UNIVERSIDAD POLITÉCNICA DE VALENCIA

DEPARTAMENTO DE INGENIERÍA ELECTRÓNICA

ADQUISICIÓN Y PROCESAMIENTO DIGITAL DE IMÁGENES PARA LA OBTENCIÓN DE LA TRAYECTORIA DE LOS VECTORES DE POSICIÓN DEL CAMARÓN Y LA JAIBA

Tesis Doctoral

Presentada por:
M.M Carlos Alberto Luján Ramírez

Dirigida por:
Dr. Francisco José Mora Más

Valencia, España, Septiembre 2012

AGRADECIMIENTOS

Gracias a DIOS, por permitirme vivir día tras día al lado de mi familia.

Agradezco al Dr. Francisco José Mora Más por sus preciados consejos me han permitido aprender, así como por su paciencia y su valioso tiempo dedicado a la dirección; por su hospitalidad que junto con su familia me hicieron más agradables mis estancias en España.

Agradezco al Dr. Ramón Atoche Enseñat, por sus enseñanzas y consejos que han quedado plasmados en esta investigación.

Agradezco al Dr. Jesús Sandoval Gio, por su tiempo que le dedico en la revisión de esta Tesis y por sus comentarios al respecto.

Gracias a mis papás por todas sus enseñanzas, amor y ejemplo con el que siempre me guiaron, así como a mi hermano Miguel.

Deseo agradecer también al Instituto Tecnológico de Mérida, a la Dirección General de Educación Superior Tecnológica, ANUIES y a la Universidad Politécnica de Valencia por haberme dado la oportunidad de llevar a cabo estos estudios.

Gracias a mi esposa Leidy, a mis hijos Margarita, Carlos y Kamila por motivarme a terminar este trabajo el cual les dedico.

A todos aquellos amigos y familiares que de alguna forma me alentaron con sus palabras.

Carlos Alberto Luján Ramirez

RESUMEN

El procesamiento digital de imágenes es un área de la ciencia y la tecnología cuyo desarrollo ha crecido enormemente durante los últimos años. Este desarrollo se debe entre otras cosas a los grandes avances en la fabricación de los circuitos integrados y a las nuevas tecnologías de computación.

En el ámbito del procesamiento de imágenes esta Tesis se avoca analizar el impacto del hardware y el software en un sistema de adquisición de imágenes en tiempo real que se usa en la investigación del comportamiento depredador-victima de animales marinos.

Esta investigación nace del estudio de la conducta de animales marinos, y en específico en el análisis de las trayectorias de escape del camarón *Litopenaeus vannamei*, el registro de los eventos se graban en video y un investigador posteriormente revisa los mismos para encontrar las interacciones entre la jaiba y el camarón, y una vez detectadas dentro de los videos, se evalúa cuadro por cuadro cada una de estas interacciones para marcar en el monitor los puntos importantes, y poder calcular ángulos (manualmente en el monitor), velocidades (manualmente, en base a distancia medida en el monitor y número de cuadros transcurridos), entre otras. La respuesta inicial de escape del animal, está determinada por el movimiento rotacional y la energía cinemática con la que se mueve. En esta Tesis se plantea una solución para resolverlo de manera automática, aumentando la eficiencia de los investigadores y reduciendo los errores debido a los métodos manuales que se están utilizando.

Esta Tesis tiene como objetivo fundamental generar una instrumentación (hardware y software) para el seguimiento en tiempo real de diferentes especies en el estudio de la conducta de animales marinos, en especifico, en esta Tesis los experimentos que se realizaron fueron para estudiar la conducta de camarones ante la presencia de un depredador natural (jaiba). Sin embargo los algoritmos desarrollados pueden ser adaptados fácilmente para el estudio de animales fusiformes y crustáceos con quelas. Debido a las exigencias en tiempo de procesamiento de las imágenes fue necesario emplear nuevos métodos para reducir el tiempo de procesamiento de las imágenes, lo cual nos ofrece más información sobre la posición y movimiento de los animales en estudio, y con esto predecir mejor sus trayectorias.

Se presenta una revisión rápida de los métodos básicos de procesamiento de imágenes, abordando de manera general los filtros que permiten resaltar características en la imagen, dando especial importancia a aquellos que permiten resaltar los bordes en la imagen, el borde es el que proporciona los datos necesarios para el cálculo de los vectores de posición y siendo una de las etapas que tarda mucho en el procesado de la imagen, es aquí en donde proponemos un método para acortar el tiempo siendo este una de las aportaciones en esta Tesis.

Se proponen mejoras en el algoritmo de localización y seguimiento de trayectorias de los individuos basado en la detección de bordes de imagen. Tomando en cuenta que el método de Susan es en uno de los más rápidos y muy confiable para la detección de bordes, se comparó este con el método propuesto y esta etapa del proceso se logró ejecutar 4000 veces más rápido, considerando que dentro de las mejoras se evita el procesado general utilizando una selección de las áreas a tratar.

Para la adquisición de los datos se analizó el protocolo de comunicación Channel Link implementándose en los FPGAs de Xilinx y Altera para analizar quien tiene el mejor desempeño y poder seleccionar al mejor. Para realizar el procesamiento de las imágenes se plantea una arquitectura mixta en la que se puede integrar a los DSPs, a los FPGAs e incluso a los microprocesadores embebidos con el fin de reducir los tiempos. En cada una de las etapas del proyecto se dan al menos dos alternativas de solución, de las que se toman las mejores características de cada tecnología con el fin de obtener los mejores resultados en el tiempo.

ABSTRACT

In the last years, a scientific and technological field which has developed greatly is the digital image processing. This development has been brought about, among other things, by the extraordinary innovations in the production of closed-circuits and also by the new computational technology.

This paper is aimed to analyze the impact that the hardware and software have in a real-time image acquisition system used in the study of the predator-prey behavior of sea animals in image processing.

This research stems from the study of the behavior of marine animals, particularly from the analysis of the escape trajectories of the shrimp, *Litopenaeus vannamei*. The events are recorded in video; and later, a researcher checks them to find interactions between crabs and shrimps. The records of the interactions are videotaped; once the interactions are detected, each one is evaluated frame by frame to mark the important points and to calculate angles (by hand in the screen), as well as the speeds (by hand based on a distance measured on the screen and the numbers of frames in an elapsed time). The initial escape response of the animal is determined by the rotational movements and the kinesthetic energy with which it moves. The purpose of this paper is to propose a solution to solve the above automatically so the efficiency of the researchers increases and the mistakes committed by the manual methods decrease.

This paper aims fundamentally to design an instrumentation (hardware and software) for the study in real time of the behavior of different species of sea animals. The specific experiments performed in this paper were performed on the response conduct of the shrimp to a natural predator (crab). Nevertheless, the algorithms developed in this paper can be adapted easily to the study of other fusiforms and chelae crustaceans . Due to the time demands in image processing, it was necessary to use new methods to decrease the time of image processing. This produces more information about the position and movement of the studied animals making possible more accurate predictions of their trajectories.

A quick review of the basic methods of image processing is introduced; presenting, in a general way, the filters that allow the highlighting of characteristics in an image, and emphasizing those that highlight the edges in an image. The edge provides the data necessary to calculate the position vectors. Since, the latter is the longest stage in the image processing; a method to shorten this time is one contribution of this paper.

Improvements in the localization algorithm and the monitoring of the individuals trajectories based on the detection of the image edges are proposed. Taking into consideration that the Susan method to detect edges is one of the fastest and most reliable, it was compared to the method proposed here. As a result, this stage of the process was performed 4000 times faster; for due to the improvements, instead of using the general process, a selection of the areas to be analyzed is used.

To obtain the data, the Channel Link communication protocol was analyzed by implementing it in the Xilinx and Altera FPGAs which performs better so the best could be chosen. To perform image processing, a mixed architecture is suggested in which the DSPs can be integrated to the FPGAs, and even to the encompassed microprocessors, to reduce times. In each phase of the project, two solution choices are given in which the best characteristics of each technology is considered so the best time results are obtained.

RESUM

El processament digital d'imatges és una àrea de la ciència i la tecnologia que s'ha desenvolupat enormement els últims anys. Aquest desenvolupament es deu, entre altres causes, als grans avanços que s'han produït en la fabricació dels circuits integrats i a les noves tecnologies de computació.

En l'àmbit del processament d'imatges, aquesta tesi pretén analitzar l'impacte del maquinari i el programari en un sistema d'adquisició d'imatges en temps real que s'usa en la investigació del comportament depredador-víctima d'animals marins.

Aquesta investigació naix de l'estudi de la conducta d'alguns animals marins, i específicament, del'anàlisi de les trajectòries de fuita de la gambeta*Litopenaeusvannamei*; el registre dels esdeveniments es grava en vídeo i un investigador posteriorment els revisa per trobar les interaccions entre el cranc de mari la gambeta, i una vegada detectades dins dels vídeos, s'avalua quadre per quadre cadascuna d'aquestes interaccions per marcar en el monitor els punts importants, i poder calcular angles (manualment en el monitor), velocitats (manualment, sobre la base de la distància mesurada en el monitor i el nombre de quadres transcorreguts), etc. La resposta inicial de fuita de l'animal està determinada pel moviment rotacional i l'energia cinemàtica amb la qual es mou.En aquesta tesi es planteja una solució per atrobar això de manera automàtica, amb la qual cosa s'augmenta l'eficiència dels investigadors i es redueixen els errors causats pels mètodes manuals que s'utilitzenavui.

Aquesta tesi té com a objectiu fonamental generar una instrumentació (maquinari i programari) per al seguiment en temps real de diferents espècies en l'estudi de la conducta d'animals marins; específicament, en aquesta tesi els experiments que es van portar a terme van tenir l'objectiu d'estudiar la conducta de les gambetes davant la presència d'un depredador natural (el cranc de mar). No obstant això, els algorismes desenvolupats es poden adaptar fàcilment a l'estudi d'animals fusiformes i crustacis amb pinces (*chelae*). A causa de les exigències en temps de processament de les imatges, va caldre emprar nous mètodes per a reduir aquest temps de processament, la qual cosa ens ofereix més informació sobre la posició i el moviment dels animals en estudi, i amb això esdevé possible predir-ne millor les trajectòries.

Es presenta una revisió ràpida dels mètodes bàsics de processament d'imatges, en què s'aborda de manera general els filtres que permeten destacar característiques en la imatge, donant especial importància a aquells que permeten ressaltar les vores en la imatge. La vora és allò que proporciona les dades que calen per al càlcul dels vectors de posició, i, atès que és una de les etapes que tarda molt en el processament de la imatge, és ací que proposem un mètode per a escurçar aquest temps: aquesta és una de les aportacions de la present tesi.

Es proposen millores en l'algorisme de localització i seguiment de trajectòries dels individus basat en la detecció de vores d'imatge. Tenint en compte que el mètode de Susan és en un dels més ràpids per a la detecció de vores,i molt segur, aquest mètode es

va comparar amb el proposat, i aquesta etapa del procés es va aconseguir executar 4.000 vegades més ràpidament, considerant que dins de les millores s'evita el processament general utilitzant una selecció de les àrees que cal tractar.

Per a l'adquisició de les dades es va analitzar el protocol de comunicació ChannelLink implementant-se en els FPGA de Xilinx i Altera per analitzar quin dóna millors resultats i així poder seleccionar el millor. Per portar a terme el processament de les imatges es planteja una arquitectura mixta en la qual es pot integrar els DSP, els FPGA i, fins i tot,els microprocessadors incrustats,afi de reduir els temps. En cadascuna de les etapes del projecte es donen almenys dues alternatives de solució, de les quals es prenen lesmillors característiques de cada tecnologia amb la finalitat d'obtenir els millors resultats en el temps.

Índice

Capítulo 1 – INTRODUCCIÓN

Capítulo 2 – ESTADO DEL ARTE

Capítulo 3 – ANÁLISIS Y DISEÑO DEL SISTEMA

Capítulo 4 – TEST Y RESULTADOS ALCANZADOS

Capítulo 5 - CONCLUSIONES Y TRABAJOS FUTUROS

Bibliografía

Anexos

Índice de Figuras

Índice de Tablas

CAPITULO 1 - INTRODUCCIÓN

1. Motivación

El procesamiento digital de imágenes es un área de la ciencia y la tecnología cuyo desarrollo ha crecido enormemente durante los últimos años. Este desarrollo se debe entre otras cosas a los grandes avances en la fabricación de los circuitos integrados y a las nuevas tecnologías de computación. Los ordenadores actuales nos permiten realizar volúmenes de procesamiento a velocidades nunca vistas. Como consecuencia los investigadores han podido incursionar en el campo de la visión artificial, en la cual se conjugan algoritmos de tratamiento de imágenes y procesos estadísticos entre otros, para poder extraer de las imágenes información de utilidad para algún proceso, dependiendo de la aplicación puede requerirse procesar las imágenes de manera independiente o realizar el proceso en una secuencia de video.

En la actualidad es posible resolver problemas relativamente simples procesando a la velocidad del video estándar en un ordenador común, sin embargo cuando se requiere resolver problemas complejos a grandes velocidades, es necesario incluir hardware especializado que apoye al ordenador en el procesamiento de las imágenes, incluso se necesita utilizar sistemas en paralelo que comparten información o bien hacer uso de la mezcla de tecnologías.

En esta Tesis se presenta la implementación de algoritmos de visión artificial como parte del desarrollo de una herramienta para la investigación de biólogos marinos de la Unidad Multidisciplinaria de Docencia e Investigación (UMDI) en Sisal Yucatán, dónde actualmente estudian el comportamiento del camarón de la especie *Litopenaeus vannamei* ante un depredador, en este caso una jaiba *Callinectes ornatus*, estos estudios se basan en experimentos que se graban bajo condiciones controladas, para posteriormente analizar las secuencias de imágenes en busca de ciertos parámetros ya establecidos, que permiten estudiar las trayectorias de escape del camarón. El análisis de las trayectorias de escape puede dar información sobre las estrategias frente a los depredadores de las especies presa y tienen aplicaciones relativas al medio ambiente. Debido a que los investigadores se pasan horas en dicho análisis para obtener estos parámetros, en esta Tesis hago uso de algoritmos de visión artificial para extraer de manera automática, los datos necesarios, de una secuencia de imágenes, de una determinada escena, de esa manera el tiempo que empleaban los investigadores queda libre para otras actividades, además de evitar el error que puede ocasionar el cansancio al permanecer varias horas en análisis de dichas imágenes.

Para realizar la extracción de los datos se planteó un proceso que consta de los siguientes pasos: la adquisición de la imagen, eliminación todo lo que no es de interés en la imagen mediante una resta de la imagen de fondo con la imagen actual, aplicar un filtro pasa bajo para eliminar cualquier desperfecto que pueda arrojar la resta de imágenes mejorando la calidad de la imagen obtenida, finalmente segmentar al resultado del filtro para extraer las características que me interesan transmitir, en este caso el centro de masa y los puntos más lejanos del centro de masa. En todos estos pasos hago un análisis de al menos dos opciones con algoritmos y hardware para elegir el mejor. El problema a vencer fue lograr hacer todo este tratamiento en lo que tarda un frame, es decir, en menos de 33.3 milisegundos.

Para la adquisición de la imagen se requería de una etapa de deserializadores, se evaluaron diferentes opciones: un circuito integrado fabricado por National Semiconductor listo para manejar el protocolo cámara link, su inconveniente fue la necesidad de una circuitería adicional para funcionar, además de un hardware para el almacenamiento de las imágenes adquirida. Otra opción fue el uso de FPGAs que tienen deserializadores embebidos pero no preparados

para el uso del protocolo cámara link por lo que complicaba su uso. Y finalmente el utilizado, la implementación dentro del FPGA con flip flops emplazados de manera manual para mejorar su desempeño, esta tiene la versatilidad de ser más general y puede implementarse en cualquier FPGA.

Se presenta una comparación de diferentes formas de implementación hardware de los algoritmos, lo cual permitió elegir la implementación óptima para cada algoritmo dentro del proceso completo. Esta comparativa permite observar que en procesos de visión artificial, la combinación óptima incluiría coprocesadores hardware para las etapas iniciales de procesamiento de imágenes, que por su naturaleza son inminentemente paralelizables, lo cual permite explotar al máximo la capacidad intrínseca de paralelizar de los FPGAs, mientras que en etapas posteriores cuyo objetivo es la extracción de listas de características de la imagen, cuyos algoritmos normalmente tienen carácter secuencial, es preferible implementarlas en software, en específico se plantea la utilización de DSPs, y se muestra que la combinación planteada permite la evaluación en tiempo real (Superior a 76 frames por segundo en resolución VGA estándar). Esta combinación no solamente optimiza el tiempo de procesado, sino que también optimiza el tiempo de implementación. Ya que realizar diseños Hardware en HDL es un proceso costoso en tiempo [Gribbon, 2005], es adecuado reducir su utilización a la etapa inicial del proceso, que normalmente involucra operaciones sencillas que deben repetirse masivamente para cada pixel de la imagen, permitiendo que la implementación hardware sea relativamente sencilla, mientras que los procesos de segmentación de alto nivel que involucran operaciones más complejas que resultaría difícil implementarlas en HDL, pueden ser fácilmente implementados en C para un DSP.

Se evaluó el uso de procesadores embebidos en el FPGA, y se comprobó que su uso es adecuado como elementos de control del proceso, pero no como procesadores de la imagen, ya que para esto resultan demasiado lentos.

Para implementar los algoritmos y lograr la ejecución de estos en el menor tiempo posible, se optimizó su implementación, tanto hardware como software, y en algunos casos se plantearon modificaciones estructurales que aceleraron grandemente la ejecución de dichos algoritmos. Las dos modificaciones más importantes son las siguientes: al comprender el proceso fue posible eliminar la transformación lineal de la imagen del formato Bayer a RGB, dado que no aportaba información adicional, reduciendo con esto en aproximadamente 38.019ms el tiempo empleado para el procesado inicial de la imagen (etapa de resta de la imagen de fondo y umbralización), con lo cual se aceleró 2.9 veces esta etapa del proceso. Otra aportación importante se realizó en el cálculo del vector de posición, en donde se realizaron optimizaciones con respecto a la implementación del método de comparación entre distancias algebraicas, así como la eliminación de la etapa de detección de bordes, cambiándola por un método de caminata sobre el borde que permitieron reducir de 116.8 ms a 0.027ms tiempo empleado, acelerando 4325 veces este proceso (comparación entre el algoritmo propuesto inicialmente y la optimización final de la etapa de extracción de los puntos más lejanos). Este algoritmo esta validado solo para determinadas regiones.

La Tesis se desarrolla en cinco capítulos, en el primero describo la problemática que presentan los investigadores al realizar el análisis de comportamiento de dos especies marinas uno como agresor y el otro como víctima; asimismo planteo los objetivos para la solución a esta problemática y el impacto que esta Tesis pueda tener.

En el capítulo 2, presento el marco teórico necesario para poder cumplir los objetivos planteados. En el capítulo 3, se muestran las implementaciones realizadas comenzando con la implementación en hardware de la adquisición de la imagen, el convertidor Bayer a RGB, los algoritmos de resta, umbralización, filtro pasa bajo y la segmentación. En el capítulo 4, exhibo los resultados de la comparativa de la implementación del hardware para la recuperación de los datos; la determinación del uso del código RGB para las imágenes; la **c**omparativa entre las diferentes alternativas de implementación de los algoritmos de resta, umbralización, filtro pasa bajo y la obtención de los puntos más lejanos del camarón y la jaiba. En el capítulo 5 se dan las conclusiones y se plantean trabajos futuros derivados de esta Tesis.

En el anexo A se muestra un resumen técnico del Bus LVDS, en el anexo B describo brevemente a un FPGA y a las dos familias en las que realizo las diferentes implementaciones de hardware, en el anexo C presento una introducción del DSP DSK6416 el que se utiliza para probar los algoritmos implementados, resaltando sus principales características.

1.2. Planteamiento del Problema

En la investigaciones sobre conducta de animales marinos llevadas a cabo en la UMDI, de la Facultad de Ciencias en la UNAM campus Sisal, y en específico en el estudio de las trayectorias de escape del camarón *Litopenaeus vannamei*, el registro de los eventos se graban en video y un investigador posteriormente revisa los mismos para encontrar las interacciones entre la jaiba y el camarón, y una vez detectadas dentro de los videos, se evalúa cuadro por cuadro cada una de estas interacciones para marcar en el monitor los puntos importantes, y poder calcular ángulos (manualmente en el monitor), velocidades (manualmente, en base a distancia medida en el monitor y número de cuadros transcurridos), entre otras. La respuesta inicial de escape del animal, está determinada por el movimiento rotacional y la energía cinemática con la que se mueve.

La investigación busca automatizar la extracción de parámetros del video, de hecho se busca alcanzar la extracción en tiempo real de dichos parámetros, para evitar los grandes volúmenes de almacenamiento que se requieren con el método actual, y con ello almacenar los videos completos de cada experimento. Lograr esto reducirá significativamente la inversión de tiempo utilizada por los investigadores biólogo-marinos en su tarea de recaudación de datos para hacer análisis estadísticos del comportamientos de alguna especie marina, en este caso el camarón de la especie *Litopenaeus vannamei.*

En esta Tesis, llevo a cabo el estudio de diferentes algoritmos de procesamiento de imágenes, de igual forma muestro las adaptaciones realizadas para adecuarlos al problema estudiado y finalmente propongo mejoras que permitan acelerar su funcionamiento con la finalidad de alcanzar una velocidad de extracción de parámetros igual a la velocidad del video estándar.

Para lograr esto propongo diferentes versiones de implementación que incluyen la mezcla de procesamiento en el FPGA, DSP y microprocesadores.

Finalmente realizo un análisis de cada una de las diferentes implementaciones y establezco comparativas entre ellas para definir la mejor opción.

1.3. Objetivos del estudio

La finalidad es generar una instrumentación basado en el co-diseño (hardware-software) que permita la monitorización de experimentos biológicos con jaibas y camarones, y que almacene los vectores de sus posiciones para el posterior estudio de las trayectorias de escape de los mismos. Se prioriza la implementación de una estructura hardware sobre una solución puramente software ya que esto último requeriría que un ordenador se dedicara exclusivamente a dicho proceso, y aún así no podría procesar por completo las imágenes a la resolución y velocidad deseadas, en cambio, una implementación hardware permitirá liberar al ordenador central de una gran carga computacional, permitiéndole dedicarse a los algoritmos de nivel superior.

Se plantea una arquitectura mixta en la que se utilizará la implementación hardware (en un FPGA) de las etapas iníciales (altamente paralelizables), un microprocesador embebido en un FPGA y un DSP para procesos que por su naturaleza son altamente secuenciales.

La investigación se implementará modularmente permitiendo así, que los módulos puedan ser utilizados posteriormente para nuevas investigaciones. Los algoritmos generados para cada módulo se realizarán al menos de dos formas diferentes para comparar los resultados de cada opción y, la que ofrezca un mejor rendimiento y ventajas será la elegida en la implementación final del proyecto.

Buscando aprovechar las mejores características de cada tecnología se pretende acelerar la velocidad de procesamiento hasta donde el hardware lo permita (la meta es lograr treinta imágenes por segundo), y proporcionar una solución que permita posteriormente una implementación compacta y de bajo consumo.

La contribución principal de esta tesis consiste en el diseño de una plataforma embebida específica para el procesamiento en tiempo real de una secuencia de imágenes para determinar el comportamiento depredador-víctima de animales marinos. Dicha plataforma logra la integración de varias tecnologías FPGAs, DSPs y MCUs mediante el co-diseño HW/SW para su procesamiento en tiempo real. El autor considera que las innovaciones realizadas en la transformación algorítmica (SUSAN) y la arquitectura diseñada para la adquisición de la imagen mediante el FPGA, son parte de las funciones realizadas por la plataforma mediante el esquema de co-diseño HW/SW.

1.4. Impacto o relevancia

Considerando las ventajas y desventajas de los distintos métodos utilizados en el análisis de conducta de animales, no es difícil pensar que podrían diseñarse herramientas que automaticen este proceso sin tener que hacer fuertes inversiones monetarias y que además, hagan posible el desarrollo de experimentos más ambiciosos en esta misma área. La tarea de analizar los videos imagen a imagen, para después manualmente realizar mediciones con programas ambiguos para visualización de fotografías, puede ser simplificada con la creación de herramientas que permitan la obtención automática de las características de interés de la imagen, lo cual como puede verse en las metodologías de [Arnott, 1998] y [Arnott, 1999] es un trabajo extenuante

que requiere de muchísimas horas hombre ya que se requería de aproximadamente 40 experimentos de alrededor de 8 horas cada uno, para poder tener una muestra estadística válida, esto además involucra la necesidad de contar con una gran cantidad de espacio de almacenamiento para poder guardar todas estas horas de video.

Poder resolver esto de manera automática, aumentará la eficiencia de los investigadores y reducirá errores debido a los métodos manuales que se están utilizando.

Lograr el reconocimiento de un objeto o animal en una imagen, en tiempo real, es la base para implementar algoritmos de seguimiento por secuencia de imágenes, basados en dos o más imágenes consecutivas de una misma escena, buscando aprovechar la variación espacio-tiempo que se genera, para obtener la máxima información posible a partir de dichas secuencias de las imágenes. Ejemplo de esto es el análisis de secuencia de imágenes satelitales, ya que algunas veces es necesario observar los cambios que se dieron en una determinada región al cabo de un tiempo, la evaluación automática mediante algoritmos de seguimiento evitaría la necesidad de tener a alguna persona vigilando que exista un cambio significativo en dicha región.

También, la inteligencia artificial basada en visión permitirá la creación de sistemas más completos que puedan utilizarse en percepción remota, para guiar robots industriales, en robótica móvil o en cualquier tarea que requiera un sistema capaz de reconocer su ambiente para moverse en el e interactuar con los objetos a su alrededor, incluso con objetos en movimiento. El lograr esto empleando tecnología electrónica de alta integración y bajo consumo, nos permitirá contar con un dispositivo compacto que pueda ser utilizado por sistemas móviles de gran autonomía.

Mientras que para los seres humanos la vista es el sentido que más utilizan y que más desarrollado tenemos, para los robots (entendiéndose por robot cualquier sistema automático que interactúa con su ambiente), su sistema de visión actualmente es muy pobre (se reduce en la mayoría de los casos a una detección de presencia).

Con esta Tesis se obtiene un instrumento importante para obtener información que permita la descripción detallada de los factores de carácter conductual (trayectorias de escape) que determinan el resultado de un ataque. Siendo así una herramienta importante para probar hipótesis relativas a las interacciones depredador-presa, aspecto fundamental para entender el efecto que dichas interacciones tienen en la estructura y funcionamiento de las comunidades estuarinas, lo cual permitirá a futuro establecer mejores políticas de protección y explotación para estas poblaciones. De igual forma, un mejor entendimiento de la conducta de las especies explotadas, permitirá mejorar los métodos utilizados en las granjas.

Otra aplicación de los métodos diseñados se encuentra en el estudio del comportamiento de los camarones ante diferentes estímulos alimenticios, en el cual se marcan zonas “invisibles” en un estanque, en las que predomina el olor a cierto alimento, y se evalúa el tiempo de permanencia del camarón en dichas zonas, determinando su preferencia. Este estudio se utiliza para buscar el alimento preferido por los camarones en diferentes etapas de su desarrollo para establecer su dieta en las granjas intensivas, con lo cual se busca obtener un producto de mejor calidad en el menor tiempo posible. La importancia de establecer métodos que permitan automatizar este estudio es lograr su utilización permanente para poder adaptar las dietas tanto a las diferentes regiones geográficas como a las diferentes épocas del año.

El proceso planteado puede ser modificado para estudiar la conducta de diferentes especies lo cual debe llevarnos al entendimiento de estas formas de vida para que podamos protegerlos y/o mejorar el aprovechamiento racional de los mismos como recurso natural renovable o bien para aprender o mejorar métodos de cultivo intensivo.

1.5. Antecedentes

El camarón blanco del pacífico *Litopenaeus vannamei* es la especie más cultivada en América y ha sido introducida en granjas del Atlántico, desde Brasil hasta Estados Unidos. En México, su cultivo continúa extendiéndose, principalmente hacia la región costera del sureste, donde habita el camarón blanco del Golfo, *L. setiferus*. Esto ha provocado la preocupación sobre riesgos ecológicos potenciales al presentarse una liberación masiva de la especie no nativa al ambiente natural. Siendo necesario estudiar su comportamiento.

En los estudios biológicos clásicos, realizados con el objetivo de conocer las características de la conducta de escape de Camarón *Litopenaeus vannamei* en respuesta al ataque de la jaiba *Callinectes ornatus*, se usa el siguiente protocolo experimental: los experimentos se llevan a cabo en una habitación de fibra de vidrio provista de un sistema de aire acondicionado con el interior pintado de color negro para tener las condiciones de iluminación controladas. Cada experimento consiste en una serie de pruebas individuales (n = 40±1 para cada especie de camarón), en las que se induce una respuesta de escape en un camarón ante ya fuera el ataque de una jaiba o un estímulo artificial. Al terminar cada experimento, se obtuvieron medidas morfométricas del camarón (longitud total LT) y de la jaiba (Largo del caparazón LC; Ancho del caparazón AC) y el dimorfismo entre las quelas (cortadora y trituradora), considerándose como diestra la jaiba que presenta la quela trituradora del lado derecho.

Antes de cada evento, las jaibas se mantuvieron en un periodo de ayuno de dos días, para inducir un mayor interés de la jaiba hacia el camarón y que el ataque se diera más rápidamente. Para determinar las características de la conducta de escape de esa especie de camarón ante la aproximación del depredador, se realiza una serie de experimentos en una “arena” circular de fibra de vidrio translúcido (1.26 m de diámetro y 7 cm de profundidad de agua, a 28±1°C de temperatura y salinidad de 39±1 ppm), iluminada por la parte inferior a una distancia de 30 cm. Dentro de la arena experimental se colocan un camarón juvenil y una jaiba, fuera de cualquier contacto físico mediante un cilindro de PVC (20 cm de diámetro y 12.5 cm de altura), durante 15 minutos con aireación constante. El cilindro presenta perforaciones para permitir el intercambio de sustancias químicas entre los organismos [Arnott, 1999].

Al iniciar cada prueba, se suspende la aireación y se retira la barrera mediante un dispositivo mecánico operado por detrás de un telón. Se filma el evento utilizando una cámara digital convencional Canon Elura 90 A KIT, la cual opera con cassettes de cinta con calidad digital, colocada a 2.50 metros del fondo de la arena experimental con el fin de obtener la imagen completa de la circunferencia. Se analiza solo el primer Tail-Flip de cada camarón, usando una resolución de 30 cuadros/seg. [Arnott, 1999]. El tiempo de grabación se extiende alrededor de 30 minutos si no ocurría un escape exitoso, o bien, terminó inmediatamente después del primer escape. Al finalizar cada prueba, se realizó un recambio del 100% del agua de la arena experimental, para evitar cualquier sesgo en la respuesta de los camarones debida al cambio en la composición química del agua.

Se mencionarán dos métodos diametralmente opuestos para extraer los datos de los videos capturados durante los experimentos, estos son los normalmente usados en la actualidad.

1.5.1. Método clásico

Se realiza la corrección de las aberraciones esféricas del sistema de video, mediante la filmación y digitalización de las dimensiones grabadas dentro de la arena. Las distancias filmadas son calibradas y siempre se referencian a un objeto de dimensiones conocidas dentro del campo de filmación (moneda de 10 pesos). Tomando en cuenta las velocidades de escape registradas para peneidos (0.26 – 1.42 m/seg.), se realiza el análisis cuadro por cuadro, utilizando una resolución de 33 cuadros por segundo. Todas las mediciones se efectúan a partir del centro de masa del camarón, norte del camarón y centro de masa de la jaiba. [Arnott, 1998].

Los videos se digitalizan en una computadora personal utilizando el programa Pinnacle Studio. Cada escape se captura en formato AVI con la máxima resolución que permite el equipo empleado (30 cuadros por segundo). Se identifican los ataques y las respuestas de escape analizando los videos cuadro por cuadro. Se registran dos imágenes en formato JPEG de cada video, a partir de las cuales se extraen todos los datos requeridos para el análisis de las trayectorias de escape. Este protocolo experimental es sumamente laborioso debido a su naturaleza puramente manual, es decir, durante el experimento completo ya sea hasta que se presente el primer Tail-Flip o bien hasta que se cumpla el tiempo máximo de 30 minutos, el investigador debe encontrarse presente y prestando plena atención a la arena donde se encuentran los animales. Es importante destacar que cualquier distracción por parte del investigador puede representar la omisión del primer Tail-Flip pues este se lleva a cabo en fracciones de segundo, y de este modo no se detendrá la grabación en el momento necesario.

1.5.2. Método moderno

Otro medio del cual se puede valer un investigador para realizar este tipo de experimentos de una manera sumamente sencilla y mayormente automática es mediante el uso de una tecnología en software llamada Ethovision desarrollada por la compañía NOLDUS (www.noldus.com/site/doc200403002). Ethovision es un sistema avanzado de seguimiento en video que realiza una grabación automática de actividad, movimiento, e interacción social de los animales localizados en cualquier recinto o escenario. Como referencia, la última versión de EthoVision XT, además del seguimiento del centro del animal, ofrece el seguimiento del punto donde se localiza la nariz y la base de la cola de las ratas y ratones. Proveyendo de un amplio rango de características para el seguimiento en video y análisis de datos, EthoVision permite la automatización de un amplio rango de pruebas de comportamiento incluyendo:

- Pruebas de reconocimiento de nuevos objetos.
- Pruebas de interacción social.
- Prueba del laberinto.
- Pruebas de nado, etc.

Entre las características más importantes de este programa comercial tenemos que cuenta con el seguimiento de las coordenadas del centro matemático del cuerpo del animal, así como

también las coordenadas de la nariz y de la base de la cola en el caso de tratarse con ratones o ratas. Puede medir diferentes grados de elongación presentados por el cuerpo de un ratón o una rata, pues el usuario puede decidir qué grado de elongación necesita que se grabe o registre como “Estirado”, “Normal”, y “Contraído”. Cuenta con un modo de medición de inmovilidad, el cual consiste en medir el grado de inmovilidad de manera independiente del desplazamiento espacial. El usuario decide el grado de movilidad que requiere sea registrado como “Inmóvil”, “Móvil”, o “altamente móvil”. Es importante aclarar que la movilidad y el movimiento son parámetros con significado diferente, ya que el movimiento mide desplazamiento espacial. EthoVision cuenta con muchas más características que lo vuelven una solución muy rápida para los experimentos etológicos con animales de diferentes especies, una muestra de este programa se muestra en la figura 1.1. [Noldus, 2006]

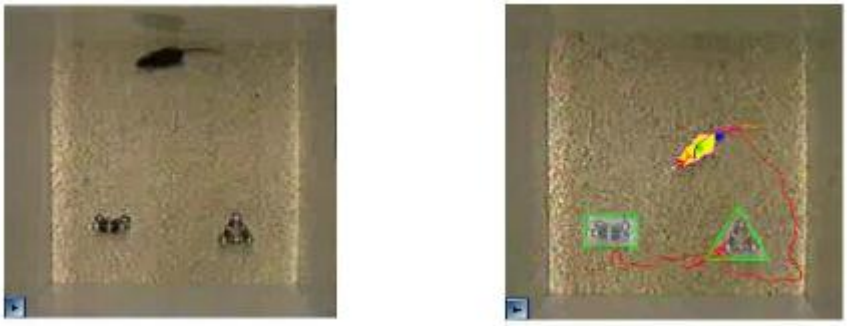


Figura 1.1 Vista de Etho visión en operación [Noldus, 2006].

El área donde se realizan los experimentos de comportamiento se le conocen como “arena”, y EthoVision está listo para ser utilizado en diferentes clases de arenas. Brinda la capacidad de definir zonas de seguimiento específicas, así como también zonas ciegas. Las zonas ciegas son las áreas de la arena en donde la cámara no puede ver al objetivo, como ejemplo, el ratón escondido debajo de un tranco.

Las operaciones antes mencionadas pueden realizarse en tiempo real o previa grabación en video, siendo requerida de manera obvia una unidad de almacenamiento cuya capacidad pudiera llegar a ser una limitante si se trata de videos de alta calidad en resolución. La principal desventaja de esta herramienta para la experimentación del comportamiento de algunos animales es su elevado precio, por lo que su compra es factible únicamente cuando la investigación es soportada por empresas farmacéuticas, laboratorios de investigación por contrato, compañías de biotecnología y nutrición, etc. Es decir, en un número muy reducido de instituciones donde se realiza investigaciones de este tipo.

1.6. Entorno de la Tesis

Esta investigación es planteada por biólogos marinos, el objetivo es conocer el comportamiento de los camarones cuando son atacados por uno de sus depredadores en este caso la jaiba, apoyándose en las investigaciones realizadas por Arnott [1998 y 1999], se ven en la necesidad de contar con una cámara que sea lo más rápida posible, y por ello se eligió una cámara fabricada por la compañía Silicon Imaging, el modelo SI-1280RGB-CL, la cual es capaz de tomar hasta 300 frames por segundo.

Esta cámara tiene una interfaz “CameraLink”, que puede ser conectada a un Módulo que convierte CameraLink a interface Ethernet que es el que se conecta a la computadora para obtener las fotografías en el formato RGB.

Sin embargo, en el desarrollo de esta Tesis se propone conectar directamente la cámara a un FPGA para no utilizar este módulo ya que de acuerdo a Egrid [2006] la interfaz Camera Link tiene mejores prestaciones que la interfaz Ethernet y además se puede trabajar directamente con la imagen a medida que esta se vaya almacenando mientras que si utilizamos el módulo, tendríamos que esperar a que la imagen se transmita totalmente, con lo cual se incrementaría la latencia.

De no conectar la cámara al módulo, se presentan al menos dos opciones para la recuperación de la imagen; una es utilizar el circuito integrado DS90CR285MTD fabricado por National Semiconductor [2004], cuya función es la deserialización de los datos provenientes de la cámara. Esta opción se descartó ya que involucraría una tarjeta adicional para el circuito integrado y los componentes que necesita éste, además de una fuente de alimentación. Adicionalmente, debido a que el circuito integrado no tiene memoria y requiere de un sistema de control, se tendría que conectar a un FPGA para almacenar los datos recuperados y posteriormente realizar el procesamiento con el FPGA o un DSP. La otra opción es la de implementar el deserializador con el FPGA como lo propone Sawyer N. [2008]. En esta Tesis se presentan algunas modificaciones en su arquitectura y en el dispositivo.

