 Formulario de configuración del sistema de visualización de resultados.

4.1.2.1.1 Protocolo de comunicación

El protocolo de comunicación forma parte del sub-sistema que permite el intercambio de información entre la interfaz de usuario y el SECPV, dicho protocolo está formado únicamente por una trama cuya estructura se adapta a cada una de las funciones que soporta la comunicación entre ambos elementos. La estructura de la trama está compuesta por cuatro campos (ver Figura 4.7). El campo ID determina el tipo de trama y a que modulo del SECPV está dirigida (consultar Tabla 5). El campo OP, en su bit 0 define la señal de habilitación para el modulo que la recibe, el bit 1 define si la trama es de actualización de configuración, el bit 2 indica si la trama de configuración está dirigida a un elemento secundario (este bit solo tiene sentido en el *administrador de captura de video*, el cual es usado para configurar el sub módulo de captura automática). El campo ND contiene el número de datos que se transmitirán. Por último el campo de datos DAT, contiene la información que se desea transferir, estos datos están codificados con diferentes tamaños de palabra; en el caso de la trama dirigida al *administrador de captura de video* su tamaño es de 40 bits, para el *administrador de algoritmos* de 24 bits y para el resto de 8 bits.

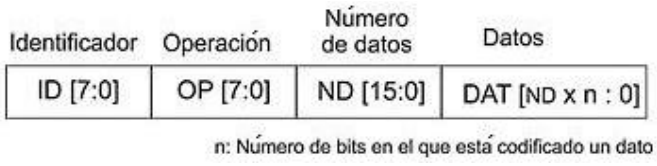

Figura 4.7 Estructura de la trama de comunicación.

En el caso de las tramas de configuración del sistema de adquisición de video, éstas se dividen en dos tipos de trama a partir del valor del bit 2 del campo OP. Cuando OP[2] = ‘1’, la trama está dirigida al elemento encargado de almacenar el video generado por la cámara, este recibe

su información a través de un solo dato de 40 bits, donde los primeros 16 bits más significativos indican el número de cuadros a almacenar y los últimos 24 bits la dirección del segmento en memoria en la cual serán alojados. En el caso que OP[2]= '0' y OP[1] ='1' indica que la trama entrante es de configuración de los parámetros de la cámara. Para la configuración de la cámara es necesaria la modificación de los registros internos de configuración de dicha cámara, mediante un protocolo serial I^2C. Para direccionar un registro en la cámara es necesario indicar la dirección de 8 bits del dispositivo, después proporcionar la dirección de 16 bits del registro y por último indicar el valor del dato para ese registro, el cual también es de16 bits. Estos tres parámetros se concatenan para formar los datos de 40 bits que integran la trama de configuración del sistema de adquisición de video.

Tabla 5 Tipo de tramas soportadas por el protocolo de comunicación.

ID	Tipo de trama
00000001	Trama de configuración del sistema de adquisición de video (Cámara A).
00000010	Trama de configuración del sistema de visualización de resultados.
00000011	Trama de configuración del administrador de algoritmos.
00000100	Trama de configuración del sistema de adquisición de video (Cámara B).
00000101	Trama de solicitud de prueba de comunicación.

Por otra parte para hacer una configuración correcta de las cámaras, los datos en la trama de configuración tienen que seguir un orden de envío donde los registros a enviar deben seguir la siguiente secuencia:

- Registro que permite identificar la cámara.
- Registros para el reinicio del micro controlador de la cámara.
- Registros para la configuración del PLL.
- Registro para habilitar la configuración del micro controlador.
- Registros que modifican los parámetros de imagen.
- Registros de efectos especiales de la cámara.
- Registro que indica la habilitación de la salida de video e inicio del proceso de transmisión de video por la cámara.

Ahora bien, en el caso de la trama de configuración del sistema de visualización de resultados, el campo de datos está integrado por ocho palabras de 8 bits cada una. La primera palabra indica el formato de color de los pixeles, las cuatro siguientes el ancho y alto del video, y las últimas tres palabras definen la localidad en memoria en que están alojados los datos de video a mostrar.

En la trama de solicitud de prueba de comunicación, el campo de datos está integrado por un solo dato de 8 bits con un valor en hexadecimal de 0xA5. Cuando el SECPV recibe esta trama responde regresando un dato cuyo valor en hexadecimal es de 0x5A para indicar que la comunicación es correcta.

Por último en la trama de configuración del administrador de algoritmos, los datos están compuestos por una palabra de 24 bits, donde los primeros 8 bits más significativos indican la dirección del registro a modificar y los 16 bits siguientes contienen la información respectiva para ese registro.

4.1.2.2 Descripción HDL del elemento central de procesamiento

El elemento central de procesamiento es la unidad principal de la herramienta propuesta, para su realización se utilizó la herramienta de diseño de sistemas digitales Xilinx ISE *Desing Suite* 14.5. Con esta herramienta se pueden crear, verificar, sintetizar e implementar diseños basados en un FPGA o CPLD. El editor que brinda esta herramienta para el diseño es el *Project Navigator*. Los diseños se pueden programar en lenguaje descriptor de hardware HDL (VHDL o Verilog) o bien, mediante esquemáticos y diagramas de estados. Una vez que se ha generado el diseño en el editor y verificado mediante simulaciones, este puede ser implementado pasando por las etapas de asignación de pines, síntesis, mapeo, ruteado, generación del programa (*Bitstream*) y por último la descarga del programa al FPGA para su validación final.

Para el modelado del elemento central de procesamiento se siguió la metodología descendente (*Top-Down).* Esta metodología parte de visualizar al sistema desde un nivel alto de abstracción e inicia la descripción en forma descendente, incrementando el nivel de detalle según sea necesario. El diseño inicial se divide en diferentes módulos, cada uno de los cuales se encuentra a su vez subdividido hasta llegar a elementos primarios. Partiendo de este enfoque, el esquema resultante en el nivel más alto de jerarquía que describe al elemento central de procesamiento es el mostrado en la Figura 4.8, en la cual se puede apreciar que la arquitectura está compuesta por seis módulos, los cuales son definidos a partir de la funcionalidad que realizan como:

- Administrador de comunicación.
- Administrador de reloj.
- Administrador de memoria.
- Administrador de visualización de resultados.
- Administrador de captura de video.
- Administrador de algoritmos.

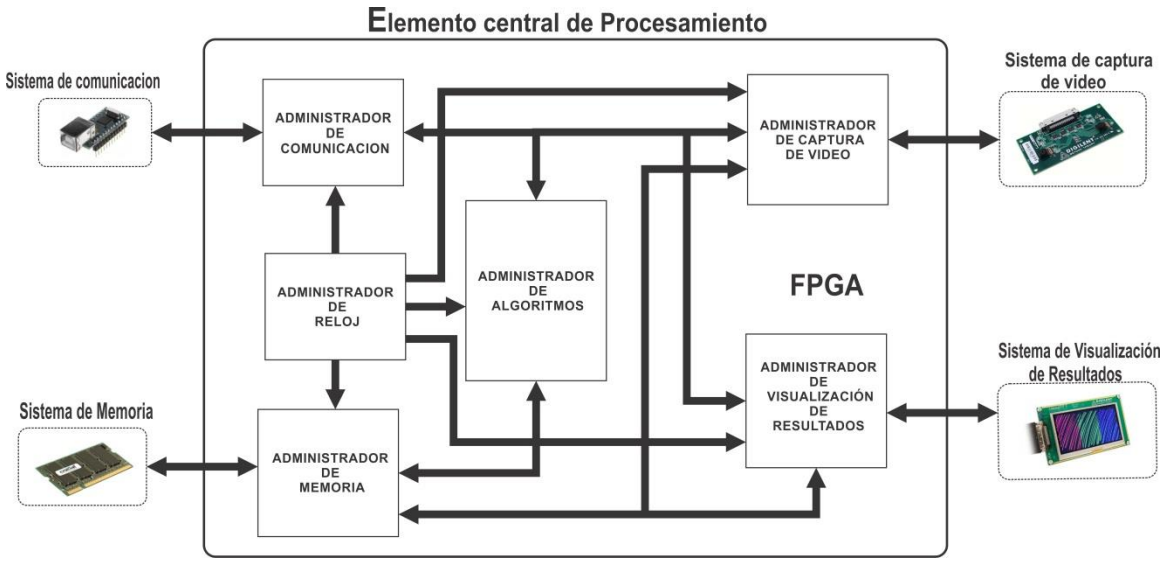

Figura 4.8 Arquitectura del elemento central de procesamiento del SECPV.

4.1.2.2.1 Administrador de comunicación

Este módulo es el encargado de brindar soporte en la comunicación entre la interfaz de usuario y los módulos que integran el elemento central de procesamiento. Sus principales funciones son la de generar las señales para el envío y recepción de datos a través del DLP-USB1232H y la de definir la dirección del flujo de datos. Para cumplir con su función, este módulo está compuesto internamente por dos módulos; el *controlador USB* y el *decodificador de trama* (ver Figura 4.9). La descripción de las señales de entrada y salida a este módulo se encuentran en la Tabla 6.

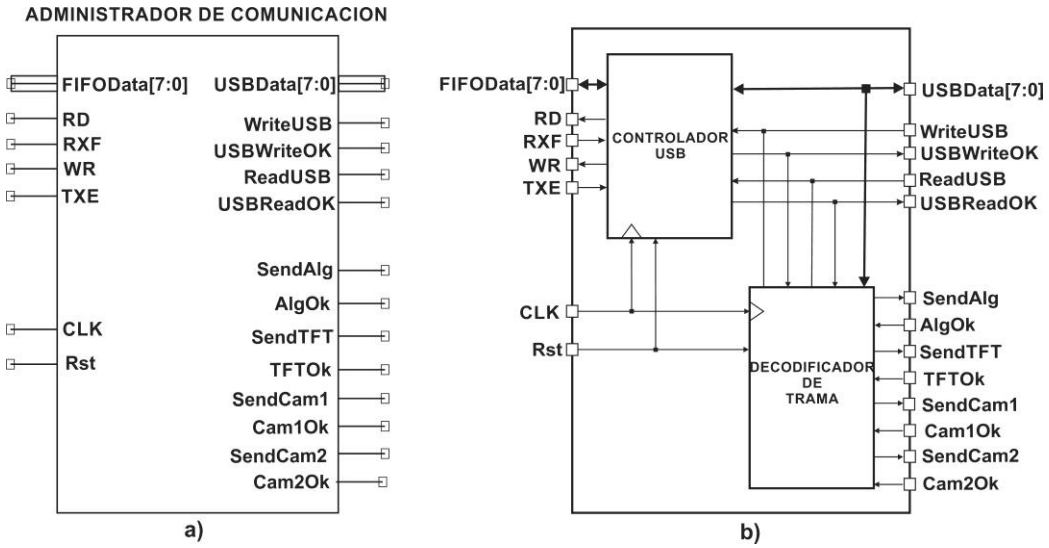

Figura 4.9 a) Símbolo del *administrador de comunicación*, b) Estructura interna del *administrador de comunicación*.

Tabla 6 Descripción de las entradas y salidas del módulo administrador de comunicación.

Señal	Descripción
CLK	(Entrada) Señal de reloj principal del sistema de 250 MHz.
Rst	(Entrada) Señal de reinicio global del sistema.
FIFOData[7:0]	(Entrada y Salida) Bus de datos bidireccional conectado al bus de datos D[7:0] del DLP-USB1232H.
RD	(Salida) Señal que permite leer un dato en la FIFO de recepción del DLP-USB1232H.
WR	(Salida) Señal que permite escribir un dato en la FIFO de transmisión del DLP-USB1232H.
TXE	(Entrada) Señal proveniente del DLP-USB1232H, cuando está en un nivel lógico alto indica que no se puede escribir en la FIFO de transmisión.
RXF	(Entrada) Señal proveniente del DLP-USB1232H, cuando está en un nivel lógico bajo indica que hay un dato en FIFO de recepción.
USBData[7:0]	(Entrada y Salida) Bus de datos bidireccional para la transmisión de datos entre los módulos del elemento central de procesamiento y el administrador del sistema de comunicación.
WriteUSB	(Entrada) Inicia el proceso de escritura de un dato al DLP-USB1232H.
USBWriteOk	(Salida) Indica que el proceso de escritura fue exitoso.
ReadUSB	(Entrada) Inicia lectura del dato proveniente del DLP-USB1232H.
USBReadOk	(Salida) Indica que hay un dato listo para ser leído.
SendAlg	(Salida) Le indica al módulo *administrador de algoritmos* que inicie su proceso de comunicación.
AlgOk	(Entrada) El módulo *administrador de algoritmos* pone en alto esta señal para indicar que finalizo su proceso de comunicación.
SendTFT	(Salida) Le indica al módulo *administrador de visualización de resultados* que inicie su proceso de comunicación.
TFTOk	(Entrada) Indica que el módulo *administrador de visualización de resultados* ha finalizado su proceso de comunicación.
SendCAM1	(Salida) Le indica al módulo *administrador de captura de video* que inicie el proceso de comunicación de la cámara 1.
CAM1Ok	(Entrada) Indica que el módulo *administrador de captura de video* finalizo el proceso de comunicación de la cámara 1.
SendCAM2	(Entrada) Le indica al módulo *administrador de captura de video* que inicie el proceso de comunicación de la cámara 2.
CAM2Ok	(Entrada) Le indica al módulo *administrador de captura de video* que inicie el proceso de comunicación de la cámara 2.

Para poder realizar una transferencia de información hacia la PC a través del controlador DLP-USB1232H la señal TXE# debe estar en un nivel lógico bajo para poder introducir los datos presentes en el bus de datos en la FIFO de transmisión mediante la señal WR siguiendo el diagrama de tiempos mostrado en la Figura 4.10, donde los tiempos que deben cumplir las señales se indican en la Tabla 7. Si la FIFO de transmisión está llena o se está realizando una

operación de escritura el dispositivo pondrá TXE# en un nivel lógico alto, no permitiendo la escritura de datos hasta que alguno de los datos de FIFO sea transferido por el USB hacia la PC.

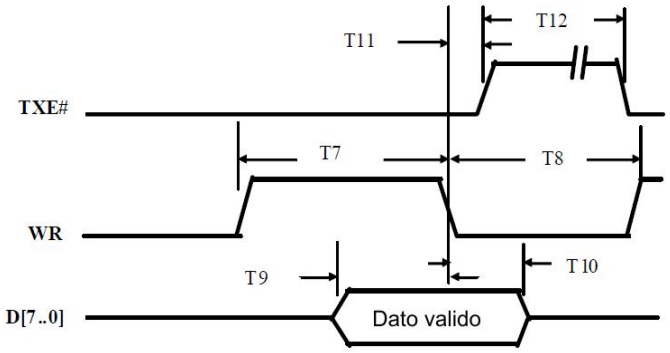

Figura 4.10 Diagrama de tiempos para el proceso de escritura del DLP-USB1232H.

Por otra parte, para la recepción de información proveniente de la PC mediante el DLP-USB1232H, se sigue el diagrama de tiempos de la Figura 4.11 Los tiempos de estas señales están presentes en la Tabla 7. Una vez que el DLP-USB1232H recibe un dato desde la PC, éste lo almacena en la FIFO de recepción e indica que hay un dato disponible para ser leído a través de la señal RXF#, por su parte el dato es leído en el bus de datos D[7:0] mediante el uso de la señal RD#.

Tabla 7 Tiempos para los procesos de lectura y escritura del DLP-USB1232H.

Tiempo	Descripción	Min	Max	Unidades
T1	Ancho de pulso activo para RD.	50	-	ns
T2	Tiempo entre habilitación y deshabilitación de RD.	T5+T6	-	ns
T3	Tiempo para un dato válido después de activar RD.	20	50	ns
T4	Mantenimiento del dato válido después de desactivar RD.	0	-	ns
T5	Tiempo para activación de RXF después de desactivar RD.	0	25	ns
T6	Tiempo de inactividad de RXF después de un ciclo de lectura.	33	67	ns
T7	Ancho de pulso activo para WR.	10	-	ns
T8	Tiempo entre desactivación y activación de WR.	50	-	ns
T9	Establecimiento del dato válido antes de desactivar WR.	20	-	ns
T10	Mantenimiento del dato válido después de desactivar WR.	10	-	ns
T11	Tiempo para activación de TXE después de desactivar WR.	10	25	ns
T12	Tiempo de inactividad de TXE después de un ciclo de escritura.	49	84	ns

En este sentido el modulo controlador USB lleva acabo estos procesos generando las señales de control en los tiempos estipulados en la Tabla 7, esto simplifica la tarea de envío y recepción de datos. Para la escritura de un dato hacia el DLP-USB1232H mediante el controlador USB, se coloca el dato a enviar en el bus de datos *USBData* y se activa la señal *WriteUSB*, la señal *USBWriteOk* indica que el proceso ha finalizado y fue exitoso. En el caso de la lectura, cuando la señal *USBReadOk* está activa, indica que el dato presente en el bus de datos *USBData* es un dato válido proveniente de la interfaz de usuario. Para indicar que este dato ya fue leído y

finalice el proceso de lectura, se activa señal *ReadUSB* durante 20ns, de esta manera se libera el bus de datos y se puede hacer otra operación de lectura o escritura.

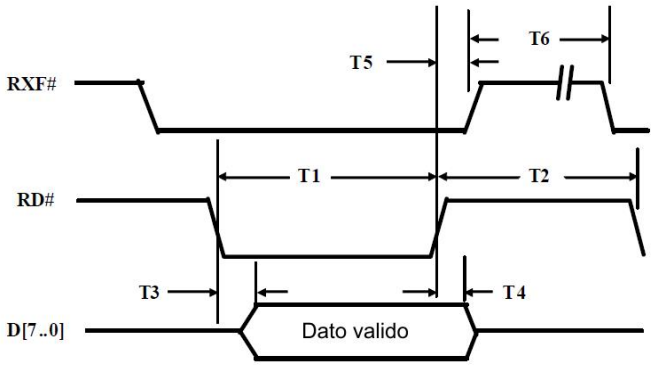

Figura 4.11 Diagrama de tiempos para el proceso de lectura del DLP-USB1232H.

Por otra parte el módulo *decodificador de trama* es el encargado de la interpretación del campo ID de la trama proveniente de la interfaz de usuario. El funcionamiento de este módulo está representado en la máquina de estados de la Figura 4.12. Donde el proceso de decodificación inicia cuando llega el primer dato de la trama de comunicación (campo ID) al módulo decodificador, dependiendo de la información que este dato contenga, la máquina de estados se dirige a un estado de espera, donde se habilita la bandera correspondiente al módulo al que va dirigido la información para que éste inicie el proceso de recepción de los datos restantes de la trama. Cuando el modulo que está recibiendo la información termina su proceso de comunicación éste se lo notifica al decodificador mediante la activación de una bandera, así, el decodificador vuelve a su estado inicial y está listo para atender otra solicitud de comunicación hecha por el usuario.

Además en este módulo también se integra la función de prueba de comunicación, que permite validar que la comunicación entre el SECPV y la interfaz de usuario se ha establecido correctamente.