Para poder conectar la cámara con el FPGA se requiere de un cable con características muy especiales, el cable tiene en el extremo un conector que no es compatible a la tarjeta de evaluación donde está el FPGA por lo que fue necesario realizar una tarjeta que adecuara la señales para poderlas introducir al FPGA. Esta tarjeta debe fabricarse con ciertos cuidados por manejar señales diferenciales. En [Knighten, 2000] se describe que el máximo skew permitido en un par diferencial es de 100 ps, mientras que el skew máximo entre pares de líneas diferenciales que es de 250 ps, otra consideración que debe tomarse en cuenta es la separación entre las líneas diferenciales debido a la generación del ruido electromagnético, [Bloomingdale, 2002] nos recomienda que el ancho de la separación de las pistas debe ser menor que el doble de ancho las líneas diferenciales.

Un estudio basado en un FPGA y un microprocesador para el seguimiento de animales en tiempo real es presentado por [Chen, 2005], en su arquitectura el FPGA es el encargado de recuperar la imagen y almacenarla en una memoria así como tener el algoritmo de seguimiento del animal, el microcontrolador es el encargado de ejecutar el algoritmo que realiza el análisis del comportamiento del animal, una de sus limitantes es el tiempo en que se ejecuta ya que el microcontrolador va a 50 MHz. En la figura 1.2 podemos ver su implementación.

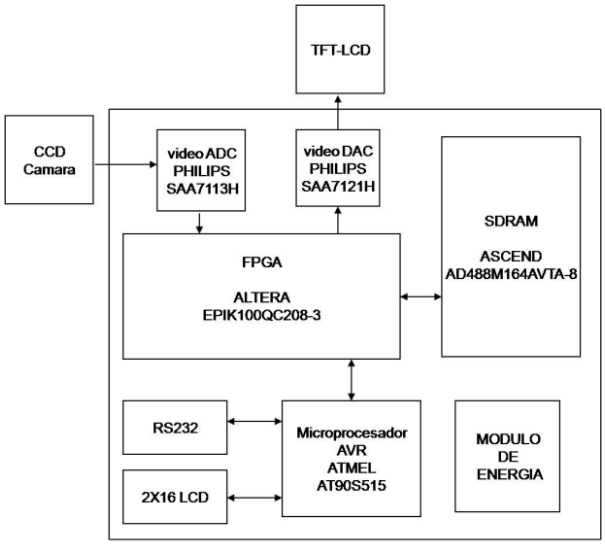


Figura 1.2 Diagrama a bloque del sistema

El sistema que se propone en lo general podría ser parecido, sin embargo el potencial de esta Tesis es el análisis de las comparativas de cada módulo, que se realiza para obtener los mejores resultados en cada etapa en el algoritmo de análisis de las imágenes.

Una vez obtenidas las imágenes con la cámara estática y almacenadas en memoria, de las cuales una representa el fondo del escenario y la otra es el fondo del escenario más los dos animales en estudio, el siguiente paso es la extracción automática de los puntos característicos. Para lograr esto, debemos apoyarnos en algoritmos del nivel más bajo de tratamiento de imágenes, la primera acción sería dejar únicamente a los animales mediante la resta de ambas imágenes como mencionan [Benezeth, 2008] y [Kumar, 2010]. Cabe mencionar que existen varios métodos para el seguimiento de objetos en tiempo real sin utilizar la resta de imágenes como mencionan en [Schoenemann, 2008], [Takaya, 2006] y [Wang, 2008], una de las ventajas que tiene el método de resta de imágenes es que a medida de que se va recibiendo la imagen se puede estar realizando la resta y esto nos ahorra tiempo, ya que los otros métodos deben esperar a que se tenga la imagen completa para procesarla. Debido a como se implementó el fondo de la imagen se presentaba un problema, los animales se veían seccionados y el algoritmo lo tomaba como otros animales presentes, por lo que se vio la necesidad de implementar un filtro pasa bajo para reagrupar las partes de los animales y se umbralizó para eliminar las partes débiles de la imagen para después segmentarla y obtener datos de la imagen. Los algoritmos de resta de imágenes, filtros pasa bajos, umbralización y el de segmentación han sido ampliamente estudiados [González 1992, Horavd 1993, Bravo 1996].

El siguiente paso fue encontrar los puntos más lejanos del animal con respecto al centro de masa; aquí se utilizaron algoritmos que permitieran encontrar los contornos de los animales, para lo cual existen también numerosos algoritmos propuestos, que han sido probados ampliamente en el campo computacional, por ejemplo S. Smith, J. Brady en [Smith, 95] y sus versiones mejoradas [Mingliang, 2010], describen un algoritmo para la localización de contornos y de esquinas, J. Canny en [Canny, 1986] también describe otro método para la extracción de contornos, este ha sido ampliamente utilizado por la comunidad científica, en lo particular es más preciso pero es muy lento; otros algoritmos basados en los operadores de gradiente como Sobel, Prewitt Robertes y otros tienen el efecto de magnificar el ruido subyacente en la imagen como lo menciona Pajares en [Pajares, 2008]. La característica principal del algoritmo de Smith

es que es muy rápido y muy fiable, debido a que para nosotros es importante la rapidez, este algoritmo fue el que mejor desempeño mostraba.

Finalmente, en base a los datos obtenidos en la segmentación y a los puntos más lejanos, utilizando principios básicos de geometría analítica se logra obtener los vectores de posición de los animales en estudio.

En la mayoría de los casos estudiados las soluciones que se obtienen son relativamente lentas (en segundos) y para acelerar el proceso utilizan ordenadores muy potentes y voluminosos e incluso ya han empezado a utilizar FPGA's en las etapas de preprocesamiento, esto se debe a que en la mayoría de las aplicaciones de visión por ordenador se requiere de un gran número de operaciones que se repiten sobre toda la imagen, lo cual implica, que aún para las capacidades de cómputo actual, se requieren ordenadores muy poderosos para lograr velocidades aceptables hasta en las labores más simples. Es por esto que la implementación de un sistema de visión por medio del tradicional software secuencial no ha sido del todo adecuada en aplicaciones que requieren operación en tiempo real.

CAPITULO 1 - INTRODUCCIÓN

2.1 Hardware para adquisición de imágenes

2.1.1 Adquisición de imágenes

Las imágenes para el seguimiento de objetos son capturadas a través de una cámara de uso industrial que presenta características de alta resolución y velocidad de captura. La cámara es de la compañía Silicon Imaging (en la figura 2.1 puede observarse la cámara utilizada) y presenta la opción de operar con sensores CCD monocromáticos modelo SI-1280FM-CL y con sensores a color RGB modelo SI-1280RGB-CL, estos sensores se pueden observar en la figura 2.2.

Figura 2.1. Cámara SI-1280F

Para esta Tesis se utilizó el modelo SI-1280RGB-CL, ya que de requerirse el color se puede obtener en cualquier momento, sin embargo por el momento se trabajará como si fuera el sensor monocromático ya que no se precisan para el reconocimiento los tres planos de color, así como también el tamaño menor de las imágenes permite una tasa de captura más grande. Con estas características esta cámara es adecuada para procesamiento de imágenes. En la figura 2.3 se da la representación de las ubicaciones de los píxeles así como la representación de las filas y columnas para formar la matriz que será la imagen.

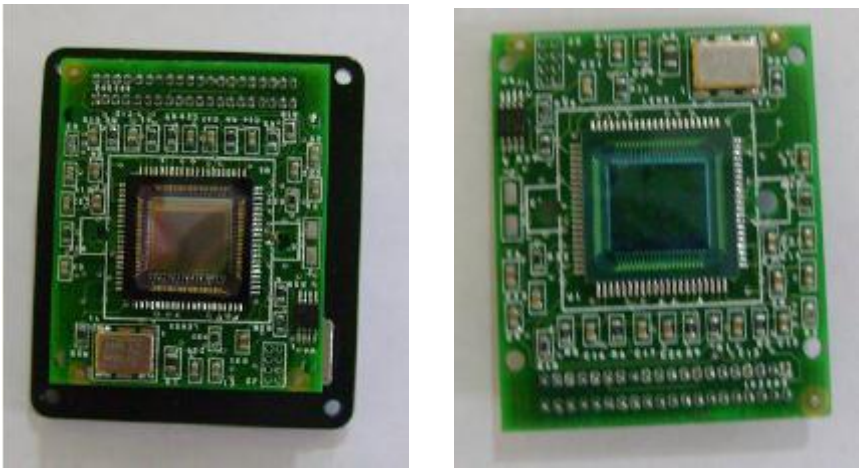


Figura 2.2. A la izquierda sensor CCD a color y a la derecha sensor CCD monocromático

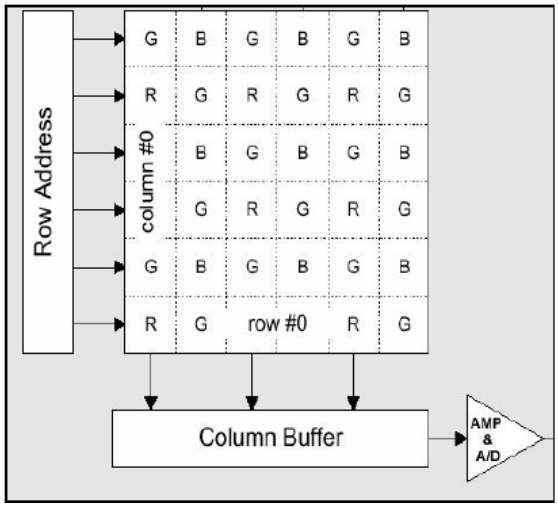


Figura 2.3. Arreglo de 1280 filas x 1024 columnas

Como accesorio de esta cámara se encuentra un modulo convertidor de Camera Link a Gigabit Ethernet y se le conoce como Interface GigE-CameraLink, mostrado en la figura 2.4. Este modulo permite capturar imágenes con las cámaras de alta velocidad Silicon Imaging con un ordenador desde una distancia de 100 metros utilizando un cable CAT-5. Sin embargo, este modulo puede transmitir imágenes a 10/100 Mbps, correspondientes a Ethernet y Fast Ethernet. En estos casos la cámara se ve obligada a reducir la tasa de captura pues la velocidad de transmisión es insuficiente para enviar *frames* por segundo nominales, en otras palabras, para operar la cámara con sus mejores prestaciones el ordenador receptor debe contar con una tarjeta de red con tecnología Gigabit Ethernet. En un inicio la captura de la imagen se realizó a través de Gigabit Ethernet posteriormente se migró a adquirir las imágenes directamente a través Camera Link, con ello se evitaba el modulo de Ethernet, y se aprovecha las prestaciones que tiene Camera Link sobre Gigabit Ethernet [Egrid, 2006]

Figura 2.4. Modulo GigE-CameraLink, interface Ethernet 10/100/1000

Entre las características más notables de esta cámara están:

- Cámara de 1.3 MP.
- 12 bits de resolución por pixel.
- 1280 x 1024 pixeles a unos 40 fps.
- Sub-muestreo y Ventaneo direccionado en X y Y.
- Interface de comunicación Camera Link & Gig-E.

Una cámara fotográfica construida con tecnología de dispositivo de carga acoplada (Charge-Coupled Device, CCD), es el dispositivo de entrada más flexible y común para los sistemas de visión por computadora. En la figura 2.5 se puede observar un sensor CCD. Cada celda convierte la energía luminosa que recibe en una carga eléctrica. Al principio todas las celdas son puestas a 0, para entonces comenzar a integrar su respuesta a la energía luminosa que reciben. Un obturador (mecanismo para permitir la entrada de la luz) puede ser o no necesario para controlar el tiempo de exposición a la energía luminosa. El plano de la imagen actúa como una memoria digital que puede ser leída renglón a renglón por un proceso de entrada a la computadora. [Shapiro, 2001]

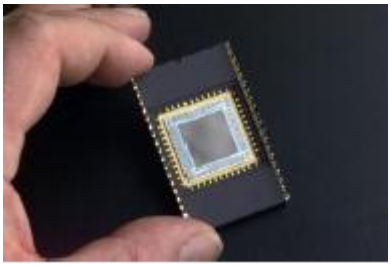


Figura 2.5. Sensor CCD

La *resolución nominal* de un dispositivo CCD es el tamaño del elemento de escena que representa un píxel en el plano de la imagen. El termino *resolución* se refiere a la precisión del dispositivo al hacer mediciones, se puede definir como el número de píxeles disponibles; esta definición tiene la ventaja de que puede expresar en cuantas partes el campo de visión será dividido, el cual proporciona la capacidad de hacer mediciones más precisas y cubrir cierta región de una escena. [Shapiro, 2001] El tamaño físico de un píxel en una imagen esta definido por la resolución espacial, la resolución espacial de una imagen se expresa como sinónimo de puntos por pulgada (dpi) o píxeles por pulgada (ppi). Para una imagen fija de una región física, un muestreo denso dará como resultado una imagen de alta resolución, con un gran número de píxeles donde cada uno contribuye a una pequeña parte de la escena, mientras que un muestreo grueso dará como resultado una imagen de baja resolución con un pequeño número de píxeles donde cada uno contribuye a una gran parte de la escena. En la figura 2.9 podemos encontrar el efecto que se produce cuando la imagen cuenta con menos elementos para representar la escena. La codificación tricromática RGB en los sistema gráficos generalmente utiliza tres bytes permitiendo 16 millones distintos de códigos de color. Para ser exactos, 16 millones de códigos y no 16 millones de colores porque los seres humanos no pueden percibir realmente tal cantidad de colores. Las máquinas pueden distinguir entre cualquier par de diversos códigos de bits, pero estos pueden o no representar diferencia significativa en el mundo real. En cada 3 bytes o 24 bits RGB se designa un byte para cada canal rojo, verde, y azul. El orden en el cual cada uno aparece en memoria puede variar; el orden es irrelevante en la teoría pero es importante en la programación. [Shapiro, 2001]

El modelo RGB está basado en la observación de que al mezclar el rojo (R), verde (G) y azul (B) en distintas proporciones es posible obtener un amplio rango de colores. Por lo tal es posible construir una imagen a color usando los componentes rojo, verde y azul para cada píxel. Entonces el color de cada píxel está determinado por el peso de cada color primario. En una imagen a color acorde con el modelo RGB, el valor para cada píxel se puede imaginar como un vector de tres componentes, denominados los valores rojo, verde y azul. Así el espacio de color puede ser definido tal que R, G, y B se encuentren colocados en los ejes ortogonales definiendo un espacio de color tridimensional, representado en términos de un cubo de color en el primer cuadrante como se muestra en la figura 2.6.

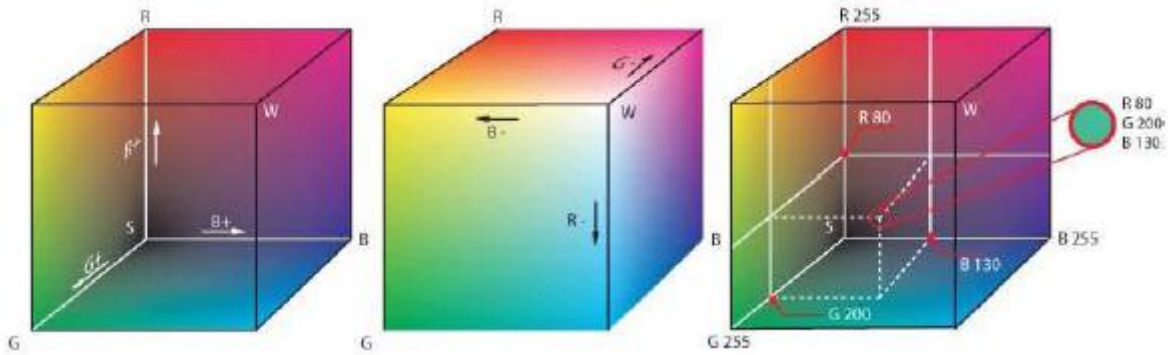


Figura 2.6. Modelo RGB (espacio de color)

2.1.2 Camera Link

Camera Link es una interfaz de comunicación para aplicaciones de visión. Esta interfaz extiende la tecnología base del Channel Link para ofrecer especificaciones más útiles para aplicaciones de visión. Durante años, el mercado de video digital científico e industrial ha carecido de un método estándar de comunicación. Tanto los fabricantes de capturadores de video (Frames Grabbers) como los de cámaras desarrollaron productos con diferentes conectores, y esto hacia que la producción de cables fuera difícil para los fabricantes y muy confusa para los consumidores. Una conectividad estándar entre cámaras digitales y capturadores de video está sumamente atrasada en su desarrollo y llegará a ser mucho más necesaria en un futuro próximo, así como el incremento en la velocidad de transferencia de datos.

Con el incremento en la diversidad de las cámaras y con las avanzadas transmisiones de señales y datos se hace aún más necesaria una conectividad estándar como Camera Link. La interfaz Camera Link reducirá el tiempo de soporte técnico, así como el coste de ese soporte técnico. El cable estándar será capaz de manejar las elevadas velocidades de las señales, y el conector estándar permitirá a los consumidores reducir sus costes a través de la compra por volumen.

Las especificaciones del Camera Link incluyen transmisiones de datos mayores a 1.2 Gb/s así como un control de la cámara y comunicación serie asíncrona. Todo en un sencillo cable con un conector de 26 pines. Sólo dos conexiones son requeridas para hacer la interfaz de la cámara digital a una multitud de frame grabbers.

Como un estándar que ha sido definido por miembros de la industria, CameraLink provée de los siguientes beneficios:

- *Una interfaz estándar*: cada producto CameraLink usará el mismo cable y señalización. Cámaras y frame grabbers pueden fácilmente ser intercambiadas usando el mismo cable.
- *Conexión simple*: Solamente dos conexiones son requeridas para comunicar la cámara y frame grabbers: alimentación y el de Camera Link.
- Bajo costo.
- Conectores y cables más pequeños.

- *Las más altas tasas de transmisión*: la tecnología usada en CameraLink tiene una tasa máxima de 2.3Gb/s, para uso de altas demandas de definición.

2.1.3 Channel Link

National Semiconductor desarrolló la tecnología Channel Link [National Semiconductor, 2006] como solución para las pantallas planas, basadas en LVDS (Low Voltage Differential Signaling) para la capa física. Luego la tecnología fue extendida en un método de transmisión de datos de propósito general. Channel Link consiste en un par transmisor (driver) y un par receptor. El transmisor cuenta con 28 señales de datos de terminación sencilla y un reloj de terminación sencilla. El dato es serializado 7:1 y las cuatro secuencias de datos y un reloj dedicado son conducidos a través de cinco pares LVDS. El receptor acepta las cuatro secuencias de datos en LVDS y el reloj también en LVDS y luego dirige los 28 bits y el reloj a la tarjeta. La figura 2.7. muestra la operación de la tecnología Channel Link.

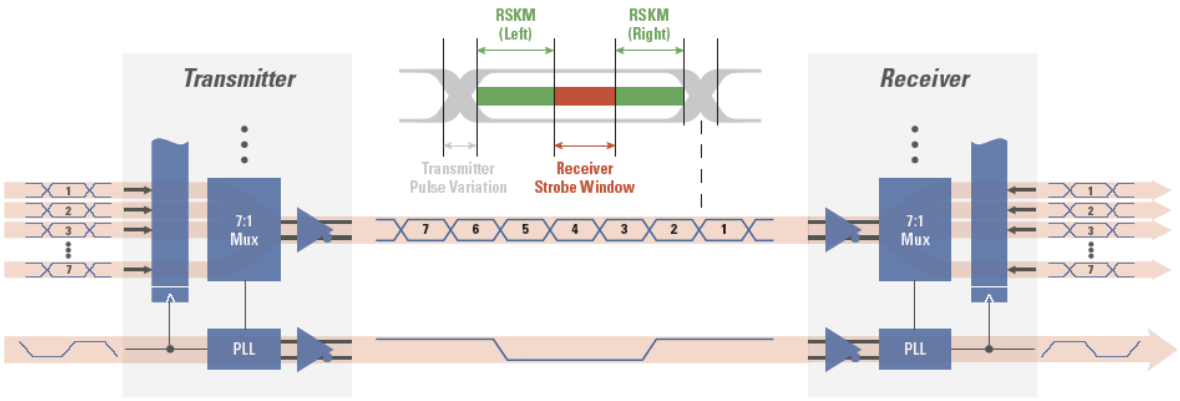


Figura 2.7. Representación de la operación de la Tecnología Channel Link

2.2 Visión por computador

Actualmente existen numerosos trabajos de investigación sobre este tema, ya que sus aplicaciones pueden ser muchas, desde un sistema de vigilancia para un cruce de calles, hasta el control fino del movimiento de un robot. Las primeras bases teóricas empiezan en los años 60, a partir de entonces se han desarrollado aplicaciones industriales, militares, aeroespaciales y médicas. [Hovard, 1993].

2.2.1 Definiciones

Para introducirnos en la visión por ordenador debemos entender primero el sentido de la vista. Cuando vemos un objeto, la información que está realmente disponible por la retina es, ni más ni menos, que una colección de puntos (alrededor de un millón).

Cada punto o píxel contiene simplemente la información que indica la cantidad de iluminación y el color que proviene del espacio ambiente y que se proyecta en ese punto de la retina. Por tanto, objetos como un teléfono, una mesa o un libro no existen para la retina, si no

que estos conceptos, son el resultado final de un proceso de interpretación que forma parte integral del sistema de visión. Además, dicho sistema debe de proporcionar la información necesaria a fin de permitir interpretaciones que no sean ambiguas.

La visión por computadora es un área de investigación de gran importancia en el campo de la Inteligencia Artificial, su influencia puede ser encontrada en varias áreas del conocimiento, entre éstas tenemos a la medicina, geología, genética, las gráficas por computadora, la robótica, la interacción Humano-Computadora, la industria la demanda para las tareas de inspección y ensamblaje, entre otras áreas. Según se relata en [Shapiro, 2001] *"La meta de la visión por computadora es hacer decisiones útiles sobre objetos físicos reales y las escenas basadas en las imágenes capturadas"*. Del término Visión por Computadora se encuentran distintas definiciones para el mismo, sin embargo tienen un denominador común, *tratar de imitar la forma de percibir las imágenes por los seres humanos*. Entre algunas definiciones tenemos las siguientes:

La Visión por Computadora es una rama de la inteligencia artificial que tiene por objetivo modelar matemáticamente los procesos de percepción visual en los seres vivos y generar programas que permitan simular estas capacidades visuales por computadora. [Lania, 1999].

El término Visión por Computadora dentro del campo de la Inteligencia Artificial puede considerarse como el conjunto de todas aquellas técnicas y modelos que nos permiten el procesamiento, análisis y explicación de cualquier tipo de información especial obtenida a través de imágenes digitales. [INAOEP, 2005].

La Visión por Computadora es el estudio de capacitar a las computadoras para interpretar las imágenes, es un amplio campo interdisciplinario, aplica al campo de la computación científica. Abarca matemáticas, ingeniería eléctrica, procesamiento de señales, óptica, física, psicofísica, teoría computacional y algoritmos. [Maxwell, 1998].

Entre algunas tareas que comprende la visión computacional se encuentran el análisis de imágenes, restauración, realce, corrección, extracción de información, reconocimiento, interpretación, modelado, etc.

Por la propia naturaleza del trabajo es necesario el uso de la Visión por Computadora, debido a que la tarea principal es el rastreo de dos animales a través de una secuencia de imágenes, y por el enfoque planteado, éste se realizará a través del reconocimiento de ciertas características de ellos visibles en la propia imagen. Adicionalmente, para poder manipular dicha información es necesario un tratamiento a las mismas, es decir, modelar la información que ahí se encuentra en base a los requerimientos que se tengan.

2.2.2 Procesamiento de imágenes

Una Imagen digital proviene comúnmente de muestrear el espacio físico de la imagen en filas y columnas. Cada muestra de la imagen corresponde a una pequeña región de la imagen física, y se llama elemento de imagen, o *píxel*. Cada pixel es convertido a uno o varios números (dependiendo del tipo de imagen de que se trate) mediante un convertidor de analógico a digital, el cual es el encargado de cuantizar el valor analógico de entrada para representarlo con un número finito de bits.

El *muestreo* es el proceso de medir el valor de la imagen física en intervalos discretos en el espacio.

La *cuantización* es el proceso de sustituir los valores continuos de la imagen muestreada con un conjunto discreto de niveles de cuantización.

Una imagen puede ser definida como una función bidimensional *f(x, y)*, donde *x* e *y* son coordenadas espaciales, y la amplitud de *f* en cualquier par de coordenadas *(x, y)* es llamada intensidad o nivel de gris de la imagen en ese punto. Cuando *x, y*, y los valores de amplitud de *f* son todos finitos, la imagen es llamada ***imagen digital***. La figura 2.8 muestra la convención de ejes utilizados. Una imagen digital está compuesta de un número finito de elementos, cada uno de los cuales tiene una posición particular y valor, estos elementos son referidos como elementos de la foto, elementos de la imagen o ***píxeles***. [González, 1992].

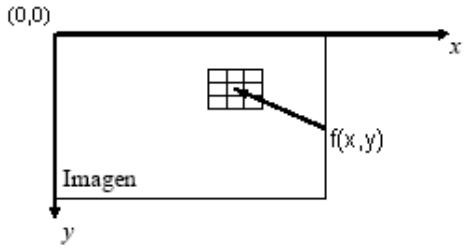


Figura 2.8. Convención de ejes utilizados para la representación de imágenes digitales [González, 1992].

Dependiendo del número de píxeles la imagen digital poseerá más o menos resolución espacial. En la figura 2.9 se muestra cuatro representaciones de la misma imagen con variación en el número de píxeles utilizados.

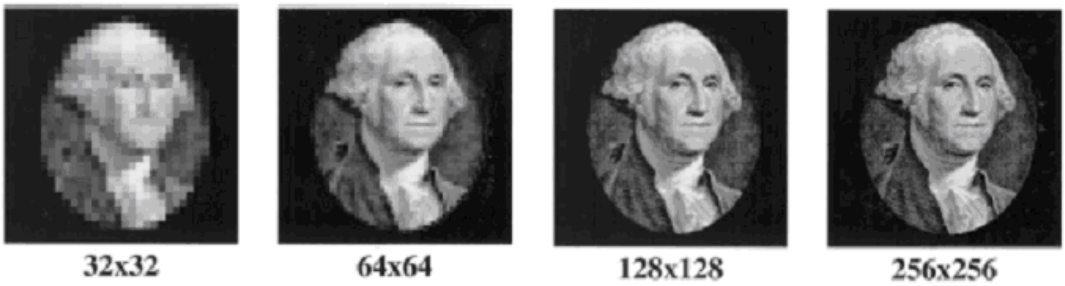


Figura 2.9. Resolución espacial en píxeles manteniendo los niveles de intensidad [Hansen, 2002].

Una *imagen en escala de grises* (Gray scale image), es una imagen monocromática digital con un solo valor de intensidad por píxel.

Una *imagen multiespectral* es una imagen que contiene un vector de valores para cada píxel. Si la imagen es a color, entonces el vector tiene 3 elementos por píxel.

Una *imagen binaria* es una imagen digital con todos los valores de sus píxeles puestos a 0 y 1. [Shapiro, 2001]

2.3 Formato Bayer para Imágenes.

En la actualidad existen un gran número de formatos para el almacenamiento de las imágenes digitales, sin embargo la mayoría de las cámaras captura la imagen en formato Bayer [Bayer, 1976], que normalmente pasa por un convertidor y nos la entrega en el formato RGB. El arreglo del filtro de color Bayer es un formato popular para la adquisición digital de imágenes de color. El dibujo de los filtros de color es mostrado en la figura 2.10. La mitad del número total de los píxeles son verdes (G), mientras que un cuarto del número total es atribuido a tanto rojo (R) como color azul (B).

G	R	G	R
B	G	B	G
G	R	G	R
B	G	B	G

Figura 2.10. Formato Bayer

Para obtener la información de color, el sensor de imagen de color está cubierto de un rojo, un verde, o un filtro azul, en un patrón repetitivo. Este patrón, o secuencia, de los filtros puede variar, pero el patrón "Bayer" es extensamente adoptado, este fue inventado en Kodak, es un arreglo de 2x2 repetitivo. En la figura 2.11 nos muestra la ubicación de los sensores y la representación del color generado en la imagen.

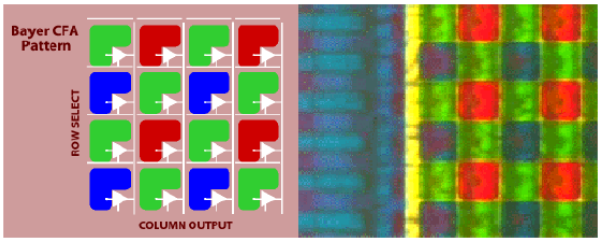


Figura 2.11. Ubicación de los sensores y su respectiva imagen

Para obtener una imagen en mapa de bit, se necesitará de un convertidor de Bayer a RGB, existiendo varios métodos como se analizan en [Hua, 2010]. Se ha elegido el método de interpolación bilineal junto con el de interpolación lineal tomando la correlación en consideración [Sakamoto, 1998] ya que es el que presenta mejores características en tiempos de ejecución. A continuación se describirá el método.

Los valores de R y B son interpolados en línea recta de los vecinos más cercanos del mismo color. Hay cuatro casos posibles, como se muestra en figura 2.12. Se interpolará los valores de R y B sobre un pixel verde, como en figura 2.12 (a) y (b), tomando los valores medios de los dos vecinos más cercanos del mismo color. Por ejemplo, en figura 2.12 (a), el valor para el componente azul sobre un pixel de G, serán el promedio de los pixeles azules por encima y por debajo del pixel de G, mientras que el valor para el componente rojo será el promedio de los dos pixeles rojos a la izquierda y la derecha del pixel de G. La figura 2.12 (c) muestra el caso cuando el valor de la componente azul es interpolado para un pixel de R. En tal caso, tomamos el promedio de los cuatro pixeles azules más cercanos rodeando el pixel R. De forma semejante, determinar el valor de la componente rojo sobre un pixel de B en figura 2.12 (d) tomamos el promedio de los cuatro pixeles rojos más cercanos rodeando el pixel de B.

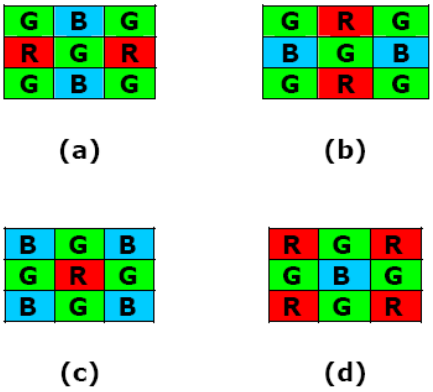


Figura 2.12. Cuatro posibles casos para la interpolación de las componentes R y B

La parte del método de interpolación lineal tomando la correlación en consideración, se resume a continuación.

En figura 2.13 (a), el valor de la componente verde es interpolado sobre un píxel R. El valor usado para el componente de G se muestra a continuación.

$$G(R) = \begin{cases} (G_1 + G_3)/2, & si \quad |R_1 - R_3| < |R_2 - R_4| \\ (G_2 + G_4)/2, & si \quad |R_1 - R_3| > |R_2 - R_4| \\ (G_1 + G_2 + G_3 + G_4)/4 & si \quad |R_1 - R_3| = |R_2 - R_4| \end{cases}$$

Para la figura 2.13 (b), el valor de la componente verde es interpolado sobre un píxel B. El valor usado para el componente de G se muestra a continuación:

$$G(B) = \begin{cases} (G_1 + G_3)/2, & si \quad |B_1 - B_3| < |B_2 - B_4| \\ (G_2 + G_4)/2, & si \quad |B_1 - B_3| > |B_2 - B_4| \\ (G_1 + G_2 + G_3 + G_4)/4 & si \quad |B_1 - B_3| = |B_2 - B_4| \end{cases}$$

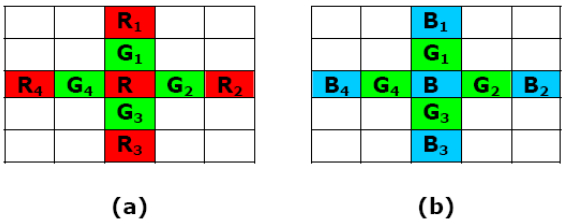


Figura 2.13. Dos posibles casos para la interpolación de la componente G

2.4 Conceptos básicos para el procesamiento de imágenes y filtrado espacial

2.4.1 Conceptos básicos para el procesamiento de imágenes

En este apartado presento los conceptos básicos que se utilizan en el procesamiento de imágenes. Es importante comprender los métodos básicos de procesamiento, que van desde el procesamiento puntual, pasando por los métodos basados en máscaras hasta métodos basados en frecuencia, ya que esta comprensión es la que nos permitirá hacer una correcta elección de la combinación optima de procesos a utilizar e inclusive poder modificar algún método existente para adaptarlo a las necesidades específicas del problema a resolver. Con esto en mente presento una breve reseña de los principales métodos existentes para el procesamiento de imágenes, hablo primero de los diferentes dominios en que pueden operarse los métodos (dominio espacial y de la frecuencia), luego presento algunos métodos del domino espacial, primero aquellos de ámbito puntual, y luego algunos métodos clásicos que utilizan máscaras.

2.4.1.1 Domino espacial y de la frecuencia

Los métodos de mejora de la imagen pueden estar basados en técnicas, bien en el dominio espacial o bien en el dominio de la frecuencia. El propósito de esta sección es desarrollar sus ideas básicas y relacionar ambas aplicaciones.

El término ***dominio espacial*** se refiere al conjunto de píxeles que componen una imagen, y los métodos en el domino espacial son procedimientos que operan directamente sobre los píxeles. Las funciones de procesamiento de la imagen en el domino espacial pueden expresarse como:

$$g(x,y) = T\left[f(x,y)\right] \qquad \text{Ec. 2.1}$$

Donde f(x,y) es la imagen de entrada, g(x,y) es la imagen procesada y T es un operador que actúa sobre f, definido en algún entorno de (x,y). Además T puede operar sobre un conjunto de imágenes de entrada. La aproximación principal para definir un entorno alrededor de (x,y) es emplear un área de subimagen cuadrada o rectangular centrada en (x,y), como se muestra en la figura 2.8, aunque a veces se empleen otros tipos de entorno, tales como aproximaciones a un circulo. A dicho entorno se le conoce también como “máscara, plantilla, ventana o filtro”. El centro de la subimagen se mueve píxel a píxel aplicando el operador en cada posición (x,y) para obtener g.

La forma más simple de T corresponde a un entorno de 1x1, en cuyo caso a menudo se conoce como procesamiento de punto o puntual.

La base de las técnicas en el dominio de la frecuencia es el teorema de convolución. Sea *g(x,y)* una imagen formada por la convolución de una imagen *f(x,y)* y un operador lineal *h(x,y)*, es decir:

$$g(x,y) = h(x,y)^{*} f(x,y) \qquad \text{Ec. 2.2}$$

Entonces, por el teorema de convolución se cumple la siguiente relación el dominio de la frecuencia:

$$G(u,v) = H(u,v)F(u,v) \qquad \text{Ec. 2.3}$$

Donde ***G***, ***H*** y ***F*** son respectivamente las transformadas de Fourier de *g*, *h* y *f*. En la terminología de la teoría de sistemas lineales, la transformación ***H****(u,v)* se denomina la función de transferencia del proceso. En una aplicación típica *f(x,y)* es conocida y el objetivo, después de calcular ***F****(u,v)*, es seleccionar ***H****(u,v)* de forma que la imagen deseada, presente resaltada alguna característica de *f(x,y)*.

$$g(x,y) = \Im^{-1}\left[H(u,v)F(u,v)\right] \qquad \text{Ec. 2.4}$$

En resumen, para mejorar una imagen, trabajando en el domino de la frecuencia, la idea es calcular la transformada de Fourier de la imagen, multiplicar el resultado por la función de transferencia de un filtro y, finalmente tomar la transformada de Fourier inversa para llegar a una imagen mejorada. Aunque existen numerosos problemas que pueden ser abordados con estas técnicas, en la práctica, las pequeñas máscaras espaciales son mucho más empleadas que las transformadas de Fourier debido a su facilidad de implementación y a su velocidad de operación. Es por este motivo que durante el presente trabajo me centro en los métodos de mejora de la imagen en el dominio espacial.

2.4.1.2 Procesamiento puntual

En esta sección consideraré algunos de los métodos de procesamiento que se basan sólo en la intensidad de píxeles individuales. En lo que sigue indicaremos como *r* y *s* la intensidad de los píxeles antes y después del procesamiento, respectivamente.

2.4.1.2.1 Negativos

El negativo de una imagen digital se obtiene empleando la función de transformación

$s = T(r) = L - r$

Donde *L* es el número máximo de niveles de gris. La idea es invertir el orden de blanco a negro, de manera que la intensidad de la imagen de salida aumente conforme la intensidad de la imagen de entrada disminuye.

2.4.1.2.2 Aumento de contraste [González 92].

Frente a una imagen de entrada con poco contraste, la idea es incrementar el rango dinámico de los niveles de gris de la imagen que se está procesando. La figura 2.14 muestra una transformación típica empleada para la mejora del contraste. La ubicación de los puntos (r_1,s_1) y (r_2,s_2) controla la forma de la función de transformación. Por ejemplo, si $r_1 = s_1$ y $r_2 = s_2$, la transformación es una función lineal que no produce cambios en los niveles de gris. Si $r_1 = r_2$, $s_1 = 0$ y $s_2 = L-1$, la transformación se convierte en una *función umbral* que crea una imagen binaria. La figura 2.15 (a) muestra una imagen con bajo contraste, la figura 2.15 (b) muestra el resultado de aumentar su contraste y la figura 2.15 (c) muestra el resultado de la umbralización de esta misma imagen.

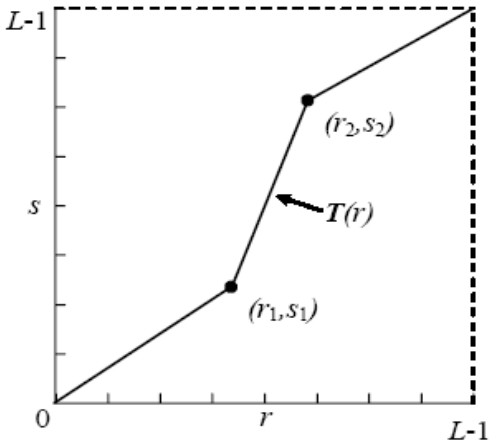


Figura 2.14. Perfil de la función de aumento de contraste [González, 1992].

Figura 2.15. a) Imagen original. b) Resultado del aumento del contraste. c) Resultado de la umbralización.