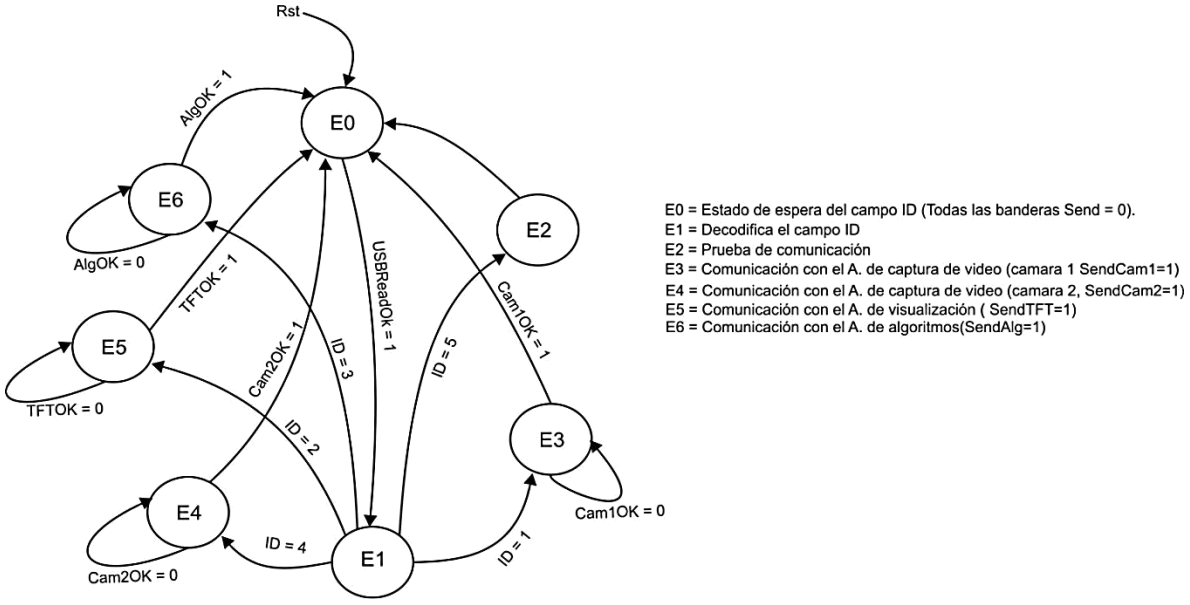

Figura 4.12 Maquina de estados del módulo *decodificador de trama.*

4.1.2.2.2 Administrador de reloj

Este módulo (Figura 4.13) hace uso de una primitiva PLL para la generación de las diferentes frecuencias que son necesarias para el funcionamiento del SECPV (en la Tabla 8 se muestra la descripción de las frecuencias sintetizadas). Un PLL (*Phase Locked Loop*) permite la generación de señales a frecuencias predeterminadas y agregar desfasamientos según la configuración de sus registros. El mismo es un circuito analógico, pero las entradas y las salidas son digitales. Para configurar las salidas del PLL se debe de realizar una multiplicación de la frecuencia de referencia y luego dividirla, para lo cual se tienen seis distintas salidas digitales.

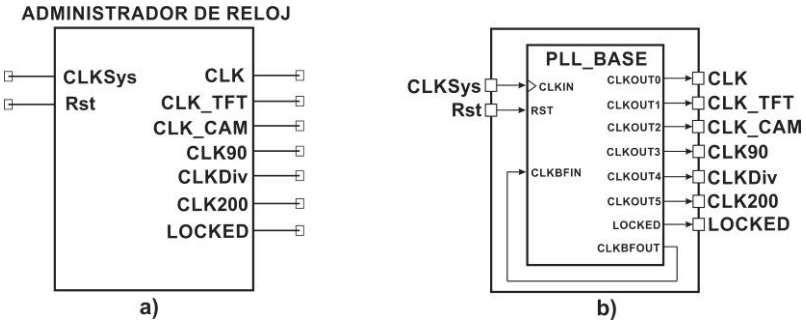

Figura 4.13 a) Símbolo del *administrador de reloj*, b) Estructura interna.

Tabla 8 Descripción de las señales de entrada y salida del *administrador de reloj*.

Señal	Descripción
CLKSys	(Entrada) Señal de reloj externa provista por la tarjeta de desarrollo.
Rst	(Entrada) Señal de reinicio.
CLK	(Salida) (250MHz) Señal de reloj principal para el funcionamiento síncrono del sistema.
CLK_TFT	(Salida) (8 MHz) Señal de reloj necesaria para el funcionamiento del a*dministrador de visualización de resultados.*
CLK_CAM	(Salida) (20 MHz) Señal de reloj necesaria para el funcionamiento del *administrador de captura de video.*
CLK90	(Salida) (250 MHz) Señal de reloj desfasada 90 grados con respecto a la señal de reloj principal, esta señal es necesaria para el funcionamiento del *administrador de memoria.*
CLKDiv	(Salida) (125 MHz) Señal de reloj cuya frecuencia es la mitad de la señal de reloj principal necesaria para el funcionamiento del *administrador de memoria.*
CLK200	(Salida) (200 MHz) Señal de reloj necesaria para el funcionamiento del *administrador de memoria.*
LOCKED	(Salida) Señal que indica que las señales de reloj están en fase y frecuencias deseadas.

El uso de la primitiva del PLL consiste en la configuración de los registros con los valores de las constantes con las cuales se realizará la multiplicación y división, los cuales no pueden ser cambiados durante su funcionamiento. La entrada de *CLKIN* es la entrada del reloj de referencia mientras que *CLKFBIN* se conecta a la salida de un reloj con el cual se está buscando la sincronía. Las seis salidas *CLKOUTX* permiten configurar los relojes a distintas frecuencias y fases simultáneamente. La salida *CLKBFOUT* permite tomar una señal resultante de referencia para mejorar la sincronización de la señal, de modo que es recomendado conectarla a

CLKFBIN. La señal *LOCKED* presenta un estado alto cuando logra la alineación de fases y frecuencias deseadas.

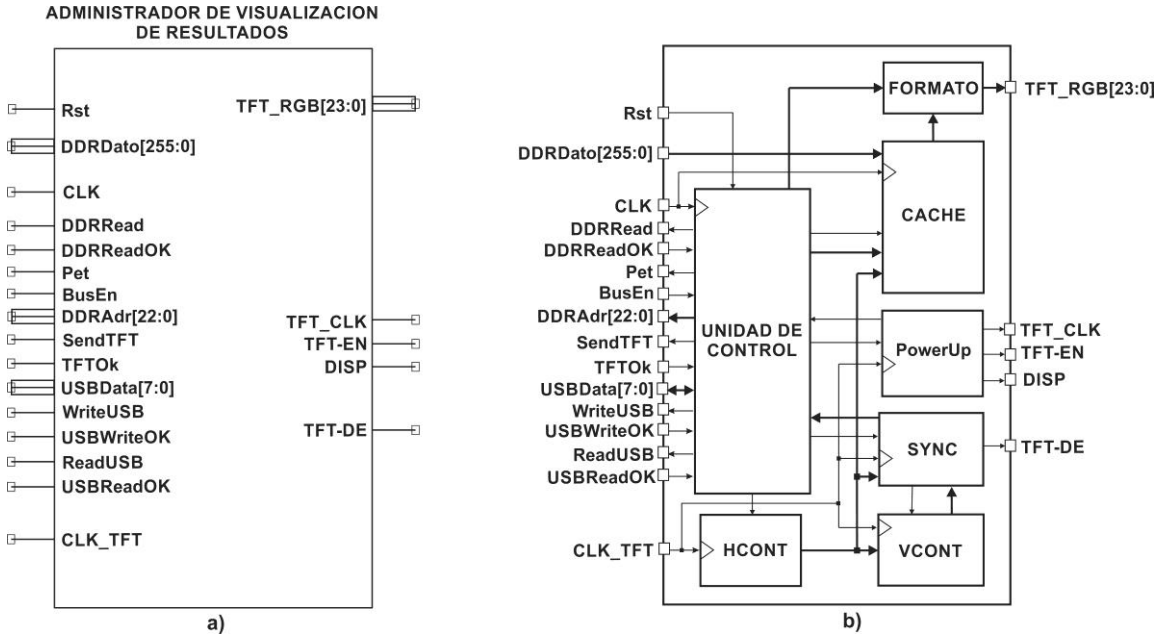

Figura 4.14 a) Símbolo del *administrador de visualización de resultados*, b) Estructura interna.

4.1.2.2.3 Administrador de visualización de resultados

En el módulo *administrador de visualización de resultados*, se implementa la lógica necesaria para la proyección del video procesado en el SECPV, en el LCD TFT que éste contiene. Su estructura interna se puede ver en la Figura 4.14, y la descripción de sus señales de entrada y salida en la Tabla 9. La *Unidad de control* recibe la configuración a través del bus de datos (*USBData*) proveniente del *Administrador de comunicación*. Estos datos contienen: la señal de habilitación del módulo, la dirección del segmento de memoria en donde se encuentra alojado el video a mostrar, el tamaño del video (Ancho y alto) y el formato de color RGB en el cual están cifrados los pixeles del video.

Tabla 9 Descripción de las señales de entrada y salida del *administrador de visualización de resultados*.

Señal	Descripción
CLK	(Entrada) Señal de reloj principal del sistema de 250 MHz, para un funcionamiento síncrono con el *administrador de comunicación* y el *administrador de memoria*.
Rst	(Entrada) Señal de reinicio global del sistema.
CLK_TFT	(Entrada) Señal de reloj necesaria para la transmisión de datos hacia el LCD AT043TN24 V.7.
DDRDato[255:0]	(Entrada y Salida) Bus de datos bidireccional para lectura/escritura de datos a la memoria.
DDRAdr[22:0]	(Salida) Bus de direccionamiento a la memoria.
Pet	(Salida) Permite hacer la solicitud de acceso a la memoria.
BusEn	(Entrada) Indica que el acceso a la memoria ha sido concedido y se pueden hacer peticiones de lectura/escritura.

Tabla 9 Descripción de las señales de entrada y salida del *administrador de visualización de resultados* (cont.).

Señal	Descripción
DDRRead	(Salida) Señal que indica una petición de lectura a una localidad en memoria direccionada por DDRAdr.
DDRReadOk	(Entrada) Indica que los datos han sido leidos de la memoria y están disponibles en el bus de datos DDRDato.
USBData[7:0]	(Entrada y Salida) Bus de datos bidireccional para la lectura/escritura de datos con el USB.
WriteUSB	(Salida) Inicia el proceso de escritura de un dato al USB.
USBWriteOk	(Entrada) Indica que el proceso de escritura fue exitoso.
ReadUSB	(Salida) Inicia lectura del dato proveniente del USB.
USBReadOk	(Entrada) Indica que hay un dato proveniente del USB, listo para ser leído.
SendTFT	(Entrada) Indica que hay una solicitud de comunicación que debe ser atendida.
TFTOk	(Salida) Le indica al *administrador de comunicación* que la petición de comunicación fue atendida y ha finalizado.
TFT_RGB[23:16]	(Salida) Componente Rojo del pixel.
TFT_RGB[15:8]	(Salida) Componente Azul del pixel.
TFT_RGB[7:0]	(Salida) Componente Verde del pixel.
DISP	(Salida) Encendido y apagado del LCD.
TFT_CLK	(Salida) Señal de reloj que delimita la frecuencia del envío de los pixeles.
TFT_DE	(Salida) Señal de sincronismo horizontal. Cuando está en un nivel lógico alto indica que los pixeles se encuentran en la zona visible del LCD.
TFT_EN	(Salida) Habilita la alimentación de energía para el LCD.

Después de ser recibida la configuración, la unidad de control envía una señal de activación a la unidad de *PowerUp*, esta unidad, se encarga de generar la secuencia de encendido que el fabricante recomienda para un funcionamiento correcto del LCD. Al finalizar esta secuencia, la unidad *PowerUp* envía una señal de finalizado a la unidad de control, con esto se inicia el proceso de envío de video hacia el LCD.

Para la transferencia de video al módulo LCD se sigue el diagrama de tiempos de la Figura 4.15, los tiempos de estas señales se pueden consultar en la Tabla 10. En él envío del video es necesario establecer un orden de barrido de la señal de video que se envía, esto consiste en enviar el primer elemento de la parte superior derecha y hacer un barrido progresivo hasta completar una línea de 480 pixeles, posteriormente se espera un periodo de 45 pixeles para generar el retorno a la siguiente línea. Los pixeles se envían por el bus de datos paralelo *TFT_RGB* el cual es síncrono con la señal de reloj *TFT_CLK* y la señal de sincronismo horizontal *TFT_DE*. Cada cuadro se compone de 272 líneas activas y 16 líneas inactivas para el borrado vertical.

Para generar la señal de sincronismo, la unidad *SYNC* monitorea el estado de los contadores (vertical y horizontal) para determinar, si activa, la señal *TFT_DE* a partir de las restricciones de la Tabla 10. El contador horizontal (*HCONT*), es un contador de 0 a 524, síncrono con la señal de reloj *TFT_CLK*, el cual indica el índice de la columna del LCD a la cual se está

teniendo acceso. El contador vertical (*VCONT*), es un contador de 0 a 287, que incrementa su valor cuando el contador horizontal llega a su nivel máximo de 524, este contador indica el índice de la fila del LCD que se está mostrando.

Para garantizar que no existan retrasos en los pixeles presentes en el bus de datos *TFT_RGB* con respecto a la señal de reloj *TFT_CLK*, el *administrador de visualización de resultados* integra un buffer de pixeles a través de la unidad *CACHE*, esta unidad está basada en una memoria embebida de doble puerto (este tipo de memorias permiten la lectura y escritura en puertos diferentes) con capacidad de almacenar una fila de 480 pixeles. El buffer es llenado por la unidad de control, la cual lee desde la memoria los valores de la fila a ser mostrados y los almacena en la unidad *CACHE*. El direccionamiento de los pixeles en la unidad *CACHE* se realiza mediante la salida del contador horizontal y son transferidos directamente al módulo *FORMATO*, en este módulo se transforma el formato de color del pixel a un formato RGB de 24bits compatible con la pantalla LCD. Estos datos transformados son finalmente transferidos al bus *TFT_RGB* para su visualización.

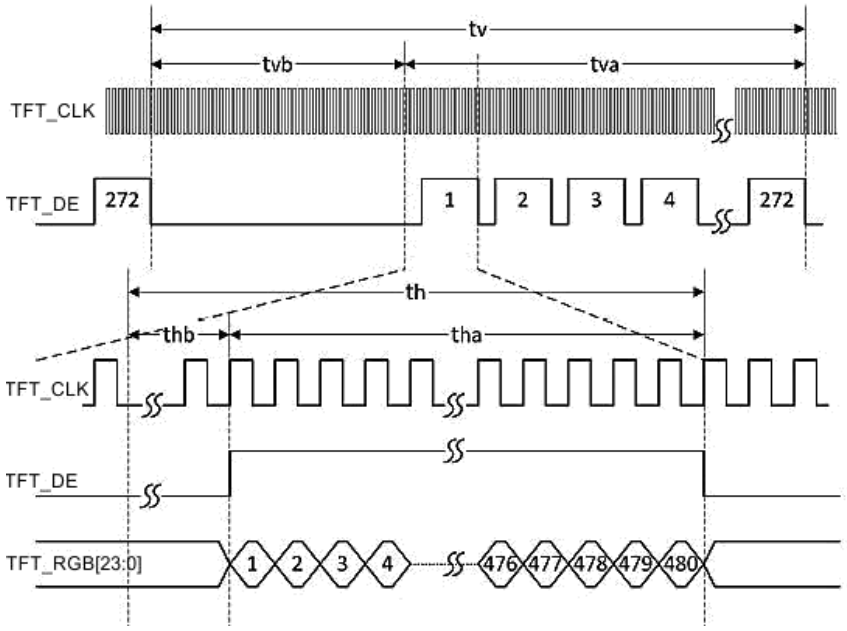

Figura 4.15 Diagrama de tiempos para el envío de video hacia el LCD AT043TN24 V.7.

Tabla 10 Parámetros para el envío de la señal de video.

Parámetro	Descripción	Valor	Unidades
CLK	Señal de reloj para el envío de los pixeles	6 a 12	MHz
tba	Periodo de líneas activas	272	Líneas
tbv	Periodo de líneas inactivas	16	Líneas
tha	Periodo de columnas activas	480	Pixeles
thv	Periodo de columnas inactivas	45	Pixeles

4.1.2.2.4 Administrador de captura de video

Este módulo es el encargado de gestionar el funcionamiento de cada una de las cámaras Aptina MT9D112 que integran al SECPV. Uno de los requisitos que debe cumplir el módulo *administrador de captura de video*, es la de generar la secuencia de encendido y reinicio, esta secuencia es importante para que la cámara opere de forma correcta. Por otra parte, para la configuración de las cámaras es necesario el envío de registros de configuración a través del protocolo serial I^2C al microcontrolador que estas cámaras poseen. Por último, después de que la cámara ha sido configurada, ésta envía la señal de video digital mediante su bus de datos paralelo de 8 bits sincronizado con las señales FV, LV y PCLK como se muestra en la Figura 4.16, debido a que solo se cuenta con un bus de datos paralelo de 8 bits, los pixeles son enviados de forma segmentada, en el caso del formato RGB un pixel se compone de dos datos recibidos y en el formato YCbCr está compuesto por cuatro datos recibidos.

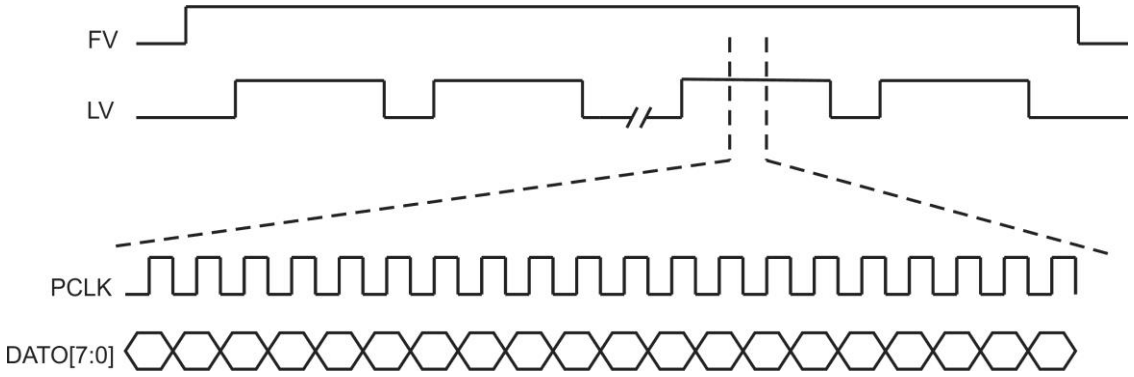

Figura 4.16 Estructura de la señal de video digital.

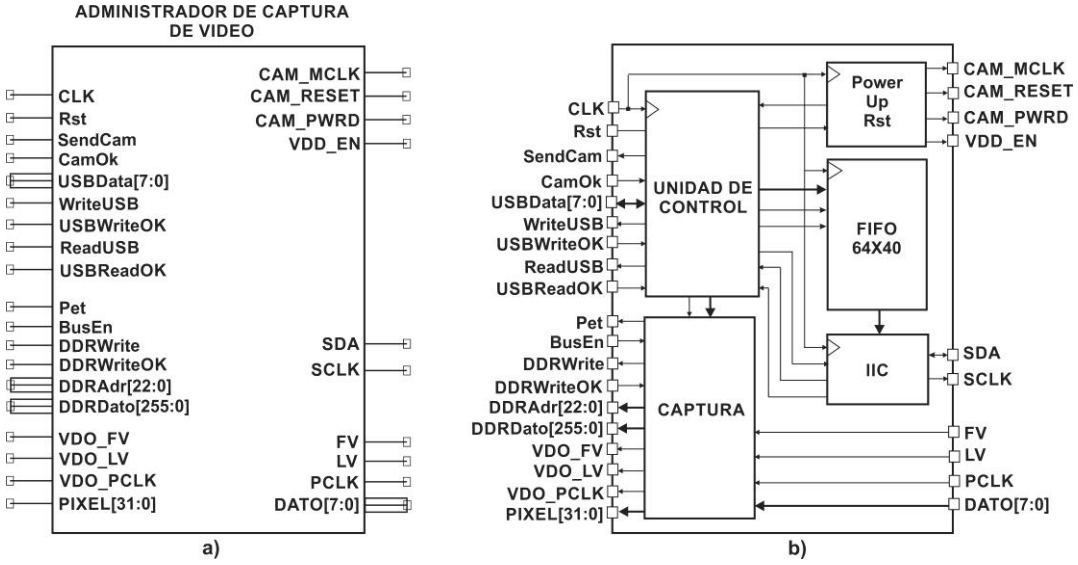

Figura 4.17 a) Símbolo del *administrador de captura de video*, b) Estructura interna.

Partiendo de los requisitos necesarios para la gestión de las cámaras la estructura interna del *administrador de captura de video* está compuesta por dos módulos independientes, donde cada uno se encarga de la gestión de una cámara. La estructura interna de cada módulo es la

mostrada en la Figura 4.17 y la descripción de las señales de entrada/salida se muestra en la Tabla 11.

Tabla 11 Descripción de las señales de entrada y salida del *administrador de captura de video.*

Señal	Descripción
CLK	(Entrada) Señal de reloj principal del sistema de 250 MHz, para un funcionamiento síncrono con el *administrador de comunicación* y el *administrador de memoria*.
Rst	(Entrada) Señal de reinicio global del sistema.
CLK_CAM	(Entrada) Señal de reloj que necesita la cámara para funcionar correctamente.
DDRDato[255:0]	(Entrada y Salida) Bus de datos bidireccional para lectura/escritura de datos a la memoria.
DDRAdr[22:0]	(Salida) Bus de direccionamiento a la memoria.
Pet	(Salida) Permite hacer la solicitud de acceso a la memoria.
BusEn	(Entrada) Indica que el acceso a la memoria ha sido concedido y se pueden hacer peticiones de lectura/escritura.
DDRRead	(Salida) Señal que indica una petición de lectura a una localidad en memoria direccionada por DDRAdr.
DDRReadOk	(Entrada) Indica que los datos han sido leídos de la memoria y están disponibles en el bus de datos DDRDato.
DDRWrite	(Salida) Señal que indica una petición de escritura de un dato en DDRDato a una localidad en memoria direccionada por DDRAdr.
DDRWriteOk	(Entrada) Indica que los datos han sido escritos en la memoria.
USBData[7:0]	(Entrada y Salida) Bus de datos bidireccional para la lectura/escritura de datos con el USB.
WriteUSB	(Salida) Inicia el proceso de escritura de un dato al USB.
USBWriteOk	(Entrada) Indica que el proceso de escritura fue exitoso.
ReadUSB	(Salida) Inicia lectura del dato proveniente del USB.
USBReadOk	(Entrada) Indica que hay un dato proveniente del USB, listo para ser leído.
SendCam	(Entrada) Indica que hay una solicitud de comunicación que debe ser atendida.
CamOk	(Salida) Le indica al *administrador de comunicación* que la petición de comunicación fue atendida y ha finalizado.
VDO_FV	(Salida) Señal de sincronismo vertical que indica que se está transmitiendo un cuadro de video.
VDO_LV	(Salida) Señal de sincronismo horizontal que indica que los datos en el bus de datos PIXEL corresponden a pixeles válidos.
VDO_PCLK	(Salida) Señal de reloj con la cual los pixeles están alineados en el flanco de subida.
PIXEL[31:0]	(Salida) Bus de datos que contienen la información de los pixeles que conforman al video.
VDD_EN	(Salida) Habilita la alimentación de las cámaras que conforman el sistema de captura de video.

Tabla 11 Descripción de las señales de entrada y salida del *administrador de captura de video* (cont.).

Señal	Descripción
VDD_EN	(Salida) Habilita la alimentación de las cámaras que conforman el sistema de captura de video.
CAM_RESET	(Salida) Señal de reinicio para la cámara.
CAM_MCLK	(Salida) Señal de reloj principal para el funcionamiento de la cámara.
CAM_PWDN	(Salida) Señal que permite apagar la cámara, esta señal es activa en un nivel lógico alto.
SCLK	(Salida) Señal de reloj para la transferencia serial de registros de control de la cámara.
SDA	(Entrada/Salida) Señal de datos seriales para la cámara.
LV	(Entrada) Esta señal indica que la salida en el bus paralelo de la cámara corresponde a una línea valida de pixeles.
FV	(Entrada) Esta señal indica que la cámara está transfiriendo una imagen.
PCLK	(Entrada) Señal de reloj con la cual están alineados los pixeles de salida de la cámara
DATO[7:0]	(Entrada) Bus de datos paralelo para la transferencia de pixeles por la cámara.

La unidad *Power Up Rst* se encarga de generar la secuencia de encendido y reinicio de la cámara, a partir de una petición hecha por la *unidad de control* interna del módulo. La secuencia de encendido consiste en esperar un tiempo de 75 µs para activar la señal de reloj *MCLK*, después de haber activado la alimentación de la cámara. El fabricante recomienda que siempre que se haga una secuencia de encendido, ésta sea seguida por una secuencia de reinicio, con la finalidad de que la máquina de estados interna del micro controlador de la cámara se encuentre en el estado inicial y de esta manera permita una configuración correcta de la cámara. La secuencia de reinicio (Figura 4.18) consiste básicamente en mandar a un nivel bajo la señal *RST* de la cámara por un periodo mínimo de 30 ciclos del reloj *MCLK*, y posteriormente hacer una espera de 6000 ciclos de reloj para poder iniciar la configuración de la cámara.

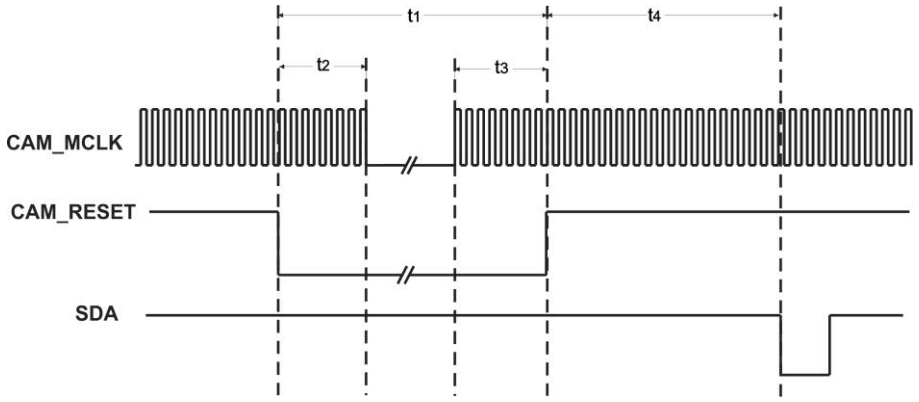

Figura 4.18 Secuencia de reinicio de la cámara Aptina MT9D112.

Por otra parte, la estructuración de los registros de configuración de la cámara es llevada a cabo en la interfaz de usuario, siendo el *administrador de captura de video* un puente de comunicación entre la interfaz de usuario y la cámara, de esta manera, la configuración se vuelve un proceso flexible para el usuario. Para la configuración de la cámara, la unidad de control recibe la información proveniente de la interfaz de usuario a través del *administrador de*

comunicación, siguiendo el protocolo interno de comunicación, donde la trama entrante es agrupada en paquetes de 40 bits e incrustada en la *FIFO* de configuración, posteriormente se hace el envío de estos registros, al micro controlador de la cámara, mediante el módulo *IIC*, el cual implementa un nodo maestro que hace uso del protocolo I^2C. Para la configuración de los registros mediante el módulo *IIC*, la unidad de control extrae un dato de la *FIFO* de configuración y le informa al módulo *IIC* que se desea enviar dicho dato mediante la activación de una bandera. El módulo *IIC* regresa una señal de error si el dato no ha podido ser enviado o una señal de finalizado si el proceso se llevó correctamente.

Para enviar y recibir información a través del protocolo I^2C es necesario especificar la dirección del nodo esclavo, el tipo de operación (lectura o escritura), la dirección del registro y el valor del dato para ese registro. En este sentido, las palabras de 40 bits alojadas en la *FIFO* están compuestas por estos campos, donde, los primeros siete bits más significativos [39:33] indican la dirección de la cámara, el bit 32 indica la operación (0=lectura, 1= escritura), los bits [31:16] determinan la dirección del registro objetivo y por último, los bits [15:0] indican el valor para ese registro. El proceso que sigue el módulo *IIC* para la escritura a un registro interno de la cámara mediante el protocolo I^2C, se compone de (ver Figura 4.19):

- La generación de la condición de inicio.
- Envío de la dirección del dispositivo esclavo más el bit de operación = ‘0’ que indica que es una escritura.
- Recepción del bit ACK.
- Envío de la parte alta de la dirección del registro a modificar.
- Recepción del bit ACK.
- Envío de la parte baja de la dirección del registro a modificar.
- Recepción del bit ACK.
- Envío de la parte alta del valor del registro direccionado.
- Recepción del bit ACK.
- Envío de la parte alta del valor del registro direccionado.
- Recepción del bit ACK.
- Generación de la condición de parada.

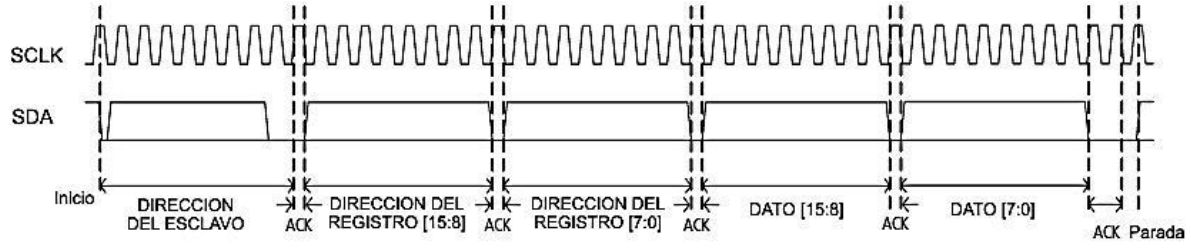

Figura 4.19 Proceso de escritura en el protocolo I^2C.

De igual forma el proceso de lectura sigue la secuencia de la Figura 4.20, la cual se compone de:

- La generación de la condición de inicio
- Envío de la dirección del dispositivo esclavo más el bit de operación = '0' que indica que se escribirá al bus.
- Recepción del bit ACK.
- Envío de la parte alta de la dirección del registro al que se quiere tener acceso.
- Recepción del bit ACK.
- Envío de la parte baja de la dirección del registro al que se quiere tener acceso.
- Recepción del bit ACK.
- Generación de la condición de inicio.
- Envío de la dirección del dispositivo esclavo más el bit de operación = '1' que indica que se leerá el valor que contiene el registro direccionado anteriormente.
- Recepción del bit ACK.
- Generación de la condición de inicio.
- Recepción de la parte alta del valor leído del registro direccionado.
- Envío del bit ACK.
- Recepción de la parte baja del valor leído del registro direccionado.
- Envío del bit NACK.
- Generación de la condición de parada.

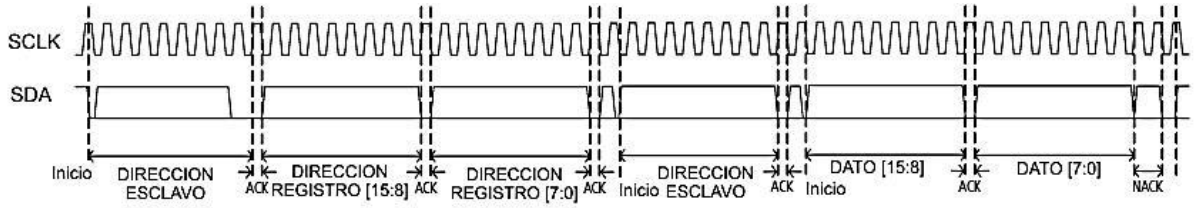

Figura 4.20 Proceso de lectura en el protocolo I^2C.

En el protocolo I^2C una transición en la señal *SDA* cuando *SCLK* está en un nivel alto no es permitido, salvo para indicar el inicio o finalización de un proceso de lectura o escritura al bus. La condición de inicio es generada por el módulo *IIC* para indicar que se efectuará un proceso de lectura o escritura, esto consiste en la transición de la señal *SDA* de un estado alto a uno bajo cuando la señal *SCLK* está en un estado alto. La condición de parada, indica que el módulo *IIC* ha finalizado el proceso de comunicación, éste consiste en la transición de *SDA* de un estado bajo a uno alto cuando *SCLK* está en alto. Para enviar los datos, éstos son colocados en la línea *SDA* bit a bit cuando la señal *SCLK* está en un nivel bajo y son leídos cuando *SCLK* está en alto.

El bit *ACK* le indica al emisor que los ocho bits enviados han sido recibidos por el receptor, cuando el módulo *IIC* está emitiendo datos hacia la cámara, éste recibe el bit *ACK* poniendo la línea *SDA* en alta impedancia, si la cámara pone en un nivel bajo la línea *SDA* indica que ha recibido el dato. De manera similar cuando el módulo *IIC* está trabajando como receptor pone la línea *SDA* en bajo cuando la cámara termina de enviar 8 bits y libera la línea poniéndola en alta impedancia.

Por otra parte el módulo de captura se encarga de concatenar los datos que integran un pixel para facilitar la interpretación de la señal de video, además genera una señal de reloj *VDO_PCLK* a partir de la señal de reloj de pixel *PCLK*, para que el valor del pixel coincida con el flanco de subida de esta señal de reloj. Otra de las funciones que lleva acabo este módulo es la de almacenar el video entrante en una localidad de la memoria a partir de una petición hecha por el usuario. El almacenamiento en memoria de los cuadros de video se puede ver como una fila circular, donde el número de cuadros de video y la dirección de memoria es indicado por el usuario. Principalmente esta función está destinada, para comprobar que la cámara está funcionando correctamente. Sin embargo el usuario puede hacer uso de esta característica para el funcionamiento de sus algoritmos.

4.1.2.2.5 Administrador del sistema de memoria.

En el módulo *administrador de memoria* (ver Figura 4.21 y consultar la Tabla 12) se implementa un protocolo de acceso al sistema de memoria, para que cualquier módulo que integra el elemento central de procesamiento tenga acceso a éste. Debido a que múltiples módulos requieren del acceso al bus de la memoria, el *administrador de memoria* integra un esquema de arbitraje de acceso al bus, el cual determina qué módulo y en qué momento puede tener acceso a la memoria, evitando colisiones en las señales de datos y de control.

El protocolo para escritura de datos en una localidad en memoria a través del *administrador de memoria* consiste en:

- Hacer la petición de acceso al bus mediante la activación de la bandera *Pet* = ‘1’.
- Esperar hasta que el árbitro de la concesión del bus mediante la activación de la bandera *BusEn*= ‘1’.
- Colocar el dato y su correspondiente dirección en el bus de datos *DDRDato* y el bus de direcciones *DDRAdr*.
- Activar la bandera *DDRWrite = ‘1’* para iniciar el proceso de escritura.
- Cuando el proceso ha finalizado, el a*dministrador de memoria* lo indica poniendo la bandera a un nivel alto *DDRWriteOK= ‘1’.*
- Para finalizar el acceso a la memoria el módulo que hiso la petición debe desactivar su bandera *Pet*= ‘0’ para indicar que ha sido liberado el bus de memoria.

La lectura de datos en una localidad en memoria a través del *administrador de memoria* consiste en:

- Hacer la petición de acceso al bus mediante la activación de la bandera *Pet* = ‘1’.

- Esperar hasta que el árbitro de la concesión del bus mediante la activación de la bandera *BusEn*= ‘1’.
- Colocar la dirección del dato que se quiere leer en el bus de direcciones *DDRAdr*.
- Activar la bandera *DDRread = ‘1’* para iniciar el proceso de lectura.
- Cuando el dato leído se encuentra en el bus de datos el a*dministrador de memoria* lo indica poniendo la bandera *DDRReadOK= ‘1’.*
- Para finalizar el acceso a la memoria el módulo que hizo la petición debe desactivar su bandera *Pet*= ‘0’ para indicar que ha sido liberado el bus de memoria.

En el esquema interno del *administrador de memoria* (ver Figura 4.21) los módulos *ARBITRO, WRITE Y READ*, se encargan del protocolo de acceso entre los módulos del *elemento central de procesamiento* y el modulo *controlador de memoria*. El modulo *controlador de memoria* es el encargado de llevar el control del sistema de memoria DDR2. Debido a la complejidad en el diseño de este tipo de controladores se decidió implementar el *controlador de memoria* a partir del código generado por la herramienta MIG (*Memory Interface Generator*) proporcionada por Xilinx. Esta herramienta permite el diseño de controladores eficientes de memoria DDR2 de una manera simple, a partir de las características de la memoria que se esté empleando.

Tabla 12 Descripción de las señales de entrada y salida del *administrador de memoria*.

Señal	Descripción
CLK	(Entrada) Señal de reloj principal del sistema de 250 MHz, con la cual están sincronizados los procesos de escritura y lectura, al igual que el arbitraje del acceso.
Rst	(Entrada) Señal de reinicio global del sistema.
CLK90	(Entrada) Señal de reloj desfasada 90 grados con respecto a la señal de reloj principal, esta señal es necesaria para la generación de las señales de control en el controlador de memoria generado por el MIG.
CLKDiv	(Entrada) Señal de reloj cuya frecuencia es la mitad de la señal de reloj principal, esta señal es necesaria para la generación de las señales de control en el controlador de memoria generado por el MIG.
CLK200	(Entrada) Esta señal de reloj es necesaria para la generación de las señales de control en el controlador de memoria generado por el MIG.
LOCKED	(Entrada) Señal que indica que las señales de reloj están en fase y frecuencias deseadas.
DDRDato[255:0]	(Entrada/Salida) Bus de datos bidireccional para lectura/escritura de datos a la memoria.
DDRAdr[22:0]	(Entrada) Bus de direccionamiento a la memoria.
Pet[3:0]	(Entrada) Señales de petición de acceso a la memoria.
BusEn[3:0]	(Entrada) Señales de concesión del bus.
DDRRead	(Entrada) Señal que indica una petición de lectura a una localidad en memoria direccionada por DDRAdr.
DDRReadOk	(Salida) Indica que los datos han sido leidos de la memoria y están disponibles en el bus de datos DDRDato.

Tabla 12 Descripción de las señales de entrada y salida del *administrador de memoria* (cont.).

Señal	Descripción
DDRWrite	(Entrada) Señal que indica una petición de escritura de un dato en DDRDato a una localidad en memoria direccionada por DDRAdr.
DDRWriteOk	(Salida) Indica que los datos han sido escritos en la memoria.
DDR2_ADR[12:0]	(Salida) provee la dirección de la fila y la columna a la cual se quiere tener acceso. El bit [10] determina si el proceso de lectura o escritura se realizara con precarga.
DDR2_BA	(Salida) Define el banco al cual se le va a aplicar el comando de activación, de precarga, de lectura o el de escritura.
DDR2_CK, DDR2_ CKN	(Salida) Reloj Diferencial. Todas las señales de control, comandos y direcciones son muestreadas en el cruce del flanco positivo de DDR2_CK y el flanco negativo de DDR2_CKN.
DDR2_CKE	(Salida) Permite la habilitación o deshabilitación de los relojes de entrada.
DDR2_DMASK[7:0]	(Salida) Máscara para escritura de datos a la memoria
DDR2_ODT	(Salida) Activación o desactivación de las resistencias terminales internas. Cuando es activada en operación normal ODT solo es aplicado a los siguientes pines: DQ, DQS, DQS#, DM y CB. La señal ODT es ignorada si es desactivada por el comando de carga de modo (LOAD MODE).
DDR2_RAS, DDR2_CAS, DDR2_WE	(Salida) Señales de control que definen el comando a enviar a la memoria.
DDR2_CS	(Salida) Habilita el decodificador de comandos. Activo en nivel bajo.
DDR2_RST	(Salida) Señal de reinicio de la memoria activa en un nivel lógico bajo.
DDR2_DQ[63:0]	(Entrada/Salida) Bus de datos bidireccionales.
DDR2_DQS[7:0], DDR2_DQSN[7:0]	(Entrada /Salida) Usados para detectar la presencia de nuevos datos en el bus. Es salida en las operaciones de lectura y entrada en las de escritura. Está alineada al flanco con los datos para las lecturas y alineada en el centro para las escrituras.

El esquema general del controlador generado por el MIG integra la capa física, la cual se encarga del acondicionamiento de las señales para la transferencia de datos e instrucciones a la memoria. La capa física incluye los bloques de entrada / salida y otras primitivas utilizadas para leer y escribir las señales de datos hacia y desde la memoria. Este módulo también incluye los elementos *IODELAY* del FPGA. Estos elementos *IODELAY* se utilizan para retrasar las señales de datos, con el fin de capturar los datos válidos en el proceso de lectura. En todas las señales de entrada y salida se compensan los retrasos de enrutamiento dentro de la FPGA. Por otra parte, la inicialización de la memoria es llevada a cabo en una secuencia predefinida de acuerdo con el estándar JEDEC para DDR2 SDRAM.