2.4.1.2.3 Planos de bits

A veces puede desearse el destacar la contribución que realizan a la imagen determinados bits específicos. Supongamos que cada píxel de una imagen viene representada por 8 bits. Imaginemos también que la imagen está compuesta de 8 planos de 1 bit, que van desde el plano 0 para el bit menos significativo hasta el plano 7 para el bit más significativo. De esta manera cada plano contiene todos los bits de igual peso en la imagen. La figura 2.16 muestra los diferentes planos de bits para la figura 2.15 b), hay que destacar que tan solo los 5 bits de mayor orden contienen los datos significativos visualmente, los otros planos de bits contribuyen a los detalles más finos de la imagen.

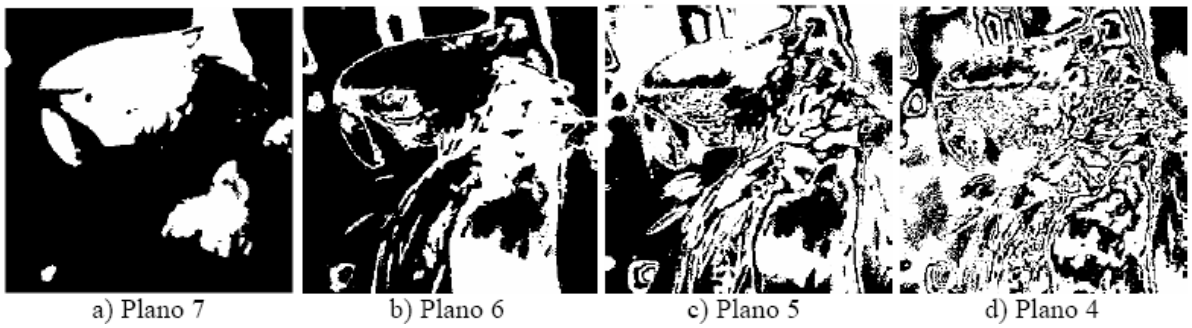


Figura 2.16. Planos de bits para la figura 2.15 b).

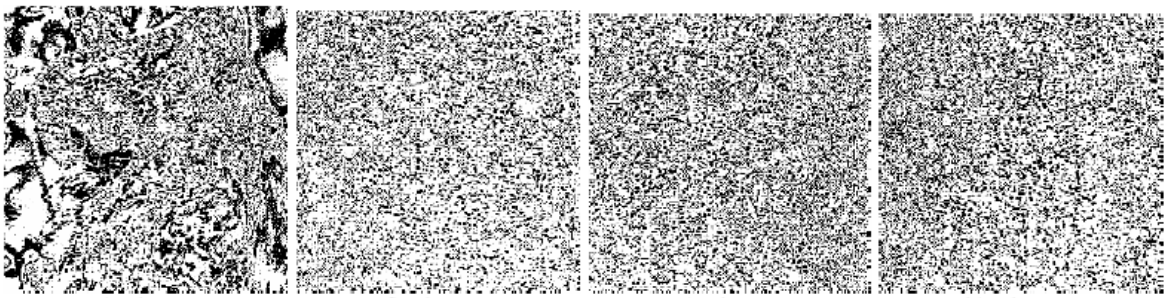


Figura 2.16 (continuación). Planos de bits para la figura 2.15 b).

2.4.1.2.4 Procesamiento de histogramas

El histograma de una imagen digital con niveles de gris en el rango [0, *L*-1] es una función discreta $p(r_k)=n_k/n$, donde r_k es el k_{esimo} nivel de gris, n_k es el número de píxeles de la imagen con ese nivel de gris, *n* es el número total de píxeles en la imagen y *k* = 0, 1, ..., *L*-1. De forma general se puede decir que $p(r_k)$ da una idea del valor de la probabilidad de que aparezca el nivel de gris r_k. La representación gráfica de esta función para todos los valores de k proporciona una descripción global de la apariencia de la imagen. Por ejemplo, la figura 2.17 muestra los histogramas para cuatro tipos básicos de imágenes, la figura 2.17 (a) muestra que los niveles de gris están concentrados hacia el extremo oscuro de la escala de grises, lo que corresponde a una imagen con una apariencia global oscura; sucede justo lo contrario con la figura 2.17 (b). La figura 2.17 (c) muestra un histograma estrecho, lo que significa que el rango dinámico es pequeño, y por tanto la imagen tiene bajo contraste, la figura 2.17 (d) corresponde a una imagen con un alto contraste. Basándonos en el histograma de una imagen podemos establecer el tipo de mejora que necesita.

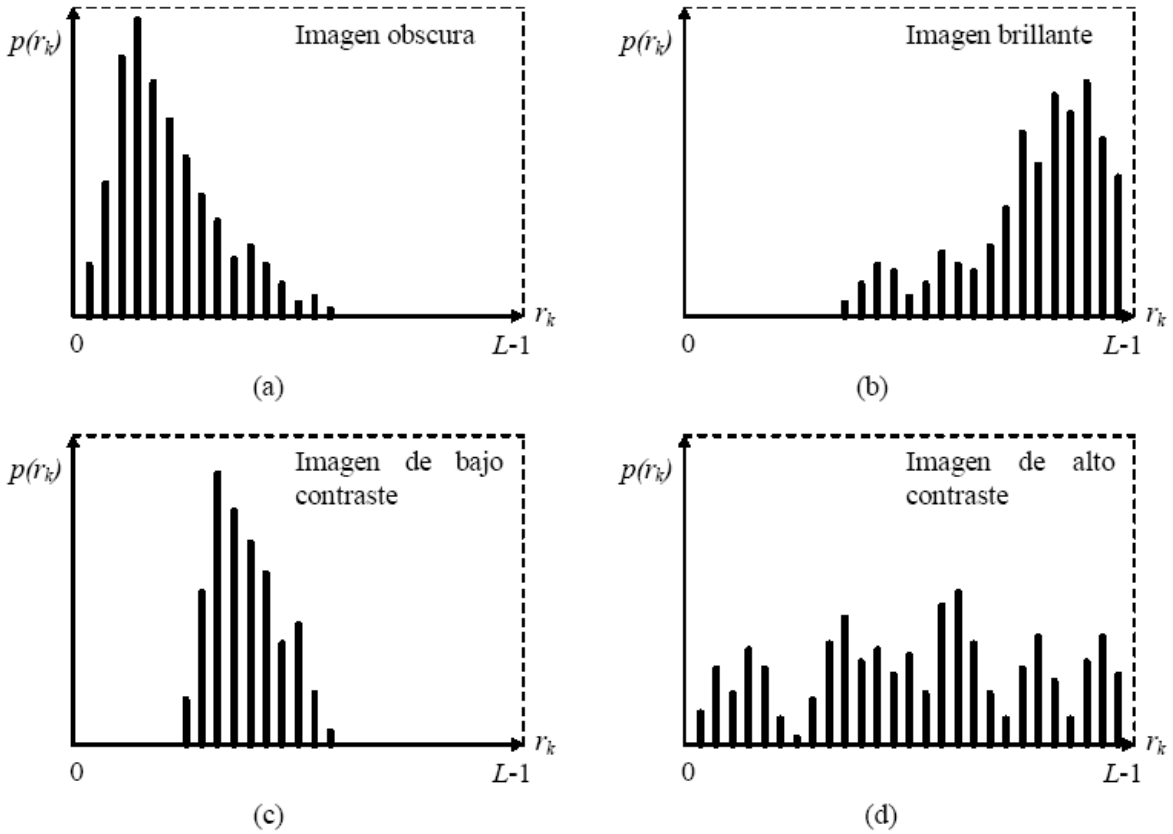


Figura 2.17. Histogramas correspondientes a 4 tipos básicos de imagen.

2.4.1.2.5 Substracción de imágenes

La diferencia entre dos imágenes *f*(*x,y*) y *h*(*x,y*), expresada de la forma:

$$g(x,y) = f(x,y) - h(x,y) \qquad \text{Ec. 2.5}$$

se obtiene calculando la diferencia entre todos los pares de píxeles correspondientes de *f* y *h*. El efecto es que solamente las áreas en las que *f*(*x,y*) y *h*(*x,y*) son diferentes aparecerán en la imagen de salida como detalles mejorados.

2.4.1.2.6 Promediado de imágenes

Consideremos una imagen con ruido *g*(*x,y*) formada por la adición de ruido aleatorio *n*(*x,y*) a una imagen original *f*(*x,y*), es decir:

$$g(x,y) = f(x,y) - n(x,y) \qquad \text{Ec. 2.6}$$

donde se ha realizado la hipótesis de que en cada par de coordenadas (*x,y*) el ruido es una función sin correlación y con valor medio cero. Si promediamos M imágenes *"estáticas"*, podremos ver que conforme aumenta M la variabilidad de los valores del píxel en el punto (*x,y*) decrece, es decir, al promediar debido al efecto aleatorio del ruido este se contrarresta, permitiendo así obtener la imagen original con un número adecuadamente grande de M.

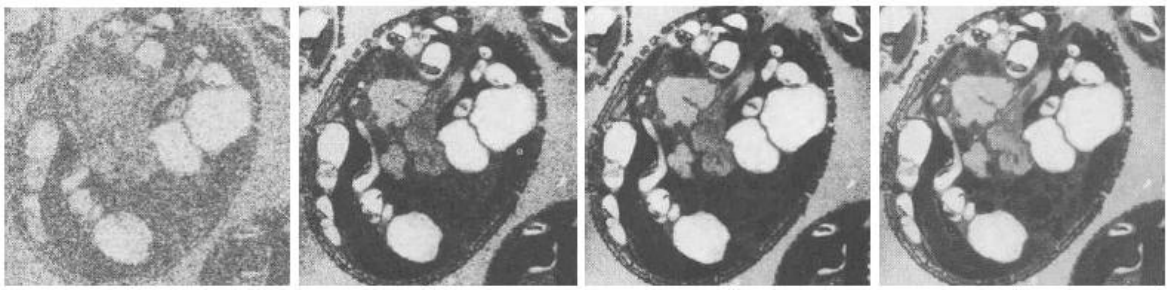


Figura 2.18. Reducción de ruido por promediado, a) imagen con ruido, b) a d) resultado de promediar 8,32 y 128 imágenes [González, 1992].

2.4.2 Filtrado espacial

El empleo de máscaras espaciales para el procesamiento de las imágenes se denomina frecuentemente *filtrado espacial* y las propias máscaras se denominan *filtros espaciales*. En esta sección se considerarán filtros lineales y no lineales para la mejora de la imagen.

El filtrado lineal consiste en sumar los productos de los coeficientes de la máscara con las intensidades de los píxeles bajo la máscara en un punto determinado de la imagen. La figura 2.19 muestra una máscara general de 3x3. Denominando a los niveles de gris de los píxeles bajo la máscara en un punto determinado como $z1$, $z2$, $z3$, ..., $z9$, la respuesta de una máscara lineal es:

$$R = w1z1 + w2z2 + ... + w9z9$$

w_1	w_2	w_3
w_4	w_5	w_6
w_7	w_8	w_9

Figura 2.19. Máscara de 3x3 con pesos arbitrarios [González, 1992].

Con respecto a la figura 2.19, si el centro de la máscara se encuentra en un punto (x,y) de la imagen, el nivel de gris del píxel situado en (x,y) se reemplaza por R. Luego se mueve la máscara hasta el siguiente píxel de la imagen y se repite el proceso hasta cubrir toda la imagen.

Los filtros no lineales operan también en entornos, sin embargo, en general su operación se basa directamente en los valores de los píxeles en el entorno en consideración y no emplean explícitamente los coeficientes en la forma descrita en la ecuación anterior.

Tanto el filtro pasa bajos como el filtro de mediana expuestos en esta sección son filtros suavizantes. Estos filtros se utilizan para hacer que la imagen aparezca algo borrosa y también para reducir el ruido. Hacer que la imagen aparezca algo borrosa puede ser útil en las etapas de preprocesado, por ejemplo en la eliminación de los detalles antes de la extracción de un objeto grande y en el relleno de pequeños espacios entre líneas o curvas antes de la extracción de contornos.

El objetivo principal de los filtros realzantes es el de destacar los detalles finos de una imagen o intensificar detalles que han sido difuminados.

2.4.2.1 Pasa bajos

Para realizar un filtro pasa bajos la máscara debe tener todos sus coeficientes positivos, como lo muestra la figura 2.20. Aunque la forma espacial del filtro pueda ser escrita por una función gaussiana (como en la figura 2.20 (b)), el requisito clave es que todos los coeficientes sean positivos. La construcción más simple consiste en una máscara en la que todos los coeficientes son 1 y el resultado se divide entre el número de elementos de la mascara (es un promedio móvil), como lo muestra la figura 2.21.

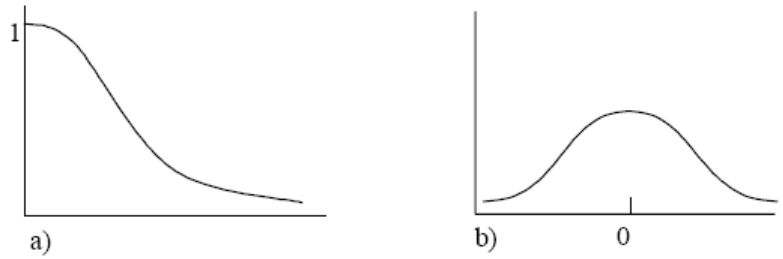


Figura 2.20. Filtro pasa bajos con simetría circular. a) Sección transversal en el dominio de la frecuencia. b) Sección transversal en el dominio espacial.

$$\frac{1}{9} \times \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Figura 2.21. Máscara de pasa bajos de 3x3.

2.4.2.2 De mediana.

Una de las características del método anterior es que difumina los bordes y otros detalles de la imagen. Cuando el objetivo es más la reducción de ruido que el difuminado, el empleo de filtros de mediana representa una posibilidad alternativa. En este caso el nivel de gris de cada píxel se reemplaza por la mediana de los niveles de gris en un entorno de ese píxel, en lugar de por la media. Este método es particularmente efectivo cuando el ruido consiste de componentes fuertes y de forma picuda, y la característica que se desea preservar es la agudeza de los bordes. Los filtros de mediana son no lineales. La figura 2.22 muestra una imagen y su tratamiento con filtros de media y mediana.

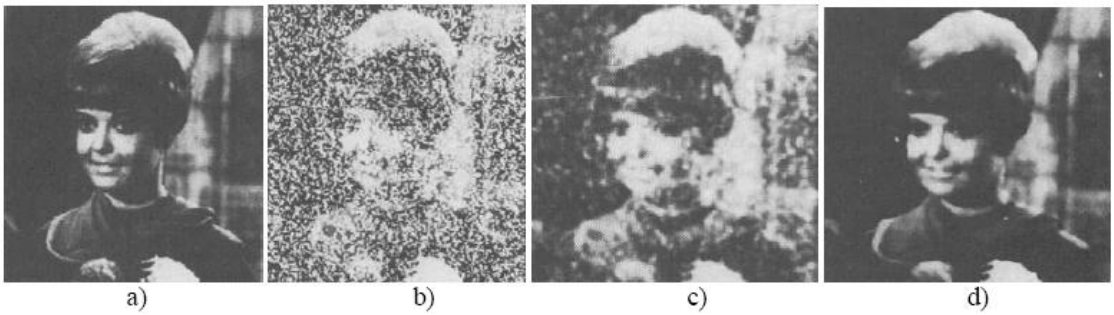


Figura 2.22. Filtro de mediana. a) Imagen original. b) Imagen corrompida por ruido en forma de impulsos. c) Resultado de promediar en un entorno de 5x5. d) Resultado de un filtro de mediana de 5x5.

2.4.2.3 Pasa alto

Para realizar un filtro pasa altos la máscara debe tener coeficientes positivos cerca de su centro y negativos en la periferia, como lo muestra la figura 2.23 b). Para una máscara de 3x3 esto se logra eligiendo un valor positivo en el centro y coeficientes negativos en el resto. La figura 2.24 muestra la implementación más clásica de un filtro de 3x3, obsérvese que la suma de los coeficientes es 0. Así, cuando la máscara esta sobre un área de nivel de gris constante o lentamente variable, la salida proporcionada por la máscara es cero o un valor muy pequeño. El hecho de haber coeficientes negativos en la máscara implica la posibilidad de obtener valores de gris negativos.

Como solo se consideran valores de gris positivos, el filtrado pasa altos necesariamente implica alguna forma de desplazamiento o cambio de escala para que al final los niveles de gris queden en el intervalo [0, *L*-1].

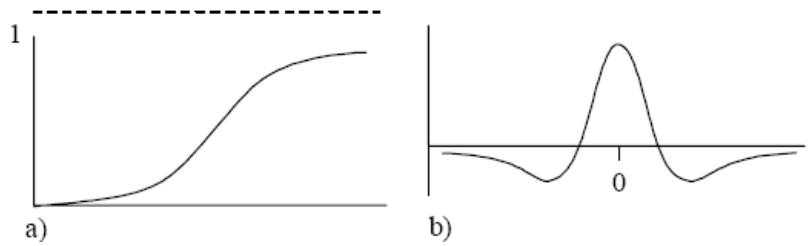


Figura 2.23. Filtro pasa Altos con simetría circular. a) Sección transversal en el dominio de la frecuencia. b) Sección transversal en el domino espacial.

$$\frac{1}{9} \times \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

Figura 2.24. Máscara Pasa Altos de 3x3.

2.5 Segmentación

El primer paso en cualquier proceso de análisis de imagen es la segmentación. Mediante la segmentación se divide la imagen en las partes u objetos que la forman. El nivel al que se realiza esta subdivisión depende de la aplicación en particular, es decir, la segmentación terminará cuando se hayan detectado todos los objetos de interés para la aplicación. En general, la segmentación automática es una de las tareas más complicadas dentro del procesado de imagen. La segmentación va a dar lugar en última instancia al éxito o fallo del proceso de análisis. En la mayor parte de los casos, una buena segmentación dará lugar a una solución correcta, por lo que, se debe poner todo el esfuerzo posible en esta etapa.

La segmentación de imágenes tiene su origen en numerosos estudios psicológicos que indican la preferencia de los humanos por agrupar regiones visuales en términos de proximidad, similitud y continuidad, para construir un conjunto de unidades *significativa*s. Existe cierta confusión en torno al concepto de segmentación ya que si bien algunos autores consideran suficiente marcar los puntos de la imagen (píxeles) con un valor indicativo de su pertenencia a determinada región o clase, otros indican que además es necesario proveer un mecanismo que permita una representación simbólica de las relaciones topológicas existentes entre las distintas unidades. Esta discrepancia se debe fundamentalmente al nivel de abstracción al cual se asocia el proceso de segmentación. Si se asocia la segmentación a un nivel de abstracción medio, ésta deberá de producir la representación simbólica anteriormente indicada, correspondiendo el marcaje a un proceso previo situado en un nivel bajo de abstracción y encaminado al objetivo de segmentar. Si bien se ha adoptado esta segunda postura, cabe reconocer que existen situaciones o aplicaciones donde no merece la pena soportar la carga computacional que supone la obtención de dicha representación simbólica. Esto suele ser cierto solamente en casos donde la tarea a realizar o el entorno están muy delimitados y controlados [Bravo 96].

En cuanto a la unidad *significativa* que rige la segmentación, ésta suele corresponder a píxeles, regiones o contornos que muestran o disciernen una similitud en cuanto a intensidad, color, textura, movimiento, etc.

2.5.1 Método basado en Píxeles [Bravo 96].

Este método de segmentación toma en cuenta solo el valor de gris de un píxel, para decidir si el mismo pertenece o no al objeto de interés. Para ello, se debe encontrar el rango de valores de gris que caracterizan dicho objeto, lo que requiere entonces la búsqueda y el análisis del histograma de la imagen.

El objetivo de este método, es el de encontrar de una manera óptima los valores característicos de la imagen que establecen la separación del objeto de interés, con respecto a las regiones que no pertenecen al mismo; debido a esta característica y si los valores de gris del objeto y del resto de la imagen difieren claramente, entonces el histograma mostrará una distribución bimodal, con dos máximos distintos, lo que debiera generar, la existencia de una zona del histograma ubicada entre los dos máximos, que no presente los valores característicos, y que idealmente fuera igual a cero, con lo cual se logrará una separación perfecta entre el objeto y la región de la imagen que lo circunda, al establecer un valor umbral ubicado en esta

región del histograma. Por lo tanto cada píxel de la imagen, es asignado a una de dos categorías, dependiendo si el valor umbral es excedido o no.

Si el valor del histograma ubicado entre los dos máximos, es distinto de cero, las funciones de probabilidad de los valores de gris del objeto y de la región restante, se solaparán, de tal manera que algunos píxeles del objeto deberán ser tomados como pertenecientes a la región circundante y viceversa. Conocida la distribución de la función de probabilidad de los píxeles del objeto y de la región circundante, es posible aplicar análisis estadístico en el proceso de buscar un umbral óptimo, con el número mínimo de correspondencias erróneas. Estas distribuciones pueden ser estimadas por histogramas locales, los cuales solamente incluyen las regiones correspondientes de la imagen.

2.5.2 Método basado en Contornos [Bravo 96].

En el método basado en píxeles, el tamaño del objeto de interés depende del nivel de umbral escogido. La variación del tamaño es una característica dada por el hecho, de que los valores de gris en el contorno de un objeto cambian gradualmente desde la región circundante hacia el mismo. El método basado en contornos puede ser usado para evitar la variación del tamaño del objeto.

Este método se basa en realizar la búsqueda del valor máximo del gradiente, sobre cada línea que forma la imagen. Cuando un máximo es encontrado, un algoritmo de trazado trata de seguir el máximo del gradiente alrededor del objeto, hasta encontrar de nuevo el punto inicial, para luego buscar el próximo máximo en el gradiente.

2.5.3 Métodos Basados en Regiones [Bravo 96].

En el método de segmentación basado en píxeles, la idea fundamental es clasificar un punto como del objeto, solamente tomando en cuenta el valor de gris que el mismo tiene asociado, lo que conlleva a que puntos aislados o pequeñas áreas puedan ser clasificadas como pertenecientes a la región de interés, es decir, allí no se toma en cuenta la conectividad como característica importante del objeto.

Métodos de segmentación basados en regiones, toman en cuenta un conjunto de puntos de la imagen, a los cuales se les analiza características como, la posición en el espacio de intensidades, las relaciones topológicas (conectividad) y las características de las fronteras entre dos conjuntos. Dependiendo de como sea analizada la posición en el espacio y las relaciones espaciales existentes entre los píxeles, se pueden encontrar métodos de clasificación y métodos por Crecimiento de Regiones.

Los métodos de Clasificación determinan primero una partición del espacio de intensidades y utilizan luego las relaciones de conectividad, para determinar una región. Los métodos de Crecimiento de Regiones utilizan de manera simultánea los dos tipos de información.

2.6 Método para la detección de contornos [González 92].

La detección de bordes es el procedimiento empleado más habitualmente para la detección de discontinuidades. Un borde se define como la frontera entre dos regiones con nivel de gris relativamente diferente. Vamos a suponer a partir de ahora que las regiones de interés son suficientemente homogéneas de modo que la transición entre dichas regiones se puede determinar empleando exclusivamente las discontinuidades en el nivel de gris. La idea básica detrás de cualquier detector de bordes es el cálculo de un operador local de derivación. En la figura 2.25 se puede ver este concepto. En la parte izquierda se puede ver una imagen de una banda clara sobre un fondo oscuro, el perfil a lo largo de una línea horizontal y la primera y segunda derivada de dicho perfil. Se puede observar que el perfil del borde se ha modelado como una discontinuidad suave. Esto tiene en cuenta el hecho de que en las imágenes reales los bordes están ligeramente desenfocados. Como se puede observar en la figura 2.25 la primera derivada es positiva para los cambios a un nivel de gris más claro, negativa en caso contrario y cero en aquellas zonas con nivel de gris uniforme. La segunda derivada presenta valor positivo en la zona oscura de cada borde, valor negativo en la zona clara de cada borde y valor cero en las zonas de valor de gris constante y justo en la posición de los bordes. El valor de la magnitud de la primera derivada nos sirve para detectar la presencia de bordes, mientras que el signo de la segunda derivada nos indica si el píxel pertenece a la zona clara o a la zona oscura. Además la segunda derivada presenta siempre un cruce por cero en el punto medio de la transición. Esto puede ser muy útil para localizar bordes en una imagen. Aunque lo que llevamos diciendo se refiere a perfiles unidimensionales, la extensión a dos dimensiones es inmediata. Simplemente se define el perfil en la dirección perpendicular a la dirección del borde y la interpretación anterior seguirá siendo válida. La primera derivada en cualquier punto de la imagen vendrá dada por la magnitud del gradiente, mientras que la segunda derivada vendrá dada por el operador Laplaciano.

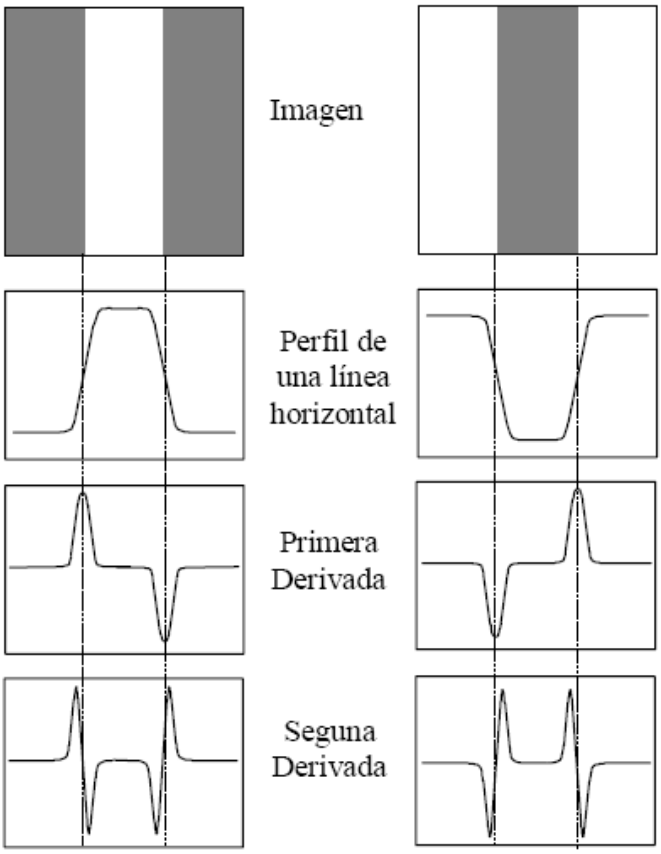


Figura 2.25. Detección de contornos empleando operadores de derivación.

2.6.1 Operador SUSAN

El operador SUSAN (Smallest Univalue Segment Assimilating Nucleus) para la detección de contornos y esquinas fue desarrollado por S. Smith y J. Brady, [Smith, 95]. En dicho trabajo se presenta un nuevo enfoque para el procesamiento de imágenes a bajo nivel. Se busca introducir un aspecto clave en el desarrollo de un sistema de visión artificial: la rapidez [Rezai, 2006][*Wang*, 2004]. Este nuevo enfoque es guiado por la premisa de obtener un sistema que extraiga características de la imagen (aristas, esquinas, uniones) en el menor tiempo posible. Por supuesto, el error cometido en la localización de estas características debe ser mínimo. Para ello evitan, en la medida de lo posible, el uso de derivadas para la obtención de las características bidimensionales. Esto debido a que la diferenciación amplifica las componentes de alta frecuencia del ruido lo que implicará una pérdida de estabilidad o continuidad [Lei, 2010].

2.6.1.1 Principio SUSAN

El principio de este método consiste en aplicar una máscara circular en cada píxel de la imagen y calcular el número de píxeles dentro de dicha máscara, con un nivel de gris similar al del píxel central. La técnica SUSAN está basada en el hecho de que cada píxel en una imagen tiene un área asociada con un nivel de gris similar al central.

El número de píxeles o área calculada con la máscara se llama USAN (Univalue Segment Assimilating Nucleus). Esta área contiene información acerca de la estructura de una imagen que rodea al píxel evaluado por la máscara y tiene un valor cercano al 50% en regiones próximas a un contorno. En la figura 2.26 podemos ver un ejemplo de esto, cuando el pixel central de la máscara queda en un punto lejano a un contorno (casos A y E), el área USAN es el 100% de la mascara (21 pixeles para una máscara de 21 pixeles); conforme se acerca la máscara al contorno el área usan decrece (caso D) hasta llegar a valores muy cercanos al 50% cuando el pixel central se encuentre exactamente sobre el contorno (caso C), sin importar de qué lado del mismo se encuentre. Mientras la máscara se encuentre sobre un contorno, el valor del área USAN se mantiene cercano al 50% sin decrecer significativamente, solamente al aproximarse a esquinas (caso B) se obtienen disminuciones significativas en el área USAN, por ejemplo al situar el pixel central de la máscara sobre una esquina en 90 grados obtendremos valores del área USAN cercanos al 25%.

Esta propiedad determina la detección de contornos y esquinas en una imagen. Para ejemplificar lo anterior, en la fig. 2.26, se muestra una imagen en la que se ha resaltado el USAN dentro de cada círculo.

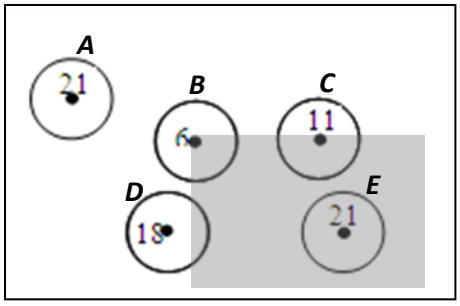


Fig. 2.26. Máscaras circulares localizadas en las diferentes regiones de una imagen. El número representa el área USAN para una máscara circular de 21 píxeles [Smith, 1995].

Para segmentar con el algoritmo SUSAN se necesita obtener una imagen de entrada, filtrar con una máscara predefinida y aplicar una serie de reglas con los datos locales para al final, obtener una imagen con los contornos resaltados.

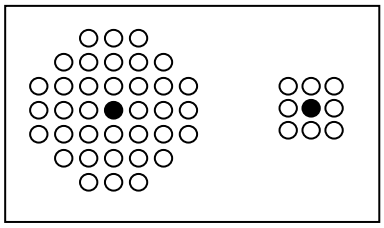


Fig. 2.27. Máscaras circulares de 37 y de 9 píxeles [Smith, 1995].

La máscara circular asegura una respuesta isotrópica (ver Fig. 2.27). La más usada contiene 37 píxeles, aunque es posible obtener buenos resultados usando una máscara pequeña de 3 x 3 píxeles. Recordemos que la máscara es puesta en cada píxel de la imagen y el valor de la intensidad de este dentro de ella, es comparado con el valor del píxel central. El resultado de la comparación es dado por la siguiente ecuación.

$$c(r,r_o)=\begin{cases}1 & si \quad |1(r)-1(r_o)|\le t\\ 0 & si \quad |1(r)-1(r_o)|> t\end{cases}$$

Donde r_o es la posición (x,y) del píxel central de la máscara, r es la posición (x,y) de cualquier otro píxel dentro de ella, I(r) es la intensidad del píxel en el punto r(x,y), t es un umbral para la diferencia de intensidades, el cual determina el contraste mínimo para detectar cambios en la imagen y c es el resultado de la comparación. Cabe aclarar que un valor de 25 es adecuado para la mayoría de las imágenes reales.

El número de píxeles con una intensidad similar al valor del píxel central puede ser obtenido con la ecuación:

$$n(r_o) = \sum_{r \in D} c(r, r_o)$$

donde *D* es el dominio de la mascara en x e y. Este valor define el área USAN para el píxel r_o

2.6.1.2 Detector de Contornos

Dada el área USAN las aristas se detectan comparando su valor con un umbral geométrico constante g, que en este caso se asigna a $3n_{max}/4$, donde n_{max} es el valor máximo que n puede tomar (el área correspondiente al dominio del filtro). El valor de g es más grande que el valor de 50% previamente discutido para la detección de contornos, con objeto de agregar capacidad de rechazo de ruido al algoritmo. Sin embargo, el umbral geométrico no es crítico para una detección de contornos precisa. La respuesta inicial de la detección de contornos es obtenida por la aplicación de la siguiente ecuación:

$$R(r_o) = \begin{cases} g - n(r_o) & si \quad n(r_o) < g \\ o & otros \quad casos \end{cases}$$

2.7 Método común en la obtención de trayectorias de escape

Existen estudios del comportamiento del camarón cuando es atacado por algún depredador, uno de ellos es el presentado por [Arnott, 1998] del que se describirá lo mas importante.

Esta investigación se basó en estudiar los camarones de cuerpo largo que median entre 11 y 69mm, cuando eran atacados por un depredador tal como el pez bacalao (Gadus morhua), ellos se ayudaron con una cámara de gran velocidad, para analizar su comportamiento.

Estos animales poseen una elongación en el abdomen que puede ser usada apropiadamente para propulsarlo a través del agua con una gran fuerza esto es lo que se conoce como el tail-flip. Este comportamiento natural es empleado principalmente como una respuesta de escape cuando sienten que son atacados por un depredador. La fuerza de propulsión generada por el tail-flip es derivada de una combinación de fuerzas reactivas (la masa adicional)

y fuerzas resistivas (resistencia aerodinámica), la combinación de ambas fuerzas desplazan el centro de gravedad.

En la investigación demuestran que el tail-flip tiene varios factores del que puede depender, uno es el tamaño del animal siendo un tamaño especial en el que el rendimiento del tail-flip es óptimo. Otros factores que afecta son el hábitat y la orientación del cuerpo.

Uno de los puntos necesarios para el análisis del tail-flip es el centro de masa (punto 4 de la figura 2.28_A), para determinarlo se utiliza un procedimiento que consiste en la suspensión de muestras congeladas (-10 ° C) con el abdomen completamente extendido (es decir, en su postura normal del cuerpo en reposo), y con su abdomen completamente flexionado a fin de determinar el cambio en la posición del centro de masa en el transcurso del tail-flip. Usando este método, definimos que el centro de masa se encuentra dentro de los límites de la mitad ventral del primer segmento abdominal cuando el camarón se encuentra en una posición completamente extendida. Cuando el cuerpo está totalmente flexionado, el centro de masa se desplaza ligeramente hacia el quinto pereiópodo.

Para el análisis solo considero los momentos de los escapes del camarón en los que tenían más de un tail- flip. De 89 filmaciones que cubrían estos requisitos seleccioné 25 para el análisis cinemático, esto fue en base a que el camarón nadó por arriba y en paralelo al objetivo.

En la figura 2.28 se muestran los puntos necesarios así como la ubicación para la medición de los ángulos para el análisis del video de alta velocidad, en (A) se muestra la toma del lateral del camarón en los que se marcan 4 puntos. El punto 1 los ojos; el punto 2 el abdomen en su punto medio en el momento que es atacado; el punto 3 extremo posterior del sexto segmento abdominal; el punto 4 el centro de masa. En (B) se muestra como podemos medir los ángulos generados cuando se realiza el tail-flip y en (C) se presenta una vista desde arriba en el que se pude apreciar la cabeza el cuerpo y la cola del camarón completamente estirado.

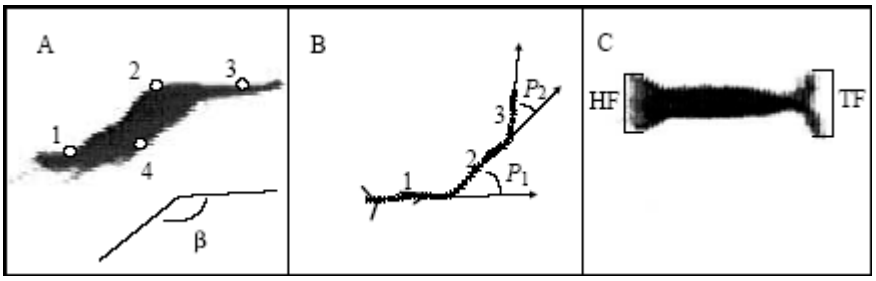


Figura 2.28. a) Toma lateral del camarón. b) Indicación de los ángulos en el momento del tail-flip. c) Vista desde arriba [Arnott, 1998].

El ángulo del cuerpo de los camarones se define como el ángulo subtendido por los puntos 1, 2 y 3 (figura 2.28(A)). Los cambios de ángulo (Ángulo β) entre las imágenes sucesivas se utilizaron para determinar velocidad angular máxima y la aceleración angular máxima durante la flexión o en las fases de re-extensión del tail-flip (es decir, el valor máximo alcanzado en un intervalo de 5 ms). La velocidad angular media de la flexión completa o de re-extensión circular del abdomen lo calcularon como: (total de ángulo Δ)/(tiempo total tomado). Los ángulos negativos fueron asignados a los movimientos de flexión y los ángulos positivos fueron para los movimientos de re-extensión del cuerpo.

En la figura 2.29 se ven dos tomas, la fila de arriba da una toma lateral mientras que la fila de abajo da una toma desde arriba del camarón en estudio. En el se puede apreciar como va disminuyendo su tamaño a medida que se va realizando el tail-flip.

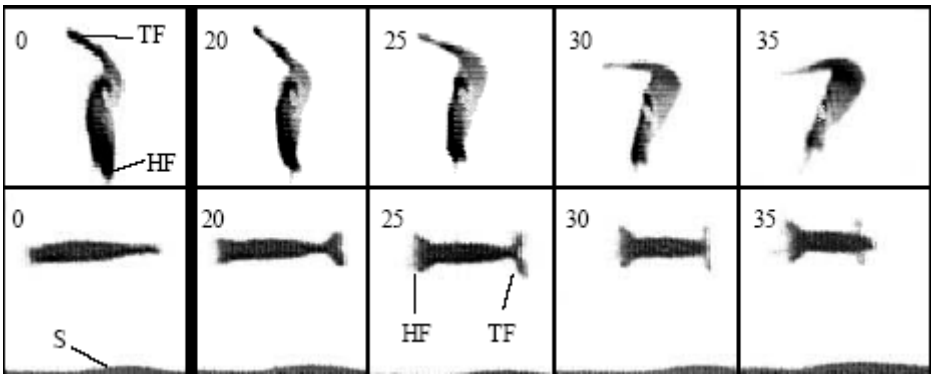


Figura 2.29. Toma lateral y de arriba para el camarón en el momento del Tail-Flip [Arnott, 1998].

El desplazamiento de los camarones se determinó midiendo la distancia recorrida por las posiciones digitalizadas de los estimados centro de masa entre un frame y el siguiente. La distancia recorrida por el tail-flip se calculó como la suma acumulada de estos valores en una flexión de cuerpo completo/un ciclo de re-extensión. La velocidad media durante un múltiplo entero del tail-flip fue calculado como: (distancia recorrida acumulada) / (tiempo transcurrido).
La velocidad en intervalos de tiempo de 5 ms se determinó de la distancia recorrida por el centro de masa entre un la imagen y la siguiente. Los valores de velocidad máxima para cada tail-flip se derivan de los valores pico alcanzados (en un intervalo de 5 ms) durante la flexión del cuerpo. Los valores de la aceleración no se han incluido debido al error asociado con el cálculo.