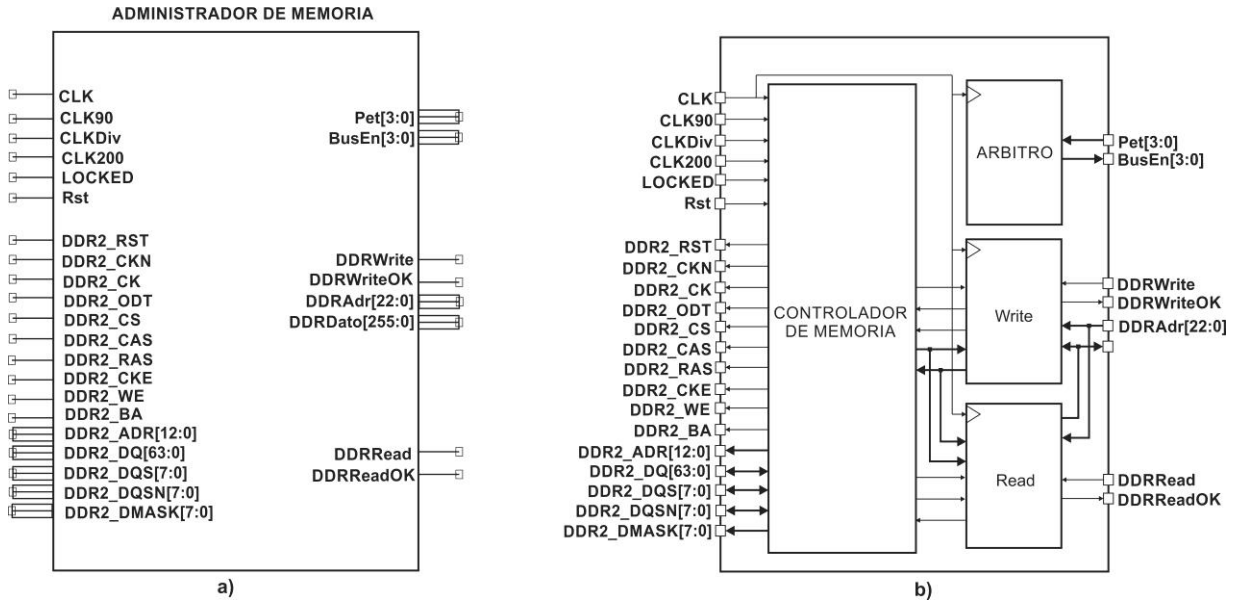

Figura 4.21 a) Símbolo del *administrador de memoria*, *b*) Estructura interna.

El controlador genera todas las señales necesarias para el reinicio de la memoria y todas las señales de control requeridas para la interfaz con la memoria DDR2 y la interfaz con el usuario. Durante el funcionamiento normal, este módulo cambia las señales de dirección y de control de la memoria, al igual que se encarga de generar los comandos necesarios para los procesos de lectura, escritura y refresco de la memoria.

La interfaz con el usuario que el MIG provee está compuesto por tres FIFOs, una para albergar los datos a escribir en la memoria, otra para almacenar los datos recibidos en la lectura y una para las direcciones y comandos de escritura o lectura. Durante una operación de escritura, el módulo de control lee la dirección y el comando, entonces los datos almacenados en la FIFO que contiene los datos a escribir, son leídos y transferidos a la capa física para ser enviados a la memoria. Del mismo modo, durante una operación de lectura, los datos de la memoria son extraídos de la dirección contenida en la FIFO de direcciones, la capa física se encarga de leer estos datos y los escribe en la FIFO destinada para la lectura. En resumen los procesos de lectura o escritura, se convierten en accesos a las FIFOs del controlador proporcionado por el MIG.

4.1.2.2.6 Administrador de algoritmos

El modulo *administrador de algoritmos* (ver Figura 4.22) tiene como objetivo albergar los algoritmos desarrollados por el usuario, este módulo está compuesto por una unidad de control y un archivo de registros. Además proporciona conexiones directas con el *administrador de memoria*, el *administrador de comunicación* y el *administrador de adquisición de video* (consultar la Tabla 13), esto con la finalidad de proporcionar todos los recursos necesarios para la implementación de un algoritmo de procesamiento de video personalizado sin restringir arbitrariamente la estructura de éste.

La función de la unidad de control es la de gestionar el funcionamiento del algoritmo a evaluar, esto mediante la modificación de los parámetros de configuración en el archivo de registros. El archivo de registros está compuesto por 16 registros de 16 bits cada uno, los cuales pueden ser utilizados a criterio del usuario. Es decir, los parámetros de configuración y su ubicación en el

archivo de registros lo determina el usuario, así como también su interpretación de éstos. Estos parámetros son enviados desde la PC por la interfaz de usuario y recibidos por la unidad de control a través del bus de datos del *administrador de comunicación.*

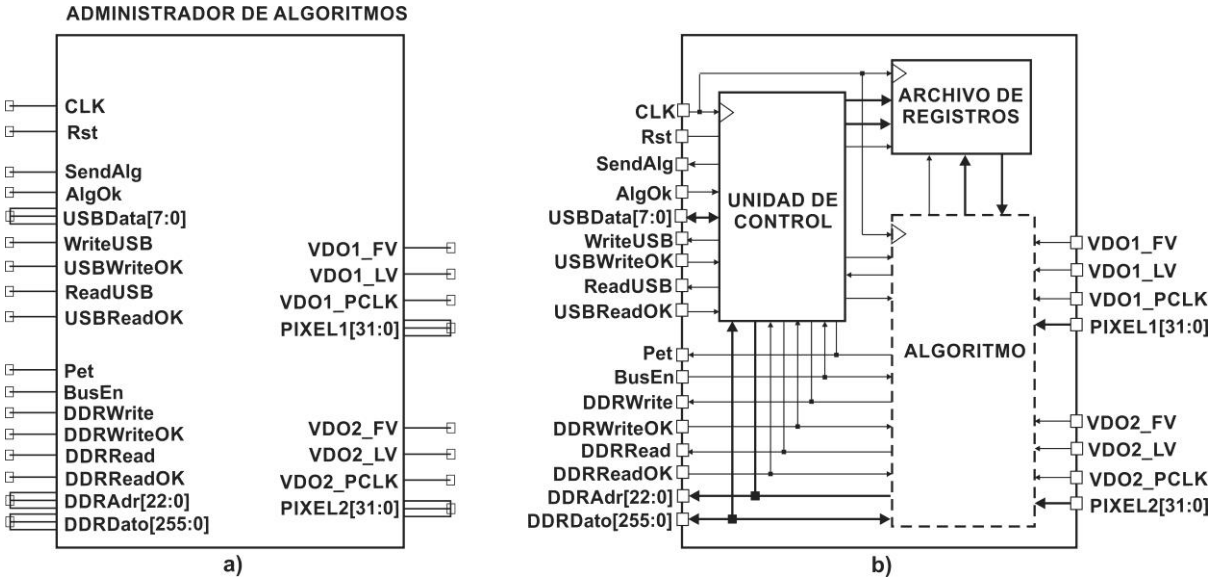

Figura 4.22 a) Símbolo del *administrador de algoritmos*, b) Estructura interna.

Tabla 13 Descripción de las señales de entrada/salida del *administrador de algoritmos.*

Señal	Descripción
CLK	(Entrada) Señal de reloj principal del sistema de 250 MHz, para un funcionamiento síncrono con el *administrador de comunicación* y el *administrador de memoria*.
Rst	(Entrada) Señal de reinicio global del sistema.
DDRDato[255:0]	(Entrada/Salida) Bus de datos bidireccional para lectura/escritura de datos a la memoria.
DDRAdr[22:0]	(Salida) Bus de direccionamiento a la memoria.
Pet	(Salida) Permite hacer la solicitud de acceso a la memoria.
BusEn	(Entrada) Indica que el acceso a la memoria ha sido concedido y se pueden hacer peticiones de lectura/escritura.
DDRRead	(Salida) Señal que indica una petición de lectura a una localidad en memoria direccionada por DDRAdr.
DDRReadOk	(Entrada) Indica que los datos han sido leídos de la memoria y están disponibles en el bus de datos DDRDato.
DDRWrite	(Salida) Señal que indica una petición de escritura de un dato en DDRDato a una localidad en memoria direccionada por DDRAdr.
DDRWriteOk	(Entrada) Indica que los datos han sido escritos en la memoria.
USBData[7:0]	(Entrada y Salida) Bus de datos bidireccional para la lectura/escritura de datos con el USB.
WriteUSB	(Salida) Inicia el proceso de escritura de un dato al USB.
USBWriteOk	(Entrada) Indica que el proceso de escritura fue exitoso.

Tabla 13 Descripción de las señales de entrada/salida del *administrador de algoritmos* (cont.).

Señal	Descripción
ReadUSB	(Salida) Inicia lectura del dato proveniente del USB.
USBReadOk	(Entrada) Indica que hay un dato proveniente del USB, listo para ser leído.
SendAlg	(Entrada) Indica que hay una solicitud de comunicación que debe ser atendida.
AlgOk	(Salida) Le indica al *administrador de comunicación* que la petición de comunicación fue atendida y ha finalizado.
VDO1_FV	(Entrada) Señal de sincronismo vertical de la cámara 1 que indica que se está transmitiendo un cuadro de video.
VDO1_LV	(Entrada) Señal de sincronismo horizontal de la cámara 1 que indica que los datos en el bus de datos PIXEL1 corresponden a pixeles válidos.
VDO1_PCLK	(Entrada) Señal de reloj de la cámara 1 con la cual los pixeles están alineados en el flanco de subida.
PIXEL1[31:0]	(Entrada) Bus de datos de la cámara 1 que contienen la información de los pixeles que conforman al video.
VDO2_FV	(Entrada) Señal de sincronismo vertical de la cámara 2 que indica que se está transmitiendo un cuadro de video.
VDO2_LV	(Entrada) Señal de sincronismo horizontal de la cámara 2 que indica que los datos en el bus de datos PIXEL2 corresponden a pixeles válidos.
VDO2_PCLK	(Entrada) Señal de reloj de la cámara 2 con la cual los pixeles están alineados en el flanco de subida.
PIXEL2[31:0]	(Entrada) Bus de datos de la cámara 2 que contienen la información de los pixeles que conforman al video.

Como resultado final en la Figura 4.23 se muestra el modelado del *elemento central de procesamiento,* este modelado fue realizado mediante el editor de esquemáticos provisto por el entorno de desarrollo del ISE *Desing Suite*. A través de este esquemático los usuarios integrarán sus algoritmos a evaluar a través del *administrador de algoritmos.* El resumen de los recursos utilizados por el *elemento central de procesamiento* en la FPGA Virtex5-LX50T se muestran en la Tabla 14, donde se puede apreciar claramente que el 78 % de los recursos lógicos de la FPGA quedan libres para ser utilizados para la implementación de los algoritmos de procesamiento de video a evaluar en la herramienta propuesta.

Tabla 14 Resumen de recursos consumidos por el *elemento central de procesamiento* en el FPGA.

Recursos	Utilizados	Disponibles	% de utilización
Slice Registers	6500	28800	22%
Slice LUTs	4715	28800	16%
LUT-FF pairs	2541	8674	29%
Pines IOBs	210	480	43%
Block RAM/FIFO	7	60	11%
BUFG/BUFGCTRLs	14	32	43%
PLL_ADVs	1	6	16%

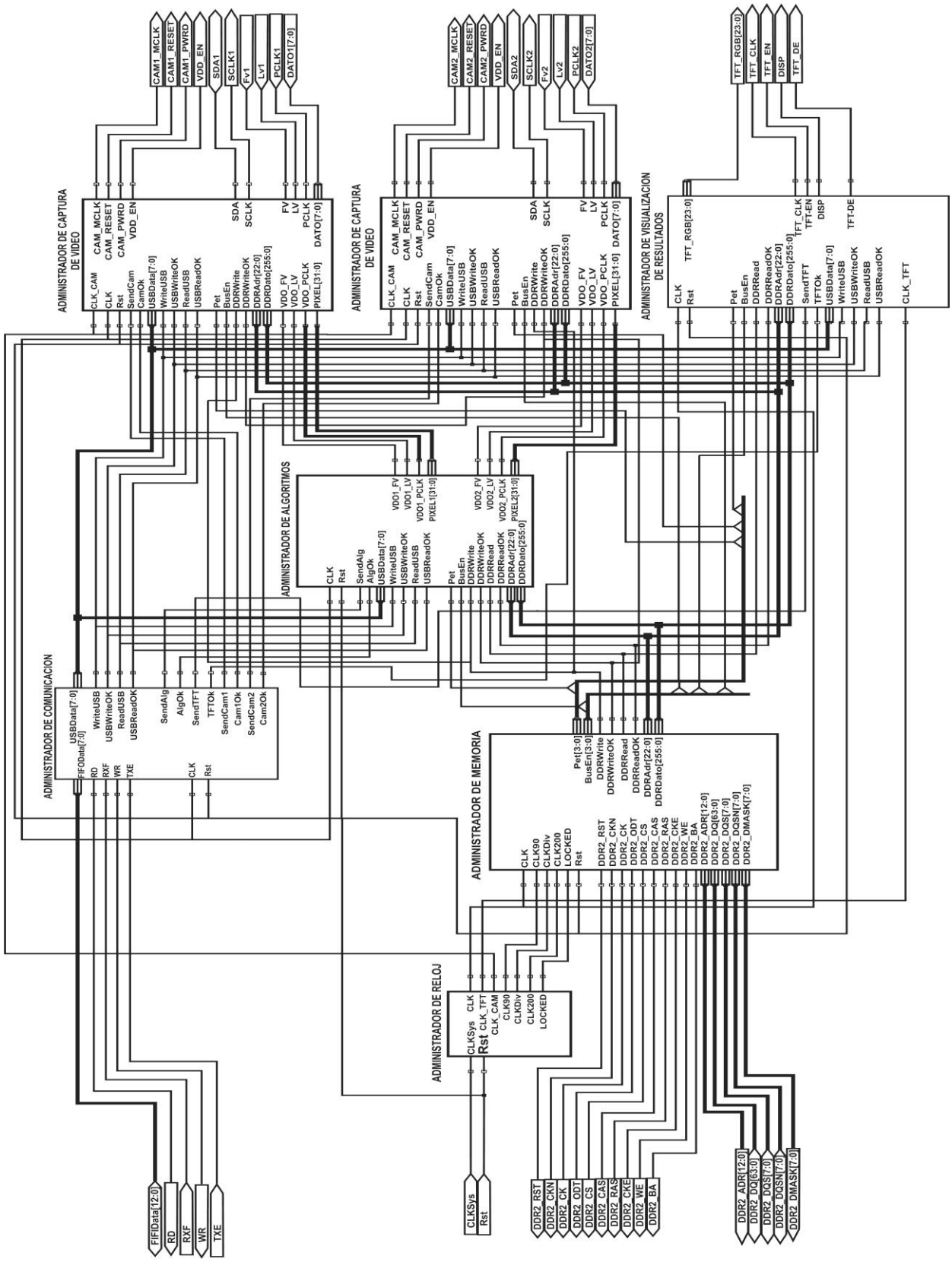

Figura 4.23 Diagrama de conexiones internas del e*lemento central de procesamiento*.

4.2 Consideraciones para el desarrollo de prácticas experimentales mediante el uso del SCPV.

En general para el desarrollo de prácticas en el contexto de un curso en áreas de las Ciencias de la Computación se sigue el marco de trabajo mostrado en la Figura 4.24. En este contexto, para el diseño y desarrollo de prácticas experimentales utilizando el SCPV es necesario considerar los aspectos necesarios para el uso de la herramienta en cada fase del marco de trabajo. Las consideraciones en cada fase de este marco de trabajo se resumen de la siguiente manera:

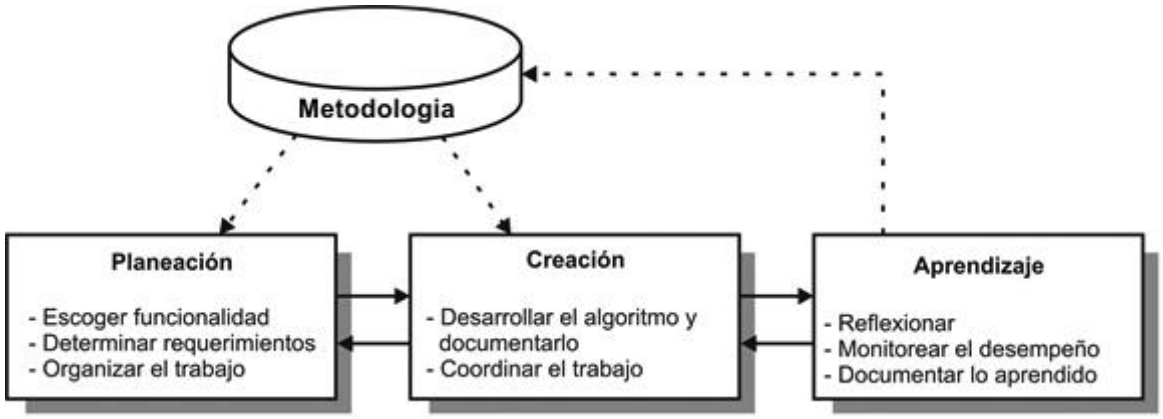

Figura 4.24 Marco de trabajo para el desarrollo de prácticas experimentales.

4.2.1 Fase de planeación

En esta fase se debe considerar la introducción del uso de la herramienta a través de los ejemplos prácticos provistos por el repertorio de algoritmos existente, para que posteriormente, los estudiantes propongan un proyecto, localicen las especificaciones necesarias para diseñar un algoritmo que lo resuelva, y organicen el trabajo a realizar. A través de estas actividades, el estudiante deberá identificar y presentar un problema, reunir la información necesaria, y generar una solución potencial.

En esta fase es de vital importancia que el profesor permita que los estudiantes se familiaricen con el uso de la herramienta HW/SW (construida como principal producto de esta tesis) a partir de su experiencia adquirida en el uso de ésta. Otra de las funciones que debe cumplir el profesor es la de orientar a los estudiantes para definir la extensión de la funcionalidad de los algoritmos propuestos, así como también, la de orientar en la determinación de las especificaciones que debe cumplir el diseño en base a los recursos provistos por la herramienta o bien definir si es necesario la modificación de estos recursos para facilitar el diseño. De forma a priori las especificaciones a considerar se resumen en:

- Definir la metodología de descripción del algoritmo y el tipo de modelado que se usará para este propósito (flujo de datos, algorítmico o estructural).
- Definir el formato de video a procesar.
- Definir el tipo de datos resultantes del procesamiento (video) o análisis (datos).

- Definir el medio para la interpretación de los resultados obtenidos; visualizar los resultados en la pantalla que integra la herramienta o bien la interpretación de estos mediante una interfaz gráfica diseñada por el usuario.
- Definir los segmentos de memoria a utilizar, si es que la aplicación lo requiere.
- Definir los parámetros que permiten o modifican el funcionamiento de la aplicación del algoritmo propuesto.

Para la organización del trabajo a realizar, se deberá definir la metodología de diseño a seguir en el desarrollo de la práctica en base a la experiencia que se tenga en el uso de dicha metodología. El propósito de dejar libre la elección de la metodología de diseño, es la de proporcionar un entorno de desarrollo flexible, en donde, el profesor y los estudiantes en lugar de preocuparse por aprender las etapas de una metodología nueva, se enfoquen en reforzar sus conocimientos con respecto a metodologías que hayan aprendido en cursos anteriores de sistemas digitales y su adaptación para la implementación de algoritmos de procesamiento de video en un dispositivo lógico programable.

4.2.2 Fase de creación

En esta fase se incluyen las actividades de diseño, implementación y documentación de los algoritmos propuestos en la fase de planeación. En esta etapa, los estudiantes construyen una nueva versión de un algoritmo ya integrado en la herramienta o bien uno completamente nuevo, que se pueda compartir con otros estudiantes o grupos. El proceso es controlado por el profesor, quien fortalece el trabajo colaborativo y formaliza el proceso de diseño en base a la metodología que se esté empleando en el desarrollo de los algoritmos destinados al procesamiento de video. El resultado a obtener en esta fase es el prototipo de la implementación del algoritmo propuesto. El prototipo desarrollado en el SCPV debe de consistir en la adecuación de la interfaz de usuario para poder gestionar el funcionamiento del algoritmo propuesto y la implementación de la lógica del algoritmo de procesamiento de video en el FPGA.

4.2.2.1 Diseño

Con base en las especificaciones obtenidas en la fase de planeación, en esta etapa se deberán establecer como serán diseñados y construidos los elementos que integran el algoritmo de procesamiento de video y la interfaz de usuario necesaria para la gestión de su funcionamiento, así como la forma en que interactúan entre sí.

Para el diseño del algoritmo de procesamiento de video se debe considerar definir:

- La metodología a utilizar para su descripción.
- La herramienta de diseño a utilizar, HDL, esquemático, editor de máquinas de estados o una combinación de éstas.

- Si se elige un HDL, es importante determinar el tipo de descripción más adecuado para el modelado de los elementos que integran al algoritmo, flujo de datos, algorítmico o estructural.
- El diseño conceptual de la entidad superior para que ésta sea compatible con el SCPV.
- Las funcionalidades internas y externas que se efectuarán en el algoritmo, las cuales definirán la estructura interna para ser modelada.
- Los recursos del SCPV que serán usados en la aplicación.

Por otra parte, para el diseño de la interfaz de usuario se debe considerar:

- La forma en que serán capturados los parámetros de configuración.
- La forma en que se presentaran los resultados en la interfaz (si es que la aplicación así lo requiere).
- La estructuración de las tramas de comunicación que se usaran.
- Las funcionalidades externas e internas que definirán la arquitectura de la interfaz de usuario.

Al finalizar esta etapa se espera contar con el diseño conceptual de los elementos a implementar para llevar a cabo su aplicación.

4.2.2.2 Implementación

En esta etapa, a partir del diseño conceptual del algoritmo de procesamiento de video y su respectiva interfaz de usuario, se desarrollan en forma paralela el modelado del algoritmo y la programación de la interfaz.

El modelado del algoritmo de procesamiento de video, consiste en seguir los procedimientos necesarios para la descripción de su arquitectura funcional mediante un lenguaje de descripción de hardware, y añadir esta arquitectura al SCPV para su evaluación. Los procedimientos mencionados deben seguir las normas establecidas por el entorno de diseño ISE *Desing Suite* 14.5, en la que esta modelada la arquitectura del *elemento central de procesamiento* que compone el SCPV. Entre los principales procedimientos se encuentran los siguientes:

- Agregar al proyecto del *elemento central de procesamiento* un nuevo módulo HDL con el nombre del algoritmo de procesamiento a implementar.
- Definir la entidad con los puertos de entrada y salida con los que se conectará este módulo con el *elemento central de procesamiento* a través el *administrador de algoritmos.*
- Modelar la estructura o comportamiento que ejecuta el funcionamiento del algoritmo de procesamiento de video mediante el lenguaje de descripción de hardware elegido.

- Una vez finalizado el modelado del comportamiento del algoritmo éste es incorporado al SCPV mediante una sentencia de instanciación que permita agregar el nuevo algoritmo al módulo *administrador de algoritmos.*
- Finalmente se genera el programa que configura al FPGA como el *elemento central de procesamiento* con el algoritmo de procesamiento a evaluar.

Por otro lado, la interfaz de usuario provista por el SCPV, es un programa realizado mediante la herramienta C++ Builder versión 6.0 usando la POO, que permite la configuración de los elementos que integran el SCPV. La implementación de la interfaz de usuario que gestiona el funcionamiento del algoritmo de procesamiento a implementar, se resume en agregar un nuevo formulario a la interfaz ya existente, siguiendo las normas de programación del entorno C++ Builder versión 6.0. Este proceso consiste en:

- Agregar o modificar un formulario al proyecto de la interfaz de usuario provista por el SCPV.
- Incorporar al formulario los elementos gráficos (cuadros de texto, botones, etc.) que permitan la captura y el envío de los parámetros de configuración para la gestión del algoritmo.
- Realizar la programación para estructurar y enviar las tramas de comunicación con la información.

Teniendo la implementación terminada se procede a efectuar las pruebas del algoritmo en el SCPV. Estas pruebas permiten evaluar si el algoritmo implementado ha cumplido con los objetivos esperados y si los resultados obtenidos concuerdan con la teoría.

4.2.2.3 Documentación

Teniendo los resultados de la implementación del algoritmo de procesamiento de video en el SCPV, se reporta el procedimiento que se siguió para el diseño e implementación de su aplicación, los resultados que fueron obtenidos y las conclusiones derivadas de su experiencia al llevar a cabo el proyecto. Además, se debe contemplar un apartado en donde se propongan mejoras a su proyecto desarrollado y/o al SCPV con la finalidad de motivar la evolución de los algoritmos y la herramienta de evaluación.

4.2.3 Fase de aprendizaje

En esta fase se contempla que se registren los resultados (código fuente, reportes, simulaciones, documentos, etc.) en el repertorio de algoritmos de la herramienta para que éstos sean compartidos con el resto de la clase, y futuros usuarios de la herramienta. En este contexto, se debe considerar exponer los problemas encontrados a lo largo del proceso y la manera en que éstos fueron solventados, al igual que se propongan mejoras a los algoritmos desarrollados con el fin de darles seguimiento en un esquema de trabajo de mejora continua. De tal forma, un resultado que no cumpla con las especificaciones planteadas no sea considerado como un

fracaso, más bien, sea considerado como una oportunidad para fortalecer el proceso de enseñanza y aprendizaje, siendo éste un medio para detectar las debilidades en el proceso y de esta manera poder corregirlas.

Para llevar a cabo este proceso de aprendizaje se plantea como recomendación que se sigan las siguientes actividades:

- En primer lugar, hacer una exposición de los proyectos realizados y los resultados obtenidos.
- Al finalizar la exposición realizar una sesión de preguntas y comentarios con el fin de obtener una realimentación.
- Finalmente realizar las conclusiones de forma grupal acerca del proyecto presentado y a través de una lluvia de ideas se proponer mejoras que puedan ser implementadas posteriormente al SCPV o a las implementaciones de los algoritmos.

Capítulo 5

Repertorio de algoritmos

Como apoyo al proceso de aprendizaje, la herramienta plantea la incorporación de un repertorio colaborativo de algoritmos, este repertorio alberga las aplicaciones que los usuarios hayan desarrollado usando la herramienta. Estas aplicaciones servirán como ejemplos prácticos para otros usuarios de la herramienta, propiciando de esta manera un ambiente colaborativo en donde el principal objetivo es compartir el conocimiento adquirido mediante la transmisión de experiencias.

Dentro del aprendizaje colaborativo, la existencia de un repertorio establece el mecanismo para la interacción entre los usuarios, propiciando el intercambio de información, de esta manera se obtiene una realimentación de los resultados obtenidos. En este sentido el repertorio tiene como objetivo motivar a los usuarios a proponer mejoras de las aplicaciones existentes dando la oportunidad de construir nuevo conocimiento a partir de conocimientos previos.

En este sentido, se presentan los resultados experimentales al utilizar la herramienta para el modelado y evaluación de algoritmos de procesamiento de video en un FPGA. El conjunto formado por estos algoritmos es la parte inicial del repertorio de algoritmos que integra la herramienta, los cuales tienen como objetivo principal servir como ejemplos prácticos que motiven al usuario a proponer sus versiones personalizadas de éstos.

5.1 Implementación del algoritmo del negativo

El algoritmo para la obtención del negativo es una técnica de procesamiento en donde se aplica una operación puntual a nivel de pixel, con el objetivo de resaltar regiones claras de interés sobre regiones oscuras dominantes. El pixel transformado a la salida se obtiene a partir de restar cada uno de los valores de los pixeles de entrada al valor máximo ($L-1$) de nivel de gris, este

valor máximo lo determina la profundidad de color que se esté empleando. La ecuación 2.4 extraída de la sección 2.2.1.1.1 muestra la transformación que se aplica a los pixeles de entrada. El resultado esperado al aplicar esta transformación a una secuencia de imágenes en niveles de gris es el mostrado en la Figura 5.1, donde el valor del pixel de salida S es inversamente proporcional al valor del pixel de entrada f.

$$S(x, y, t) = (L - 1) - f(x, y, t) \tag{2.4}$$

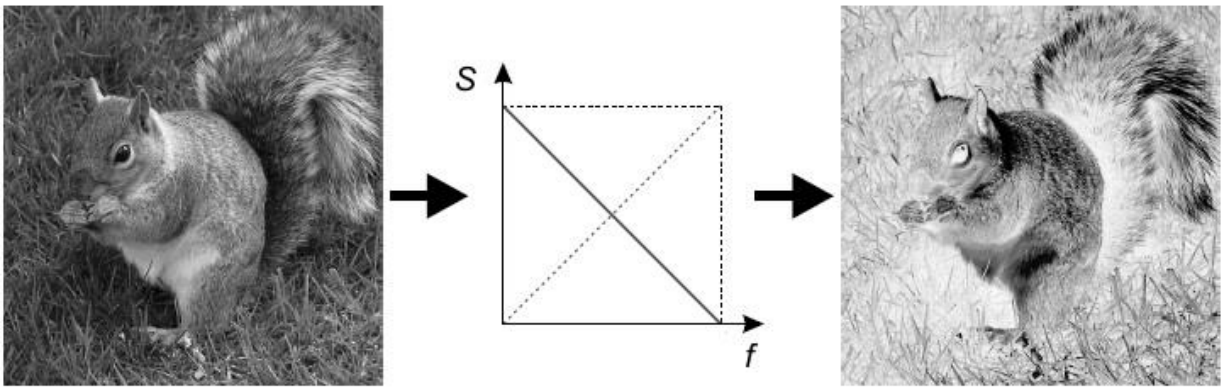

Figura 5.1 Negativo de una imagen.

Para la implementación del algoritmo del negativo en la herramienta se definen las especificaciones que ésta debe cumplir en base a la teoría del algoritmo propuesto y su adecuación para ser evaluado en la herramienta. Estas especificaciones se enlistan a continuación:

- El prototipo debe aplicar la transformación mostrada en la ecuación 2.4 a los pixeles provenientes de la cámara de video integrada.
- El formato de color de los pixeles que conforman la señal de video de entrada y salida estarán codificados en formato RGB 565, soportado por la cámara y el sistema de visualización de resultados.
- El tamaño de video de entrada es de 240 × 272 pixeles con una frecuencia de 30 cuadros por segundo.
- Con la finalidad de poder visualizar el video original y el video procesado al mismo tiempo, el prototipo debe generar a su salida una secuencia de video de 480 × 272 pixeles, donde las primeras 240 columnas de pixeles representarán el video original y las siguientes 240 el video procesado.
- Los resultados del procesamiento serán almacenados a partir de la localidad 0x000000 de la memoria para su visualización en el LCD que la herramienta provee.

La metodología de diseño descendente (*Top Down*) fue la elegida para el desarrollo de esta implementación, como se ha mencionado anteriormente esta metodología consiste en visualizar

al sistema a desarrollar en un módulo desde un nivel alto de abstracción, para luego dividirlo en módulos jerárquicamente inferiores a partir de las funcionalidades que se deben cumplir. El modelado mediante esta metodología comienza con la representación abstracta del sistema a implementar, donde este se visualiza como una caja negra y únicamente se definen las señales de entrada y salida que necesita para funcionar a partir del análisis de las especificaciones. Posteriormente se definen los módulos internos que integran al sistema y se definen sus funcionalidades. En este sentido en la Figura 5.2 se muestra el modulo resultante que implementa el algoritmo del negativo, el cual está constituido por la *unidad de control*, el módulo de *captura de resultados* y el módulo *negativo.*

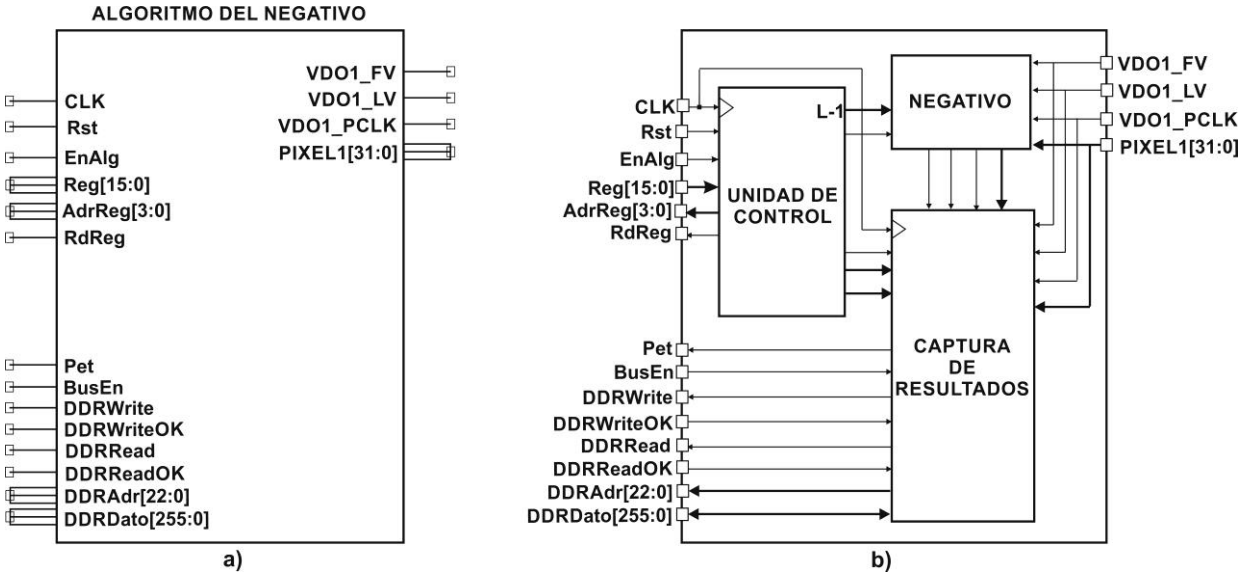

Figura 5.2 a) Símbolo del algoritmo del negativo, b) Estructura interna del símbolo.

Cuando la señal de habilitación del algoritmo (*EnAlg*) es activada la *unidad de control* lee los parámetros de configuración almacenados en el *archivo de registros* e inicia el procesamiento de la señal de video proveniente de la cámara. Este módulo recibe a través de los registros 0 y 1 la dirección de la localidad de memoria donde serán almacenados los resultados y en los registros 2 y 3 el tamaño del video a procesar, estos registros de configuración son transferidos al módulo de c*aptura de resultados* el cual permite el almacenamiento del video original y el video procesado para su posterior escritura en la localidad de memoria destinada para almacenar los resultados. Por último el modulo *negativo* ejecuta la transformación del negativo a los pixeles provenientes de la cámara siguiendo el circuito mostrado en el esquema de la Figura 5.3, el cual representa la ecuación 2.4.

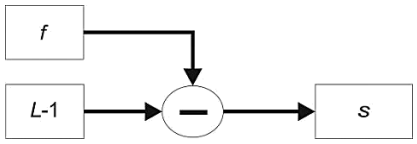

Figura 5.3 Circuito implementado para la obtención del negativo.

Por otra parte, en la Figura 5.4 se presenta la interfaz generada para la configuración de los parámetros que gestionan el funcionamiento del algoritmo, esta interfaz permite la captura de los parámetros de tamaño de video y la localidad en memoria para alojar los resultados, esta

información de configuración se organiza conforme a la trama de configuración del *administrador de algoritmos* de acuerdo al protocolo de comunicación provisto por la herramienta y es enviado a través del sistema de comunicación mediante el método *Enviar()* provisto en la clase *PuertoUSB.* Para la configuración del sistema de captura de video y el sistema de visualización de resultados se hace uso de los formularios provistos por la interfaz de usuario.

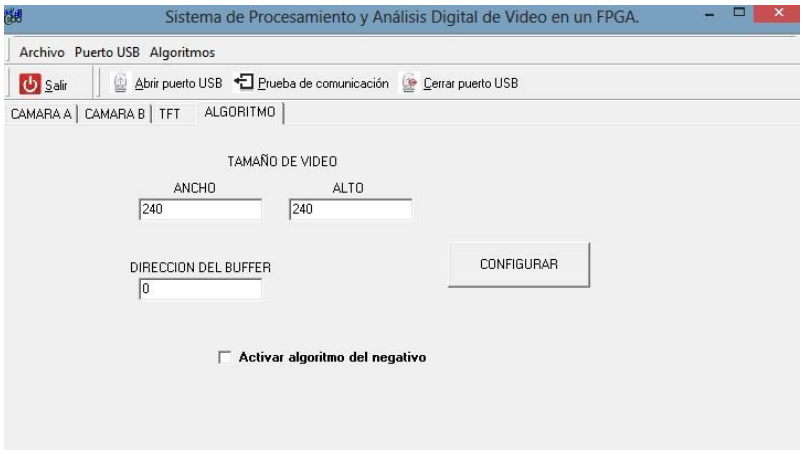

Figura 5.4 Formulario de configuración del algoritmo del negativo.

En la Figura 5.5, se muestran los resultados al aplicar el algoritmo del negativo a la señal de video provisto por la cámara, los cuales concuerdan con la teoría.

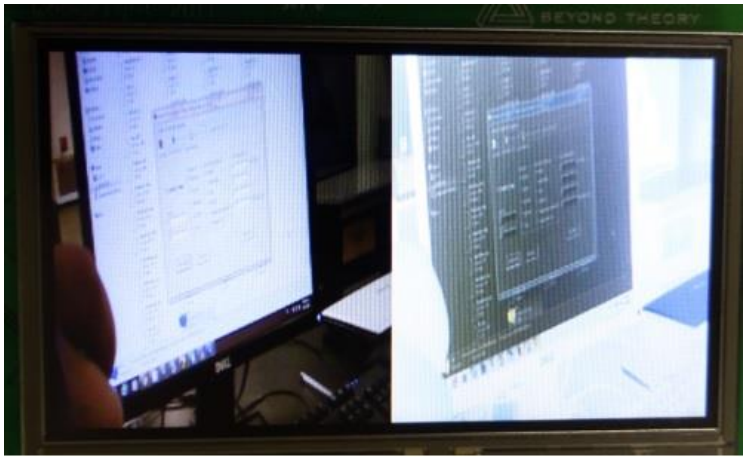

Figura 5.5 Resultado al aplicar el algoritmo del negativo.

5.2 Implementación del algoritmo de la binarización por umbral.

La técnica de binarización, al menos en su forma básica, es una operación puntual. Donde para obtener una imagen binaria se hace una transformación no lineal de la imagen de entrada, obteniéndose una imagen de salida en la cual cada pixel puede tomar alguno de dos valores: 0 y 1 (negro y blanco). Para esto, se toma un valor de umbral t_h, de forma que:

$$S(x,y,t) = \begin{cases} 0 \; si \;\; f(x,y,t) < t_h \\ 1 \; si \;\; f(x,y,t) \geq t_h \end{cases} \tag{5.1}$$

La binarización tiene una gran utilidad en el procesamiento automático de una imagen pues reduce enormemente la cantidad de datos de la imagen de una forma muy sencilla. Si se parte de imágenes bien contrastadas, la binarización permite con muy poco procesamiento un análisis fiable de la imagen. La forma más inmediata de representar una imagen de niveles de gris mediante una imagen binaria es considerar únicamente el bit más significativo del nivel de gris de cada pixel. O lo que es lo mismo, fijar un umbral en la mitad de la escala de gris que servirá de referencia para asignar en la imagen binaria un 0 si el nivel de gris del pixel es inferior o un 1 si supera este umbral. Por ejemplo, en una imagen típica de 256 niveles representada por 8 bits por pixel tomar el primer bit es lo mismo que poner a 0 (negro) todo aquel pixel que presente un nivel de gris por debajo de 128 y a 1 (blanco) el resto. Aunque el dividir la escala de grises en dos partes iguales para asignar 0 o 1 puede parecer la forma más correcta de binarizar, en realidad la mayoría de las veces esto no es así. El rango dinámico de una imagen no siempre se extiende a todo el intervalo de niveles de gris posible y, aun así, muchas veces es más adecuado fijar el umbral de binarización en otro punto distinto del valor medio de la escala de grises. En la Figura 5.6 se muestra un ejemplo del resultado esperado al aplicar la transformación de binarización a una imagen en escala de grises.

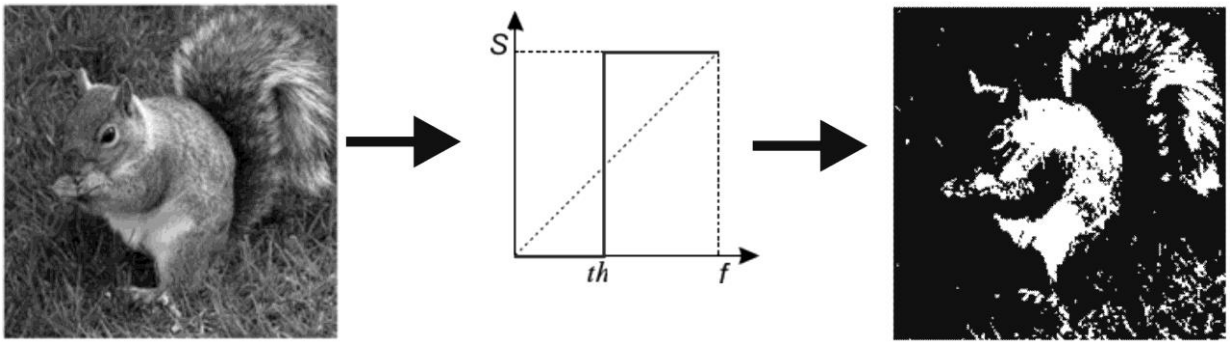

Figura 5.6 Resultado de aplicar la transformación de binarización.