Durante el tail-flip, el empuje se produce cuando los extremos se mueven a través del agua con respecto al centro de la masa. En 12 secuencias, las posiciones de los puntos 1 y 3 con respecto al punto 4 (centro de masas) se determinaron en el comienzo y final de cada fase de flexión. A partir de estos datos, se calculó la distancia recorrida durante la flexión del cuerpo, por el movimiento de la cabeza y la cola con respecto al centro de masa (note que los puntos 1 y 3 se encuentran en la base de la cabeza y de la cola, respectivamente).

Los camarones respondieron a un estímulo visual o por vibración, ya sea cualquiera de los dos, un simple tail-flip comprende un solo ciclo de flexión abdominal y re-extensión, o más comúnmente, una serie de múltiples tail-flip (denominado combate de natación de escape). La latencias de escape (el tiempo entre el primer momento del estimulo hasta el primer momento visible de movimiento del camarón) tenían entre 10 y 15 ms

Los resultados de estas investigaciones son para camarones que miden 11mm de longitud tienen un promedio de 30 a 50 ms de duración del tail-flip. La velocidad media del centro de masa para camarones que midieron 8mm fue de 0.26 m/s y fue la más baja mientras que la más alta fue de 1.42 m/s para camarones que midieron 46mm. Durante la fase de flexión en el tail-flip las velocidades extremas del centro de masa fueron: la más baja 0.59m/s para un camarón de 20mm, y la más alta 2.31 m/s para un camarón de 57mm.

En otra investigación presentada por Arnott [Arnott, 1999] se describen las posibles rutas de escape del camarón cuando es atacado por algún depredador. Los experimentos llevados ahí muestran nuevos comportamientos en las trayectorias de escape.

Se trabajó tanto con estímulos naturales como artificiales en donde las trayectorias de escape se ven afectadas más bien por el ángulo de ataque que por el tipo de estímulo. Ya que los camarones deben de procesar información sensorial sobre la posición del atacante para activar el control del motor y poder realizar el tail-flip.

Los resultados demuestran el grado en el cual el depredador puede predecir la dirección de escape inicial del camarón dependiendo del ángulo de ataque. En la figura 2.30 se muestra a un camarón cuando es atacado por un depredador. La flecha indica al atacante siendo la parte negra la posible área de ataque y las partes punteadas las rutas que no utiliza para escapar siendo la posible ruta de escape la parte que no tiene marcas.

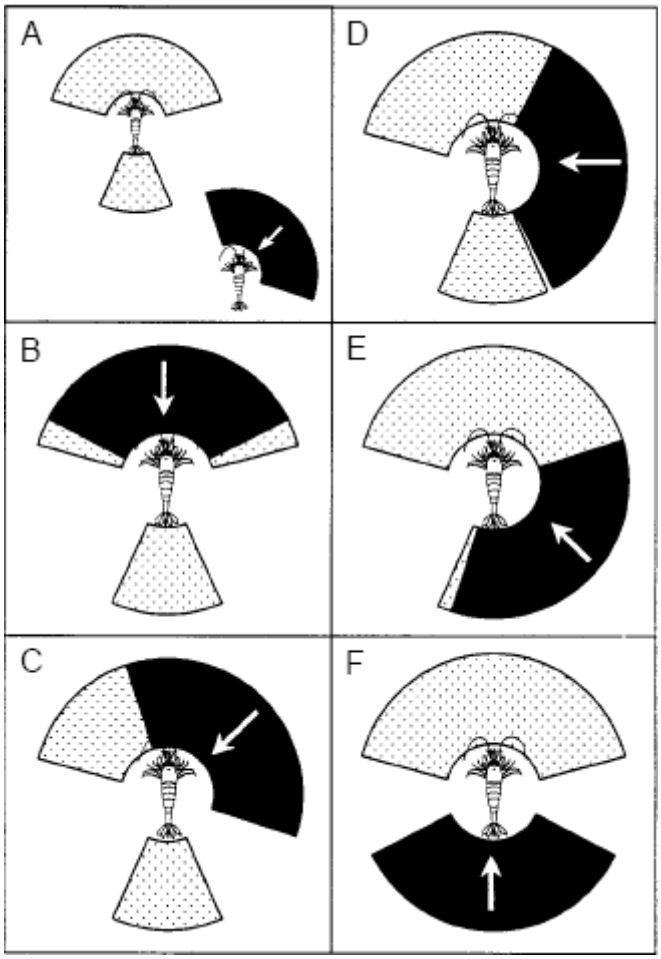


Figura 2.30. Posibles rutas de escape cuando el camarón es atacado por un depredador. A) A 63 grados. B) A cero grados. C) A 45 grados. D) A 90 grados. E) A 135 grados. F) A 180 grados [Arnott, 1999].

Para esta Tesis el aspecto más importante con respecto a estas investigaciones es la manera en que evalúan la posición y orientación del camarón. Apoyados en estas investigaciones y en conjunto con investigadores del área se llegó al acuerdo de utilizar el centro de masa y el punto localizado en la cola del camarón, ya que basados en el análisis de estos puntos los investigadores pueden obtener los parámetros que requieren para el estudio del comportamiento del camarón. Es importante comentar que fue necesario realizar adaptaciones al método de análisis ya que en el experimento se tienen condiciones de observación diferentes, contando únicamente con la vista superior de los especímenes.

CAPITULO 3 – ANÁLISIS Y DISEÑO DEL SISTEMA

3. Introducción

Para recuperar la imagen que envía la cámara en el formato Camera Link es necesario deserielizar los datos y almacenar la imagen. Para efectuar esta operación propongo el diagrama de bloques que se muestra en la figura 3.1; en las páginas siguientes se describirán en detalle cada uno de los bloques.

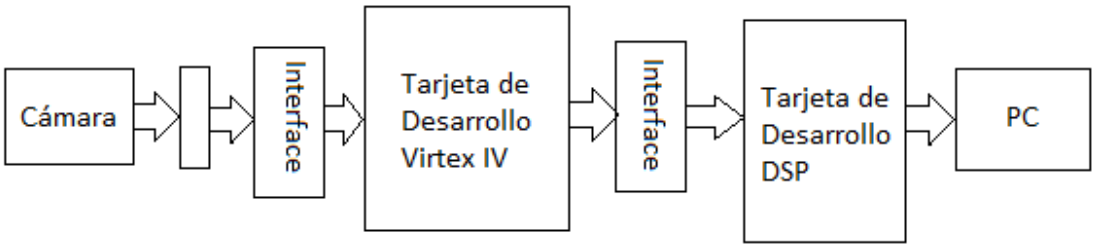


Figura 3.1. Diagrama a bloques de las conexiones

El algoritmo requiere contar con una imagen del fondo, para ello se estableció como procedimiento, que al iniciar el sistema la cámara se encuentre dirigida hacia la tina en que se realizarán los experimentos, y en la cual no deben haber animales. El procedimiento se inicia almacenando la imagen del fondo, acto seguido se preparara el experimento, colocando los animales en la tina y ejecutando el proceso de captura de imágenes. Debido a que la cámara entrega las imágenes directamente en formato Bayer [Bayer, 1976], el primer paso para poder procesar las imágenes es convertirlas de dicho formato al formato RGB estándar, que es el formato que utilizan los algoritmos conocidos. Para eliminar el fondo se realiza una resta entre la imagen actual y la imagen de fondo que se guardó previamente. Para definir bien el resultado obtenido se umbralizó, debido a que en varias ocasiones, durante las pruebas realizadas el cuerpo del animal se dividía; para unir estas divisiones se difuminó y nuevamente se umbralizó. Después se segmentó la imagen para obtener datos como el centro de masa y el área. Para evitar falsos positivos se vio en la necesidad de determinar los estándares en tamaño del camarón y la jaiba, datos que se utilizaron en una etapa de discriminación que elimina las regiones que no pertenecen a los animales en estudio, una vez determinadas las áreas de los animales se dispuso a encontrar el punto más lejano del centro de masa del camarón para definir el vector de posición, después determinamos los dos puntos más lejanos del centro de masa en la jaiba para calcular su vector de posición, para finalizar con el envío de datos al ordenador.

En la figura 3.2 se representa cada una de las etapas del software que propongo y más adelante se describirá cada uno de los siguientes bloques.

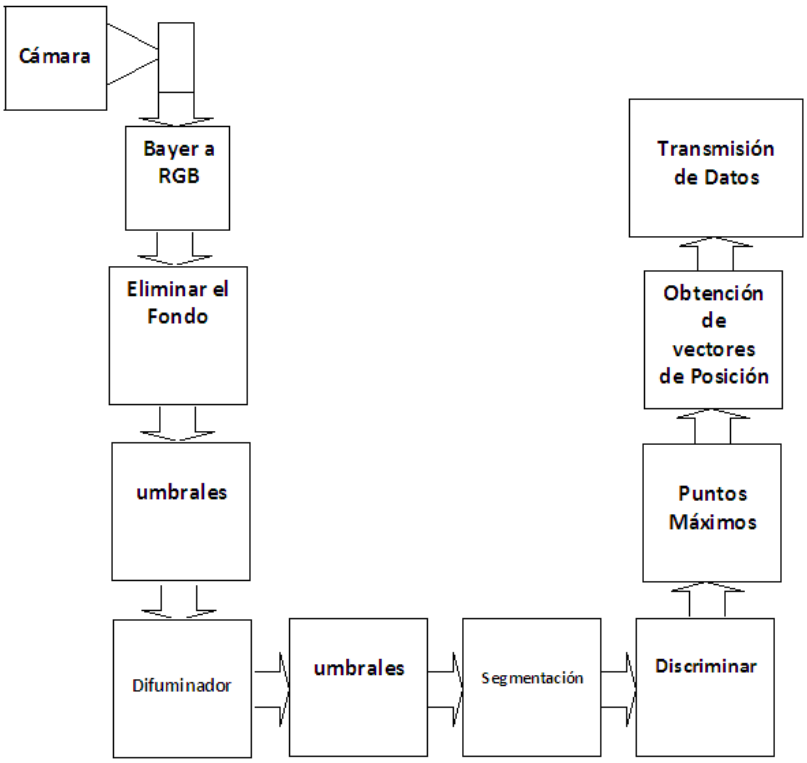


Figura 3.2.Diagrama a bloques del software

3.1 Conexión Camera Link al FPGA

3.1.1.Tarjeta SI 1280–FGPA

Tomando en cuenta que se utiliza la tarjeta de evaluación LX60 de Avnet y una cámara SI-1280 Silicon Imaging es necesario el diseño de una tarjeta que permita interconectar estos dos elementos. Para ello se diseñó una tarjeta que pudiera servir de puente y transfiera las señales de la cámara al FPGA. La tarjeta constaría de dos elementos claves: un conector MDR-26 de alta densidad para conectar la cámara y un conector AMP 5-179010-6 de 140 pines para conectar a la tarjeta de evaluación LX60 de Avnet. El desarrollo de esta tarjeta se realizó mediante la herramienta OrCAD de la empresa Cadence. La tarjeta consta de dos caras, una para las señales y otra para un seudo-plano de tierra. Las pistas debido a que son diferenciales se mantienen en parejas con una distancia entre señal positiva y señal negativa de ≈0.2mm y un ancho de pista de 0.3mm.

En base a las recomendaciones realizadas en el anexo D1, se evitaron los ángulos rectos, se calcularon las distancias entre las líneas que forman el par diferencial así como la distancia entre pares diferenciales, respetando la tolerancia permitida. Las líneas con mayores requerimientos son las cuatro líneas de datos y la señal de reloj. Las seis líneas restantes permiten mayor tolerancia debido a que cuatro de ellas son para el control de la cámara y dos de ellas para la comunicación serie basada en el protocolo RS232 que utiliza para su configuración. Estas seis líneas se utilizan de forma independiente, situación que las hace menos exigentes.

En la figura 3.3 y en la figura 3.4 se muestra la capa de tierra y de señales para el circuito impreso de la tarjeta cámara- FPGA.

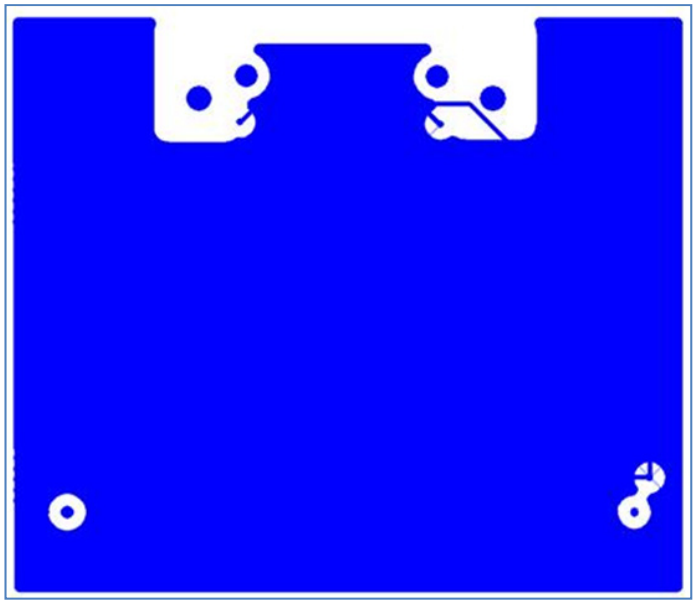


Figura 3.3.Capa de tierra para el circuito impreso (Bottom)Cámara-FPGA

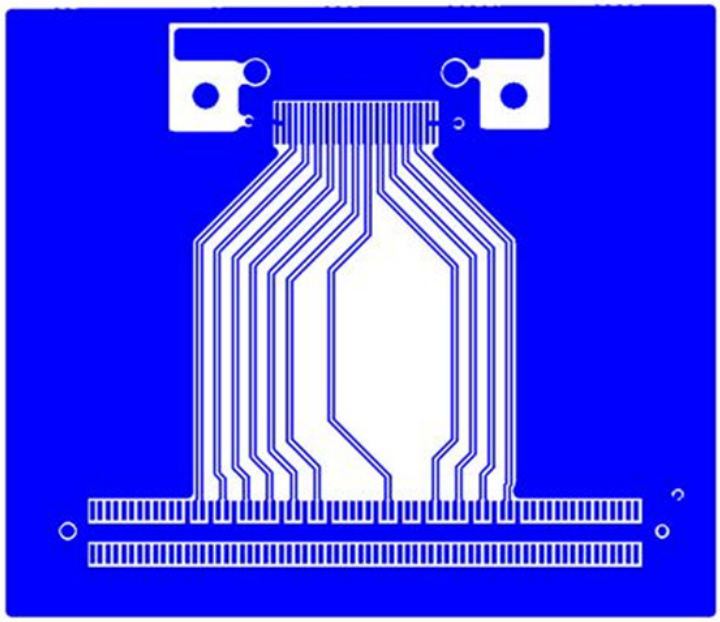


Figura 3.4.Capa de señales para el circuito impreso (TOP) Cámara-FPGA

En la tabla 3.1, se puede observar las características eléctricas que se generan al fabricar el circuito impreso, así como la longitud de las líneas para las señales, todos estos datos los tomamos de la herramienta Hyperlynx 7.5.

Par diferencial	Longitud	Z0 minima	Z0 maxima	Capacitancia	Resistencia total	Z0 total	Crosstalk pico	Retraso total
X0	8.257 cm	63 ohms	63 ohms	17 pF	0.138 ohms	63 ohms	419 mV	0.4777 ns
X1	8.246 cm	63 ohms	63 ohms	17 pF	0.138 ohms	63 ohms	378 mV	0.4771 ns
X2	8.245 cm	63 ohms	63 ohms	17 pf	0.138 ohms	63 ohms	357 mV	0.4770 ns
X3	8.245 cm	63 ohms	63 ohms	17 pF	0.138 ohms	63 ohms	306 mV	0.4770 ns
XCLK	8.246 cm	63 ohms	63 ohms	17 pF	0.138 ohms	63 ohms	330 mV	0.4771 ns
SERTC	8.456 cm	63 ohms	63 ohms	17 pF	0.138 ohms	63 ohms	157 mV	0.4892 ns
SERTFG	8.248 cm	63 ohms	63 ohms	17 pF	0.138 ohms	63 ohms	164 mV	0.4772 ns
CC1	8.454 cm	63 ohms	63 ohms	17 pF	0.138 ohms	63 ohms	172 mV	0.4891 ns
CC2	8.455 cm	63 ohms	63 ohms	17 pF	0.138 ohms	63 ohms	185 mV	0.4892 ns
CC3	8.241 cm	63 ohms	63 ohms	17 pF	0.138 ohms	63 ohms	195 mV	0.4769 ns
CC4	8.047 cm	63 ohms	63 ohms	17 pF	0.138 ohms	63 ohms	220 mV	0.4656 ns

Tabla 3.1.Características de las señales de la tarjeta Cámara-FPGA

Las señales primordiales en esta tarjeta son las señales de datos (X0, X1,X2 y X3) y la señal de reloj (XCLK) provenientes de la cámara, por este hecho es necesario realizar un estudio acerca del comportamiento de las señales a través de estas pistas y con ello conocer el deterioro de la señal y el ruido que se acopla durante su recorrido hacia el FPGA. Para conocer estos datos es necesario realizar un análisis de diagrama de ojo que permita conocer el comportamiento de estas señales. Los resultados de este análisis se resumen en los diagramas de ojo para cada una de las señales mostradas en las figuras 3.5, 3.6, 3.7, 3.8 y 3.9.

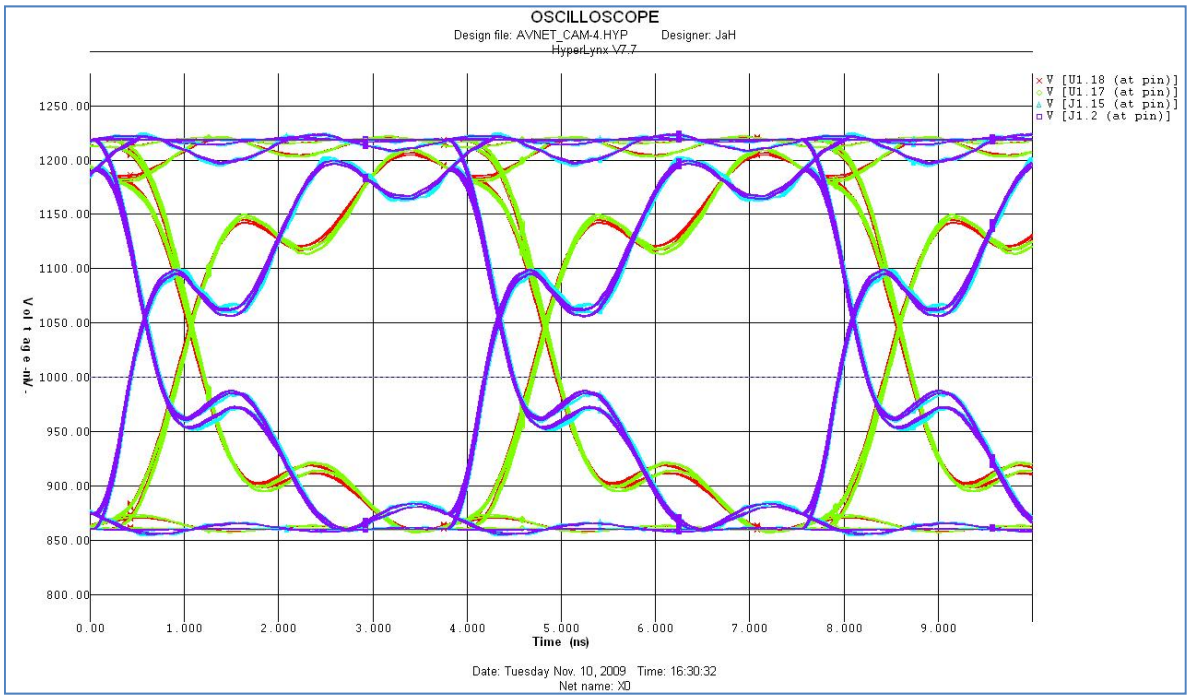


Figura 3.5.Diagrama de ojo para la señal X0

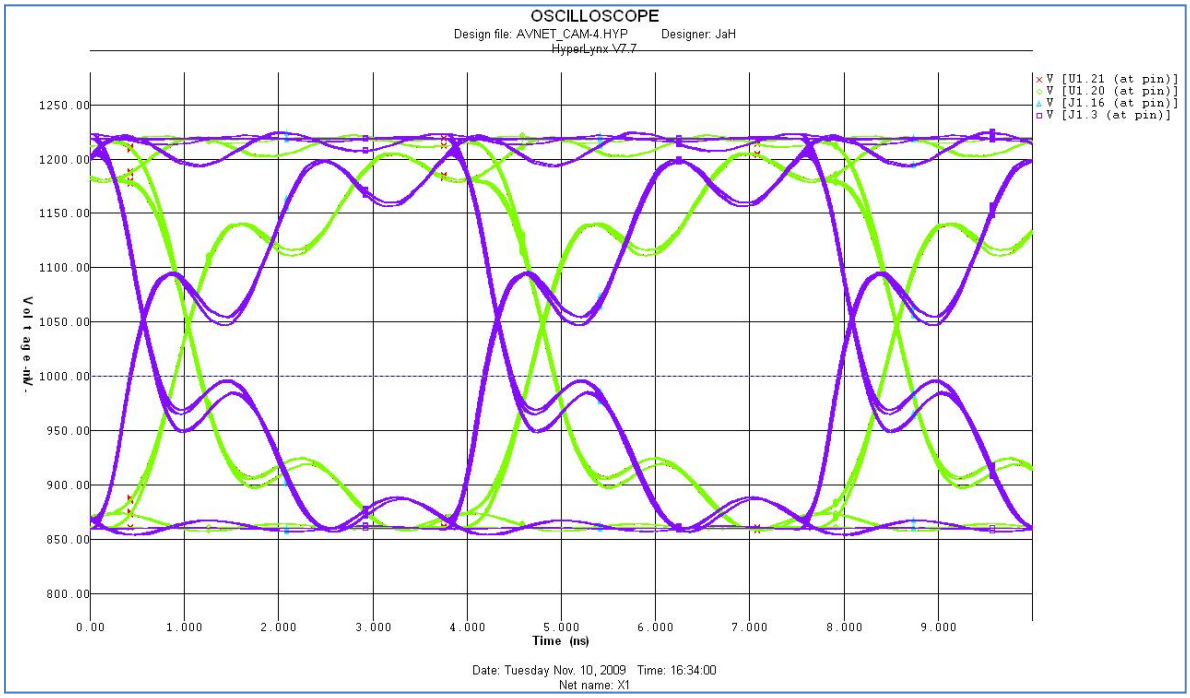


Figura 3.6.Diagrama de ojo para la señal X1

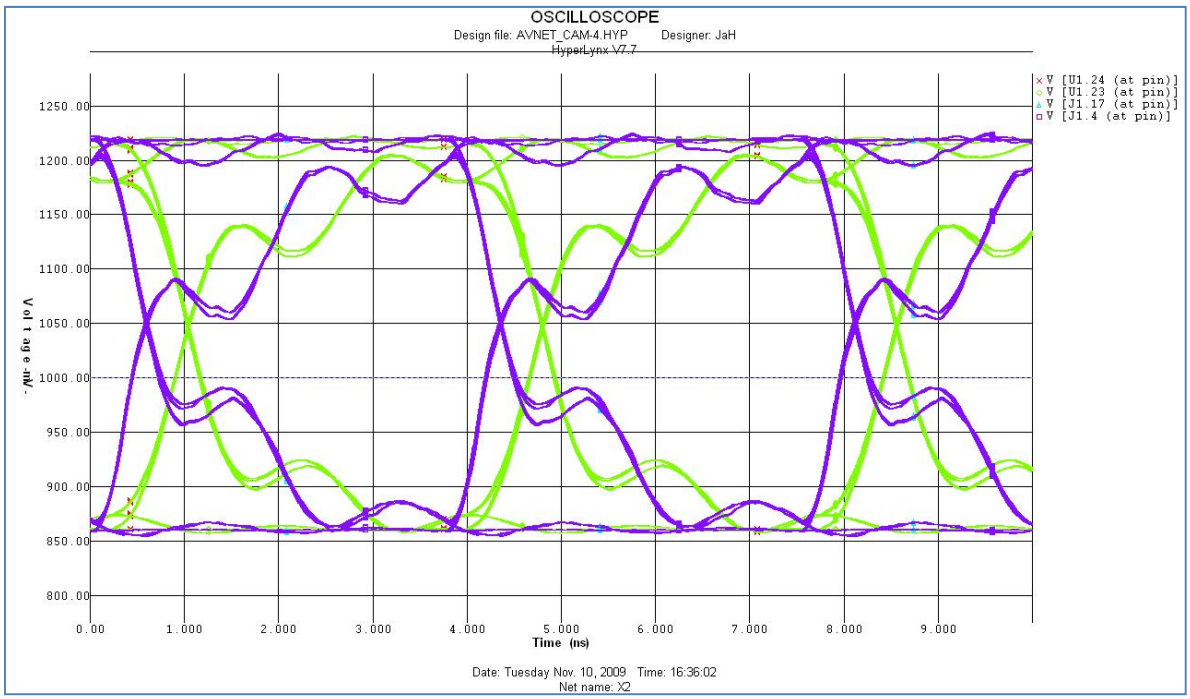


Figura 3.7.Diagrama de ojo para la señal X2

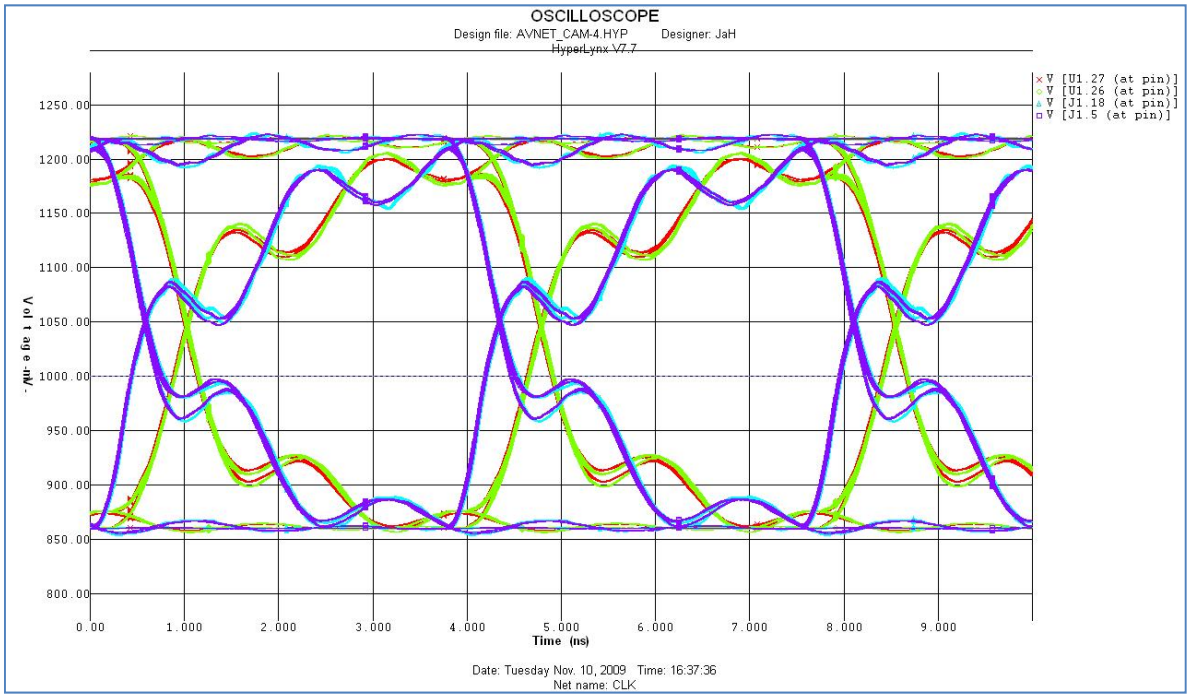


Figura 3.8.Diagrama de ojo para la señal XCLK

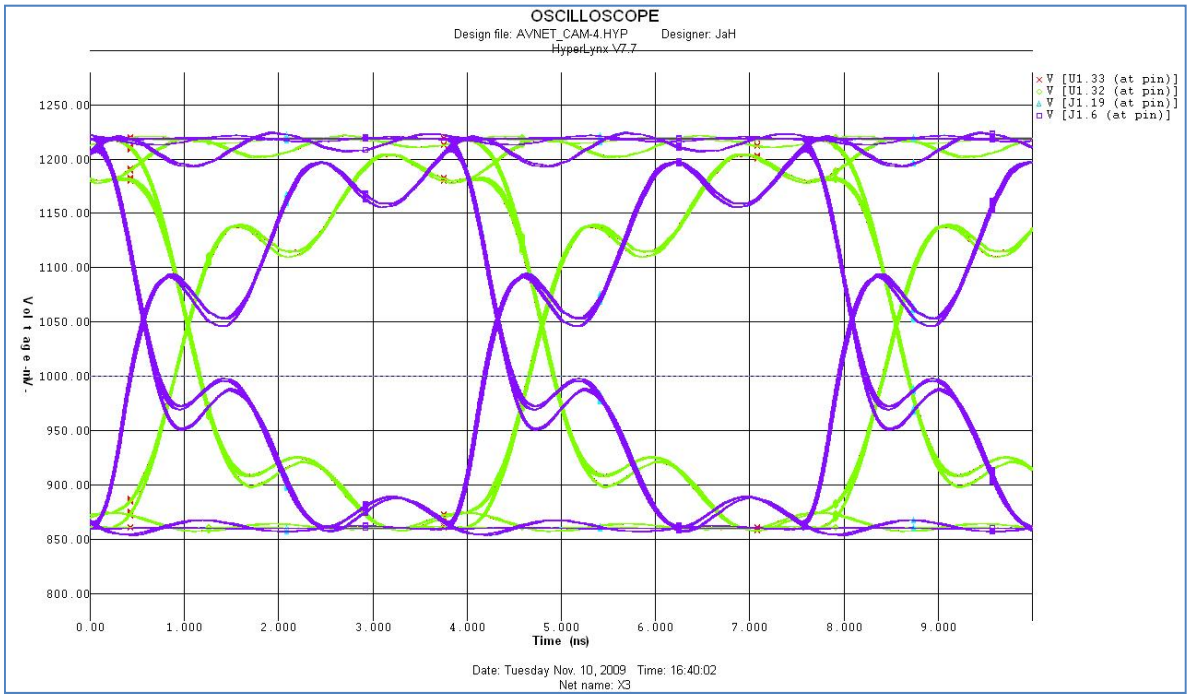


Figura 3.9. Diagrama de ojo para la señal X3

Considerando las simulaciones y los datos de la tarjeta en general podemos confirmar que la tarjeta es apta para implementarse en el sistema.

3.1.2Tarjeta DSP-FPGA

El sistema de adquisición de datos contempla una comunicación con un DSP mediante el puerto EMIF. Por este motivo se requiere el diseño de una tarjeta que sirve para comunicar el FPGA con el DSP. El motivo de este enlace es el de utilizar el DSP para el procesamiento general de la información enviada por la cámara y adquirida por el FPGA. A partir del paso de deserialización dentro del FPGA el sistema almacena los datos adquiridos en una memoria de doble puerto para que pueda ser accedida por el DSP. El puerto EMIF del DSP permite acceder a esta memoria e ingresar los datos al DSP de forma directa y sencilla. La tarjeta diseñada contempla dos conectores esenciales, el conector de expansión de la tarjeta de evaluación DSK6416 y el conector AMP 5-179010-6 de 140 pines de la tarjeta de evaluación LX60 de Avnet, además de la interconexión de estos dos elementos se provee de una serie de pines para las interrupciones. La tarjeta diseñada cuenta con dos caras, la cara superior del circuito impreso (top) sirve para rutear las pistas del EMIF y la cara de abajo del circuito impreso (Bottom) para las interrupciones. Las pistas cuentan con un ancho de 0.35mm a excepción de las líneas de alimentación y tierra que son de 0.5mm.En las figuras 3.10 y 3.11, muestra las caras del circuito impreso DSP-FPGA.

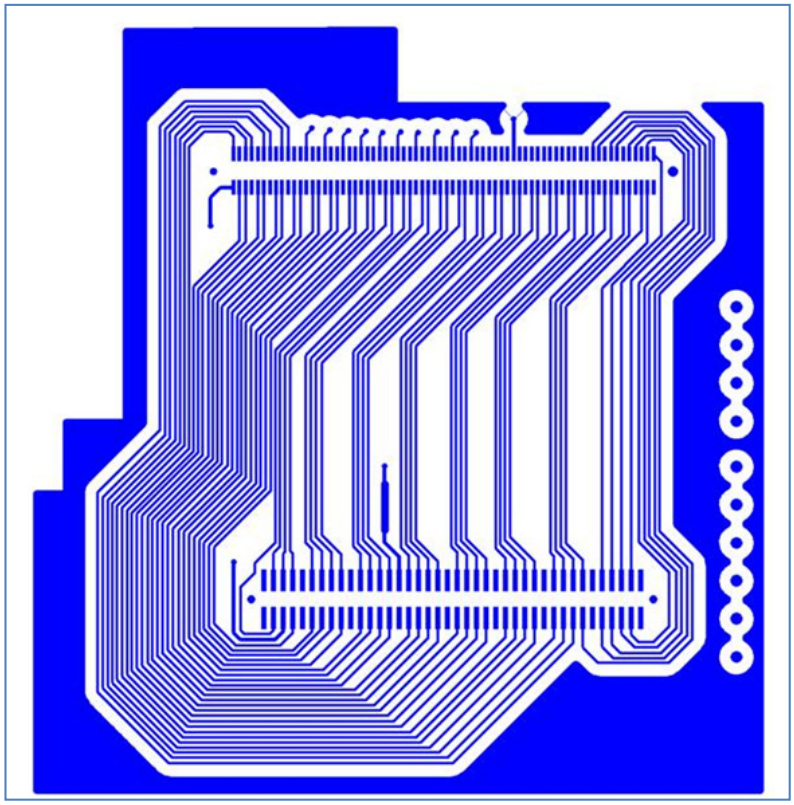


Figura 3.10. Vista de la parte superior del circuito impreso (Top) DSP-FPGA

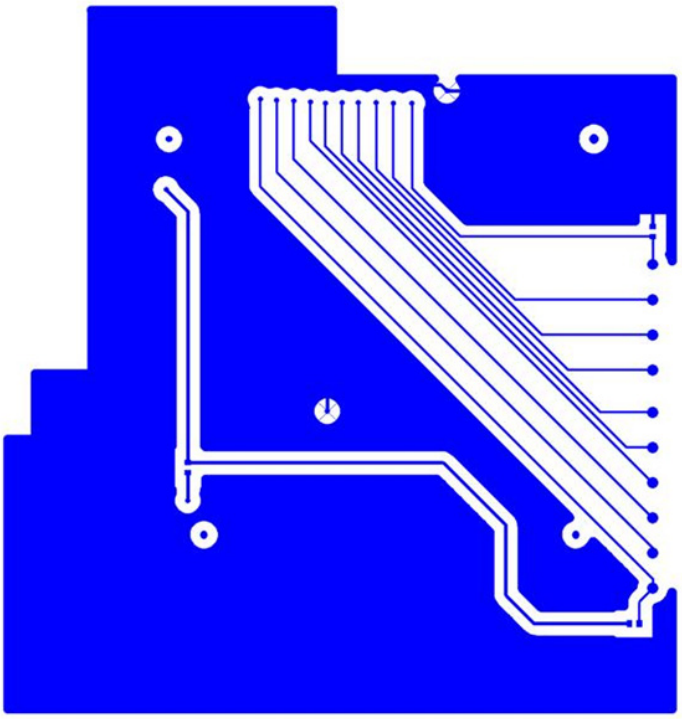


Figura 3.11. Vista de la parte de abajo del circuito impreso (bottom) DSP-FGPA

El grupo de señales de datos tienen una importancia crítica debido a la frecuencia en que serán recuperados por el DSP. Para este tipo de señales single-ended es posible realizar varios estudios, comparando la señal en cada uno de los conectores para saber cuánto tiempo de skew tienen entre cada punto. De igual forma puede realizarse un estudio de crosstalk para saber que tanto afectan las señales vecinas a una señal víctima. Estos estudios pueden proveer de datos importantes para conocer si nuestro diseño cumple con las características y tiempos que buscamos y si la implementación de una terminación puede ayudar o no en la mejora de la integridad de la señal. En las figuras 3.12 y 3.13, se muestra los resultados de las simulaciones con la herramienta HyperLynx del comportamiento de estas señales.

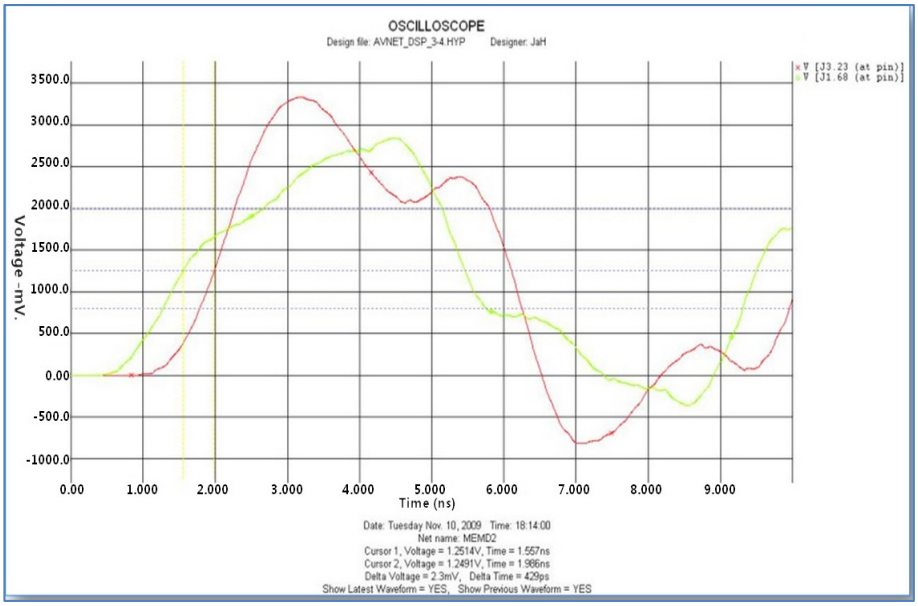


Figura 3.12. Imagen del osciloscopio para la señal MEMDATA 2 sin terminación

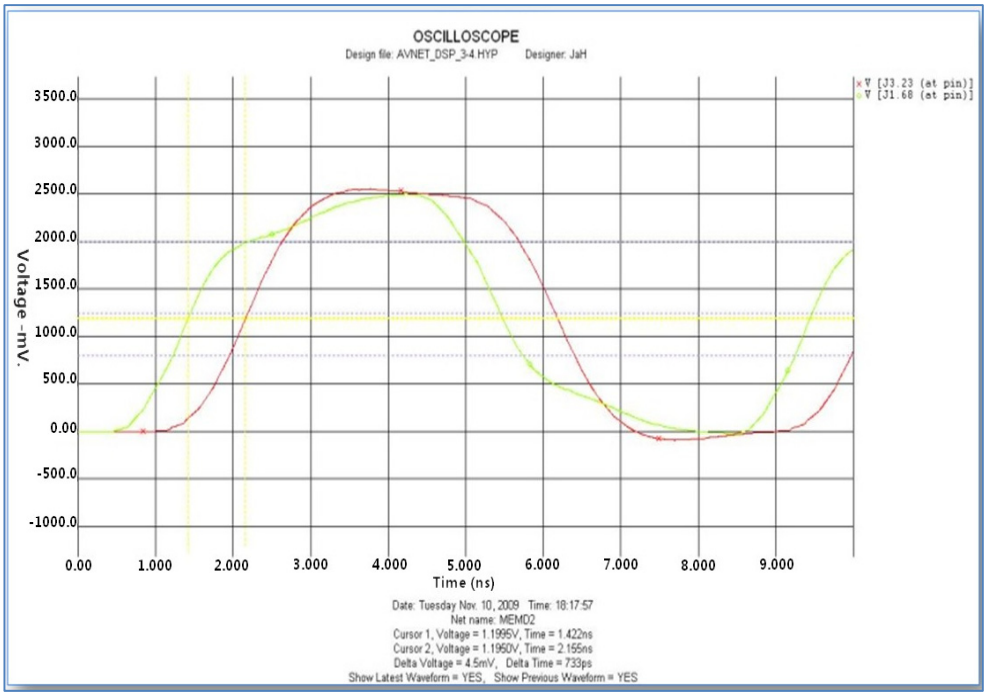


Figura 3.13.Imagen del osciloscopio para la señal MEMDATA2 con una terminación en serie de 33.3Ω

La herramienta de verificación de integridad de señal nos muestra el efecto que tendría la adición de una terminación en serie de 33.3Ω en la línea MEMDATA2 del diseño se puede apreciar comparando las figuras 3.12 y 3.13 el uso de la resistencia elimina el overshooting tanto positivo como negativo pero aumenta el skew entre las señales. En el diseño primamos la reducción de skew, el overshooting se eliminará en los buffers de entrada de datos que contiene la tarjeta de evaluación del DSP.

3.1.3 Diseño Hardware para la recuperación y almacenamiento de los datos provenientes de la cámara

La cámara SI-1280, la fuente de los datos, es una cámara de altas prestaciones con capacidad de 1.3 megapíxeles y una tasa de transferencia máxima de 50 imágenes por segundo a su máxima resolución. Cuenta entre sus características un scan progresivo, una resolución de 12 bits por píxel y una frecuencia máxima de 60 MHz. En la figura 3.14 se da el diagrama a bloques de la cámara SI-1280.

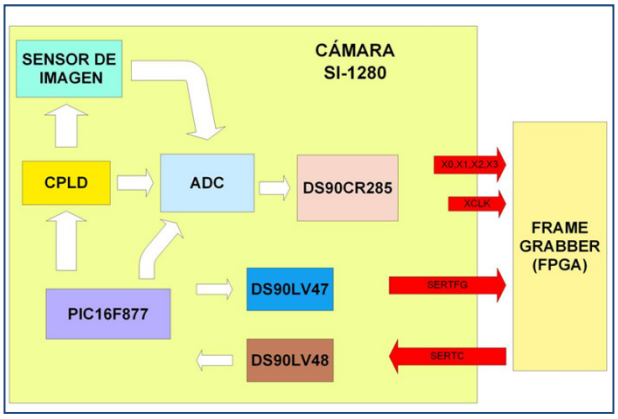


Figura 3.14.Diagrama a bloques de la cámara SI-1280

La cámara utiliza un sensor de imagen CMOS el cual maneja el estándar Bayer para la captura de imágenes, esto significa que la construcción del sensor es en base a celdas que capturan un determinado componente RGB (Red-rojo, Green-verde, Blue-azul) y en base a una combinación de estas celdas se obtiene el píxel. El sensor de la cámara tiene una proporción 1:2:1 (figura 3.15); una celda de rojo, dos celdas de verde y una celda de azul, en los píxeles. El sensor de imagen proporciona en cada ciclo de reloj principal los datos de una celda en específico, por ejemplo, en un ciclo determinado da la información de una celda azul y en el siguiente da los datos de la celda adjunta a esta celda azul. El scan se realiza de forma lineal, esto significa que el sensor envía los datos de línea en línea.

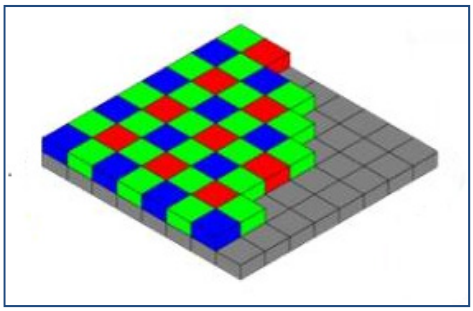


Figura 3.15. Construcción de un sensor de imagen Bayer

La resolución de cada dato producido por el sensor de imagen es de 12 bits, esto es debido a la utilización del convertidor ADS807 y es un convertidor analógico a digital de un canal con una máxima frecuencia de 53 MHz. Esta resolución nos provee un mayor rango dinámico en las imágenes.

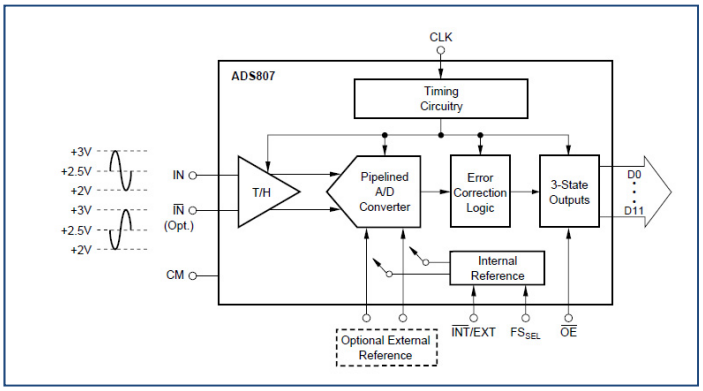


Figura 3.16. Diagrama a bloques del convertidor ADS807

El serializador DS90CR285 es el encargado de transformar los datos provenientes del convertido analógico a digital al formato CameraLink, en la figura 3.17 se muestra su diagrama. Es el encargado de hacer la serialización 7:1 en cuatro canales de datos y un canal de reloj, todo esto para conformar un enlace físico CameraLink Básico, tal como describe este estándar industrial.

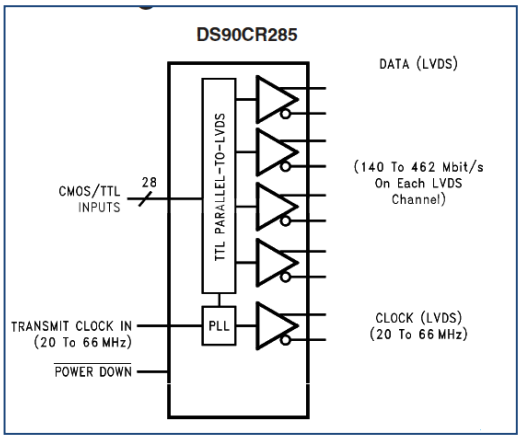


Figura 3.17. Diagrama del DS90CR285

La trama de datos entregados por este integrado tienen las características propias del estándar CameraLink, por cada ciclo de reloj envía 7 datos serializados en cada canal, como se muestra en la figura 3.18, obteniéndose una máxima transferencia de datos de 420 Mbits/s a una máxima frecuencia de 60 MHz, esta transferencia está limitada por el sensor de imagen y no por el integrado DS90CR285, debido a que este último puede llegar a una frecuencia máxima de 66 MHz. Los datos producidos por el integrado DS90CR285 se encuentran en una forma específica, y acorde con el estándar CameraLink 1.0.

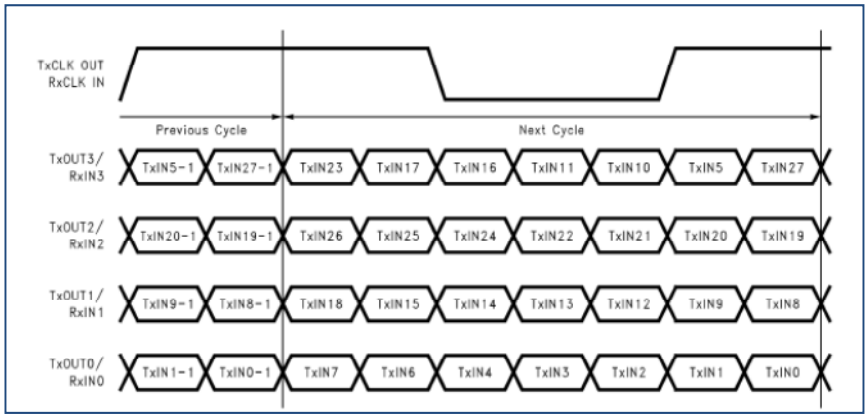


Figura 3.18. Trama de datos del DS90CR285

De la figura 3.18 se aprecia que los datos no están en fase con el ciclo de reloj principal, sino que un nuevo paquete se inicia casi a medio ciclo de trabajo. Igualmente se observa que los datos no se encuentran ordenados en forma correlativa, por ejemplo que en el TxOut0 se envíen los bits 0 al 6, en el TxOut1 se envíen los bits 7 al 13 y así sucesivamente. La correcta deserialización y el ordenamiento de los datos serán trabajo del frame grabber y sus receptores. Otro aspecto importante a recalcar es que, debido a la utilización del estándar CameraLink, a los datos provenientes del sensor de imagen debemos de anexarle cuatro datos más, los cuales son las señales de sincronía de la imágenes, estas son producidas por un CPLD, que trabaja en conjunto tanto con el sensor de imagen como con el ADC y con el micro controlador de la

cámara, para proveernos una correcta estructura de datos que permita la reconstrucción de las imágenes.

El estándar genérico del sistema CameraLink Básico define tres grupos de datos de 8 bits y cuatro señales de control dentro del conjunto de 28 bits. Debido a que la cámara utiliza una resolución de 12 bits el fabricante cambio la disposición de los datos para compensar este hecho; se tiene dividido en dos grupos de 12 bits y cuatro señales de control. El sensor de imagen es un sensor Single Tap, significa que por cada ciclo provee los datos de un solo píxel, pero debido a que existen frames grabbers que realizan la lectura en el segundo tap el fabricante dispuso la repetición del tap en cada ciclo, es decir, en cada grupo de datos se encuentran dos veces el mismo píxel, por lo que los puertos DE y DO contienen los mismo datos en cada ciclo de reloj, como se muestra en la tabla 3.2.

Pin del DSCR285	Asignación CameraLink	Asignación SI-1280
TxIn0	Port A0	DO-0
TxIn1	Port A1	DO-1
TxIn2	Port A2	DO-2
TxIn3	Port A3	DO-3
TxIn4	Port A4	DO-4
TxIn5	Port A7	DO-7
TxIn6	Port A5	DO-5
TxIn7	Port B0	DO-8
TxIn8	Port B1	DO-9
TxIn9	Port B2	DO-10
TxIn10	Port B6	DE-10
TxIn11	Port B7	DE-11
TxIn12	Port B3	DO-11
TxIn13	Port B4	DE-9
TxIn14	Port B5	DE-8
TxIn15	Port C0	DE-0
TxIn16	Port C6	DE-6
TxIn17	Port C7	DE-7
TxIn18	Port C1	DE-1
TxIn19	Port C2	DE-2
TxIn20	Port C3	DE-3
TxIn21	Port C4	DE-4
TxIn22	Port C5	DE-5
TxIn23	Spare	Spare
TxIn24	LVAL	LVAL
TxIn25	FVAL	FVAL
TxIn26	DVAL	DVAL
TxIn27	Port A6	DO-6
TxClk In	Strobe	RxClk

Tabla 3.2. Asignación de señales

Para la creación de las señales de control serial que define el estándar CameraLink, la cámara utiliza un microcontrolador PIC16F777, además de realizar esta función da una variedad de señales de control para el CPLD, el sensor de imagen y el ADC. La conversión de señales del microcontrolador a LVDS es realizada por dos drivers diferentes, uno para la recepción de datos (señal SERTC) para ello se utiliza el integrado DS90LV48 y otra para el envío de las respuestas del sistema (señal SERTFG) utilizando el integrado DS90LV47. El sistema de control serial maneja dos velocidades, la señal definida en el estándar CameraLink la cual es de 9600 bauds y otra de velocidad superior a 57kbs.

El sistema de adquisición de datos se puede desintegrar en cinco elementos: enlace físico (drivers de entrada y salida), módulo de reloj y señales de control, receptores, módulo de acomodo de datos almacenamiento y módulo de comunicación serial. En la figura 3.19 se muestra el diagrama a bloques del sistema implementado.

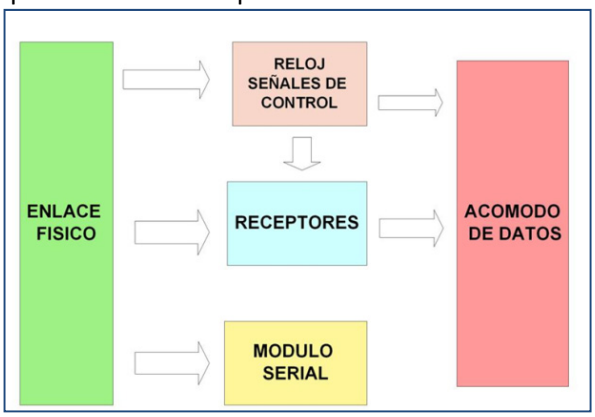


Figura 3.19. Diagrama a bloques del Sistema implementado en un FPGA

3.1.3.1. Enlace físico

Las señales provenientes de la cámara tienen el estándar LVDS y de esa forma no pueden ser procesadas internamente por el FPGA. La solución de este problema es implementar un driver que permita convertir estas señales LVDS en señales single-ended y viceversa.

Los FPGA´s Xilinx cuentan entre sus múltiples propiedades la capacidad de trabajar con diferentes estándares de señalización gracias a una serie de drivers E/S. En este caso el driver de entrada utilizado es el llamado IBUFGDS este es un buffer diferencial que transforma ciertos estándares diferenciales en una señalización single-ended. Además cuenta con la opción de creación de la resistencia diferencial si es que no se cuenta con esta en la entrada del FPGA o, aunque se tenga esa resistencia fuera del FPGA todavía se tiene demasiado ruido acoplado en la señal. Para el driver de salida la mejor opción es el driver OBUFGDS, un driver que permite un cambio de señalización single-ended hacia un estándar diferencial. Para ambos casos el estándar utilizado es el LVDS_25, debido a que la tarjeta de evaluación utilizada contempla solamente el uso de este nivel de LVDS. La señal XCLK, es el reloj principal proveniente de la cámara, recibe un trato especial ya que tras la transformación de señalización se conecta a un buffer BUFG cuya función es mejorar la distribución del mismo. La implementación de estos drivers está representada en la figura 3.20.

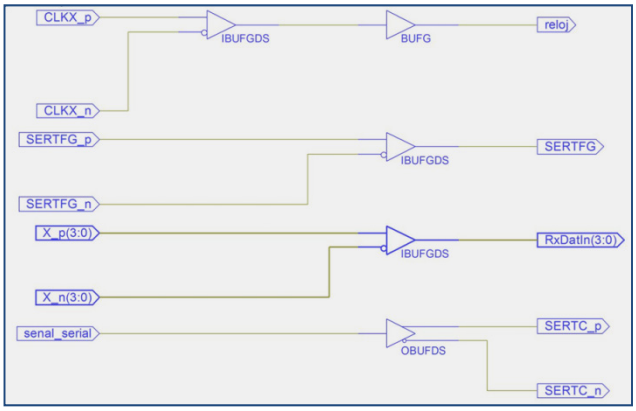


Figura 3.20. Señales de entrada y salida LVDS del FPGA

En los FPGAs de Altera utilizo la herramienta de programación Quartus II para la implementación del driver que permite convertir las señales LVDS en señales single-ended y viceversa. Quartus II permite realizar esta labor de forma gráfica mediante la asignación de pines como se muestra en la figura 3.21. En el código se declara el pin positivo del par y en el apartado de asignación de pines se declara este pin con el estándar LVDS y el pin negativo se implementa de forma automática.

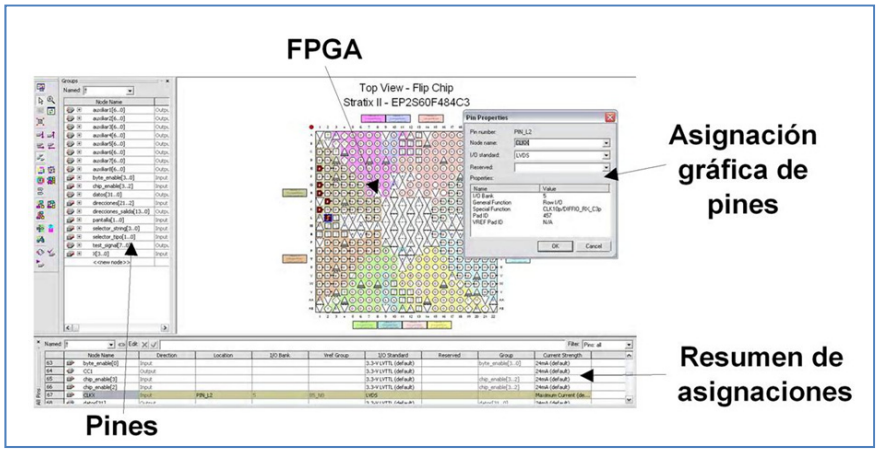


Figura 3.21. Interfaz grafica para la asignación de pines de la herramienta Quartus II

La herramienta de programación que suministra la fabricante Altera para la programación de los Stratix da una cierta ventaja debido a que simplifica el código al eliminar completamente los drivers de E/S, tanto manejando el estándar LVDS como otros diferentes tipos de estándares. También permite la adecuada configuración de las E/S, como la implementación de resistencias terminales y el establecimiento de una corriente fija.

3.1.3.2. Módulo de reloj y señales de control

El sistema para su correcto funcionamiento requiere de un grupo de señales de reloj y de control. El interfaz CameraLink provee un reloj base y es a través de éste que se crean las señales necesarias. Para la deserialización de los datos provenientes de la cámara se necesitan tres señales primordiales: una señal de reloj con un factor de multiplicación 3.5 con respecto al reloj base, una señal de activación de carga en paralelo y una señal de control de los multiplexores de los receptores. Además de estas señales para la adquisición de datos se requiere una señal extra para el almacenamiento de estos datos en una memoria.

La lógica del FPGA de Xilinx se puede dividir en dos partes: módulo sintetizador y módulo de creación de señales de control.

El módulo sintetizador es básicamente un DCM (digital clock manager) configurado de forma que multiplique el reloj base proveniente de la cámara. El factor de multiplicación es de 3.5 (m = 7, d = 2) por la serialización 7 : 1 que marca el estándar Cameralink, significa que por cada pulso de reloj base se encontrara 7 datos en cada una de las líneas de datos. En la figura 3.22 se muestra su implementación.

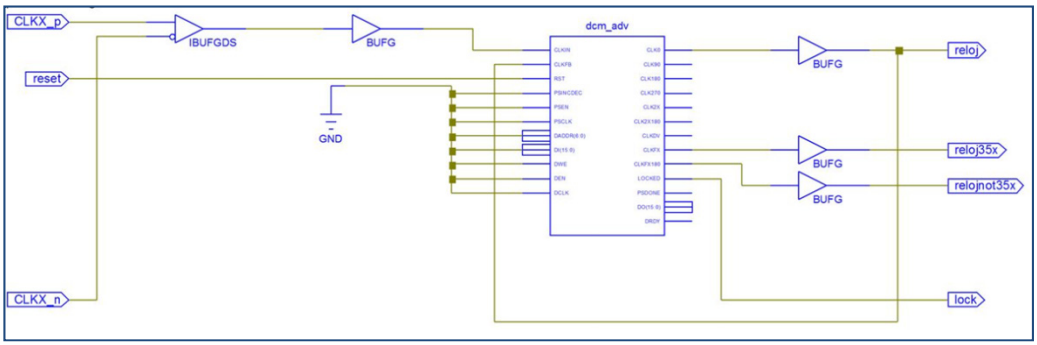


Figura 3.22. Módulo sintetizador

El módulo que genera las señales de control consta de un conjunto de compuertas y flip flops del que obtenemos dos señales; la primera es la señal de control de los multiplexores para ir acomodando los datos en los receptores ya que en medio ciclo recibe la mitad de los datos y en el otro medio ciclo el resto de los datos, y la segunda es la señal de habilitación de carga en paralelo de los mismos receptores. La primera señal se obtiene mediante un latch que va cambiando cada nuevo ciclo del reloj base. La segunda señal se obtiene mediante un arreglo de compuertas y flip flops y es válida solamente al final del reloj 3.5x, para que la carga en paralelo no se ejecute más de una vez en un ciclo.

En la figura 3.23 se muestra la máquina de estados, en la figura 3.24 el módulo de creación de las señales de control y en la figura 3.25 el diagrama de tiempos de las señales del módulo de creación de señales de control.

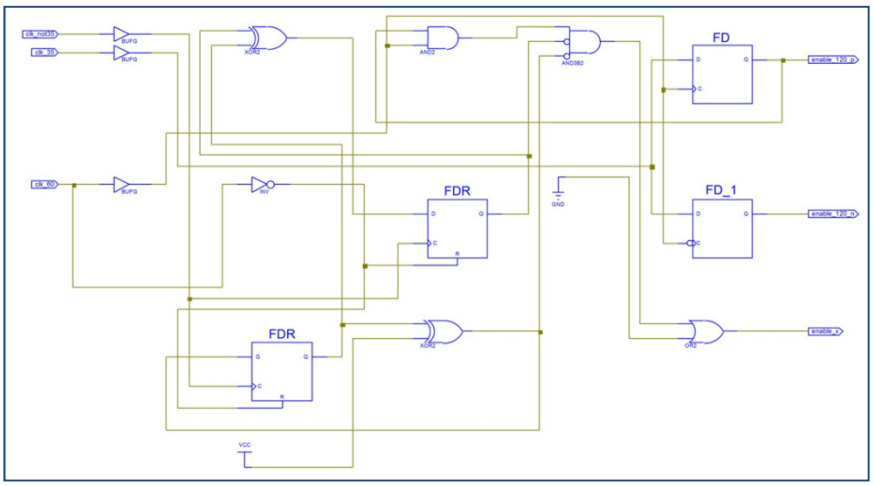


Figura 3.23. RTL de la máquina de estados

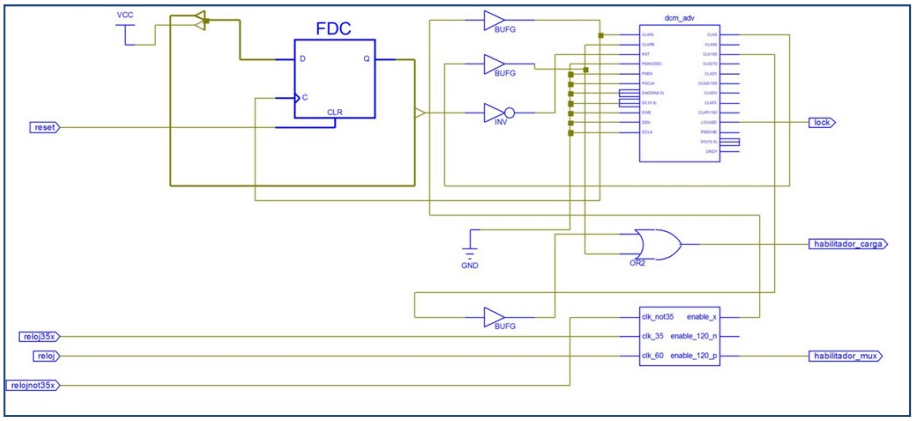


Figura 3.24. Módulo de creación de señales de control

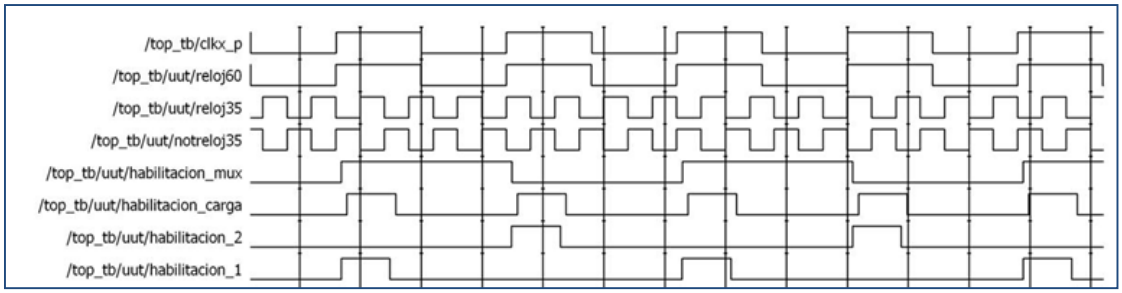


Figura 3.25. Diagrama de estados de las señales del módulo de creación de señales de control

Los FPGAs de Altera cuentan con PLL que tienen la capacidad de tener hasta 6 salidas casi totalmente independientes como se ve en la figura 3.26,por lo que la frecuencia de cada salida puede llegar a sintetizarse independientemente, adicionalmente pueden controlarse la fase y el ciclo de trabajo de las salidas. Las salidas pueden convertirse en totalmente independientes utilizando la versión no compensada en las salidas, esta provee un menor jitter pero crea un skew entre las señales de entrada y salida. Por este motivo se optó por la opción de modo normal o señal compensada que permite a las señales de salida estar en fase con la señal de reloj de entrada para que el skew existente entre estas señales sea mínimo o casi nulo. La señal compensada será la señal base para las salidas, por lo que cualquier cambio de fase en esta señal se proyectara en las demás señales de salida, por ejemplo, si le agregamos un desfase de 10˚ a nuestra señal compensada, este desfase se reflejara en cada una de las señales de salida. Los aspectos de frecuencia y ciclo de trabajo de la señal compensada no se reflejan en las demás señales de salida.

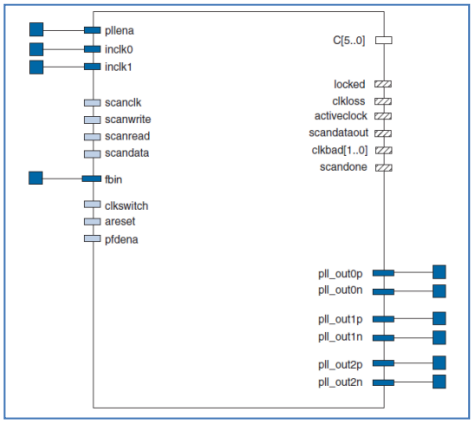


Figura 3.26. Puertos de un PLL enhanced.

El reloj con factor de multiplicación 3.5 se obtiene a través de una salida del PLL (factor 3.5; M= 7, D= 2). La señal de activación de carga en paralelo se implementa mediante el uso de dos salidas del PLL. Una salida con frecuencia igual a la señal de reloj base y un ciclo de trabajo de 30% que permite realizar la carga en un ciclo de reloj base especifico, y la otra salida de características similares con la única diferencia de un pequeño desfase que permite realizar la carga en el siguiente ciclo de reloj, ya que la deserialización en su etapa serial se realiza de forma DDR (Double Data Rate) por lo que permite realizar esta en flancos positivos como flancos negativos del reloj serial, con esto se evita manejar frecuencias mayores. Teniendo estas dos señales se procede a realizar la depuración de las mismas mediante el uso de un latch controlado por el reloj base que controla un multiplexor donde ingresamos ambas señales de control y su salida es la señal de control que permitirá realizar la carga en paralelo en todos los ciclos de reloj donde se tengan datos. La señal de salida del latch controla también a los multiplexores de los receptores. La variedad de frecuencias que maneja la cámara (de 20 MHz a 60 MHz) produce un cierto retraso en las señales de carga en paralelo. Los PLL de Altera permiten controlar este defecto mediante la opción de Clock Phase Shift que habilita el desfasamiento de la señal tanto en forma de retraso como en forma de adelanto. La última salida del PLL permite controlar la escritura de las memorias para el correcto almacenamiento de los datos deserializados. Esta implementación se muestra en la figura 3.27.

En resumen las señales de salida del PLL son las siguientes:

1.-Reloj base
2.-Reloj sintetizado 3.5x
3.-Reloj sintetizado not3.5x
4.-Habilitador 1
5.-Habilitador 2
6.-Reloj de escritura de las memorias RAM

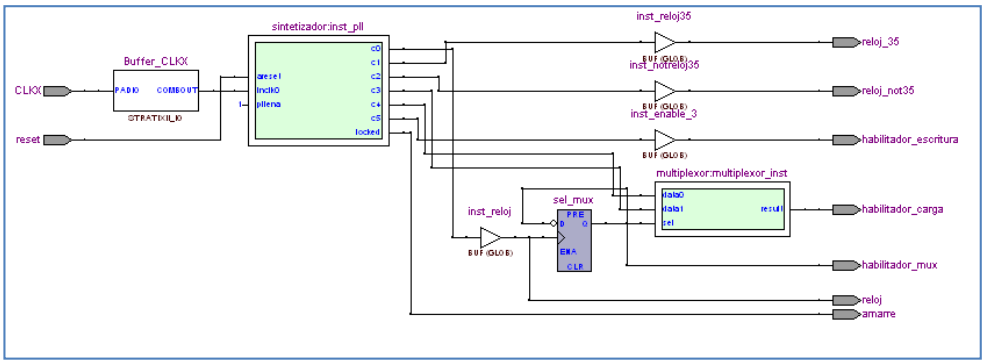


Figura 3.27. Módulo de reloj y creación de señales de control

3.1.3.3. Receptores

Teniendo las señales de reloj y de control propiamente implementadas se puede realizar la adquisición de datos. Esta adquisición es realizada por un conjunto de cuatro receptores diseñados mediante un arreglo de flip-flop y multiplexores. Este módulo se divide en dos partes debido al ingreso de datos a los flip-flops: carga serie y carga en paralelo.

En la figuras 3.28 y 3.29 se muestran las implementaciones del módulo de recepción de datos en los FPGAs de Xilinx y de Altera respectivamente.

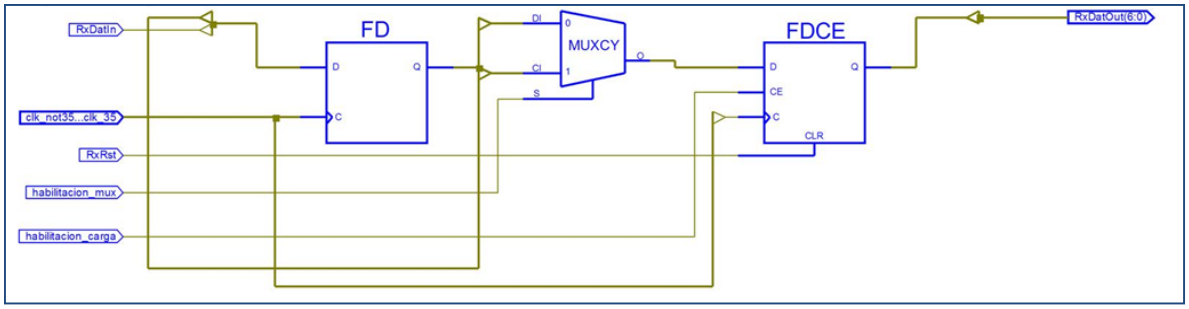


Figura 3.28. Módulo de recepción de datos

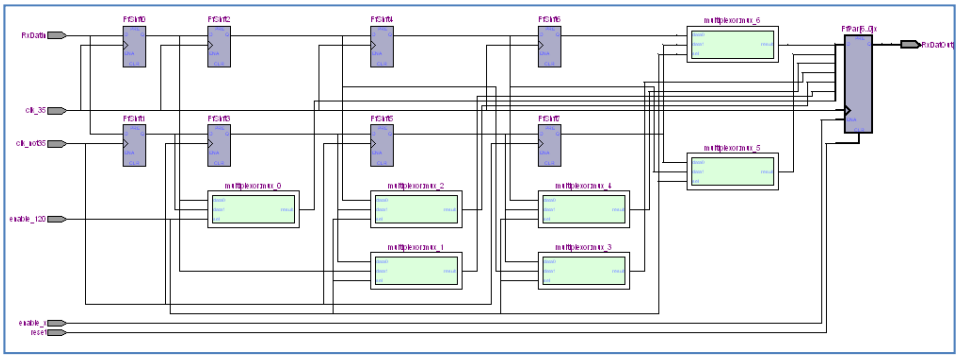


Figura 3.29. Receptor implementado en un FPGA Altera

El sistema de carga serie consta de un grupo de 8 flip-flops ordenados en grupos de cuatro, interconectados cada uno de ellos de forma serie. Cada grupo responde a un diferente cambio del reloj 3.5x, esto es que el grupo 1 se activa mediante el flanco de subida y el grupo 2 mediante el flanco de bajada. De esa manera utilizamos las ventajas del sistema DDR de manejar una menor frecuencia y como consecuencia un menor consumo del dispositivo así como una menor cantidad de errores en la adquisición.

Este arreglo de flip-flops se muestra implementado en los FPGA´s de Xilinx y Altera, figuras 3.30 y 31 respectivamente.

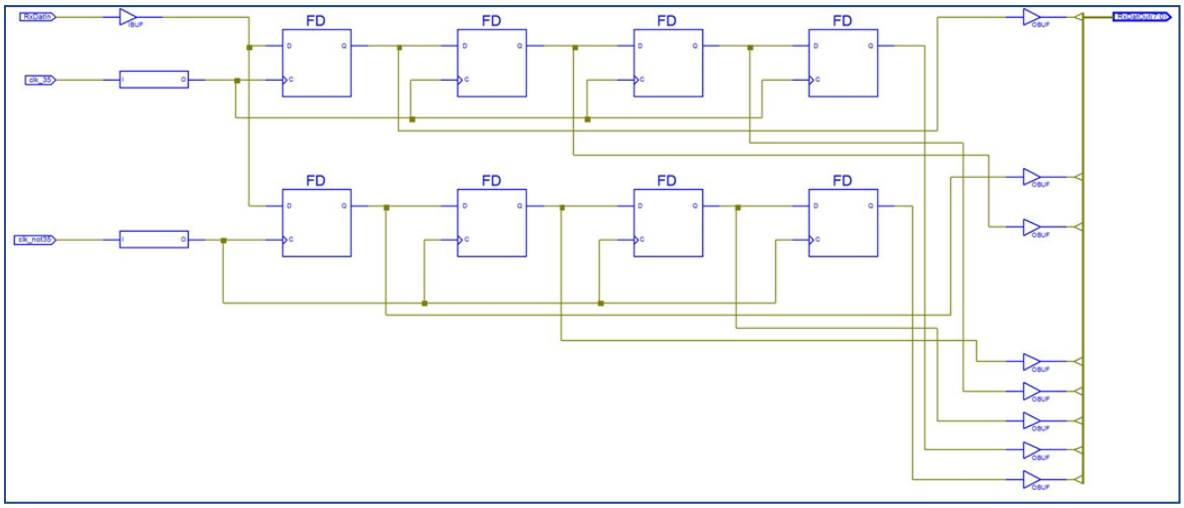


Figura 3.30. Etapa de registros serie de un receptor.

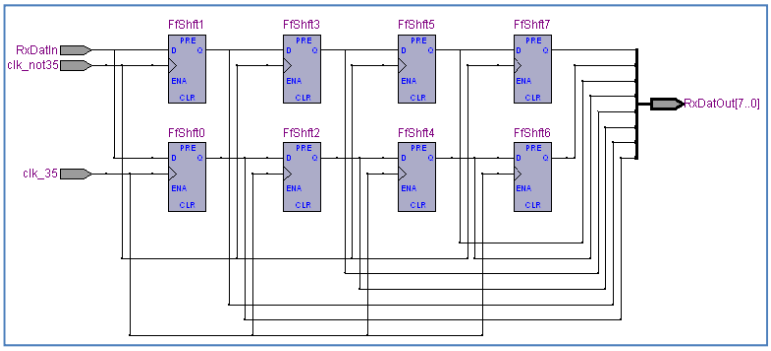


Figura 3.31. Etapa de registros serie implementados en un FPGA Altera

Los datos provenientes de la cámara son "almacenados" en los flip-flops a cada cambio de flanco, es decir, iniciando con flanco de subida, el primer dato quedara en el primer flip-flop (primero del grupo 1), llegado el flanco de bajada el segundo dato es almacenado en el segundo flip-flop (primero del grupo 2), el flanco de subida siguiente produce un desplazamiento del primer dato hacia el tercer flip-flop (segundo del grupo 1) y la llegada del tercer dato en el primer flip-flop (primero del grupo 1), a la llegada del siguiente flanco de bajada se produce el desplazamiento del segundo dato hacia el cuarto flip-flop (segundo del grupo 2) y el almacenamiento del cuarto dato en el segundo flip-flop (primero del grupo 2) y así sucesivamente hasta llegar al último flip-flop. Al momento de llenarse los flip-flops con los datos, estos son conectados en el peso binario que le corresponde para ser recuperados en la parte de carga en paralelo. Las implementaciones de la etapa de carga en paralelo en los FPGA's se muestra en la figura 3.32 para Xilinx y en la figura 3.33 para Altera.