Derivado de la teoría del algoritmo propuesto, las especificaciones propuestas para aplicar este algoritmo en la herramienta se resumen en:

- El algoritmo debe aplicar la transformación de binarización, tomando como referencia un umbral que sea variable por el usuario.
- El módulo debe ser capaz de procesar una señal de video en escala de grises en formato RGB 565 y entregar los resultados con el mismo formato de color.
- El tamaño de video a procesar es de 240 × 272 pixeles.
- Con la finalidad de poder visualizar el video original y el video procesado al mismo tiempo, el prototipo debe generar a su salida una secuencia de video de 480 × 272 pixeles, donde las primeras 240 columnas de pixeles representarán el video original y las siguientes 240 el video procesado.
- Almacenar los resultados del procesamiento en una localidad de memoria para ser visualizados en el *sistema de visualización de resultados*.

– Diseñar una interfaz de usuario que permita modificar el umbral.

A partir de los requisitos planteados y tomando en consideración que el procesamiento a efectuar es a nivel de pixel, muy similar al algoritmo del negativo previamente implementado, se sigue la filosofía de reutilización de código. De esta manera a partir del módulo y la interfaz del algoritmo del negativo se realizan los cambios necesarios para la implementación del algoritmo de binarización por umbral, reduciendo de esta manera los tiempos de desarrollo del algoritmo planteado. En la Figura 5.7 se muestra el esquema del módulo que implementa la binarización.

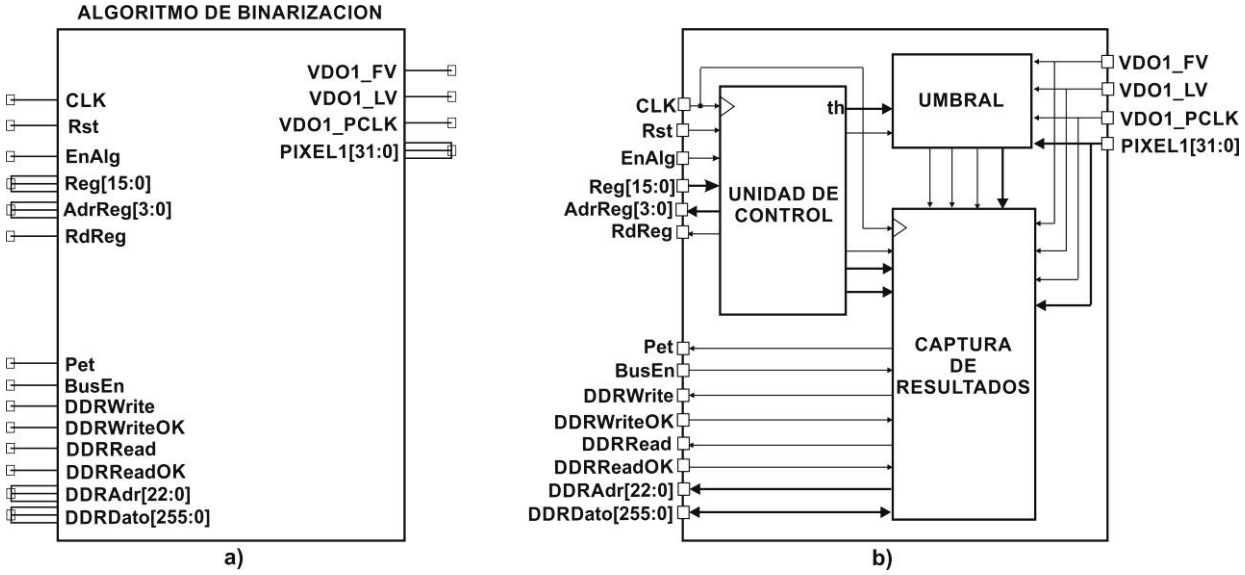

Figura 5.7 a) Símbolo del algoritmo de binarización, b) Estructura interna.

En el módulo *Umbral* se implementa el circuito mostrado en la Figura 5.8, para la obtención de la señal de video binarizada a parir del umbral t_h provisto por la unidad de control. Esta señal de video procesada es enviada al módulo de *captura de resultados* junto con la señal de video original para ser almacenados en el *buffer* de memoria destinado para la visualización de los resultados. La *unidad de control* transfiere los parámetros de configuración almacenados en el *archivo de registros.* En los registros 0 y 1 se encuentra la dirección en memoria para almacenar los resultados, en los registros 2 y 3 el tamaño del video alto y ancho respectivamente, y en el registro 4 el valor del umbral t_h.

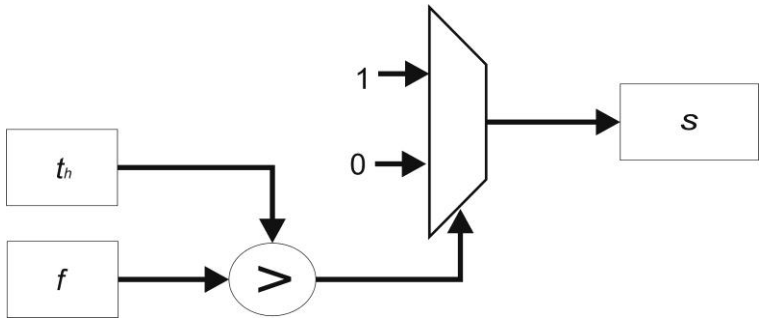

Figura 5.8 Circuito que implementa la binarización por umbral.

En la Figura 5.9 se muestra el formulario implementado para la configuración de los parámetros de configuración necesarios para el funcionamiento del algoritmo de binarización, el parámetro umbral permite el ajuste del umbral por parte del usuario. Por otra parte en la Figura 5.10 se muestra la señal de video procesado por el módulo desarrollado usando un umbral con un valor de 128 en la escala de niveles de grises.

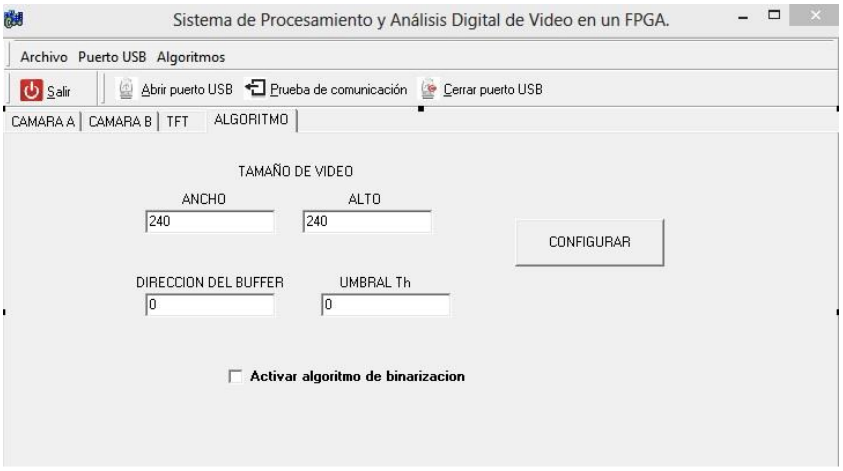

Figura 5.9 Interfaz de usuario para la configuración del algoritmo de binarización.

Figura 5.10 Resultado del procesamiento de binarización en la herramienta.

5.3 Algoritmo de sustracción de fondo

La detección de objetos en movimiento en secuencias de video es el primer paso relevante para la extracción de información para varias aplicaciones de visión, incluida la video vigilancia, la dirección de los vehículos autónomos, para la compresión eficaz de video, para el seguimiento inteligente de objetos en movimiento, para el reconocimiento automático de objetos, etc.

Hay muchas dificultades asociadas a la detección de objetos en movimiento que pueden surgir si existe un movimiento relativo del observador con respecto a los objetos y el fondo. Cuando el fondo esté fijo y el video es adquirido por cámaras fijas, la detección de objetos en movimiento puede ser solventada a través de la sustracción de fondo o diferenciación de cuadros, aun cuando el fondo se mueva con una trayectoria conocida, por ejemplo, cuando el movimiento de

la cámara se ve limitada por una trayectoria fija se pueden aplicar los métodos de sustracción de fondo y diferenciación de cuadros modificados. En cambio, la segmentación de objetos en movimiento se hace más crítica cuando el video es adquirido por una cámara en movimiento libre y sin restricciones con un movimiento desconocido.

La configuración con cámara fija con objetos en movimiento, es el caso más básico, ya que las ambigüedades inherentes de la configuración de la cámara en movimiento no se consideran y la estructura de la escena se puede descartar desde el principio. Sin embargo, debe haber un tratamiento para los cambios de iluminación gradual y súbita, los objetos de fondo de alta frecuencia, y el ruido proveniente de la cámara.

Asumiendo que la cámara está fija, existen diversos métodos para llevar a cabo la sustracción del fondo; el más sencillo consiste en hallar la diferencia entre la imagen actual y la imagen anterior de la secuencia, la diferencia $FD(x, y, t)$ entre el cuadro t y el cuadro $t - 1$ se define como:

$$FD(x, y, t) = f(x, y, t) - f(x, y, t - 1) \tag{5.2}$$

La información del objeto en movimiento se obtiene a partir de la diferencia pixel a pixel entre las dos imágenes. Para distinguir las diferencias debidas a un cambio en la escena, se puede lograr una segmentación umbralizando la imagen de la diferencia como:

$$Z(x, y, t) = \begin{cases} 1 & |FD(x, y, t)| > T \\ 0 & |FD(x, y, t)| \leq T \end{cases} \tag{5.3}$$

Posteriormente la mascara $Z(x, y, t)$ se aplica mediante una multiplicación pixel a pixel a la secuencia de imágenes entrante y con ello se obtiene la segmentación del objeto que haya sufrido un cambio en la escena.

$$S(x, y, t) = f(x, y, t) \times \mathrm{Z(x, y, t)} \tag{5.4}$$

En este sentido las funcionalidades propuestas para esta implementación son descritas a continuación:

- Se debe considerar el uso de un buffer temporal para el almacenamiento de un cuadro anterior en la secuencia, el cual fungirá como modelo de fondo.
- El modulo debe ser capaz de procesar una señal de video a color en formato RGB 565 y entregar los resultados con el mismo formato de color.
- Generar una salida de video de 480 × 272 pixeles, donde las primeras 240 columnas de pixeles representarán el video original y las siguientes 240 el video procesado.
- El tamaño del video a procesar es de 240 × 272 pixeles.
- Almacenar los resultados del procesamiento en una localidad de memoria para ser visualizados en el *sistema de visualización de resultados*.
- Diseñar una interfaz de usuario que permita el envío de los parámetros de configuración.

En la Figura 5.11, se muestra el modulo implementado para realizar la operación de sustracción de fondo, este módulo está integrado por su *unidad de control* que permite la configuración de

los parámetros de operación designados por el usuario a través del *archivo de registros*, y la unidad de *sustracción* que modela el comportamiento del algoritmo propuesto. El diagrama a bloques de la Figura 5.12 muestra el modelo funcional de la arquitectura propuesta para implementar las ecuaciones para la sustracción de fondo.

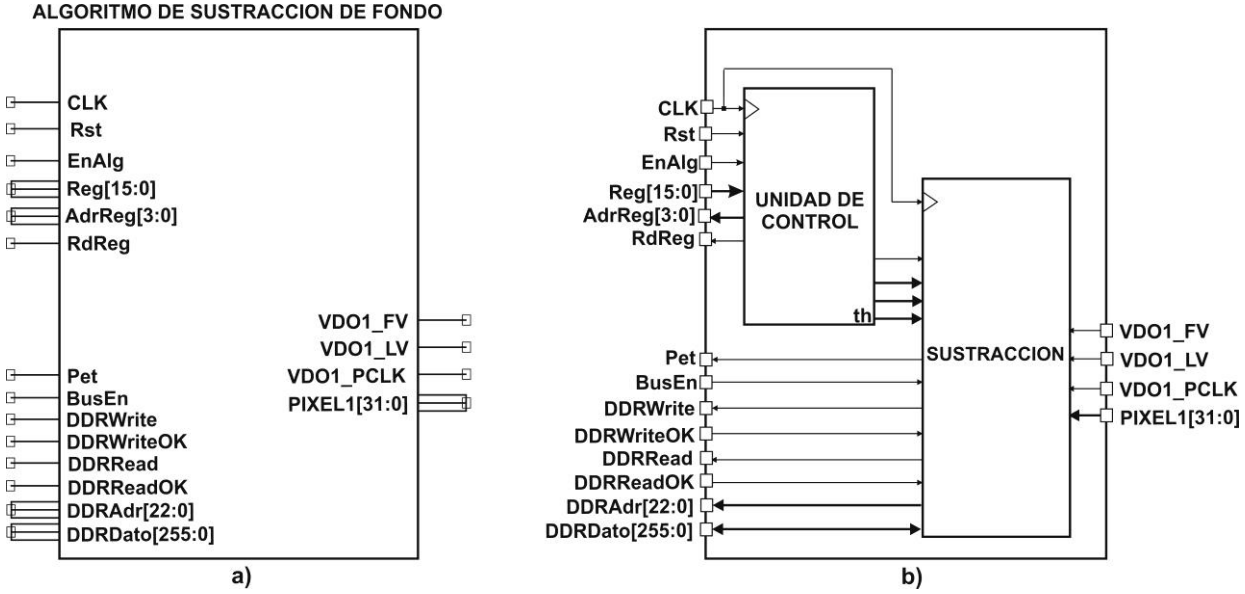

Figura 5.11 a) Símbolo del módulo que implementa el algoritmo de sustracción de fondo, b) Estructura interna.

El proceso inicia con la llegada de un pixel proveniente de la cámara, en ese instante se extrae del *buffer* temporal el pixel correspondiente a su posición en la imagen anterior de la secuencia y se actualiza el *buffer* temporal con el nuevo valor del pixel entrante. Teniendo el valor del pixel actual y el pixel de la imagen anterior se procede a calcular el valor absoluto de la diferencia de ambos pixeles. El valor absoluto de la diferencia previamente calculado pasa por un proceso de umbralización para obtener la máscara que segmenta al objeto que haya sufrido un cambio en la escena. Finalmente la máscara resultante se aplica al pixel actual y con esto se logra la segmentación del objeto en movimiento, este resultado se envía directamente al *buffer* de resultados para su posterior visualización en el LCD.

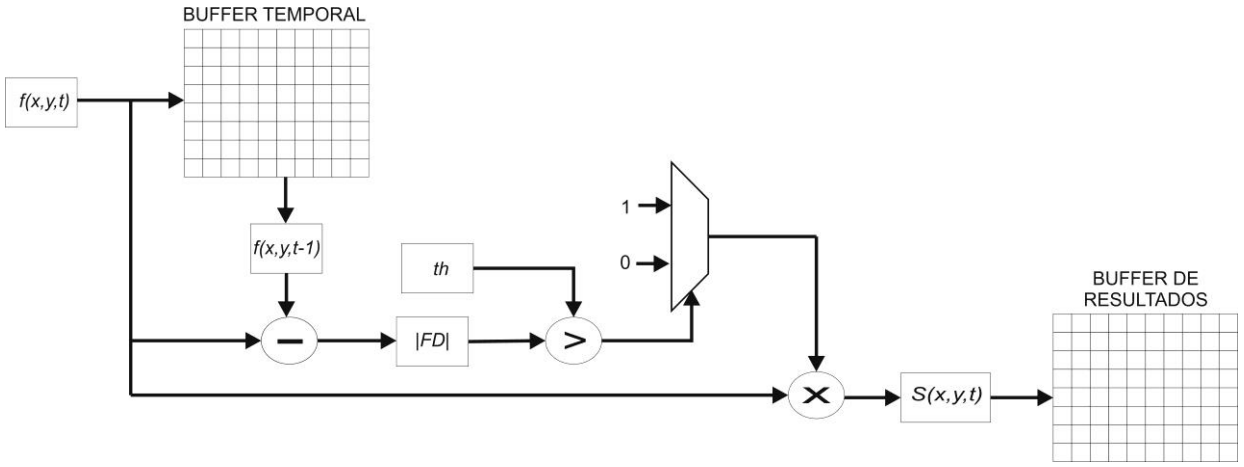

Figura 5.12 Diagrama funcional del módulo *sustracción*.

Como consecuencia de la aplicación de este esquema de segmentación en la herramienta de procesamiento de video se obtiene a la salida del LCD una imagen que aísla al objeto que esta cambiante en la escena (ver Figura 5.13), la sensibilidad a los cambios en la posición de los objetos depende del valor del umbral th y el control de los cambios de iluminación y enfoque de la cámara. Por otra parte en la Figura 5.14 se muestra la interfaz de usuario utilizada para el envío de la configuración con la cual trabaja el algoritmo implementado, en ella se define el tamaño de video a procesar la localidad en memoria donde se almacenará el *buffer* temporal y el *buffer* de resultados.

Figura 5.13 Resultados del esquema de sustracción de fondo.

Figura 5.14 Interfaz de usuario para la configuración del algoritmo de sustracción de fondo.

5.4 Algoritmo de extracción de bordes mediante el operador Sobel

Los bordes de una imagen digital se pueden definir como transiciones entre dos regiones de niveles de gris significativamente distintos. Suministran una valiosa información sobre las fronteras de los objetos y puede ser utilizada para segmentar la imagen, reconocer objetos, etc. La mayoría de las técnicas para detectar bordes emplean operadores locales basados en distintas aproximaciones discretas de la primera y segunda derivada de los niveles de grises de la imagen.

Los operadores de gradiente, en general, tienen el efecto de magnificar el ruido subyacente en la imagen, no obstante, el detector de Sobel se puede ver como la combinación de un filtro de

suavizado del ruido con un operador de aproximación del gradiente. Como ya se ha mencionado en la sección 2.2.3.1, utiliza dos máscaras de 3×3, una para el gradiente en la dirección *x* y otra para la dirección *y*, estas máscaras se mueven pixel a pixel, calculando el valor del gradiente para cada uno de ellos (esto mediante un proceso de convolución). Como ya se ha explicado anteriormente, una vez obtenido el valor del gradiente para cada dirección se obtiene la aproximación de la magnitud resultante mediante la suma del valor absoluto de ambos gradientes calculados. Finalmente se decide si es un borde o no en función de un umbral prefijado, de esta manera todos los pixeles que presentan un gradiente mayor al umbral son declarados contornos.

Partiendo de lo anterior los requisitos que se establecen para poder implementar el sistema de detección de bordes en la herramienta, son:

- Implementar el método para realizar la operación de convolución aplicando máscaras de 3×3 a la señal de video proveniente de la cámara.
- Integrar un sistema para el cálculo de la magnitud del gradiente a partir de sus componentes *x* y *y*.
- Incorporar un elemento de umbralización para la obtención de los bordes.
- El tamaño de video de entrada es de 480 × 640 pixeles y un formato de color en escala de grises RGB565.
- Los resultados del procesamiento se visualizarán en el LCD integrado en la herramienta.

Con respecto a los requisitos previamente establecidos, se puede observar que el principal reto es la implementación de la convolución. En principio la operación de convolución se puede definir para distintos espacios, tanto discretos como continuos y de cualquier dimensión. En este caso se usara la convolución discreta bidimensional porque las imágenes digitales de la secuencia de video son discretas, es decir, tienen un número finito y numerable de puntos. Además las imágenes son señales bidimensionales, pues tienen un ancho y un alto. El proceso de convolución aplica una ventana rectangular a la imagen de las dimensiones de la matriz de coeficientes de la máscara, la cual es centrada en cada uno de los pixeles de la misma en cada iteración, es decir, se toma un pixel de la imagen y todos los vecinos que lo rodean según sea el tamaño de la ventana que se esté usando (ver Figura 5.15).

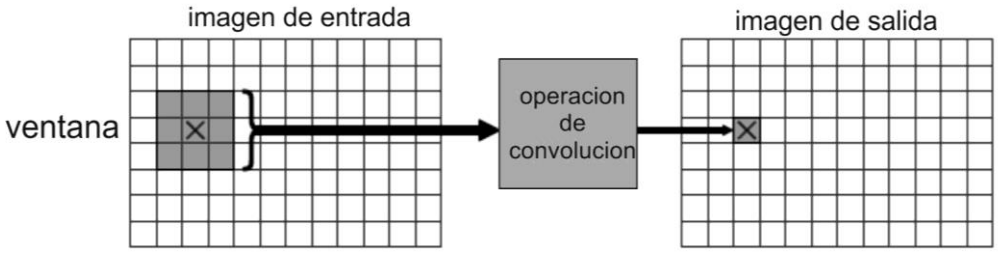

Figura 5.15 Operación de convolución.