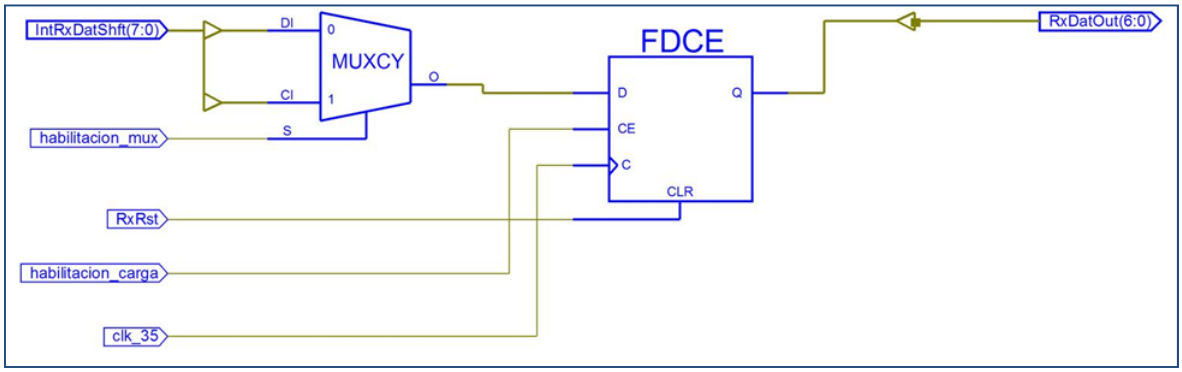


Figura 3.32. Etapa de carga en paralelo del un receptor

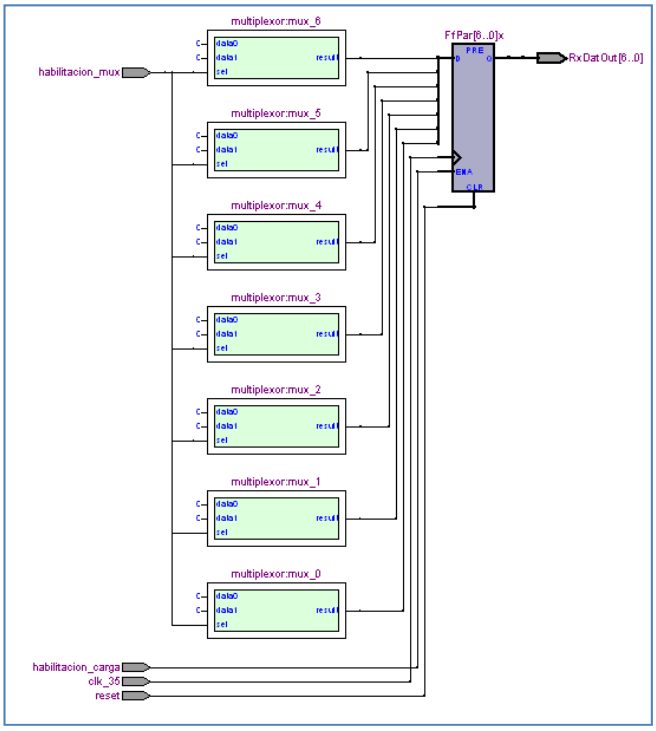


Figura 3.33. Etapa de carga en paralelo implementado en un FPGA Altera

El sistema de carga en paralelo es un conjunto de multiplexores y flip-flops que producen la palabra de 7 bits correspondiente a la línea de datos ingresada. El estándar cameralink además de especificar una serialización de 7:1 específica la estructura de los datos que se deben de enviar, esta estructura de datos describe un desfase de dos datos con respecto al flanco de subida del reloj base para el inicio de una nueva palabra, el inicio de cada palabra se produce al final del ciclo de trabajo del reloj y no al inicio, el flanco de subida es el encargado deponer en fase otra vez a los datos. Para la correcta adquisición de datos necesitamos emplear un sistema que contemple este detalle, por ello se utilizan los multiplexores. Estos multiplexores están conectados de forma específica con los flip-flops de carga serial los cuales están de forma constante recibiendo los datos de la línea. El pin selector de los multiplexores está conectado con la señal de habilitación de multiplexores producida por el módulo de señales de control, esto permite una nueva configuración en cada ciclo de reloj base. Las salidas de datos de estos multiplexores están conectadas con los flip-flops de carga en paralelo. Estos flip-flops se encargan de convertir los datos serie en un conjunto de 7 bits. Los flip-flops están conectados con la señal de reloj 3.5x producida por el sintetizador y para su habilitación cuentan con la señal de habilitación de carga paralela producida por el módulo de señales de control.

3.1.3.3.1. Implementación Xilinx de la etapa de los Receptores

La implementación del sistema se realiza en un FPGA Virtex4 LX60 FF668 -10, integrado en una placa base de evaluación Avnet (figura 3.34) y cuyas características están resumidas en la tabla 3.3. Este hecho limita la designación de pines debido que se deben de utilizar los pines previamente especificados por la fabricante Avnet.

Device	Configurable Logic Blocks (CLBs)(1)				XtremeDSP Slices(2)	Block RAM	
	Array(3) Row x Col	Logic Cells	Slices	Max Distributed RAM (Kb)		18 Kb Blocks	Max Block RAM (Kb)
XC4VLX60	128 x 52	59,904	26,624	416	64	160	2,880

DCMs	PMCDs	PowerPC Processor Blocks	Ethernet MACs	RocketIO Transceiver Blocks	Total I/O Banks	Max User I/O
8	4	N/A	N/A	N/A	13	640

Tabla 3.3 – características principales del FPGA XC4VLX60

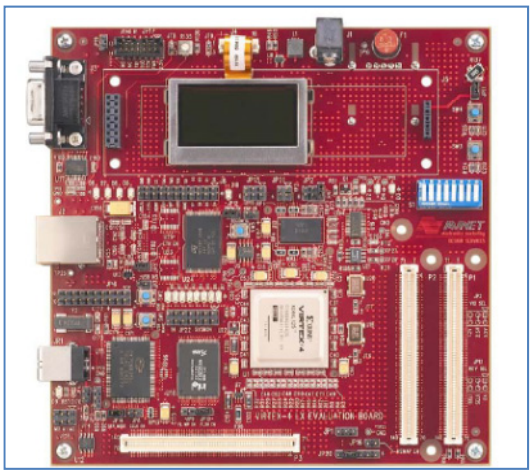


Figura 3.34.Tarjeta de evaluación Avnet LX60

Implementación sin emplazamiento manual pero con elementos de eliminación de skew de los Receptores

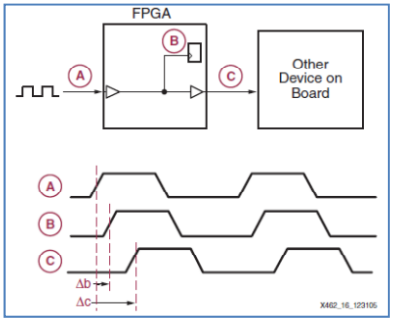


Figura 3.35. Ejemplo del Skew en un sistema síncrono

El atributo CLKOUT_PHASE_SHIFT permite elegir, entre 5 opciones posibles, qué tipo de desfase se aplicará en las salidas del DCM. El atributo PHASE_SHIFT describe la cantidad específica de desfase que se aplicará en las salidas, este atributo está conectado de forma directa con el atributo CLKOUT_PHASE_SHIFT debido a que dependiendo de la opción elegida en él se tiene un mínimo y un máximo en el atributo PHASE_SHIFT.

El sistema implementado consigue la recuperación de datos a una frecuencia máxima de 7.14 MHz. Esta frecuencia es menor que la mínima frecuencia de trabajo de la cámara SI-1280, por lo cual se requiere una depuración del sistema para lograr una frecuencia dentro del rango de trabajo de la SI-1280. Los recursos utilizados se muestran en la tabla 3.4.

Device Utilization Summary				[-]
Logic Utilization	**Used**	**Available**	**Utilization**	**Note(s)**
Total Number Slice Registers	243	53,248	1%	
Number used as Flip Flops	228			
Number used as Latches	15			
Number of 4 input LUTs	504	53,248	1%	
Number of occupied Slices	363	26,624	1%	
Number of Slices containing only related logic	363	363	100%	
Number of Slices containing unrelated logic	0	363	0%	
Total Number of 4 input LUTs	561	53,248	1%	
Number used as logic	504			
Number used as a route-thru	57			
Number of bonded IOBs	110	448	24%	
IOB Flip Flops	1			
IOB Master Pads	2			
IOB Slave Pads	2			
Number of BUFG/BUFGCTRLs	15	32	46%	
Number used as BUFGs	15			
Number of FIFO16/RAMB16s	32	160	20%	
Number used as RAMB16s	32			
Number of DCM_ADVs	2	8	25%	
Number of RPM macros	1			
Average Fanout of Non-Clock Nets	5.20			

Tabla 3.4. Recursos utilizados para el sistema sin emplazamiento y con herramientas de deskew

Se puede apreciar, a través de la figura 3.36, que el algoritmo de implementación de la herramienta de programación ISE Design Suite realiza una implementación aleatoria y sin ningún tipo de orden, por ese motivo es que la frecuencia máxima de recuperación es tan baja.

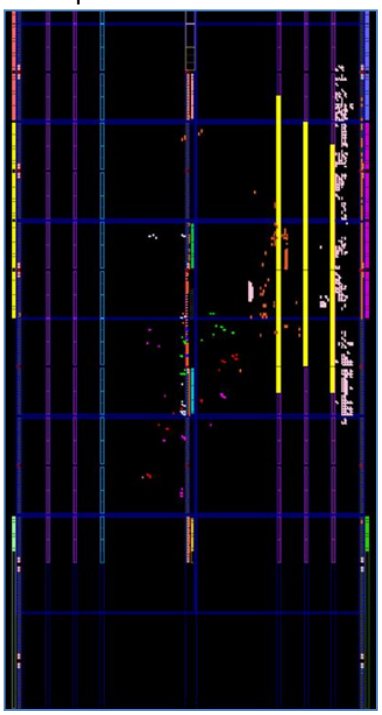


Figura 3.36. Floor pannel para el sistema sin emplazamiento y con herramientas de deskew

Implementación con emplazamiento manual de los Receptores

El fabricante Xilinx, a través de la herramienta de programación ISE Design Suite, provee de una aplicación llamada PlanAhead [Xilinx, 2007]que nos permite la ubicación específica de componentes de forma gráfica y sencilla. A través de esta herramienta podemos realizar la ubicación manual de cualquier elemento del sistema, además de poder integrar restricciones de tiempo y algunas otras especificaciones que ayuden en el mejoramiento global del sistema. Otra ayuda es la aplicación de atributos dentro del código o dentro de nuestro UCF. Atributos como RLOC, RLOC_ORIGIN o AREA_GROUP, entre otros, son esenciales en la aplicación de la técnica de emplazamiento manual.

Atributo RLOC [Xilinx, 2009].

Realiza el agrupamiento de componentes lógicos en conjuntos. Mediante este atributo se puede establecer cualquier componente lógico en un cierto lugar con respecto a otros componentes. Por ejemplo al establecer un conjunto de 4 flip-flops en un formato de columna, la herramienta de implementación mantendrá esta columna y moverá los flip-flops como si fuera un solo elemento. Al establecer una relación específica de los componentes del módulo receptor se asegura la igualdad en las distancias entre componentes así como la posible manipulación de los mismos en beneficio del sistema.

Atributo RLOC_ORIGIN [Xilinx, 2009].

Este atributo describe el origen físico de cualquier instancia. A través de este atributo es posible establecer un origen dentro de nuestro FPGA pero no un rango o límites de un componente. Por ejemplo se puede establecer que el componente “prueba” se establezca a partir del slice “X10Y10” por lo que nuestro componente da inicio en esta localidad física pero podrá abarcar el resto del FPGA.

Atributo AREA_GROUP [Xilinx, 2009].

Este atributo describe un rango físico donde alguna instancia será implementada. Por ejemplo, se tiene la instancia “prueba” y se describe un AREA_GROUP con límites en “X0Y0” y “X10Y10”, la herramienta de implementación buscara realizar el emplazamiento de la instancia “prueba” únicamente dentro de este rango. Con este atributo podemos especificar un área específica donde deseamos implementar cualquier componente. Este atributo puede ser implementado mediante la herramienta PlanAhead donde al realizar la implementación de los llamados PBlocks podemos crear el AREA_GROUP.

Atributo BEL [Xilinx, 2009].

Atributo de emplazamiento avanzado. Mediante este atributo podemos establecer símbolos lógicos (ILOGIC’S, OLOGIC’S, IO’S, etc.) en un elemento básico especifico (BEL = Basic ELement). Difiere de atributo RLOC en que el atributo RLOC funciona en un nivel de componente y este en un nivel de elemento básico.

Implementación

A través de las diferentes opciones que se tiene para la depuración del sistema se logró implementar un diseño que cubra el rango de frecuencias de trabajo de la cámara SI-1280 (20 MHz a 60 MHz) y además que pudiera llegar a frecuencias mayores para un posible enlace con una cámara más avanzada. La frecuencia máxima lograda es de 85 MHz. En la figura 3.37 se

muestra el Floor pannel del sistema con emplazamiento manual y con las herramientas de deskew. En la tabla 3.5 se muestra los recursos utilizados para esta implementación.

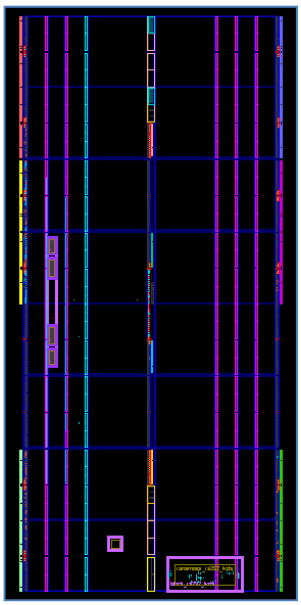


Figura 3.37. Floor pannel del sistema con emplazamiento manual y herramientas de deskew

Device Utilization Summary				
Logic Utilization	**Used**	**Available**	**Utilization**	**Note(s)**
Total Number Slice Registers	244	53,248	1%	
Number used as Flip Flops	229			
Number used as Latches	15			
Number of 4 input LUTs	505	53,248	1%	
Number of occupied Slices	365	26,624	1%	
Number of Slices containing only related logic	365	365	100%	
Number of Slices containing unrelated logic	0	365	0%	
Total Number of 4 input LUTs	562	53,248	1%	
Number used as logic	505			
Number used as a route-thru	57			
Number of bonded IOBs	130	448	29%	
IOB Flip Flops	2			
IOB Master Pads	2			
IOB Slave Pads	2			
Number of BUFG/BUFGCTRLs	16	32	50%	
Number used as BUFGs	16			
Number of FIFO16/RAMB16s	32	160	20%	
Number used as RAMB16s	32			
Number of DCM_ADVs	2	8	25%	
Number of RPM macros	5			
Average Fanout of Non-Clock Nets	5.16			

Tabla 3.5. Recursos utilizados para el sistema con emplazamiento manual y herramientas de deskew

3.1.3.3.2. Implementación en Altera de la etapa de los Receptores

El sistema se implementa en un FPGA Stratix II EPE2S60F484C3 el cual está dentro del rango de recursos del FPGA Virtex 4 LX60 de la implementación Xilinx. De forma similar a la implementación Xilinx, esta implementación en Altera se realiza con elementos de eliminación de skew en dos etapas: una etapa sin emplazamiento manual y otra etapa con emplazamiento manual. En la tabla 3.6 se muestran las características principales del FPGA.

Feature	EP2S60
ALMs	24,176
Adaptive look-up tables (ALUTs) *(1)*	48,352
Equivalent LEs *(2)*	60,440
M512 RAM blocks	329
M4K RAM blocks	255
M-RAM blocks	2
Total RAM bits	2,544,192
DSP blocks	36
18-bit × 18-bit multipliers *(3)*	144
Enhanced PLLs	4
Fast PLLs	8
Maximum user I/O pins	718

Tabla 3.6. Características principales del FPGA EPE2S60F484C3

Implementación con elementos de eliminación de skew y sin emplazamiento manual de los Receptores

La eliminación del skew en los sistemas implementados en FPGAs Altera es posible debido a una característica de los PLL´s, denominada PhaseShift (ver figura 3.38). A través de este atributo podemos describir la cantidad de desfase que queremos en nuestras salidas y con ello eliminar el skew existente en las líneas.

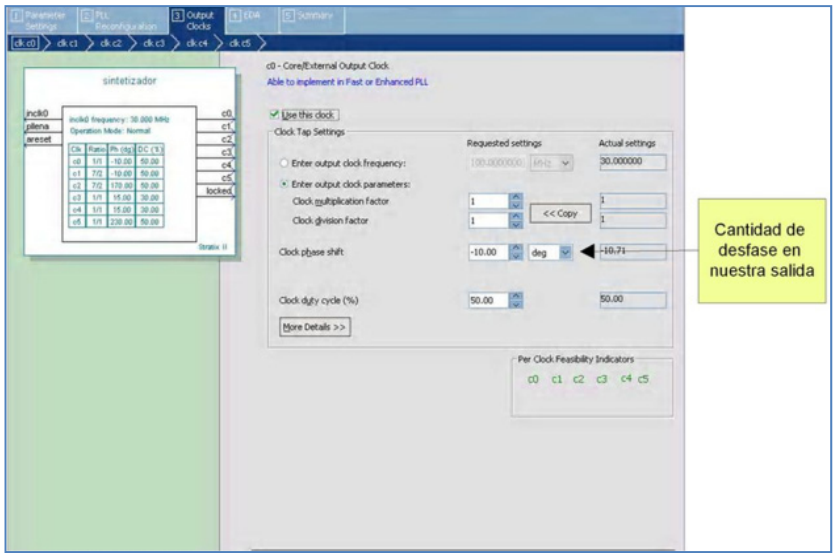


Figura3.38. Mega wizard plug-in manager ALTPLL

El sistema implementado logró una frecuencia máxima de recuperación de datos de 30.86 MHz, ubicada dentro del rango de trabajo de la cámara SI-1280, aunque no llega al máximo

necesario de 60 MHz. En la tabla 3.7 se muestra los recursos utilizados en esta implementación y en la figura 3.39 se muestra el Floor pannel de la implementación de este sistema.

Flow Status	Successful - Mon Nov 23 09:36:31 2009
Quartus II Version	8.0 Build 215 05/29/2008 SJ Full Version
Revision Name	top
Top-level Entity Name	top
Family	Stratix II
Device	EP2S60F484C3
Timing Models	Final
Met timing requirements	No
Logic utilization	1 %
Combinational ALUTs	541 / 48,352 (1 %)
Dedicated logic registers	212 / 48,352 (< 1 %)
Total registers	212
Total pins	171 / 335 (51 %)
Total virtual pins	0
Total block memory bits	524,288 / 2,544,192 (21 %)
DSP block 9-bit elements	0 / 288 (0 %)
Total PLLs	1 / 6 (17 %)
Total DLLs	0 / 2 (0 %)

Tabla 3.7. Recursos utilizados para el sistema altera sin emplazamiento y con herramientas de deskew

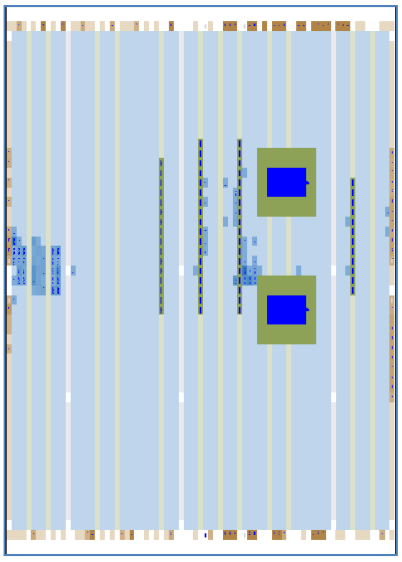


Figura 3.39. Floor pannel del sistema altera sin emplazamiento manual y con herramientas de deskew

Implementación con emplazamiento manual de los Receptores

El emplazamiento manual dentro de la herramienta Quartus II de Altera puede ser implementado de varias formas. Una de ellas es mediante el uso de Custom Regions a través del Assigment Editor. Mediante la especificación de un área concreta dentro del FPGA podemos enlazarla con cualquier instancia que se desee, y con ello poder localizar a los componentes en un lugar específico del FPGA. Dentro del mismo apartado del Assigment Editor podemos emplazar tanto componentes lógicos como celdas básicas (ver figura 3.40).

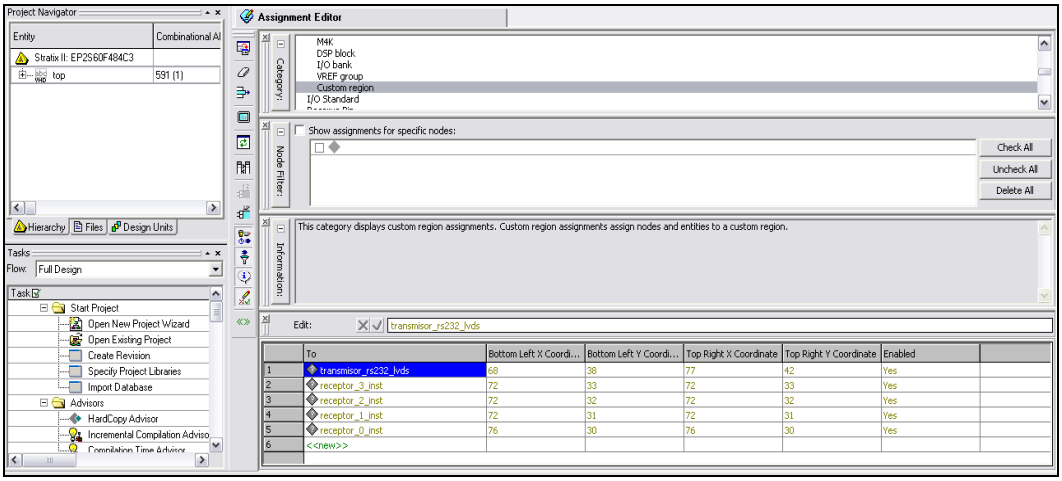


Figura 3.40. Assignment Editor de la herramienta Quartus II

La implementación con emplazamiento manual obtuvo un incremento en la frecuencia de recuperación de datos, alcanzando los 70.32 MHz. En la figura 3.41 se muestra el Floor panel del sistema implementado en Altera con emplazamiento manual y con la herramienta de deskew. En la tabla 3.8 se muestra los recursos utilizados para la implementación del sistema.

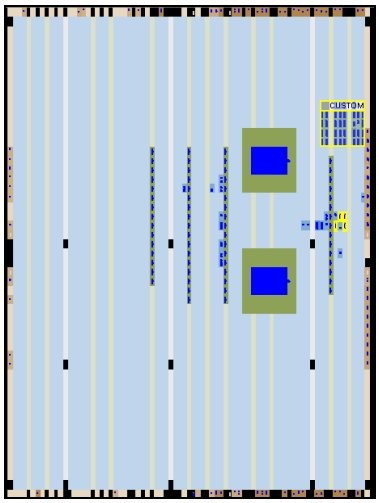


Figura 3.41. Floor pannel del sistema implementado en Altera con emplazamiento manual y con herramientas de deskew

Flow Status	Successful - Mon Nov 23 11:10:48 2009
Quartus II Version	8.0 Build 215 05/29/2008 SJ Full Version
Revision Name	top
Top-level Entity Name	top
Family	Stratix II
Device	EP2S60F484C3
Timing Models	Final
Met timing requirements	No
Logic utilization	1 %
Combinational ALUTs	533 / 48,352 (1 %)
Dedicated logic registers	213 / 48,352 (< 1 %)
Total registers	213
Total pins	171 / 335 (51 %)
Total virtual pins	0
Total block memory bits	524,288 / 2,544,192 (21 %)
DSP block 9-bit elements	0 / 288 (0 %)
Total PLLs	1 / 6 (17 %)
Total DLLs	0 / 2 (0 %)

Tabla 3.8 Recursos utilizados para el sistema altera con emplazamiento y con herramientas de deskew

3.1.3.4. Módulo de acomodo de datos

El sistema Cameralink provee además de un conjunto de bits para los datos, un conjunto de bits para las señales de control que permiten reconocer los grupos de datos validos así como la composición de las imágenes en general. Los receptores implementados sirven para la adquisición correcta de los 7 bits de cada transmisor, pero debido a la estructura de la interfaz cameralink se requiere un acomodo de datos que nos permita separar los datos de control (LVAL, FVAL, DVAL, SPARE) de los datos del píxel. Con un simple concatenamiento obtenemos los 12 bits que nos describen el píxel del ciclo; la separación de los datos de control se obtiene mediante una asignación de señales con los nombres específicos de cada dato.

Este módulo también permite el correcto almacenamiento de los datos válidos. Para el almacenamiento se utilizan memorias RAM de 8 bits y doble puerto.

La estructura de almacenamiento está diseñada para utilizar datos dentro del FPGA en un posible procesamiento y ser manipulada por un dispositivo externo, en este caso un DSP, mediante el EMIF del mismo. El sistema es un conjunto de cuatro memorias RAM, con las características anteriormente descritas, que mediante un arreglo lógico almacenan los píxeles. Estas utilizan las señales de habilitación de los receptores como relojes de escritura, así como las señales de control del interfaz cameralink en conjunto con las señales de habilitación del sistema completo para la habilitación de las memorias. En la figura 3.42 se muestra un diagrama de bloques del módulo implementado para cualquier FPGA.

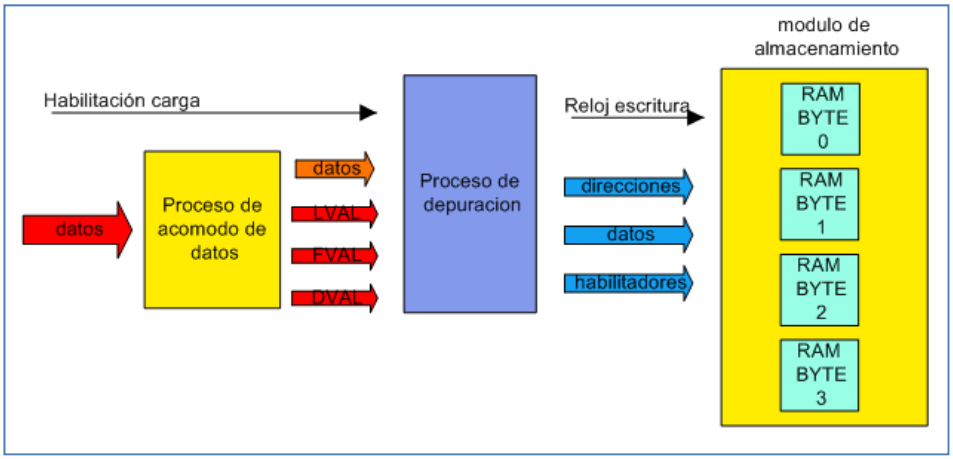


Figura 3.42. Módulo de acomodamiento y almacenamiento de datos

Conexión con el EMIF

Las memorias de almacenamiento al ser de doble puerto pueden manejar un proceso de escritura en un puerto al mismo tiempo que un proceso de lectura en el otro, permitiendo acceder a la información de la imagen al momento de almacenarla sin tener que esperar que el proceso de escritura se detenga. Para una correcta conexión con el EMIF, el cual es de 32 bits, las memorias concatenan sus salidas en una señal de 32 bits la cual se conecta directamente a la señal de datos del EMIF.La señal de AARE (asyncreadenable) da la pauta de la lectura de datos. Las señales de habilitación AAOE (async output enable) y de byte enable dan la pauta para relacionar cada memoria con cada byte del EMIF habilitado para lectura. Con estas conexiones se puede habilitar una comunicación efectiva entre cualquier FPGA y el DSP. En la figura 3.43 se muestra la interconexión de las memorias con puerto EMIF así como las señales de control.

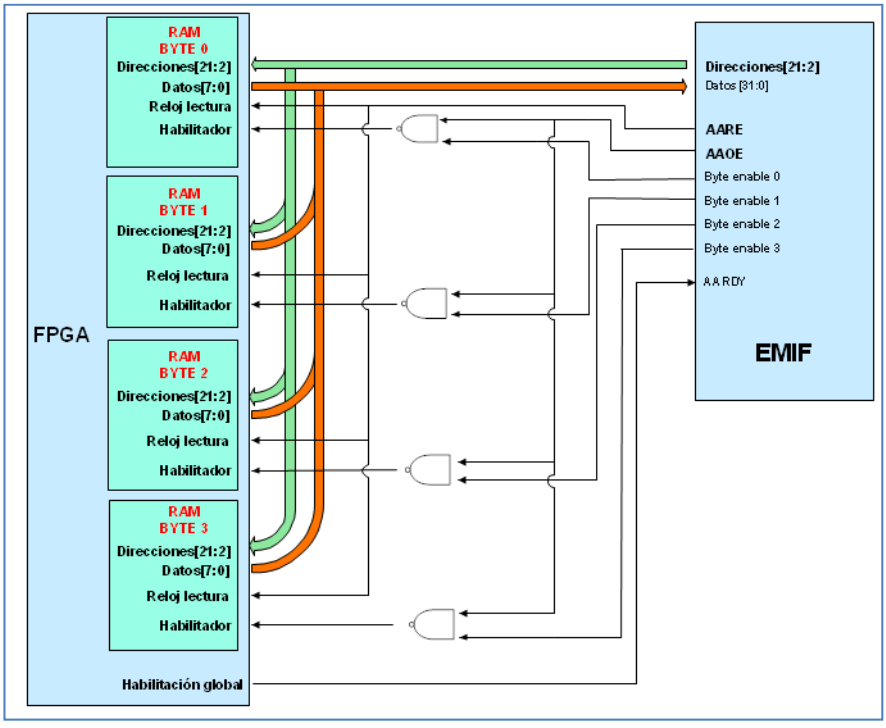


Figura 3.43. Interconexión de la memoria con el EMIF de un DSP

3.1.3.5. Módulo serie

El interfaz cameralink define un par de señales de control las cuales manejan el protocolo de transmisión RS-232 con los niveles de señalización LVDS. De igual forma, especifica una velocidad de 9600 bps mínimo, la cual puede ser variada de acuerdo al fabricante. Para nuestro caso específico tenemos que la cámara maneja tanto la velocidad mínima como una velocidad superior de 57 Kbps. El módulo implementado permite la comunicación entre la cámara y el FPGA. El módulo contiene almacenadas todas las opciones de programación de la cámara (frecuencia, tamaño de imagen, tiempos, etc.) y conecta la señal SERTFG, señal de respuesta de la cámara, hacia la PC para poder verificar que la cámara recibió el comando de forma correcta. El sistema es controlado mediante una serie de interruptores (dip switch) los cuales permiten escoger entre cada una de las opciones de comandos que se tiene y, en el instante que se genera la combinación correcta de interruptores un push button permite enviar el comando a la cámara.

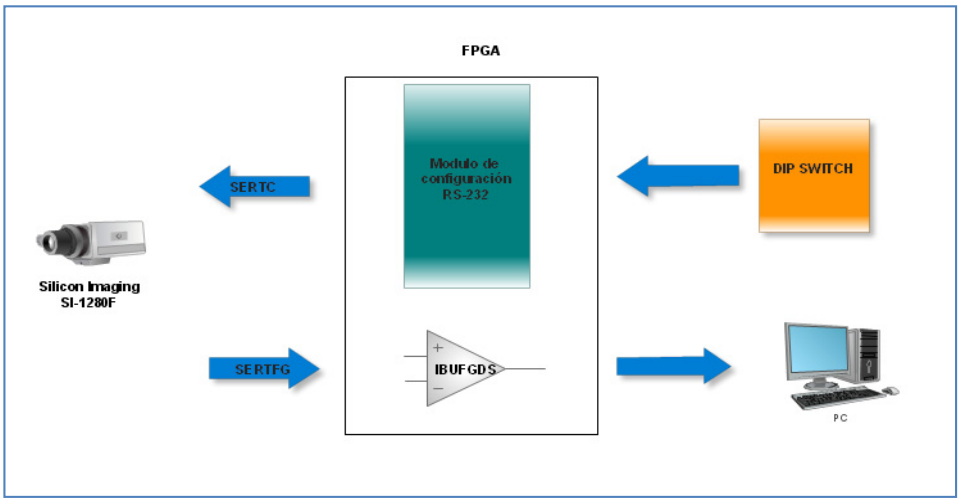


Figura 3.44. Diagrama a bloques de la interconexión serial del sistema

En las figuras 3.45 y 3.46 se muestra la implementación del módulo serial en las FPGAs de Xilinx y Altera respectivamente.

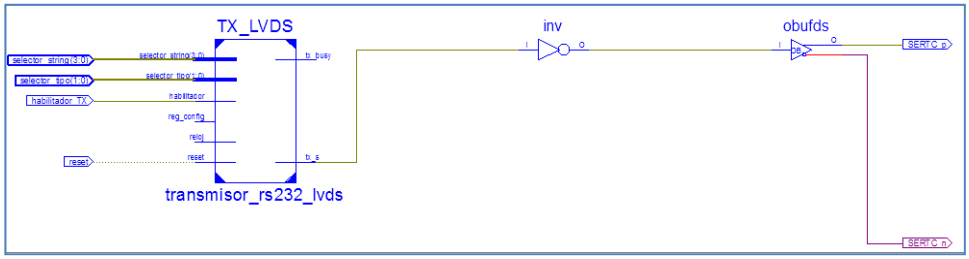


Figura 3.45. Top del módulo de configuración serial

.

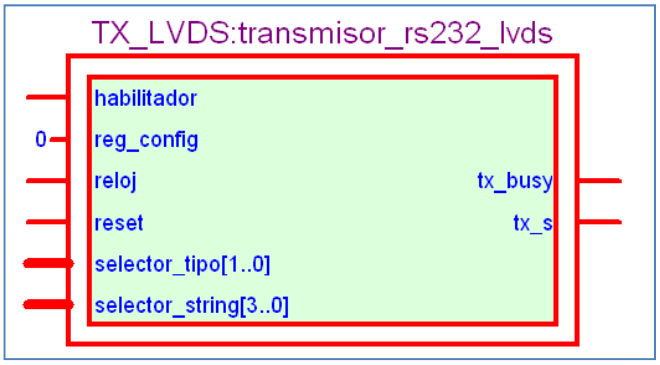


Figura 3.46. Módulo serial implementado en un FPGA Altera

3.2 ALGORITMOS

Como se mencionó al inicio del capítulo, aquí describiré las etapas del software.

3.2.1 Conversión formato Bayer a RGB entre imágenes

La cámara entrega la foto en formato Bayer, en la figura 3.47 se muestra un ejemplo de la imagen tomada por la cámara y almacenada en la memoria del FPGA.

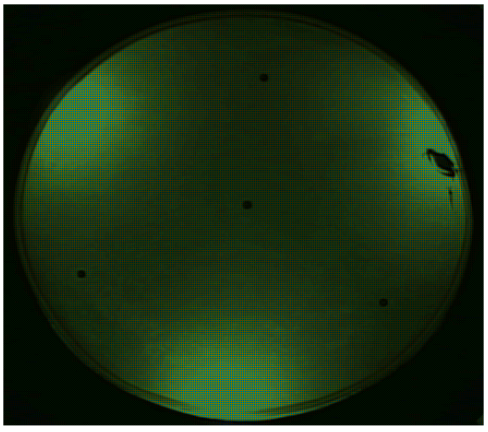


Figura 3.47. Fotografía de los animales en estudio en formato Bayer

En la figura 3.47, se puede apreciar que la foto está casi en verde, esto se debe a que la mitad de los píxeles son verdes un cuarto rojo y otro cuarto azules. Si se recorta una fracción de la imagen y se agranda se aprecia los tres colores base que intervienen en la imagen, como se muestra en la figura 3.48.

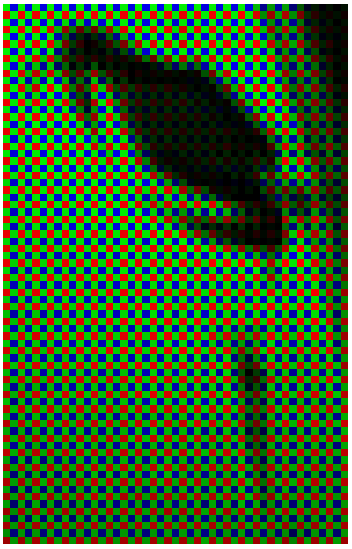


Figura 3.48. Fotografía recortada que muestra los 3 píxeles de colores

Las imágenes antes mencionadas requieren un procesamiento para que se puedan ver como se ven en la realidad, a este procesamiento se le denomina convertidor Bayer a RGB. Basándose en el apartado 2.3 se dispuso realizar este convertidor en un DSP con las características mencionadas en el apartado 2.7.

Partimos de una imagen de 480 x 720 en formato Bayer, que requiere de una matriz de 480 x 720 para su almacenamiento y que tras pasar por el convertidor generará 3 matrices de 480 x 720, una matriz para cada color, como se muestra en la figura 3.49.

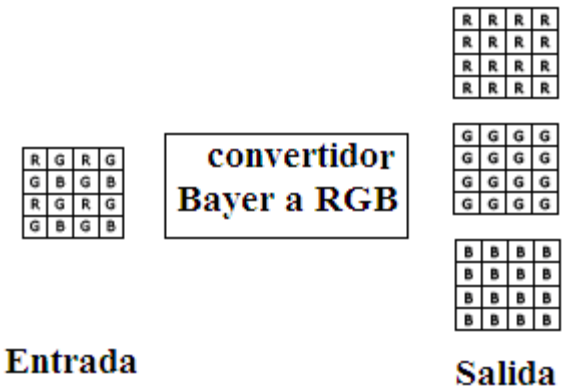


Figura 3.49. Diagrama a bloques de la entrada y salida del convertidor Bayer a RGB

Al implementar el convertidor la necesidad de memoria para almacenar la imagen se triplica, además de requerir de un tiempo para ejecutar el algoritmo de conversión.

La implementación del convertidor requiere que para cada píxel que entrega la cámara, se generen tres píxeles: verde, rojo y azul; para lograr esto se toma como referencia el píxel central, y se identifican cuatro casos.

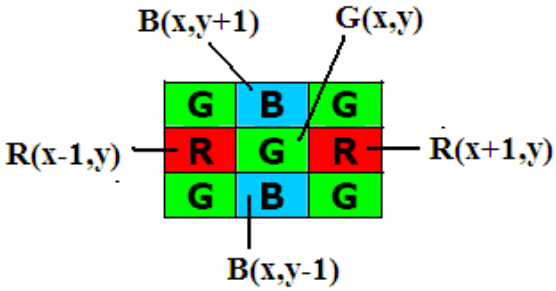
Primer caso: es cuando el píxel central es verde (en fila de rojos) y se requiere encontrar el rojo y el azul. Para el cálculo del píxel rojo se tomará el promedio de los dos píxeles que están en la misma fila, y para el cálculo del píxel azul se tomará el promedio de los píxeles que están en la misma columna. De esta manera se tienen los tres píxeles centrales necesarios para el formato RGB.	

Tabla 3.9. Primer caso de identificación de patrones para el formato Bayer

Las ecuaciones implementadas en el algoritmo quedaron así:

$$G(x,y) = G(x,y)$$

$$R(x,y) = \frac{R(x-1,y) + R(x+1,y)}{2}$$

$$B(x,y) = \frac{B(x,y+1) + B(x,y-1)}{2}$$

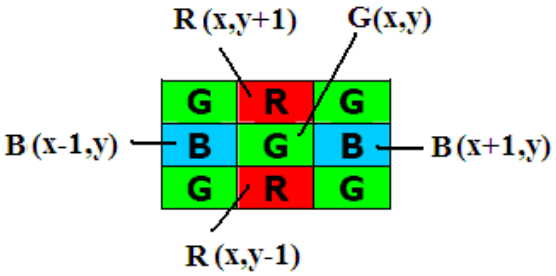
Segundo caso: es muy parecido al anterior, siempre el píxel central es verde (en fila de azules) la diferencia es la posición de los píxeles rojo y azul ya que intercambiaron su posición. Para el cálculo del píxel rojo se tomará el promedio de los dos píxeles que están en la misma columna, y para el cálculo del píxel azul se tomara el promedio de los píxeles que están en la misma fila. De esta manera se tienen los tres píxeles centrales necesarios para el formato RGB.	

Tabla 3.10. Segundo caso de identificación de patrones para el formato Bayer

Las ecuaciones implementadas quedaron así:

$$G(x,y) = G(x,y)$$

$$R(x,y) = \frac{R(x,y+1) + R(x,y-1)}{2}$$

$$B(x,y) = \frac{B(x-1,y) + B(x+1,y)}{2}$$

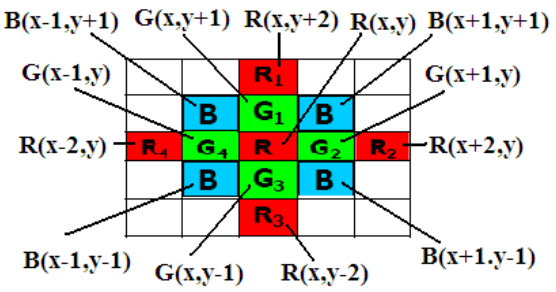
Tercer caso: cuando el píxel central es rojo, para calcular el píxel azul se tomaron el promedio de los cuatro píxeles azules alrededor del píxel rojo, y para calcular el píxel verde se calculó cual era el promedio de menor peso entre el promedio de los píxeles de la fila y la columna y en caso de que fueran iguales se tomaba un promedio de los cuatro píxeles que estaban alrededor del píxel rojo.	

Tabla 3.11. Tercer caso de identificación de patrones para el formato Bayer

Las ecuaciones implementadas en algoritmo quedaron así:

$$R(x,y) = R(x,y)$$

$$B(x,y) = B(x-1,y-1) + B(x+1,y+1) + B(x+1,y-1) + B(x-1,y-1)$$

$$G(x,y) = \begin{cases} (G_1 + G_3)/2, & if \quad |R_1 - R_3| < |R_2 - R_4| \\ (G_2 + G_4)/2, & if \quad |R_1 - R_3| > |R_2 - R_4| \\ (G_1 + G_2 + G_3 + G_4)/4 & if \quad |R_1 - R_3| = |R_2 - R_4| \end{cases}$$

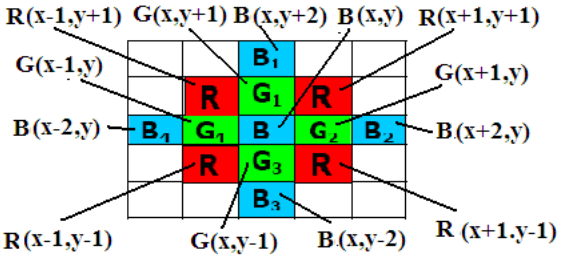
Cuarto caso: cuando el píxel central es azul, para calcular el píxel central de color rojo se promediaron a los cuatro píxeles rojos que están alrededor del píxel azul, y para calcular el píxel central verde se calcularon los promedios de los dos píxeles de la fila y de la columna y el que resultara más débil era el que determinaba el píxel central verde, en caso de que resultaran iguales se calculaba el promedio de los cuatro píxeles verdes que estaban alrededor del píxel central.	

Tabla 3.12. Cuarto caso de identificación de patrones para el formato Bayer

Las ecuaciones implementadas en algoritmo quedaron así:

$$B(x,y) = B(x,y)$$

$$R(x,y) = R(x-1,y-1) + R(x+1,y+1) + R(x+1,y-1) + R(x-1,y-1)$$

$$G(x,y) = \begin{cases} (G_1 + G_3)/2, & if \quad |B_1 - B_3| < |B_2 - B_4| \\ (G_2 + G_4)/2, & if \quad |B_1 - B_3| > |B_2 - B_4| \\ (G_1 + G_2 + G_3 + G_4)/4 & if \quad |B_1 - B_3| = |B_2 - B_4| \end{cases}$$

Como en todo tratamiento de imágenes los bordes siempre son un problema ya que las ecuaciones anteriores no funcionan, por lo que se tuvo que identificar la primera fila, la última fila, la primera comuna y la última columna para replantear nuevas ecuaciones.

Viendo la imagen mostrada en la tabla 3.10. En la primera fila se consideraron tres casos. El primero cuando se está en el píxel (0,0), el píxel verde, rojo y azul son:

$$G(0,0) = G(0,0)$$
$$R(0,0) = R(1,0)$$
$$B(0,0) = B(0,1)$$

En el caso dos, para los píxeles (n,0) y n es impar, el píxel verde, rojo y azul son:

$$G(n,0) = \frac{G(n-1,0) + G(n+1,0)}{2}$$

$$R(n,0) = R(n,0)$$

$$B(n,0) = \frac{B(n-1,1) + B(n+1,1)}{2}$$

En el caso tres, para los píxeles (n,0) y n es par, el píxel verde, rojo y azul son:

$$G(n,0) = G(n,0)$$

$$R(n,0) = \frac{R(n-1,0) + R(n+1,0)}{2}$$

$$B(n,0) = B(n,1)$$

Viendo nuevamente la imagen mostrada en la tabla 3.10. En la primera columna los píxeles (0,m), siendo m impar se tiene que el píxel verde, rojo y azul son:

$$G(0,m) = \frac{G(0,m-1) + G(0,m+1)}{2}$$

$$R(0,m) = \frac{R(1,m-1) + R(1,m+1)}{2}$$

$$B(0,m) = B(0,m)$$

Siendo m par se tiene que el pixel verde, rojo y azul son:

$$G(0,m) = G(0,m)$$

$$R(0,m) = R(1,m)$$

$$B(0,m) = \frac{B(0,m-1) + B(0,m+1)}{2}$$

Y la última esquina con coordenada (n,m) que terminara en verde, el azul y rojo serían los píxeles más cercanos de sus respectivos colores, quedando así:

$$G(n,m) = G(0,0)$$

$$R(n,m) = R(n,m-1)$$

$$B(n,m) = B(n-1,m)$$

Después de ver el comportamiento de la primera fila y la primera columna, analizamos el resto de los píxeles de la primera fila, los píxeles pares son verdes, para calcular los píxeles rojos sobre estos píxeles verdes se calcula el promedio de los dos píxeles continuos al píxel verde, y para calcular el píxel azul repetimos la segunda fila del píxel azul quedando las siguientes ecuaciones:

$$G(xpar,0) = G(xpar,0)$$

$$R(xpar,0) = \frac{R(xpar-1,0) + R(xpar+1,0)}{2}$$
$$B(xpar,0 = (B(xpar,1)$$

Para los píxeles impares de la primera fila, que son rojos, el píxel verde se calcula con el promedio de los tres píxeles que rodean al píxel rojo, para el calcular el píxel azulse toma el promedio de los dos píxeles de la segunda fila que estaban más cerca al píxel rojo, a continuación se muestran las ecuaciones obtenidas:

$$R(ximpar,0) = R(ximpar,0)$$

$$G(ximpar,0) = \frac{G(ximpar-1,0) + G(ximpar+1,0) + G(ximpar,1)}{3}$$

$$B(ximpar,0) = \frac{B(ximpar-1,1) + B(ximpar+1,1)}{2}$$

De la misma forma se analizó los otros tres bordes y se obtuvieron ecuaciones equivalentes.

Una vez identificado cada caso se realizó el barrido de toda la imagen, teniendo como resultado la imagen mostrada en la figura 3.50.

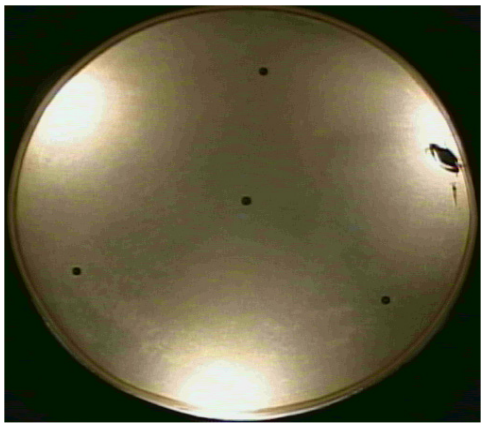


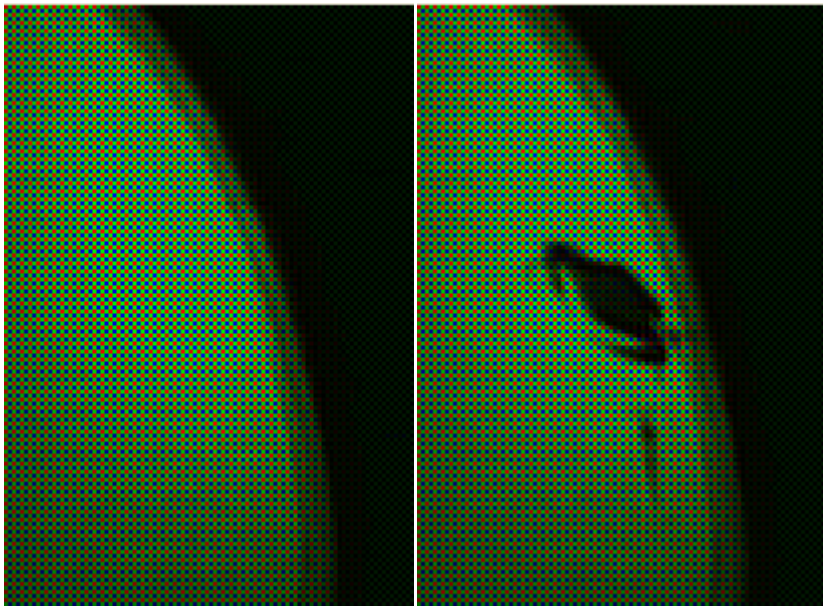
Figura 3.50. Fotografía en formato RGB obtenida del convertidor Bayer a RGB

En la implementación se requirió 2440 bytes de memoria para el almacenamiento del programa, el tiempo de ejecución fue de 38.019 ms. Con un DSP C6416 que funciona a 1GHz, y la imagen convertida tenía una dimensión de 480x720.

3.2.2 Algoritmo de resta y umbralización de imágenes

En el siguiente apartado se realizará la resta y umbralización bajo tres diferentes sistemas; microprocesador embebido, máquinas de estados implementados en un FPGA, y un procesador digital de señales. La imagen será de 102 X 150 píxeles para los tres.

En la figura 3.51 se pueden apreciar las imágenes que se desean restar para poder detectar los objetos en este caso la jaiba y el camarón.



a) b)

Figura 3.51. En (a) se tiene la imagen de fondo, en (b) la imagen con los objetos que se desean detectar

Independientemente de cualquier sistema que se implemente, lo primero que se desea lograr es separar el fondo de los objetos que se desean detectar. El método utilizado para la eliminación del fondo es el de la resta de imágenes. La imagen resultante conservará únicamente a los agentes que no se encuentran en la imagen de fondo y en este caso se trata de los objetos. Los píxeles no pertenecientes a los objetos se mantienen en un rango de valores muy cercanos a cero y se aprecian en la imagen como colores oscuros o muy cercanos a negro.

La diferencia entre dos imágenes *f(x,y)* y *h(x,y)*, se expresa de la forma:

$$g(x,y) = |f(x,y) - h(x,y)|$$

El efecto es que solamente las áreas en las que *f(x,y)* y *h(x,y)* son diferentes aparecerán en la imagen de salida como detalles mejorados.

El siguiente paso es binarizar la imagen por medio de una umbralización. De manera concreta, esto consiste en fijar un valor de brillo constante, el cual servirá como umbral para discernir entre las secciones de la imagen perteneciente al fondo y las que pertenecen a los objetos de interés. Si el nivel de brillo del píxel en análisis supera el umbral fijado su valor será modificado y establecido con el valor máximo de brillo en una imagen con resolución de 8 bits y este valor es de “255”. En el caso en donde el valor del brillo no supere el umbral, su valor será modificado al valor mínimo de brillo y correspondiente al negro profundo, es decir, “0”.

3.2.2.1 Implementación con un procesador embebido en el FPGA

El siguiente objetivo es procesar imágenes en formato Bayer con un microprocesador embebido de 32 bits llamado "Microblaze" con la finalidad de poder detectar objetos, para este caso una jaiba y un camarón.

Para lograr este objetivo se realizó un sistema electrónico completo que pueda cargar las imágenes necesarias al sistema y desplegar los resultados del procesamiento en un monitor. Se optó por utilizar imágenes en formato Bayer, por el menor espacio que ocupan en memoria y porque la cámara que se emplea, transmite las imágenes que captura en este formato. El sistema se diseñó partiendo de que las imágenes ya están cargadas en memorias de doble puerto dentro del FPGA, tal como se indica en el apartado 3.1.

El sistema electrónico consta principalmente de:

Dos memorias RAM. En donde se guardan las imágenes de entrada que se van a procesar, a una le llamamos imagen de fondo y a la otra imagen con los objetos, estas imágenes han sido obtenidas por la etapa de captura y almacenamiento descrito en el apartado 3.1.3.

Divisores de frecuencia. Sirven para dividir la frecuencia del reloj de entrada a una frecuencia deseada.

Módulo de VGA. Es un hardware electrónico diseñado para leer una imagen de una memoria RAM y transmitirla por el puerto VGA.

Memoria RAM de doble puerto. Es la memoria donde se guarda la imagen final o sea el resultado del procesamiento. Este posee dos puertos uno lo utiliza el procesador para escribir y el otro lo utiliza el módulo del VGA para leer la imagen.

Microprocesador. Programado en lenguaje C con la herramienta EDK (Embebed Development Kit), es el encargado de ejecutar el algoritmo de procesamiento.

En la figura 3.52 podemos observar el diagrama electrónico completo que consta de los bloques antes mencionados.

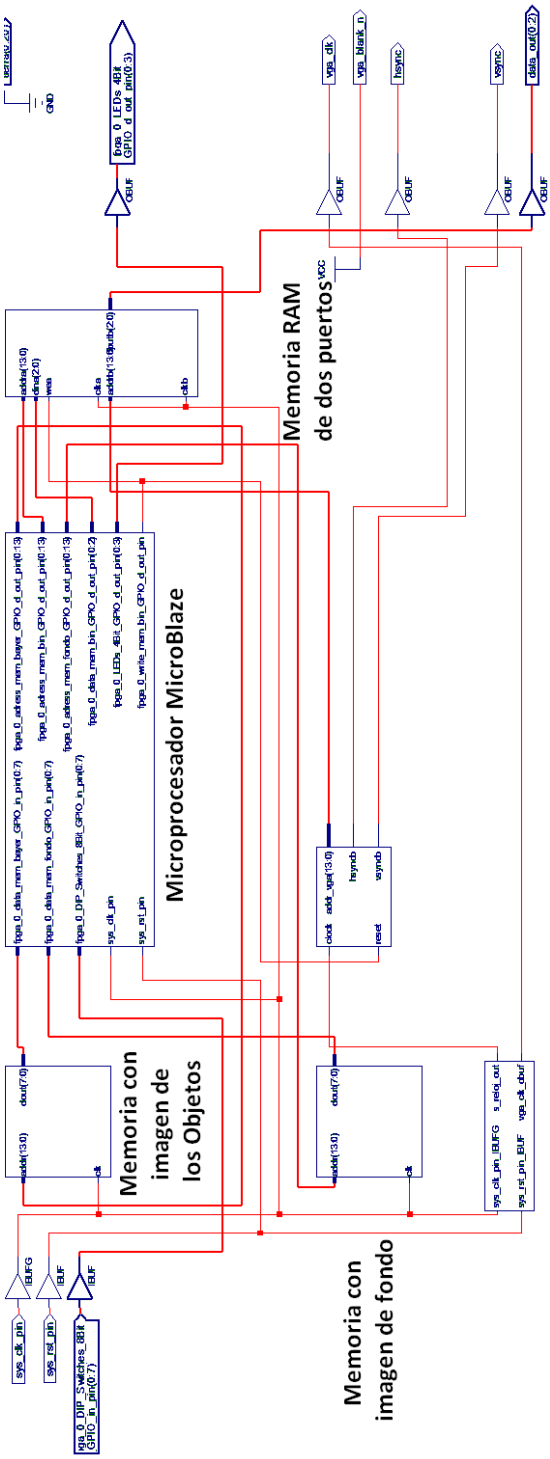


Figura 3.52. Sistema electrónico completo

A continuación se presentan las características de cada uno de los bloques utilizados. En la tabla 3.13 se especifica las características del core usado para las memorias que guardan las imágenes de entrada, cada una es un block Ram de un solo puerto. Estas memorias fueron generadas con el core generador dentro del ISE Foundation.

Totalmente sincronizado en el módulo de Virtex ™, Virtex-II, Virtex-II Pro, Virtex-4, Spartan-II ™, Spartan-IIE, Spartan-3, y Spartan 3EFPGAs.
Soporta las tres opciones de modo de escritura: leer después de escribir, leer antes de escribir, no leer en la escritura (Disponible sólo para implementar en Virtex-II, Virtex-II Pro, Virtex-4, Spartan-3 y Spartan-3E).
Soporta conectores de funciones ROM y RAM.
Soporta anchos de datos de 1 a 256 bits.
Soporta profundidades de memoria de 2 a 1 millón de palabras dependiendo de la arquitectura seleccionada.
Incorpora tecnología Xilinx Smart-IP ™ para la máxima parametrización y óptima implementación.
Soporta núcleos diseñados para la optimización de área o usando una simple primitiva como SelectRAM+™ o SelectRAM II
Soporta diferentes polaridades de los pines para distintas señales de control: reloj, habilitadores, habilitadores en ciclos de escritura y pin de inicialización de salida.
Disponible en el Xilinx CORE ™generator system v7.1i SP1 y versiones superiores.

Tabla 3.13 Características del Block RAM de un solo puerto

Esta memoria se usó dentro del proyecto como una ROM. También dentro del Core generator se puede generar una memoria RAM pero con dos puertos A y B los cuales funcionan de una manera muy similar a los bloques RAM de un solo puerto. En la tabla 3.14 se dan las características de una memoria de doble puerto que se utiliza.

Módulo para los FPGAsVirtex ™, Virtex-E, Virtex-II, Virtex-II Pro, Virtex-4, Spartan ™-II, Spartan-IIE, Spartan-3 y Spartan-3E.
Soporta las tres opciones de modo de escritura: leer después de escribir, leer antes de escribir, no leer en la escritura (Disponible sólo para implementar en Virtex-II, Virtex-II Pro, Virtex-4, Spartan-3 y Spartan-3E).
Soporta conectores de funciones ROM y RAM.
Soporta anchos de datos de 1 a 256 bits.
Soporta profundidades de memoria de 2 a 1 millón de palabras dependiendo de la arquitectura seleccionada.
Soporta diferentes polaridades de los pines para distintas señales de control: reloj, habilitadores, habilitadores en ciclos de escritura y pin de inicialización de salida.
Disponible en el Xilinx CORE ™ generator system v7.1i SP1 y versiones superiores
Soporta funciones ROM, lo que permite operaciones de lectura simultáneas de la misma localidad.
Admite funciones de RAM, lo que permite operaciones de escritura simultáneas para separar las localidades y las operaciones de lectura simultáneas de la misma localidad.
Los puertos son completamente independientes entre sí.
Soporta configuraciones asimétricas para los puertos Ay B.
Soporta núcleos diseñados para la optimización de área o usando una simple primitiva como SelectRAM+™ o SelectRAM II
Incorpora tecnología Xilinx Smart-IP ™ para la máxima parametrización y óptima implementación

Tabla 3.14. Características del Block RAM de doble puerto

El bloque de memoria Dual-port está compuesto de uno o más bloques de 4Kb llamados Select.RAM+^{TM}. Un módulo de memoria tiene dos puertos independientes que habilitan el acceso a un solo espacio de memoria compartido. Ambos puertos son funcionalmente idénticos, cada puerto provee de un acceso de lectura y escritura a la memoria. Lecturas simultáneas de la misma localidad de memoria pueden ocurrir, pero si se desea leer y escribir simultáneamente en la misma localidad de memoria da como resultado en una correcta escritura del dato pero una invalida lectura de él.

Después de haber estudiado acerca de los tiempos (timing) que maneja el protocolo de comunicación VGA, ahora procederemos a adecuar los tiempos para nuestro diseño el cual usa una Virtex IV con un reloj principal de 100MHz, necesitamos configurar el proceso que enviará los píxeles al monitor con los tiempos de sincronía correctos.

Para encontrar cuantos píxeles se necesitan enviar para completar un tiempo de línea igual a 25.17µs con un periodo de reloj de 50ns se usó la siguiente relación.

$$\#\ Píxeles = \frac{Tiempo\_de\_linea}{Periodo\_de\_reloj} = \frac{25.17\ \mu s}{50ns} \approx 503$$

Por tanto, se necesita enviar 503 píxeles. Siendo la imagen de 102 pixeles, pondré a cero (blanking) los 401 valores restantes de la trama.

Se necesita dejar un espacio de tiempo entre cada trama de los píxeles de las líneas donde la sincronización vertical cambia de nivel, por lo tanto se calculó los píxeles equivalentes en un tiempo de 6.6us.

$$\#\ Píxeles\_Blanks = \frac{Tiempo\_de\_blank}{Periodo\_de\_reloj} = \frac{6.6\ \mu s}{50\ ns} = 132$$

Si se suma 503 con 132 nos arroja el resultado de 635 que son los tiempos de píxeles necesarios para poder enviar un pulso de sincronización horizontal semejante a los 31.77µs requeridos.

$$31.77\mu s \approx 635 \times 50ns$$

De una manera similar se hace para la lograr la sincronización vertical. Los resultados totales obtenidos se ven en la figura 3.53. Se observa que se tiene un tiempo de línea de 25199ns y un tiempo de blanking de 6599.9ns para la sincronía horizontal con lo cual se aproxima a los tiempos de 25.17µs y 6.6µs respectivamente.

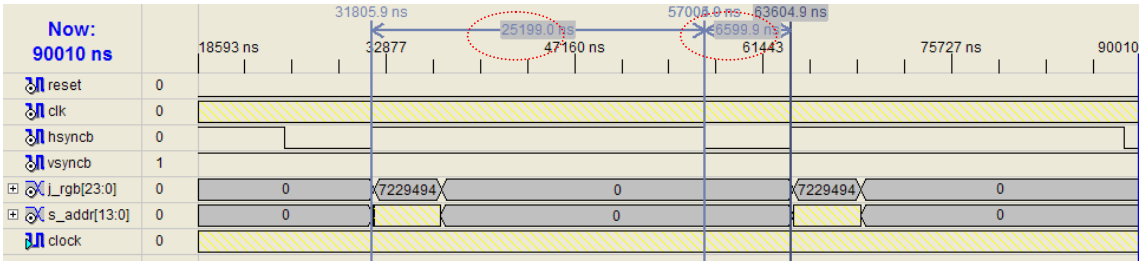


Figura 3.53. Ejemplo de los tiempos del controlador VGA

El módulo del VGA está pensado para direccionar una memoria RAM. Los datos de la memoria RAM, los formamos en líneas de píxeles, que son conformados y transmitidos al monitor con los pulsos apropiados de sincronía vertical y horizontal. En la figura 3.54 se ve los pines principales del controlador del VGA. Donde clk es el reloj a 20 MHz para esta aplicación, reset es el que reinicia la secuencia de transmisión de la imagen, hsyncb es la señal que controla la sincronización horizontal (líneas) y vsyncb es la señal que controla la sincronización vertical (frames)

Figura 3.54. Top del controlador VGA

El esquemático del proyecto se ve en la figura 3.55. Este controlador se diseñó en vhdl.

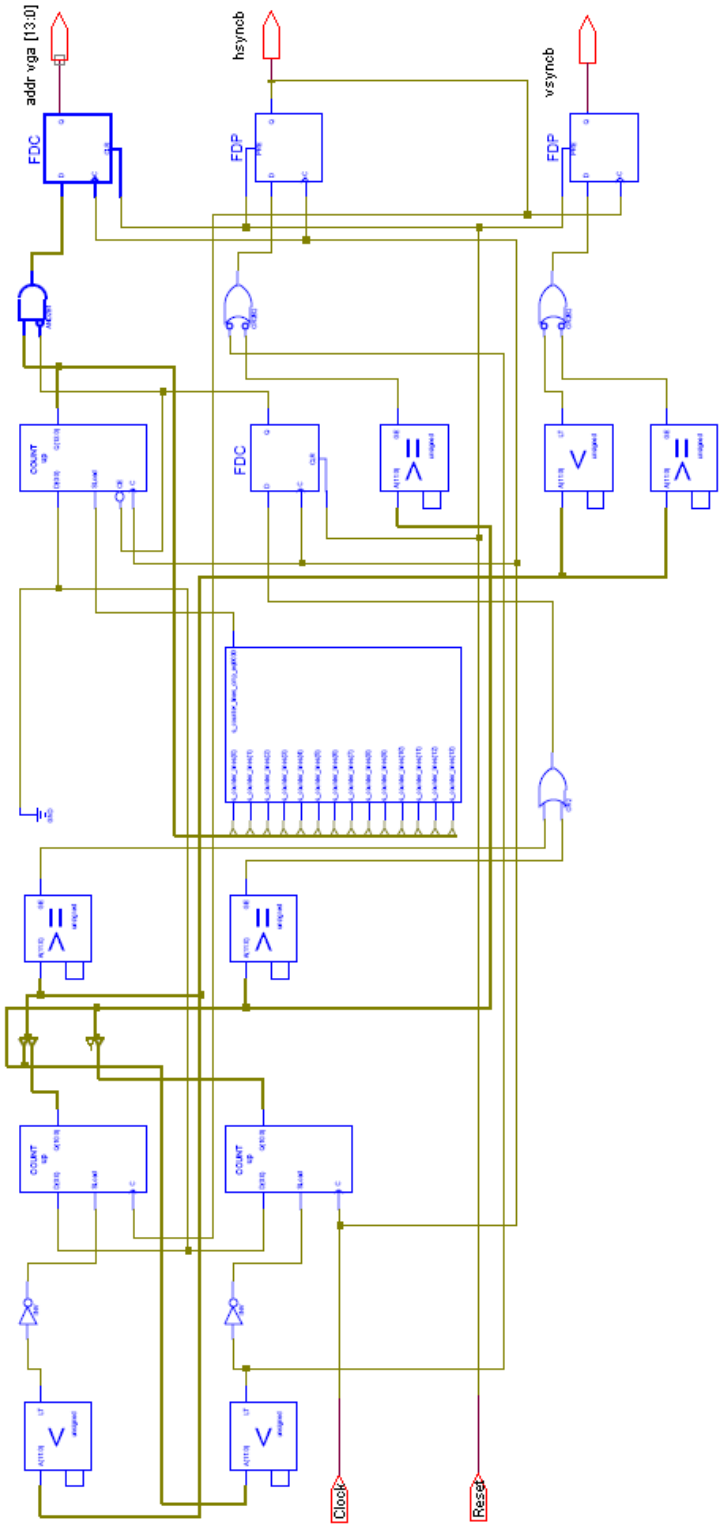


Figura 3.55. Esquemático del controlador VGA

El controlador VGA está formado, principalmente, por contadores, comparadores y registros. Estos componentes ayudan a generar correctamente los tiempos de sincronía vertical y horizontal y también el direccionamiento de la memoria RAM donde se almacena los píxeles.

Los divisores fueron desarrollados en VHDL y son simplemente unos contadores con comparadores. Para las aplicaciones que se implementaron se necesitaban dos frecuencias diferentes a la del sistema de 100 MHz que son de 20MHz para el controlador de VGA y de 50 MHz para el DAC del VGA así que se dividió la frecuencia principal en cinco y en dos.

El módulo tiene cuatro pines, los de entrada son el reloj de entrada y el reset, los de salida son las dos frecuencias divididas, en la figura 3.56 se muestra su implementación.

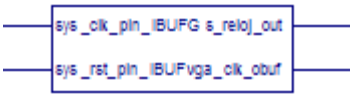


Figura 3.56. Top de los divisores de frecuencia

El código en C de este algoritmo (de la resta y umbralización) es ejecutado por el microcontrolador para detectar los objetos en la imagen, lo que hace básicamente es direccionar tanto la memoria de la imagen de fondo como la memoria de la imagen del objeto y luego guarda los valores de cada píxel en una variable, a continuación se restan los valores entre los píxeles de cada imagen para después umbralizarla ó binarizarla a la imagen, esta debe visualizarse en un monitor. En la figura 3.57 se muestra la secuencia de procesos que ejecuta el Microprocesador.

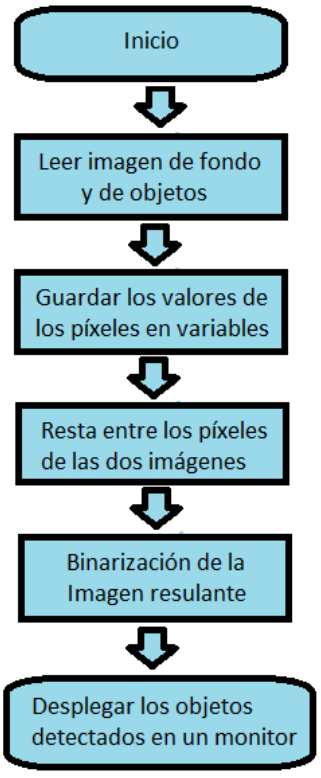
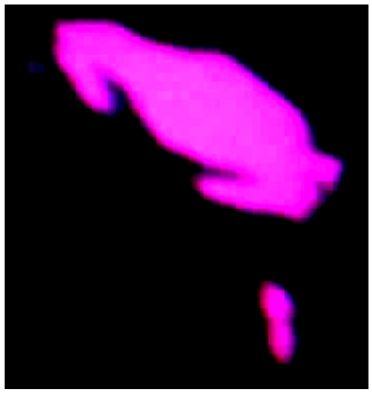


Figura 3.57. Diagrama de flujo del proceso de Segmentación

Como parte final de esta sección se muestra en la figura 3.58. Los resultados de los procesamientos con el Microblaze.



3.58. Imagen resultante en la resta y umbralización

La imagen presentada se tomó directamente del monitor y se amplió la zona de interés para su mejor visualización.

3.2.2.2 Implementación en un FPGA

En esta sección se explicará acerca de la implementación del hardware para la segmentación de imágenes Bayer, la cual se realizó bajo la plataforma del ISE foundation.

Desarrollo del Hardware

Tomando en cuenta la teoría de segmentación explicada en el capítulo II, se implementó un hardware que satisfaga las mismas exigencias aplicadas al anterior sistema de procesamiento. Para fines de esta tesis se implementó la resta de dos imágenes Bayer y la umbralización del resultado, logrando una segmentación con una mayor velocidad de procesamiento.

Las imágenes iniciales que se utilizaron son las mismas que las del procesamiento con el EDK, y de la misma manera se cargaron las imágenes a los bloques de memoria RAM.

En el ISE Foundation se diseñó un procesador para realizar los procesos planteados sobre las imágenes Bayer de entrada. En la figura 3.59 se ven los puertos que se requieren para lograr el procesamiento.

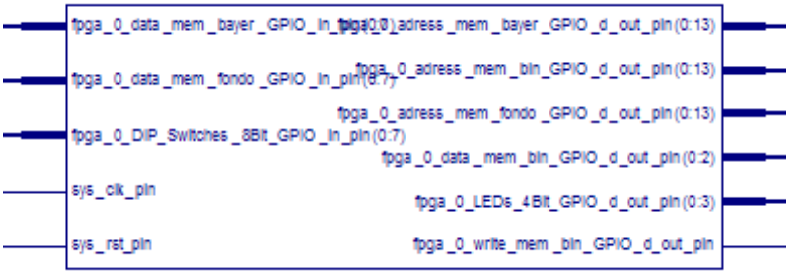


Figura 3.59. Puertos del sistema de procesamiento diseñado en VHDL

Como se observa en la figura 3.59 se tiene los mismos números de puertos que el sistema diseñado con el Microprocesador Microblaze y sus funciones son las mismas, pero la diferencia

entre ambas es la arquitectura interna. También por conveniencia se les llamó de la misma manera a los puertos de entrada y salida. Una descripción de los puertos se puede ver en la tabla 3.15.

Data_mem_bayer: Es el puerto de entrada de los datos provenientes de la memoria y contiene los objetos que se desean segmentar
Data_mem_fondo: Es el puerto de entrada de los datos provenientes de la memoria que contiene la imagen de fondo.
Dip_switches_8bit: Puerto de entrada de los dip_switches
Sys_clk_pin: Entrada del Reloj Principal
Sys_rst_pin: reset del sistema.
Adress_mem_bayer: Puerto de salida, cuya función es direccionar a la memoria ROM donde esta almacenada la imagen bayer con los objetos.
Adress_mem_bin: Puerto de salida, cuya función es direccionar un puerto de la memoria de dos puertos donde se va a almacenar la imagen resultado.
Adress_mem_fondo: Puerto de salida, cuya función es direccionar a la memoria ROM donde esta almacenada la imagen bayer de fondo.
Data_mem_bin: Puerto de salida donde se envía los datos del resultado del procesamiento al bus de datos de la memoria de dos puertos.
Leds_4bit: Puerto de salida para controlar los leds.
Write_mem: Puerto de salida que sirve para poder habilitar la escritura de la memoria de dos puertos

Tabla 3.15. Puertos de entrada y salida del procesador

El hardware cuenta con tres bloques principales como se puede apreciar en la figura 3.60.

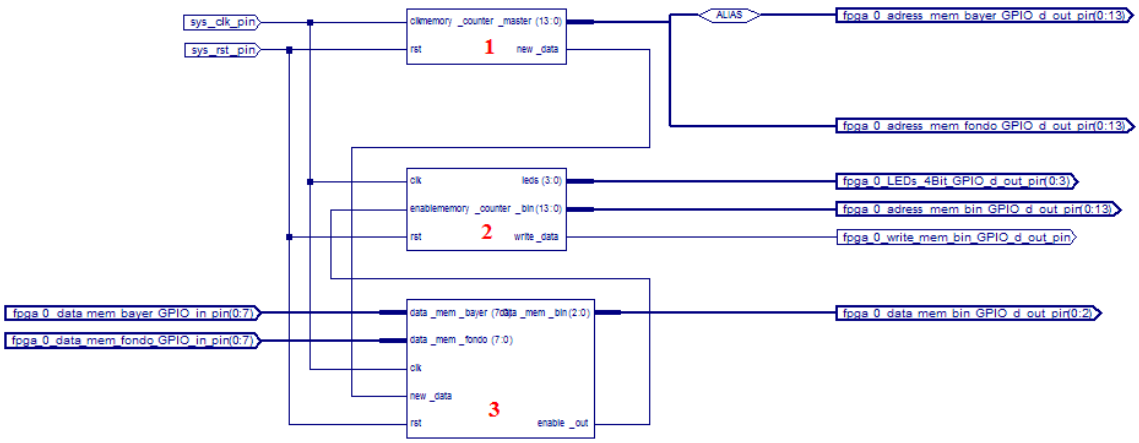


Figura 3.60. Arquitectura del Procesador

El bloque número uno se trata de un simple contador que va del cero al número de direcciones que tenga la memoria de las imágenes, para este caso las imágenes que se utilizaron son de 102 x 150 por tanto este contador va del cero hasta el 15300.En resumen este bloque va a servir para direccionar a las memorias que contienen las imágenes.

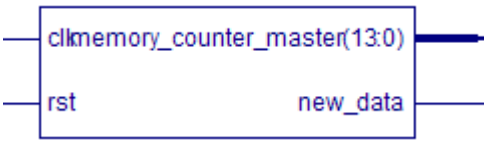


Figura 3.61. Módulo para direccionar a las memorias de entrada de datos

En el diagrama de tiempos de la figura 3.62, considerando un reloj de 100MHz, se puede observar que cuando el rst se deshabilita al empezar el conteo, cada 10ns se está solicitando una lectura de una dirección de memoria. Además se tiene un puerto de salida (new_data) que tiene la función de avisar el momento en que se pueden considerar que los datos son válidos.

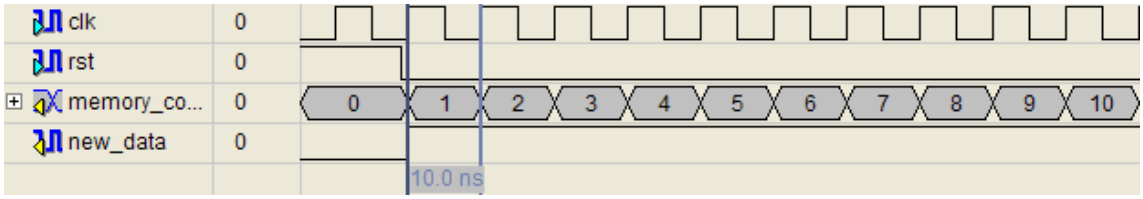


Figura 3.62. Diagrama de tiempos del bloque 1

El bloque número dos se trata igualmente de un contador que va del cero al número de direcciones que tenga la memoria que va a almacenar la imagen resultante, con la diferencia que cuenta con tres puertos de más, como se ve en la figura 3.63.

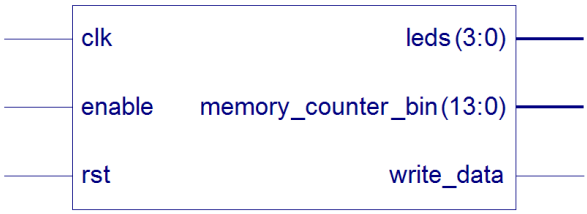


Figura 3.63. Módulo que direcciona a la memoria con la imagen resultante

Del diagrama de tiempos de la figura 3.64 se puede deducir que cuando la señal rst está deshabilitada y el enable habilitado, comienza este módulo a direccionar la RAM. El pin de write_data tiene la función de poner en modo de escritura a la memoria que se direcciona y los leds sirven para obtener información acerca del tiempo de procesamiento del sistema.