El esquema que modela el funcionamiento para la aplicación de la convolución utilizado en esta implementación se muestra en la Figura 5.16, donde los coeficientes *w* representan la máscara de convolución que se aplicará a la imagen de entrada f. Los *buffer* de fila son memorias configuradas para trabajar como una FIFO, de esta manera se obtiene la ventana para aplicar la operación de convolución.

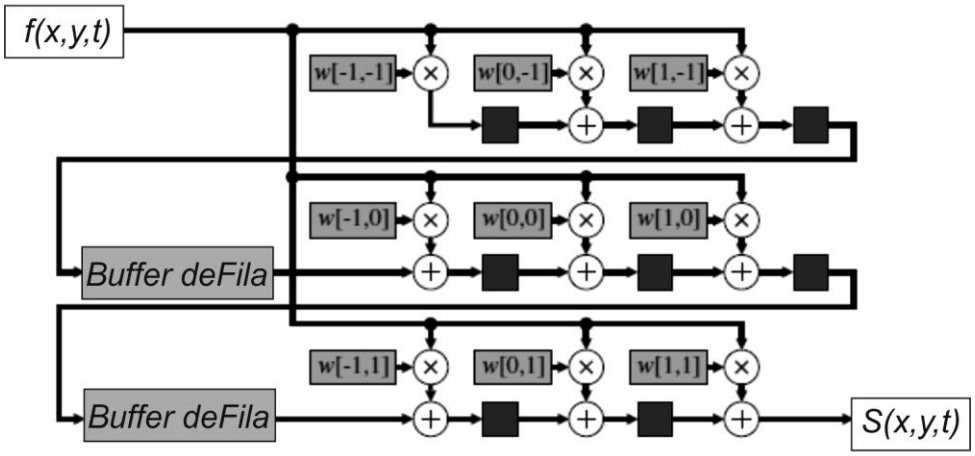

Figura 5.16 Circuito de convolución con una máscara de 3×3.

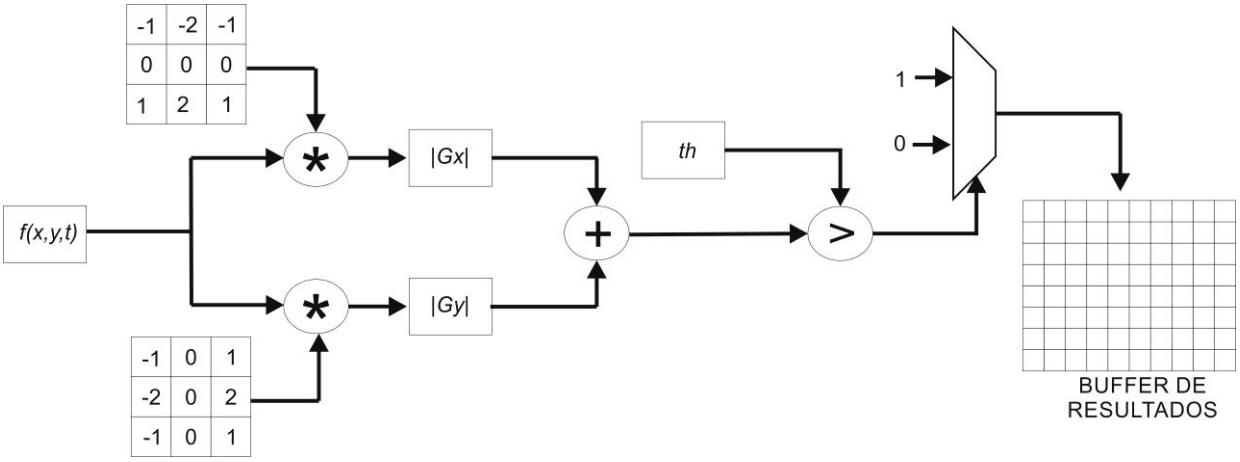

Figura 5.17 Circuito implementado para la obtención de bordes mediante el operador Sobel.

El esquema mostrado en la Figura 5.17 representa el modelo funcional para el algoritmo de extracción de bordes implementado, en donde al flujo de pixeles provenientes de la cámara, se les aplica la operación de convolución para la obtención del gradiente en la dirección *x*, y en la dirección *y*. Posteriormente se calcula la magnitud de cada uno de los gradientes y se suman con la finalidad de obtener la magnitud resultante. Finalmente se pasa la magnitud del gradiente por el proceso de umbralización obteniendo el video que contiene los bordes, el cual se almacena en el *buffer* de resultados para su posterior visualización. Como resultado de la implementación de este esquema de extracción de bordes en la herramienta se obtuvo el módulo mostrado en la Figura 5.18, por otra parte los resultados del video procesado se muestran en la Figura 5.19.

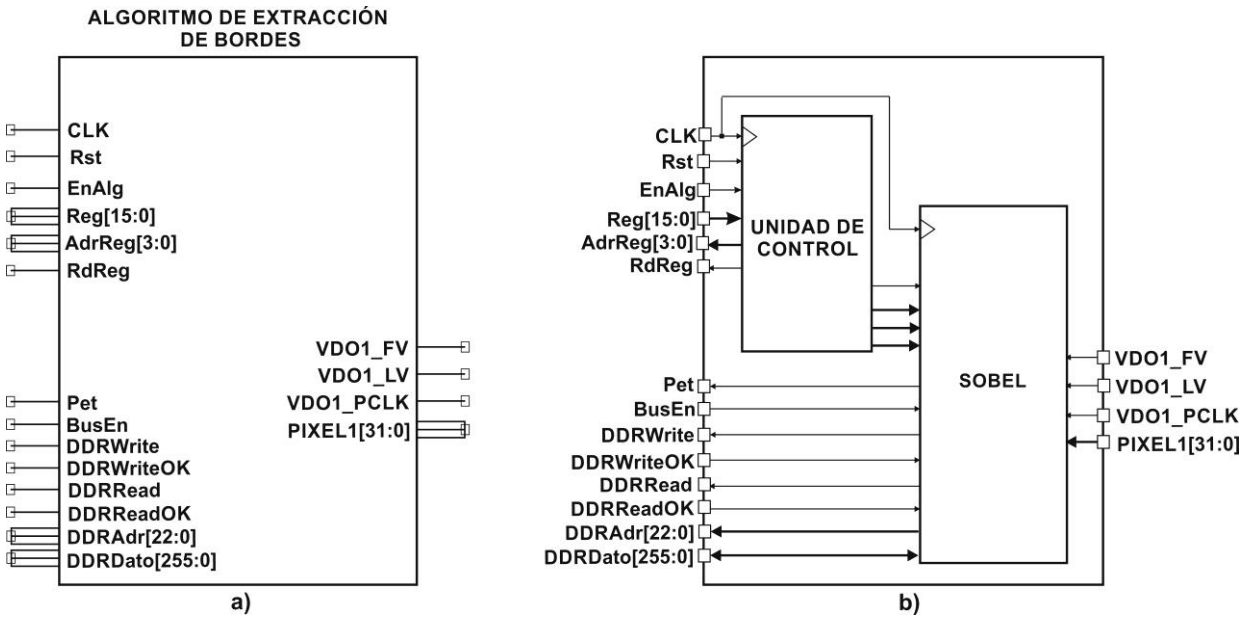

Figura 5.18 a) Símbolo del módulo de extracción de bordes, b) Estructura interna del símbolo.

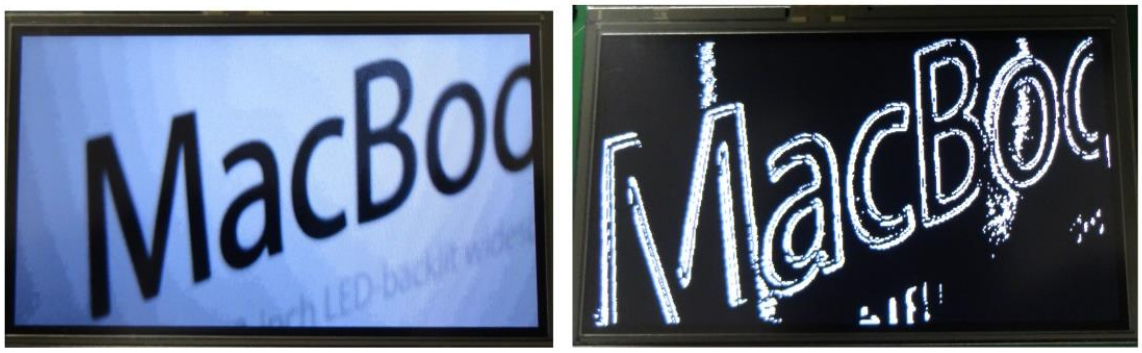

Figura 5.19 Resultados de la implementación del extractor de bordes en la herramienta.

Capítulo 6

Conclusiones y trabajos futuros

En este apartado se proporcionan las conclusiones obtenidas en el desarrollo de este trabajo y se mencionan algunos trabajos futuros por realizar.

6.1 Conclusiones

En este trabajo de tesis se ha propuesto una herramienta de adquisición y procesado de video basado en un FPGA, con el objetivo de que ésta sea incorporada en un futuro como apoyo en procesos de enseñanza/aprendizaje bajo un enfoque constructivista. Por tal motivo la herramienta fue diseñada en base a los fundamentos que establece este enfoque. Como resultado se obtuvo una herramienta con una arquitectura completamente modular mediante un esquema de código abierto, esto permite la optimización en el tiempo de desarrollo de aplicaciones de procesamiento y/o análisis de video sobre un FPGA. Como se puede ver en la Figura 6.1, el tiempo de desarrollo de una aplicación de procesamiento de video es una tarea altamente demandante, sin embargo este tiempo puede ser reducido en gran medida al introducir la herramienta propuesta (ver Figura 6.2), donde los usuarios se abstraen de tareas tales como la adquisición, almacenado y visualización de la señal de video, enfocándose solamente en el diseño de las funcionalidades del algoritmo que es objeto de estudio. En base a lo anterior los usuarios tienen mayor probabilidad de éxito en el desarrollo de sus proyectos, enfocando todo su esfuerzo y tiempo en su realización. En este sentido, su experiencia se enriquece al poder experimentar con una mayor cantidad de algoritmos en un tiempo reducido, propiciando de esta manera un aprendizaje más significativo.

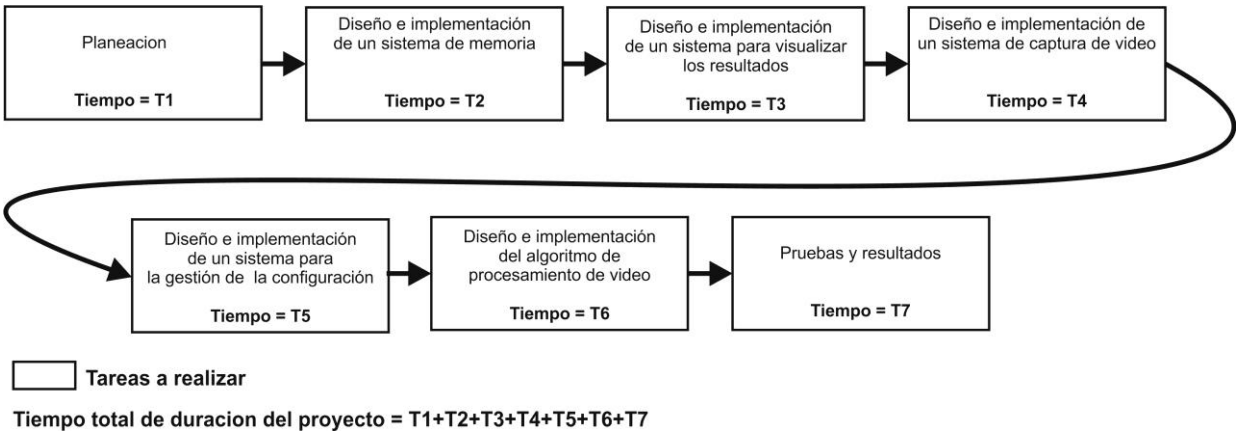

Figura 6.1 Tiempo estimado para la realización de un sistema de procesamiento de video.

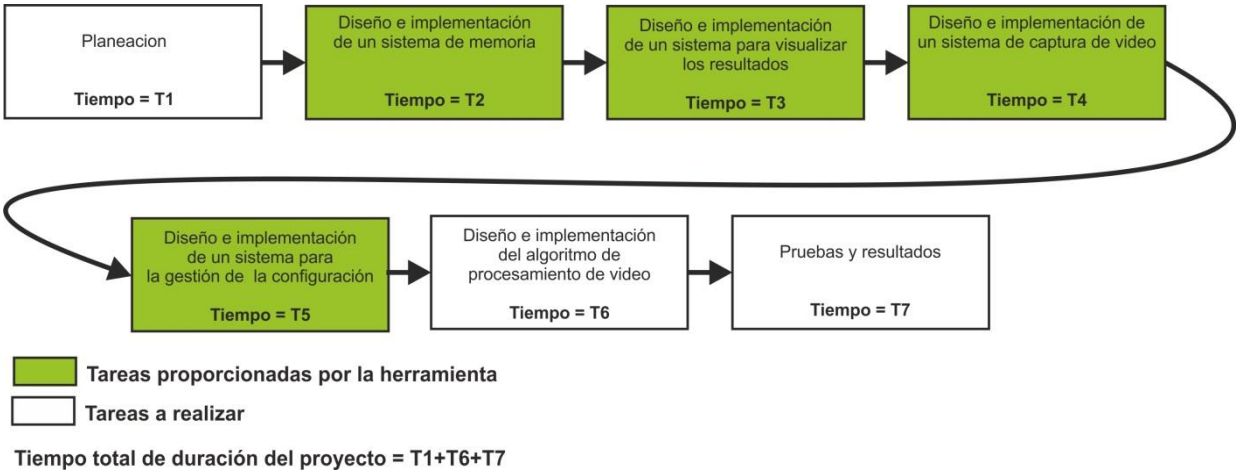

Figura 6.2 Tiempo estimado para la realización de un sistema de procesamiento de video basado en la herramienta propuesta.

La teoría constructivista proporciona orientación teórica y eficaz para el diseño y el desarrollo de herramientas de apoyo a la enseñanza universitaria en las áreas de Ciencias de la Computación. En este sentido, el estudio teórico realizado sobre esta teoría permitió extraer características esenciales con las que debe contar una herramienta de apoyo a la docencia. Estas características se tradujeron en especificaciones de diseño que se agregaron a la herramienta para su compatibilidad con el enfoque de enseñanza constructivista. La herramienta resultante proporciona un mecanismo ilustrado y simplificado para procesar y analizar algoritmos de procesamiento de vídeo, de esta forma se fomenta la construcción de conocimientos de forma activa e intencional en escenarios auténticos.

Por otra parte, la flexibilidad que provee la estructura de código abierto permite la evolución de la herramienta a partir de las aportaciones de los usuarios. De tal forma que es posible agregar o mejorar funcionalidades en la herramienta bajo un enfoque de mejora continua.

Adicionalmente, la herramienta generada cumple la característica de ser portable. En este contexto, el estándar del lenguaje de descripción de hardware VHDL, con el cual se modeló el elemento central de la herramienta, permite que los proyectos desarrollados sean fácilmente transportados a otras arquitecturas FPGAs. En este mismo sentido el modelado de los nuevos algoritmos puede ser desarrollado en otro lenguaje estandarizado, como es el caso de Verilog, o

utilizando otras herramientas o formas de modelado, como esquemático, Matlab, LabView, descripción de *cores*, por mencionar algunos.

Mediante la incorporación del repertorio de algoritmos, se permite un acercamiento al uso de la herramienta a partir de ejemplos prácticos, además durante el desarrollo de los algoritmos que integran el repertorio inicial, se pudo validar el funcionamiento de la herramienta propuesta e incorporar nuevas funcionalidades que no se habían contemplado en la planeación de la misma como: el árbitro para el acceso a la memoria y la incorporación de un módulo de captura de pixeles en el *administrador de captura de video*.

Independientemente del enfoque educativo con el cual fue desarrollada la herramienta, esta puede ser utilizada en el desarrollo de aplicaciones de visión tanto en el ámbito educativo, de investigación e industrial, esto es debido a la modularidad con la que fueron desarrollados sus componentes, los cuales pueden ser adaptados a una aplicación final o incluso eliminados si la aplicación no los requiere.

Por otro lado, para el uso adecuado de la herramienta es necesario que el usuario tenga experiencia en el desarrollo de proyectos de descripción de hardware mediante el lenguaje estandarizado VHDL (en el entorno de diseño ISE *Desing Suite*) y en el desarrollo de aplicaciones en lenguaje C/C++ usando la POO (en la herramienta Borland C++ Builder versión 6.0). Siendo esto un requisito para la introducción de la herramienta en las prácticas de procesamiento digital de video sobre lógica reconfigurable.

6.2 Trabajos futuros

Considerando el enfoque evolutivo de la herramienta no cabe duda que ésta requerirá de mejoras que permitan su evolución, además el carácter educativo en el cual está enfocado su funcionamiento implica una gran cantidad de trabajos futuros por realizar, los cuales se resumen a continuación:

- El enriquecimiento del repertorio de algoritmos es sin duda uno de los principales trabajos por realizar, siendo éste el que determinará qué nuevas funcionalidades son necesarias.
- Incorporar la herramienta a los cursos impartidos en la Universidad Tecnológica de la Mixteca con la finalidad de enriquecer el proceso de enseñanza/aprendizaje.
- Incorporar la herramienta a un laboratorio remoto, en el cual los usuarios puedan hacer uso de esta a través de Internet y tener acceso al repertorio de algoritmos a través de un blog en donde los usuarios puedan compartir sus experiencias.
- El estándar HDMI es sin duda uno de los medios principales para la transmisión de video HD, por tal motivo se propone la incorporación de un sistema de captura y visualización de video mediante este estándar.

Referencias

[ACK99] Ackenhusen J. G. *Real-Time Signal Processing: Design and Implementation of Signal Processing Systems*. Prentice Hall. Englewood Cliffs. USA, 1999.

[AHM74] Ahmed N., Natarajan T. and Rao K. R. "Discrete Cosine Transform", *IEEE Transaction on Computers*, Vol. 23, No. 1, pp. 90-93, 1974.

[ALO09] Alonso F., Manrique D., and Viñes J. M. "A moderate constructivist e-learning instructional model evaluated on computer specialists", *Computers & Education*, vol. 53, No. 1, pp. 57-65, 2009.

[APP10] Appavoo P. "The constructivist' approach to teaching computing", *International Journal of Continuing Engineering Education and Life Long Learning*, Vol. 20, No. 3-5, pp. 407-417, 2010.

[ASH08] Ashenden Peter J. *The Designer's Guide to VHDL*. Morgan Kaufmann. Tercera Ed. USA, 2008.

[ATI04] Atitallah A. B., Kadionik P., Ghozzi F., Nouel P., Masmoudi N. and Marchegay P. "Hardware platform design for real-time video applications", *Proceeding ICM 2004. The 16th International Conference on Microelectronics*, pp. 722-725, 2004.

[ATI11]Atitallah A. Ben, Loukil H. and Masmoudi N. "Fpga Design For H.264/Avc Encoder", *International Journal of Computer Science, Engineering and Applications (IJCSEA),* Vol.1, No.5, pp. 119-138, 2011.

[AUS78] Ausubel D. P., Novak J. D. and Hanesian H. *Educational psychology: A cognitive view*. Holt Rinehart and Winston. 2nd Ed. New York, USA, 1978.

[AZI11] Aziz E. S. "Teaching and learning enhancement in undergraduate machine dynamics", *Computer Applications in Engineering Education,* Vol. 19, No. 2, pp. 244-255, 2011.

[BAR00] Barberá E., Bolivar A., Calvo J., Coll C., Fuster J., García M., Grau R., Cabañas A., Manuel J., Marrero M., Mollá J., Navarro M., Onrubia J., Pozo J., Rodríguez F., Segura J., Soler M., Teberosky A., Torres M. and Yábar J. *El constructivismo en la práctica*. Editorial Graó. Barcelona, España, 2000.

[BLI10] Blikstein P. and Wilensky U. "MaterialSim: A constructionist agent-based modelling approach to engineering education", *Designs for learning environments of the future: International perspectives from the learning sciences.* Springer. pp. 17-60. New York, USA, 2010.

[BOV00] Bovik A. *Handbook of Image and Video Processing.* Academic Press. USA, 2000.

[BOV09] Bovik A. *The Essential Guide to Video Processing.* Academic Press. Ed. Elsevier Inc. Austin, Texas, USA, 2009.

[BRO05] Broers H., Caarls W., Jonker P., and Kleihorst R. "Architecture Study for Smart Cameras", *Proceedings of the European Optical Society Conference on Industrial Imaging and Machine Vision.* pp. 39–49, 2005.

[CAR97] Carretero M. *Constructivismo y educación*. Editorial Progreso. Mexico,1997.

[CAS96] Castleman K. *Digital Image Processing.* Prentice Hall. USA, 1996.

[CAV11] Cavallotti C., Merlino P., Vatteroni M., Valdastri P., Abramo A., Menciassi A. and Dario P. "An FPGA-based versatile development system for endoscopic capsule design optimization," *Sensors and Actuators A*, Vol. 172, No. 1, pp. 301–307, 2011.

[CHA08] Chase J., Nelson B., Bodily J., Wei Z., and Lee D.J. "Real-Time Optical Flow Calculations on FPGA and GPU Architectures: A Comparison Study", *Proc. 16th Int'l Symp. Field-Programmable Custom Computing Machines*, pp. 173-182, 2008.

[CHO09] Cho J., Mirzaei S., Oberg J., and Kastner R. "FPGA-based face detection system using Haar classifiers," *Proc. 17th ACM/SIGDA Int. Symp. Field-Programm. Gate Arrays*, pp. 103–112, 2009.

[CHR04] Christophersen H.B., Pickell W. J., Koller A. A., Kannan S.K., and Johnson E.N. "Small Adaptive Flight Control Systems for UAVs using FPGA/DSP Technology", *Proceedings of the AIAA Unmanned Unlimited Technical Conference*, 2004.

[COP10] Cope B., Cheung P., Luk W., and Howes L. "Performance Comparison of Graphics Processors to Reconfigurable Logic: A Case Study", *IEEE Transactions on Computers*, Vol. 59, No. 4, pp. 433-448, 2010.

[DES09] Desmouliers C., Oruklu E., and Saniie J., "FPGA-Based Design Of a High-Performance and Modular Video Processing Platform", *IEEE International Conference on Electro/Information Technology*, pp. 393 – 398, 2009.

[DES10] Desmouliers C., Oruklu E., Saniie J., and Martinez F.V. "HW/SW co-design platform for image and video processing applications on Virtex-5 FPGA using PICO", *IEEE International Conference on Electro/Information Technology (EIT)*, pp. 1-6, 2010.

[DES12] Desmouliers C., Oruklu E., Saniie J., and Martinez F.V. "Image and Video Processing Platform for FPGAs Using High-Level Synthesis", *Computers & Digital Techniques, IET*, Vol. 6, No. 6, pp. 414-425, 2012.

[DIA06] Díaz J., Ros E., Pelayo F., Ortigosa E.M., and Mota S. "FPGAbased real-time optical-flow system," *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 16, No. 2, pp. 274– 279, 2006.

[DON01] Dong K., Hu M., Ji Z., and Fang B. "Research on Architectures for High Performance Image Processing", *Proceedings of the Fourth International Workshop on Advanced Parallel Processing Technologies*, 2001.

[DOU95] Dougherty E. and Laplante P. *Introduction to Real-time Imaging*. SPIE Press/IEEE Press Understanding Science & Technology Series. USA, 1995.

[DUB84] Dubois E. and Sabri S. "Noise reduction using motion compensated temporal filtering", *IEEE Transaction on Computers.* Vol. 32, No. 7, pp. 826–831, 1984.

[FOU22] Fourier J. B. *Theorie Analytique de la Chaleur*. Chez Firmin Didot, père et fils. Paris, 1822.

[FRE02] Fresse V., Déforges O. and Nezan J.F., "AVSynDEx: A Rapid Prototyping Process Dedicated to the Implementation of Digital Image Processing Applications on Multi-DSP and FPGA Architectures", *Journal on Applied Signal Processing*, Vol. 9, pp. 990-1002, 2002.

[GAR11] Garcia I. A., and Cano E. M. "Designing and implementing a constructionist approach for improving the teaching–learning process in the embedded systems and wireless communications áreas", *Computer Applications in Engineering Education.* Vol. 22, No. 3, pp. 481-493, 2011.

[GAR13] Garcés L., Sanchez S., Brox P., and Cabrera A., "Prototipado rápido de sistemas de procesado de video basados en el VFBC de Xilinx", *RIELAC Revista de Ingeniería Electrónica, Automática y Comunicaciones*, Vol. 34, No. 1, pp. 100-109, 2013.

[GEN11] Genovese M. and Napoli E. "FPGA-based architecture for real time segmentation and denoising of HD video", *Journal of Real- Time Image Processing*, Vol. 8, No. 4, pp. 389-401, 2011.

[GIL03]Gillet, D., Geoffroy, F., Zeramdini, K., Nguyen, A. V., Rekik, Y., & Piguet, Y. "Thecockpit: An effective metaphor for web-based experimentationin engineering education", *International Journal of Engineering Education*, Vol. 19, No. 3, pp. 389-397. 2003.

[GON02] Gonzales R. and Woods R. *Digital Image Processing*. Prentice-Hall. USA, 2002.

[GOP12] Gopal J., Purushottam S., Karmakar A., and Shekhar C. "Platform-Based Extensible Hardware-Software Video Streaming Module for a Smart Camera System", *International Journal of Modeling and Optimization*, Vol. 2, No. 4, pp. 482-487, 2012.

[GUP04] Gupta N. and Sinha P. "FPGA Implementation of Fuzzy Morphological Filters", *Proceedings of SPIE-IS&T Electronic Imaging Conference on Real-Time Imaging, SPIE*. Vol. 5297, pp. 220–230, 2004.

[HAY02] Haynes S. D., Epsom H. G., Cooper R. J., and McAlpine P.L. "UltraSONIC: A Reconfigurable Architecture for Video Image Processing", *Lecture Notes in Computer Science: Field-Programmable Logic and Applications: Reconfigurable Computing Is Going Mainstream*, Vol. 2438, pp. 482-491, 2002.

[HIR10] Hiraiwa J., Vargas E. and Toral S. "An FPGA based Embedded Vision System for Real-Time Motion Segmentation", I*WSSIP 17th International Conference on Systems, Signals and Image Processing*, pp. 360-363, Rio de Janeiro, Brazil, 2010.

[HIR13] Hiraiwa J. and Amano H. "An FPGA Implementation of Reconfigurable Real-Time Vision Architecture", *27th International Conference on Advanced Information Networking and Applications Workshops (WAINA),* pp. 150-155, Barcelona, España, 2013.

[HOR81] Horn B.K.P. and Schunck B.G. "Determining Optical Flow", *Artificial Intelligence*. Vol. 17. pp. 185-203. 1981.

[HUF52] Huffman D. "A method for the construction of minimum redundancy codes," *Proceedings of the IRE*, Vol. 40, No. 9, pp. 1098–1101, 1952.

[HUN02] Hundhausen C.D. "Integrating algorithm visualization technology into an undergraduate algorithms course: ethnographic studies of a social constructivist approach", *Computers & Education*, Vol 39. Pp. 237–260. 2002.

[HUN02a] Hundhausen C.D., Douglas S.A., and Stasko J.T. "A meta-study of algorithm visualization effectiveness", *Journal of Visual Languages and Computing*, Vol 13, pp. 259–290, 2002

[HUN03] Hunter H. and Moreno J. "A New Look at Exploiting Data Parallelism in Embedded Systems", *Proceedings of the International Conference on Compilers, Architectures, and Synthesis for Embedded Systems*, pp. 159–169, 2003.

[HUR06] Hurts J. and Bull L. "A neural learning classifier system with self-adaptive constructivism for mobile robot control", *Artifical Life*, Vol. 12, No. 3, pp. 353-380, 2006.

[ILL00] Illgner K., "DSPs for image and video processing", *Signal Processing*, Vol. 80, No. 11, pp. 2323-2336, 2000.

[IRI07] Irick K., DeBole M., Narayanan V., Sharma R., Moon H., and Mummareddy S. "A unified streaming architecture for real time face detection and gender classification", *Procedings International Conference Field Programmable Logic Appl.* pp. 267–272, 2007.

[JAC05] Jack K. *Video Demystified.* Elsevier Newnes, 4 Ed., USA, 2005.

[JAI79] Jain R.C. y Nagel H. H. "On the analysis of accumulative difference pictures from image sequences of real world scenes", *IEEE Transactions on Pattern Analysis and Machine Intelligence, PAMI.* Vol.1. No.2. pp. 206–214, 1979.

[JIN10]Jin S., Cho J., Pham X. D., Lee K. M., Park S. K., Kim M., and Jeon J. W. "FPGA Design and Implementation of a Real-Time Stereo Vision System". *IEEE Transactions On Circuits And Systems For Video Technology*, Vol. 20, No. 1, 2010.

[KEH04] Kehtarnavaz N. *Real-Time Digital Signal Processing Based on the TMS320C6000*. Elsevier. Amsterdam, 2004.

[KIR04]Kirschner P. A. "Design, development, and implementation ofelectronic learning environments for collaborative learning" *Educational Technology Research and Development,* Vol. 52, pp. 39–46, 2004.

[KOK98]Kokaram A. C. *Motion Picture Restoration: Digital Algorithms for Artifact Suppression in Degraded Motion Picture Film and Video*. Springer Verlag. New York, 1998.

[KRE07] Kreijns K. P., Kirschner A., Jochems W., and Buuren V. H.,"Measuring perceived sociability of computer-supported collaborative learning environments", *Computers & Education,* Vol. 49, pp. 176–192, 2007.

[LAI07] Lai H.C., Savvides M. and Chen T. "Proposed FPGA hardware architecture for high frame rate (> 100 fps) face detection using feature cascade classifiers", *Procedings First IEEE International Conference on Biometrics: Theory, Applications, and Systems,* pp. 1–6, 2007.

[LAP03] Laplante P. and Neill C. "A Class of Kalman Filters for Real-Time Image Processing", *Proceedings of SPIE-IS&T Electronic Imaging Conference on Real-Time Imaging,* Vol. 5012, pp. 22–29, 2003.

[MAR11] Marques O. *Practical Image and Video Processing Using MATLAB.* Wiley IEEE Press, 1ra Ed., USA, 2011.

[MEY04] Meyer B. U. *Digital Signal Processing with Field Programmable Gate Arrays*. Springer, USA, 2004.

[MOE12] Moeslund T. B. *Introduction to Video and Image Processing: Building Real Systems and Applications*. Springer. USA, 2012.

[MOR07] Moreno L., Gonzalez C., Castilla I., Gonzalez E. and Sigut J. "Applying a constructivist and collaborative methodological approach in engineering education". *Computers & Education,* Vol. 49, No.3, pp. 891-915, 2007.

[MUR95] Murat T. A. *Digital Video Processing.* Prentice hall Signal Processing Series. USA, 1995.

[NAV03] Navabi Z. *Video Codec Design, Developing Image and Video Compression Systems*. John Wiley and Sons. Gran Bretaña, 2003.

[PAT06] Patten B., Arnedillo Sánchez I. and Tangney B. "Designing collaborative, constructionist and contextual applications for handheld devices". *Computers & Education,* Vol 46, pp. 294-308, 2006.

[PAV88] Pavlidis T. y Liow Y. "Integrating Region Growing and Edge Detection", *Procedings of Computer Vision and Pattern Recognition,* pp. 208-214, 1988.

[PIA52] Piaget J. *The Origins of Intelligence.* International Universities Press, Inc. New York, USA. 1952

[PIA70] Piaget J. *Genetic epistemology.* Norton. New York, USA. 1970

[PRI06] Price A., Pyke J., Ashiri D., and Cornall T., "Real time object detection for an unmanned aerial vehicle using an FPGA based vision system", *Proceedings of International Conference on Robotics and Automation,* pp. 2854-2859, 2006.

[RAO96] Rao P.K. and Hwang J.J. *Techniques and Standards for Image Video and Audio Coding*. Prentice Hall, USA, 1996.

[REE04] Reed T. R. *Digital Image Sequence Processing, Compression, and Analysis.* CRC Press, USA, 2004.

[ROS01] Roschelle J., Pea R. D., Hoadley C. M., Gordin D. N. and Means B. "Changing How and What Children Learn in School with Computer-Based Technologies". *The Future of Children*. Vol. 10, No.2, pp.76-101. 2001.

[ROT98] Roth C. H. *Digital Systems Design Using VHDL*. PWS Plushing Company. USA, 1998.

[SAI12] Said Y., Saidani T., Elhamzi W., and Atri M. "HW/SW Co-design for FPGA based Video Processing Platform" Archives Des Sciences, Vol. 65, No. 12, pp. 504-515, 2012.

[SAM10] Samper D., Santolaria J., Pastor J. J., and Aguilar J. J. "Teaching camera calibration by a constructivist methodology", *IEEE Transactions on Education*, Vol. 53, No. 4, pp. 646-652, 2010.

[SEZ93] Sezan M. I. and Lagendijk R. L. *Motion Analysis and Image Sequence Processing.* Kluwer Academic Publishers, Boston, 1993.

[SHA14] Shahriar A. *Digital Video Concepts, Methods, and Metrics: Quality, Compression, Performance, and Power Trade-off Analysis*. Apress Open, USA, 2014

[SHA48] Shannon C. E. "A mathematical theory of communication", *Bell System Technical Journal*, Vol. 27, No. 7, pp. 379–423, 1948.

[SHA89] Shalkoff R. J. *Digital Image Processing and Computer Vision.* Ed. John Wiley & Sons, USA, 1989.

[SHI08] Shi Y. Q. and Sun H. *Image and Video Compression for Multimedia Engineering: Fundamentals, Algorithms, and Standards*. CRC Press, 2nd Ed., USA, 2008.

[SHY12] Shyr W. J., "Teaching mechatronics: An innovative group project-based approach", *Computer Applications in Engineering Education*, Vol. 20, No. 1, pp. 93-102, 2012.

[TAB98]Tabatabai A. L., Jasindchi R. S., and Veen T. N. "Motion estimation methods for video compression- A Review", *Elsevier Science Ltd. J. Franklin Inst*., Vol. 335, No. 8, pp. 1411–1441, 1998.

[TER11] Terkowsky C., Jahnke I., Pleul C. and Erman Tekkaya A. "Platform for E-Learning and Telemetric Experimentation (PeTEX) – TeleOperated Laboratories for Production Engineering Education". *Proceedings of the 2011 IEEE Global Engineering Education Conference (EDUCON) – Learning Environments and Ecosystems in Engineering Education*. IAOE. pp 491-497, 2011.

[TSA11] Tsai W. T., Li W., Elston J., and Chen Y. "Collaborative learning using wiki web sites for computer science undergraduate education: A case study", *IEEE Transactions on Education*, Vol. 54, No. 1, pp. 114-124, 2011.

[VEN05] Venugopal S., Castro-Pareja C., and Dandekar O. "An FPGA-Based 3D Image Processor with Median and Convolution Filters for Real-Time Applications", *Proceedings of Electronic Imaging Conference on Real-Time Imaging, SPIE*. Vol. 5671, pp. 174–182, 2005.

[VON94] von Glasersfeld, E. "A constructivist approach to teaching". *Constructivism in education*. In L. P. Steffe & J. Gale (Eds.). New Jersey, USA. 1994.

[VON95] von Glasersfeld E. *Radical constructivism: A way of knowing and learning*. The Falmer Press, London, 1995.

[VYG78] Vygotsky L. *Mind in society: The development of higher psychological pro-cesses.* Cambridge MA: Harvard University Press. USA, 1978.

[WAN02]Wang Y., Ostermann J. and Zhang Y.Q. *Video Processing and Communications.* Prentice Hall, Signal Processing Series. USA, 2002.

[WIL99] Wilensky U. "NetLogo". Center for Connected Learning and Computer-Based Modeling. http://ccl.northwestern.edu/netlogo: Evanston, IL. 1999.

[WON05] Wong P. and Bhavana M. "Hardware in Process: Mobile Handset Cameras Challenge Image Processors", *Optical Engineering Magazine.* Vol. 5, No. 9, pp. 15-17, 2005.

[WOO12]Woods J.W. *Multidimensional Signal, Image and Video Processing and Coding*. Academic Press, 2nd edición. USA, 2012.

[XIL85] Xilinx, "XC2000 Logic Cell Array Families", 1985.

[YAL81] Yalamanchili S. and Aggarwal J.K. "Motion and image differencing", *Pattern recognition and image processing. Proc. IEEE Computer Society conference*. Vol. 1595 no.8. pp. 211–216, New York, USA, 1981.

[YOU82] Young S. *Real time languages design and development*. Ellis Horwood Ltd, USA, 1982.

[ZIL01] Ziliani F. and Cavallaro A. "Image Analysis for Video Surveillance Based on Spatial Regularization of a Statistical Model-Based Change Detection", *Journal of Real-Time Imaging.* Vol. 7, No. 5, pp. 389-399, 2001.

[ZUR04] Zurita, G. and Nussbaum, M. "A Constructivist Mobile Learning Environment Supported by a Wireless Handheld Network", *Journal of Computer Assisted Learning*, Vol. 20, pp 235-243, 2004.

Acrónimos

A*CK*	*Acknowledgement.*
ASIC	*Application Specific Integrated Circuit.*
CL	*Column address strobe Latency.*
CLB	*Configurable Logic Blocks.*
CMOS	*Complementary Metal Oxide Semiconductor.*
DCT	*Discrete Cosine Transform.*
DDR	*Double Data Rate.*
DFT	*Discrete Fourier Transform.*
DLL	*Dynamic Link Library.*
DLP	*Data Level Parallelism.*
DSP	*Digital Signal Processor.*
EDA	*Electronic Design Automation.*
FIFO	*First In, First Out.*
FPGA	*Field Programmable Gate Array.*
FTDI	*Future Technology Devices International.*
FV	*Frame Valid.*
GPU	*Graphics Processing Unit.*
HDL	*Hardware Description Language.*

HDMI	*High Definition Multimedia Interface.*
HDTV	*High Definition Television*
HLS	*High Level Synthesis.*
HW	*Hardware.*
I2C	*Inter Integrated Circuit.*
IDCT	*Inverse Discrete Cosine Transform.*
IDFT	*Inverse Discrete Fourier Transform.*
IFP	*Image Flow Processor.*
ILP	*Instruction Level Parallelism.*
IP DMA	*Intellectual Property Direct Memory Access.*
ISO	*International Organization for Standardization*
ITU	*International Telecommunication Union*
IVPP	*Image and Video Processing Platform*
JTAG	*Joint Test Action Group.*
LC	*Logic Cells.*
LCD	*Liquid Crystal Display.*
LUT	*Look-Up Table*
LV	*Line Valid.*
MAE	*Mean Absolute Error.*
MB	*Macro Bloques.*
MPEG	*Moving Picture Experts Group.*
MSE	*Mean Squared Error.*
NTSC	*National Television System Committee.*
PBL	*Project-based learning.*
PC	*Personal Computer.*
PeTEX	*Platform for E-Learning and Telemetric Experimentation*
PLD	*Dispositivos Lógicos Programables.*
PLL	*Phase Locked Loop*
PNSR	*Peak Signal to Noise Ratio*
POO	*Programación Orientada a Objetos*
RGB	*Red Green Blue*
RISC	*Reduced Instruction Set Computer.*
SCPV	*Sistema de Captura y Procesamiento de Video*
SDRAM	*Synchronous Dynamic Random-Access Memory.*
SECPV	*Sistema Embebido de Captura y Procesamiento de Video basado en un FPGA*
SPVTR	*Sistemas de Procesamiento de Video en Tiempo Real.*
SODIMM	*Small Outline Dual In line Memory Module*

SPI	*Serial Peripheral Interface.*
SW	*Softhware.*
TFT LCD	*Thin Film Transistor-Liquid Crystal Display*
TI	*Tecnologías de la Información*
UART	*Universal Asynchronous Receiver-Transmitter.*
USB	*Universal Serial Bus.*
VCP	*Virtual Com Port.*
VGA	*Video Graphics Array.*
VHDCI	*Very High Density Cable Interconnect.*
VHDL	*Very High Speed Integrated Circuit Hardware Description Language*
VPP	*Video Processing Platform*