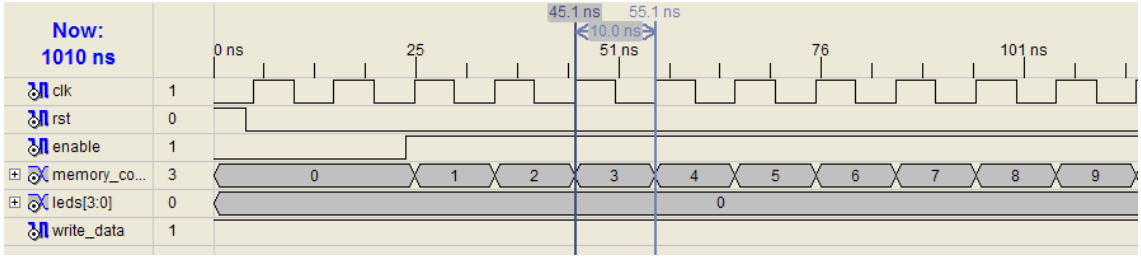


Figura 3.64. Diagrama de tiempos del módulo que direcciona a la memoria

En el bloque número tres se encuentra la unidad donde se realiza la resta y la umbralización. Se decidió poner un umbral con un peso de 39.

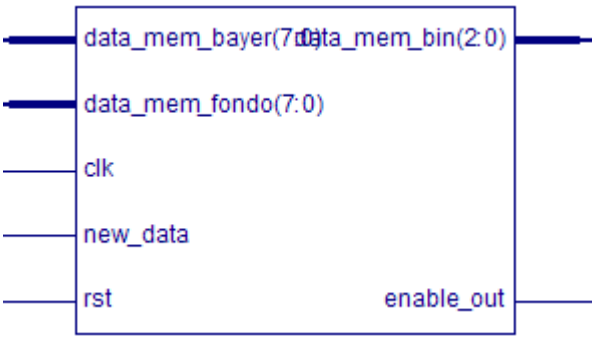


Figura 3.65. Módulo de resta y umbralización

Del diagrama de tiempos de la figura 3.66 se observan los resultados de un test bench que se realizó para poder comprobar el funcionamiento del módulo. Las dos entradas de datos representan la cantidad de energía de los píxeles de las imágenes que se van a restar. La entrada new_data habilita todo el módulo al indicar que empieza una nueva imagen o frame. La salida enable_out tiene la función de indicar que el dato de salida está listo para leerse. La salida importante es data_mem_bin donde se envía el resultado de la umbralización entre el resultado de la resta y el umbral, si el resultado es mayor que el umbral se envía un siete (correspondiente a blanco) si no un cero (correspondiente a negro). Para poder visualizar mejor el proceso se agregó una señal llamadas_resta donde se puede ver los resultados entre las restas de los dos píxeles.

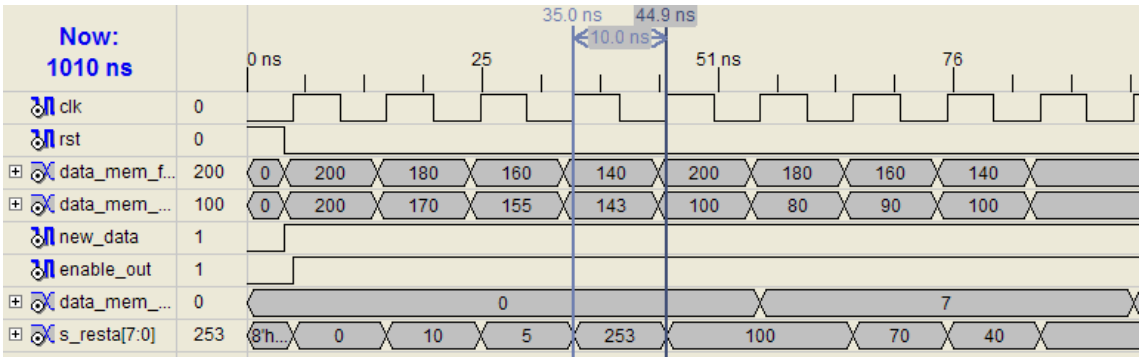


Figura 3.66. Diagrama de tiempos para el módulo de resta y umbralización

Vale la pena destacar que se manejaron números sin signo (unsigned) para ello se utilizó un rango de umbralización

$$pixel\_salida = \begin{cases} 7 & umbral < resta < 255 - umbral \\ 0 & otro\ caso \end{cases}$$

Para el caso de 140 – 143 el módulo arroja el resultado de 253 (ver figura 3.66) pero como el número es mayor al rango de aceptación el dato de salida es un cero.

Los archivos fuentes que se usaron para poder generar el top dentro del ISE foundation son los que se presentan en la figura 3.67.

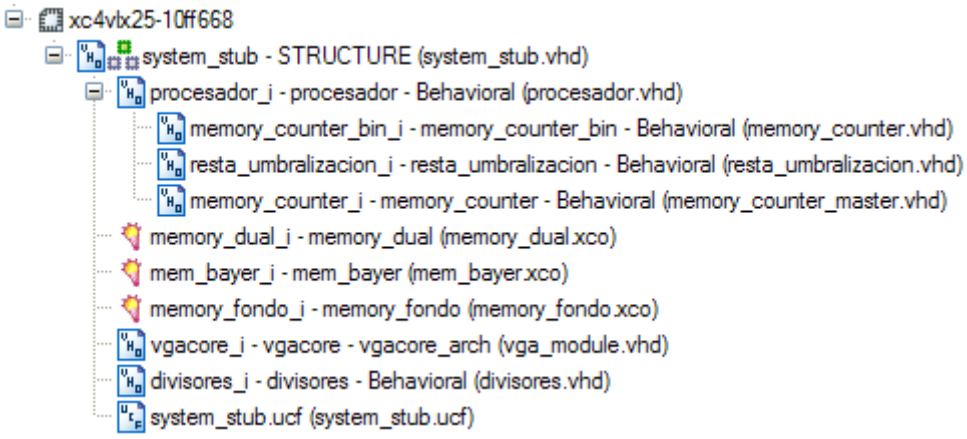


Figura 3.67. Archivos fuente dentro del proyecto

La descripción de los archivos fuente se especifican en la tabla 3.16.

system_stub.vhd: Es un archivo en lenguaje hdl cuya función es de instanciar todo el sistema electrónico.
procesador.vhd: Es un archivo en hdl en el que se instancia el procesador que consta de tres partes.
memory_counter_master.vhd: contiene la descripción en hdl del direccionador de las imágenes de entrada
resta_umbralizacion.vhd: Archivo que contiene el módulo que realiza la resta y la umbralización dentro del procesador.
memory_counter.vhd: contiene la descripción en hdl del direccionador de la memoria de la imagen resultado.
memory_dual.xco: Es un archivo creado por el Xilinx core generador, contiene una memoria RAM de dos puertos.
memory_bayer.xco: Es un archivo creado por el Xilinx core generador, contiene una memoria ROM con una matriz Bayer precargada
memory_fondo.xco: Es un archivo creado por el Xilinx core generador, contiene una memoria ROM con una matriz Bayer precargada.
vga_core.vhd: este archivo contiene la descripción del hardware del controlador de vga.
divisores.vhd: archivo fuente donde se encuentran los divisores de frecuencia.
system_stub.ucf: Es un archivo al que se le asigna a los puertos del top un periférico de la tarjeta virtex 4 ML 401.

Tabla 3.16. Descripción de los Archivos fuente

En la figura 3.68 Se observa la imagen resultante de la resta y la umbralización con la implementación en el FPGA.

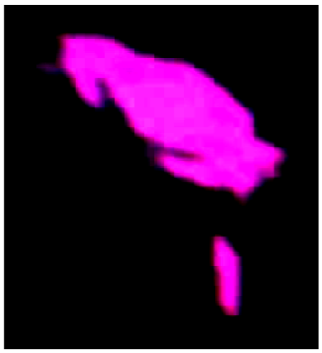


Figura 3.68. Imagen resultante

3.2.2.3 Implementación en un DSP

Se desea procesar imágenes con formato Bayer con la herramienta de programación C++ bajo el ambiente Code Composer Studio 3.1 (CCS 3.1) con la finalidad de poder detectar objetos.

Para lograr separar el fondo de los objetos que se desean detectar se procedió a recuperar las imágenes. La primera imagen que era el fondo se transfirió de la memoria implementada en el FPGA a la memoria RAM de la tarjeta de evaluación del DSP, la imagen donde estaban los animales se conservó en la memoria implementada en el FPGA.

Como parte primordial de esta sección se presentarán los resultados obtenidos de las implementaciones de los algoritmos (resta y umbralización) para la detección de objetos que se realizó con la herramienta CCS 3.1.

Se observa en la imagen el resultado de la Resta que eliminó el fondo del escenario. Se percibe de manera fácil la huella de los animales en el fondo y todo lo que los rodea queda en niveles de grises muy cercanos a cero, resaltando a la vista los objetos de interés como se observa en la figura 3.69.

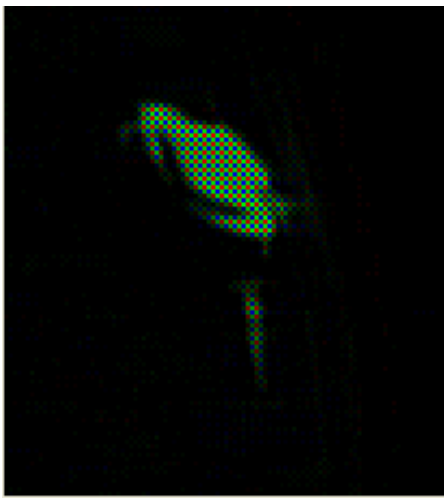


Figura 3.69. Resultado de la resta entre la imagen de fondo y la que contiene los objetos

Después de umbralizar la imagen de la resta se observa que este proceso sirvió para eliminar por completo las regiones de la imagen con valores relativamente cercanos a los de las

siluetas de los animales, y que no corresponden a las secciones de la imagen en donde se encuentran posicionados los cuerpos del camarón y la jaiba. No obstante todavía tenemos algunos inconvenientes, debido a que hay unos pequeños puntos blancos separados del cuerpo de la jaiba como se aprecia en la figura 3.70, que pueden ser identificados como otros objetos y por lo tanto será necesario eliminarlos.

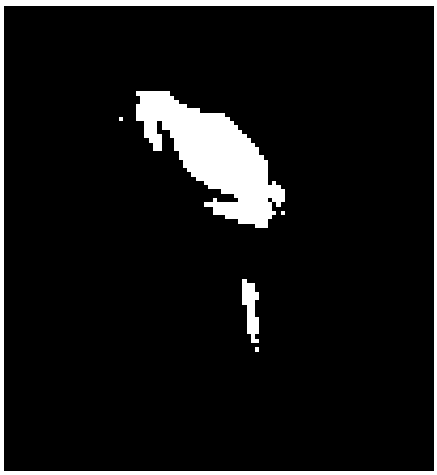


Figura 3.70. Resultado de umbralizar la imagen resultante de la resta

El desarrollo de este trabajo, implementado con relativa rapidez gracias a las bondades dadas por la herramienta de programación CCS 3.1, logró probar algunos algoritmos conocidos dentro del campo de procesamiento de imágenes pero aplicados a imágenes con formato Bayer. Se puede concluir, al observar los resultados, que la teoría de segmentación se aplica perfectamente a las imágenes Bayer.

3.2.3 Algoritmo del filtro pasa baja

Con el fin de poder comparar el desempeño del filtro pasa bajo, se implementaron en dos tecnologías diferentes, el microcontrolador embebido en un FPGA y un procesador digital de señales.

Debido a que este procesamiento se realiza después de haber realizado la resta y umbralización, la dimensión de la imagen resulta de 102X150 píxeles y es tomada de los resultados generados por las etapas previas.

3.2.3.1 Implementación con Microblaze

El filtro paso bajos funciona como un difuminador, el cual causará que las líneas de color más oscuro dentro de los cuerpos de los objetos que se deseen detectar eleven su valor de brillo en una magnitud muy moderada. Como se trata de un promediador, el filtro no modificará las áreas de la imagen que tengan valores constantes, como lo son las áreas alejadas de los objetos de interés (fondo), ni tampoco el interior de las siluetas de los objetos que tienen un valor constante de brillo más elevado.

Básicamente el algoritmo del filtro paso bajo se implementa con una máscara de 3X3 que promedia los nueve píxeles de la máscara y sustituye el píxel central por el valor promedio, esto se realiza a través de toda la imagen.

El siguiente paso es binarizar nuevamente la imagen por medio de una umbralización, como se realizó después de la resta. El valor de umbral que se utiliza en este proceso es 39, que es suficiente para eliminar totalmente el fondo.

El código en C de este algoritmo es ejecutado por el microcontrolador para detectar los objetos en la imagen. Lo que hace, básicamente, es direccionar tanto la memoria de la imagen de fondo como la memoria de la imagen del objeto y luego guarda los valores de cada píxel en una variable, a continuación se restan los valores entre los píxeles de cada imagen para después difuminar la imagen resultado y por último se umbraliza ó binariza la imagen, esta debe poder visualizarse en un monitor. En la figura 3.71 se muestra la secuencia de procesos que ejecuta el Microprocesador.

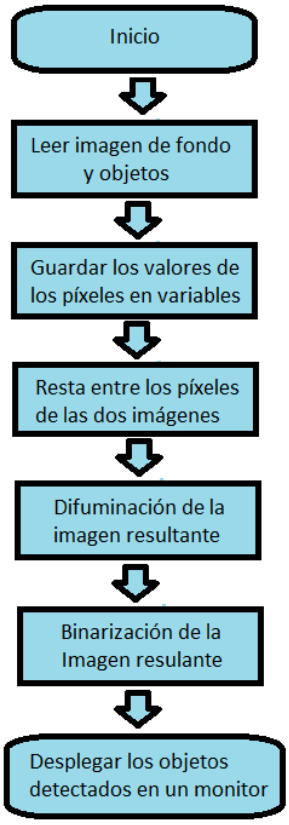
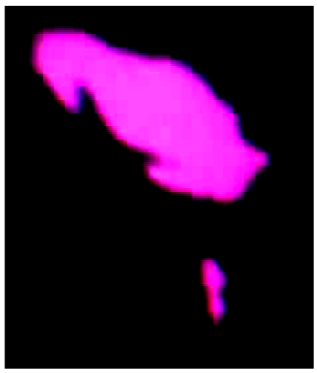


Figura 3.71. Diagrama de flujo del proceso de Segmentación

Como parte final de esta sección se muestra en la figura 3.72 los resultados de los procesamientos con el Microblaze.

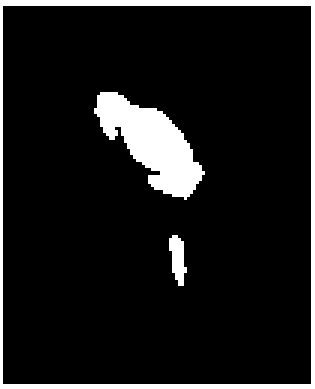


3.72. Imagen resultantes en la segmentación completa

3.2.3.2 Implementación con DSP

Básicamente al igual que en el sistema con el microcontrolador embebido (sección 3.2.3.1), el código para el DSP es casi el mismo, ya que ambos utilizan compiladores de C, varía principalmente en la manera de acceder a las imágenes, además de que los resultados fueron diferentes en los tiempos de ejecución.

Los resultados obtenidos se pueden apreciar en la figura 3.73.Los resultados de tiempo se presentan en el capítulo 4.



3.73. Imagen resultante al aplicarse el filtro pasa bajo

3.2.4 De segmentación en regiones

3.2.4.1 Calculo de centros de masas y áreas de los animales

En esta etapa del procesamiento se pretende extraer los centros de masa de cada uno de los objetos de interés, mediante un algoritmo relativamente sencillo a partir de la imagen resultado del filtro pasa bajo.

Primero es importante definir lo que en la rama de procesamiento de imágenes representa un centro de masa. El ***centro de masa*** de un objeto en una imagen es el promedio de las coordenadas de los píxeles que lo conforman, en otras palabras, para obtener el centro de masa de un objeto en una imagen se tiene que promediar las coordenadas en X de los píxeles que representan su área y de esta manera obtener la coordenada en X respectiva a su centro de masa, el mismo procedimiento corresponde a la obtención de la coordenada en Y del centro de masa. En la figura 3.74 se presenta una imagen ejemplo de lo que son los centros de masa en una imagen, los polígonos representan diferentes objetos en la imagen, estos tienen un cierto número de píxeles en sus respectivas áreas y como se distingue los puntos en sus interiores son los centros de masa o centroides.

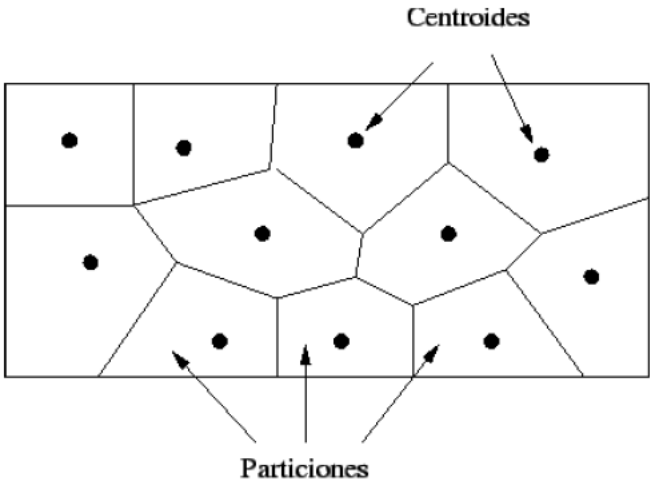


Figura 3.74. Centros de masa o centroides

Como se vio en la sección anterior, el resultado de la segmentación fue una imagen binaria donde los píxeles pertenecientes a los animales tienen el valor de "255" y los pertenecientes al fondo y al resto de la imagen tienen un valor de brillo igual a "0". Entonces el algoritmo comienza con un proceso de búsqueda general en toda la imagen, es decir inicia con un barrido de toda la imagen tratando de encontrar los píxeles pertenecientes al objeto (los que tiene el valor de "255"). Una vez encontrado un píxel del objeto se procede a buscar más de estos en su vecindad 8, pues si se trata de un objeto real sus píxeles se encuentran juntos como se puede observar en la figura 3.73. En la figura 3.75 se puede ver una ilustración de lo que se conoce como Vecindad 8 de un píxel. En el orden en el que están numerados se pueden analizar los píxeles para comprobar si estos también tienen valor "255" de brillo y por consiguiente saber si forman parte del objeto a obtener su centro de masa.

Figura 3.75. Vecindad 8

Las coordenadas de los píxeles detectados como miembros son introducidos en dos pilas respectivas a coordenadas en X y en Y para la posterior verificación de sus vecinos. Esto representa una tarea recursiva en donde se verificará cada una de las vecindades de los píxeles guardados en las pilas de direcciones. El algoritmo llega a su fin cuando todas las direcciones ingresadas en las pilas han sido verificadas y ya no se ingresan nuevas direcciones en ella, lo anterior es debido a que los píxeles que conforman al objeto han sido registrados por completo. Ahora bien, de manera simultánea al registro de los píxeles se realiza el sumatorio de las coordenadas en X y en Y de manera separada. Cuando se detecta un píxel miembro se ingresa el valor de sus coordenadas tanto en las pilas como en las variables que albergan a los sumatorios y se lleva un conteo de los píxeles registrados (cuya cuenta final determina el área), de esta forma, al terminar de recorrer todos los píxeles del objeto se divide el sumatorio entre el número total de píxeles que conforman a las huella del objeto, lo cual no es otra cosa que la obtención del promedio de todas las direcciones.

Es importante mencionar que una vez añadidas las direcciones de un píxel perteneciente a un objeto de interés este es pintado de un valor diferente al de la imagen binarizada con el fin de no detectar más de una vez el mismo cuerpo. Y de esta manera se aprovecha para segmentar la imagen por regiones, es decir cada animal encontrado tendrá un tono de gris diferente. Una vez terminada el recorrido de un objeto completo, se almacenan en un arreglo de registros: las coordenadas del centro de masa de la región, el área de la misma, y el tono de gris asignado a ella. Acto seguido, se incrementa el contador de regiones y se continúa con el escaneo general de la imagen en búsqueda de otros objetos a fin de que estos sean registrados totalmente sin excepción de alguno.

Finalmente, con el promedio de las direcciones tenemos las coordenadas del centro de masa del animal analizado. En la figura 3.76 se muestra el resultado del cálculo de los centros de masa para la jaiba y tres camarones, los centros de masa han sido pintados en la imagen de color rojo para la jaiba y de color verde para los camarones.

Figura 3.76. Centros de masa para la jaiba y los camarones

Es importante de mencionar que esta obtención del centro de masa es por vía matemática (promedio de coordenadas) por lo que este método determina el centro respecto a la disposición de las quelas (y demás extremidades) de la jaiba, ya que si estas están tendidas hacia un lado en particular el centro de masa se desplazara hacia ese lado, lo cual contrasta con la técnica de determinación manual de este parámetro. En el método manual el centro de masa es establecido sin realizar cálculos con coordenadas, en el que se fija el centro de masa de manera arbitraria por la persona que realiza el análisis, después de una simple inspección visual del objeto en la imagen y suponiendo que el centro se localiza a la mitad del eje de simetría transversal, pero esto solo a nivel de aproximaciones pues nada se calcula de manera exacta.

Esta diferencia en los métodos para calcular la localización de los centros de masa, ocasiona que para una misma imagen analizada por los dos métodos, no sea posible coincidir en las coordenadas de este parámetro, lo que conllevará a un margen de discrepancia en el cálculo de los demás parámetros (solicitados por el usuario para el futuro análisis estadístico) que utilizan como dato de entrada para su obtención la ubicación del centro de masa.

3.2.4.2 Cálculo de los vectores de posición de los animales

Una vez segmentada la imagen en regiones y con la obtención de los centros de masa, se pasa a la etapa de discriminación por área, debido a que son animales en estudio se tiene toda la

información relacionada con ellos, se toman los datos estadísticos de sus respectivas áreas y se relacionan con los tamaños de los camarones y las jaibas.

Los tamaños para los camarones están dentro de un rango de 30 hasta 150 píxeles y para las jaibas por encima de los 400 píxeles, todo lo que esté fuera de estos rangos será eliminados de los registros, de esa manera se discriminan los datos y se puede eliminar ruido asociado a la imagen.

Una vez que la imagen está libre de falsas regiones se procede al cálculo de los vectores de posición de los animales en estudio.

3.2.4.2.1 Obtención del norte del camarón

Esta es una de las etapas más importantes del software para la automatización de la adquisición de datos de las trayectorias de escape, ya que el análisis de comportamiento realizado por los investigadores biólogo-marinos está más dirigido hacia el camarón que a la jaiba. Un dato importante es la línea de vista del camarón antes de ser atacado por su predador natural y así poder generar hipótesis acerca de los medios de los que se vale el camarón para escapar exitosamente de un ataque. Uno de los aspectos visuales que tornaban en un reto esta tarea fue el constante cambio en la forma de la silueta de camarón, lo cual anulaba la opción de guiarnos en la simetría de su cuerpo para determinar donde se localizaba su norte; aunado a esto, la dispersión causada por la iluminación en ciertos momentos hacia disminuir grandemente el área de la silueta del camarón terminando en una modificación importante en su forma.

El algoritmo que resultó más sencillo de implementar y que presentó un comportamiento más robusto consistía en dos pasos principales: El primero consiste en evaluar las coordenadas de cada uno de los píxeles pertenecientes al cuerpo del camarón en cuestión de su distancia al centro de masa del mismo. Mediante la ecuación para el cálculo de las distancia entre dos puntos se buscaba cual era el píxel del cuerpo del camarón que se encontraba a mayor distancia del centro de masa.

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Una vez escaneado todo el área del animal, se guardaban las coordenadas del píxel más lejano en una variable para su posterior uso. A continuación se procedía a considerar como punto de partida el píxel hallado anteriormente y se iniciaba la búsqueda de un píxel que se encontrara más alejado a éste (el píxel más lejano al centro de masa). En la figura 3.77 se muestra una ilustración de los puntos que salen seleccionados usando este algoritmo.

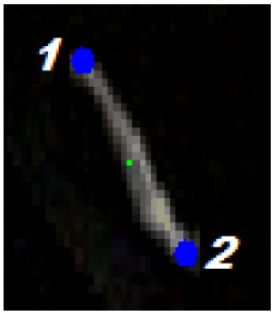


Figura 3.77. Detección del norte (2) y sur (1) del camarón

Los puntos obtenidos con el algoritmo han sido marcados con puntos azules y con un número según su orden de obtención. El principio de funcionamiento de este algoritmo radica en que uno de los patrones no cambiantes en la apariencia del camarón es el de presentar un grosor mayor en la parte de su cuerpo más cercana a su cabeza (norte), con esto aseguramos que abarcará un mayor número de píxeles en su parte superior que en la inferior (sur). Gracias a ello, estamos seguros de que el centro de masa se localizará en una posición más tendida hacia el norte, y no justo en el medio del eje transversal del cuerpo, ya que al promediar las coordenadas el mayor número de píxeles cercanos al norte "jalaran" el norte hacia ese lado en particular.

Por consiguiente, cuando se busca el píxel más lejano al centro de masa, éste se hallara en la cola del camarón (punto número 1 en la figura 3.77); sin embargo, lo que se busca es el norte y no el sur del camarón. Así, se procede a buscar el píxel más lejano al sur, tratándose de la cabeza del camarón (punto número 2 en la figura 3.77), mediante la ecuación de la distancia entre dos puntos buscando el valor máximo entre todos los píxeles de la silueta analizada.

En la figura 3.78 se muestra la forma que dibuja la línea de vista del camarón, y se indica la posición del mismo con un cuadrado de tamaño pequeño alrededor de él y uno de mayor tamaño alrededor de la jaiba, con el fin de que el usuario puede cerciorarse de que el software está operando de manera correcta y de que está reconociendo y discerniendo entre las dos clases diferentes de animales.

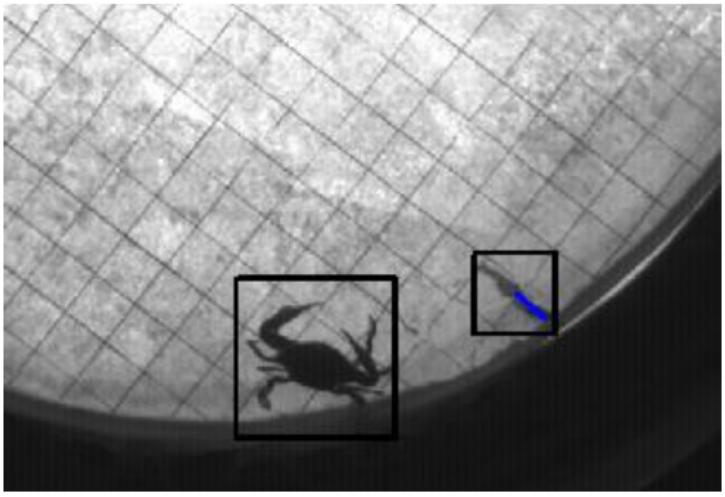


Figura 3.78. Indicadores de reconocimiento de los animales

Los datos adquiridos mediante el procesamiento de imágenes han sido los centros de masa de ambos animales y el norte del camarón. Con base a estos datos se realizan los cálculos de los demás parámetros requeridos por los investigadores biólogo-marinos.

El centro de masa y el norte del camarón sirven para calcular los parámetros de las trayectorias de escape del camarón, los cuales son:

- Distancia de reacción.
- Angulo de ataque.
- Distancia de escape.
- Angulo de escape.

Para calcular los parámetros es necesario contar con la información obtenida de las imágenes en dos tiempos, la primera imagen corresponde al tiempo inicial del ataque (Inicio del tail-flip), definido como el instante inmediatamente antes de que el camarón comience a girar el cuerpo hacia un lado y todavía se encuentre con el dorso hacia arriba; y la segunda imagen corresponde al tiempo final (Fin del tail-flip), definido como el instante inmediatamente después de que el camarón ha terminado de realizar el primer tail-flip y se encuentra de nuevo con el dorso hacia arriba.

A continuación se explican a detalle cada uno de estos parámetros y se propone un método para que el ordenador principal los calcule.

La ***distancia de reacción*** del camarón, corresponde a la distancia que hay entre el centro de masa del camarón y el centro de masa de la jaiba en el tiempo inicial. [Arnott, 1998, 1999]

La ***distancia de escape***, que es la distancia entre el centro de masa del camarón en el tiempo inicial y en el centro de masa del camarón en el tiempo final.

El ***ángulo de ataque*** (cuya magnitud máxima será de 180º sin importar el sentido), a partir del ángulo que se forma entre los vectores:

Norte del camarón — Centro de masa del camarón;
y Centro de masa del camarón — Centro de masa de la jaiba.

El ***ángulo de escape*** se calcula en el método clásico (manual), a partir de la imagen sobrepuesta del tiempo inicial y del tiempo final (tomando en consideración para su registro el ángulo agudo), a partir del ángulo que se forma entre los vectores:

Norte del camarón — Centro de masa del camarón en el tiempo inicial;
y Centro de masa del camarón en Ti — Centro de masa del camarón en tiempo final.

En el método computacional propuesto, el ángulo de escape no sobrepone las imágenes para su medición, pues esto se vuelve innecesario una vez que ya se ha obtenido el centro de masa del camarón en tiempo inicial. Este dato se almacena en una variable y es utilizado como entrada en las ecuaciones para la estimación de este parámetro indispensable en el escape.

Para la conversión de las distancias en píxeles a centímetros se usó la cuadricula pintada en la arena, donde cada cuadro es de 5cm por lado y cada lado tiene una longitud de 36 píxeles. Con esto llegamos a la constante de conversión de ***1 píxel = 0.138888 centímetro***, lista para ser usada para transformar todas las medidas obtenidas en píxeles a una unidad real, gracias a la calibración de la cámara.

Para el cálculo de los ángulos comprendidos entre los vectores antes mencionados se utilizó el ***producto escalar de vectores***. Definido de la siguiente forma cuando se trabaja en un campo vectorial de dos dimensiones ($\Re^2$):

$$\vec{A} \cdot \vec{B} = \left|\vec{A}\right|\left|\vec{B}\right| \cos\theta$$

$$\vec{A} \cdot \vec{B} = a_i b_i + a_j b_j$$

Antes de aplicar las ecuaciones anteriores se procedió a obtener los vectores unitarios formados con los centros de masa y con el norte del camarón en la forma antes mencionada. Para lo cual solo se necesitaba encontrar el módulo de los vectores por el método común con el Teorema de Pitágoras. Utilizando vectores unitarios la primera ecuación del producto punto se reduce únicamente a:

$$\vec{A} \cdot \vec{B} = \cos\theta$$

Como se conocen totalmente las componentes de estos vectores, se obtiene el resultado del producto escalar con la segunda ecuación de manera directa. Y finalmente el ángulo comprendido entre los dos vectores (en este caso A y B) se obtiene con la ecuación ya despejada:

$$\theta = \cos^{-1}\left(\vec{A} \cdot \vec{B}\right)$$

En la figura 3.79 se muestra una ilustración de la obtención del ángulo comprendido entre dos vectores aunado con la ecuación para su cálculo.

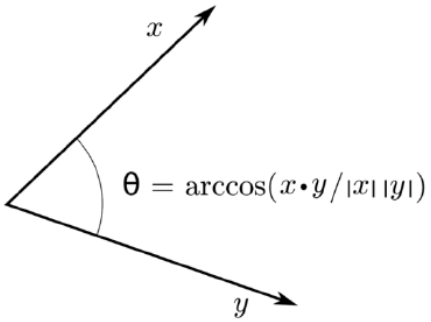


Figura 3.79. Angulo entre dos vectores

Con esto quedan descritos los métodos por los cuales se extrajeron de manera automática los datos útiles para el análisis de las trayectorias de escape incluyendo las ecuaciones matemáticas usadas.

3.2.4.2.2 Obtención del vector de posición de la jaiba con el algoritmo de SUSAN

A continuación se presenta la propuesta de un método que emplea un detector de bordes sumamente común para la determinación de la línea de vista de la jaiba. El procedimiento consiste en utilizar como entrada la imagen binaria que es resultado de la segmentación (tomando en cuenta las dos etapas de umbralización y la de difuminación entre ellas), a una nueva etapa de filtrado para detección de bordes conocido como operador SUSAN.

La imagen de entrada al detector de esquinas SUSAN es la imagen binarizada de la jaiba. Debido a que ya se tiene localizado el centro de masa de este objeto, ya no es necesario el re-escaneo de toda la imagen haciendo posible así el ahorro de tiempo de procesamiento.

Haciendo uso de los métodos antes mencionados para la detección más selectiva de las esquinas en una imagen, se filtra la imagen de los experimentos biológicos dejando únicamente las esquinas de la imagen más acentuadas con la intención de que sean detectadas las puntas de las quelas de las jaibas. Con el contexto previo de que al atacar las jaibas, estas proceden

extendiendo las quelas para así atrapar a su presa (el camarón), se considera la línea de vista del predador como la recta que une el centro de masa de este animal con el punto medio de la línea comprendida entre las esquinas detectadas en las puntas de las quelas.

El filtrado de SUSAN proporciona una imagen con bordes adicionales a las puntas de las quelas. Para desechar aquellos que no tienen importancia en el algoritmo se procede a buscar los píxeles marcados como esquinas que se encuentren más alejados del centro de masa. Se considera que los puntos esquina más alejados se encontraran en las puntas de las extremidades de la jaiba, ya que en vísperas de atacar este animal las tiene extendidas de manera importante. En la figura 3.80 se presenta la ilustración de los puntos mencionados en este algoritmo, es decir, las esquinas en las respectivas quelas de la jaiba, la recta que las une, el punto medio de esta recta y el centro de masa.

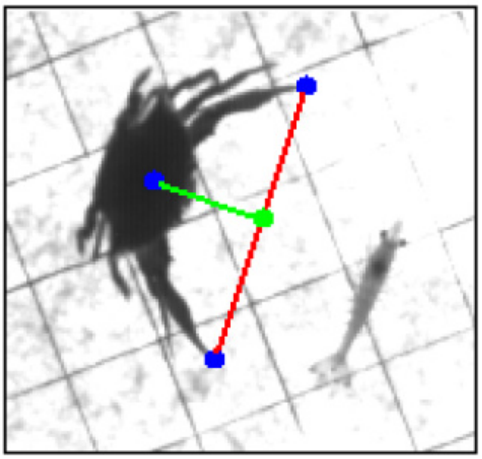


Figura 3.80. Ilustración del algoritmo usado para la detección del norte de la jaiba.

El método es similar al usado para hallar el norte del camarón, la única variante radica en que el píxel base para buscar a los más alejados es el centro de masa. Es importante mencionar que el detector de esquinas SUSAN no marcaba como píxel esquina a uno solo de ellos, en otras palabras, una esquina estaba marcado con aproximadamente 3 píxeles adyacentes o sumamente cercanos, por lo tanto, se tuvo que añadir otro candado a la búsqueda de las puntas de las quelas. El candado consistía en que la segundo esquina a encontrar no solo debía encontrase lejos del centro de masa sino también lejano a la primera quela encontrada. Con lo anterior se garantizaba que al final se quedarían registradas las coordenadas de las puntas de las quelas extendidas al momento del ataque. En la figura 3.81 se presenta la imagen con la identificación del norte de la jaiba hecha por el software.

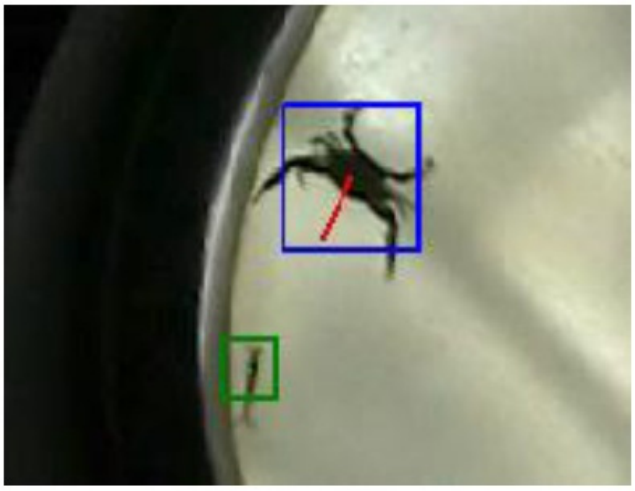


Figura 3.81. Indicaciones hechas por el software para detección de la línea de vista de la jaiba

La principal desventaja de este algoritmo es su poca robustez y estabilidad cuando se trata de monitorear experimentos completos. Esta debilidad radica en que el despliegue de las quelas no siempre se hace de manera simétrica y se presentan casos en donde la jaiba solo extiende una de ellas y la otra la deja plegada. En esta situación, la línea de vista se verá desviada debido a la falta de simetría en la posición de las quelas, sin embargo, con los datos extraídos y almacenados, bajo conocimiento que la línea de vista será detectada de manera confiable justo al momento del ataque, el método representa una herramienta muy útil para el análisis de comportamiento de la jaiba.

Obtención de contornos

Una vez analizado cómo funciona el operador de SUSAN (apartado 2.6) se sacaron los requerimientos para su implementación en el DSP, tomando en cuenta que el sistema ya implementado proporciona una imagen segmentada en regiones como se muestra en la figura 3.82.

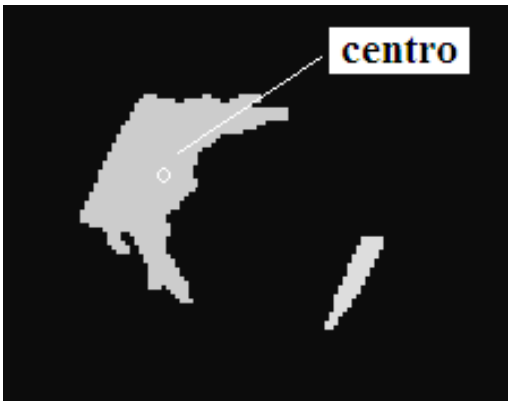


Figura 3.82. Imagen segmentada en regiones

El sistema además entrega algunos datos que puede requerirse en el proceso de obtención de los puntos máximos. Estos datos son el área y las coordenadas del centro de masa para cada uno de los animales en estudio.

El resultado obtenido después de ejecutar el algoritmo de SUSAN es una tabla de datos donde están almacenadas las coordenadas del contorno, tomando en cuenta que no son todas

ya que con el umbral geométrico se puede discriminar las muestras significativas. Si representamos estos datos en una imagen resultaría lo mostrado en la figura 3.83.

Figura 3.83. Imagen resultado del detector de bordes SUSAN

Tomando los puntos que delimitan el borde de la jaiba procedemos a encontrar los puntos más lejanos al centro de masa que se ha calculado con anterioridad utilizando la fórmula de la distancia entre dos puntos.

Esta fórmula se aplica a todos los puntos del contorno de la jaiba, hasta que un caso es lo que se representa en la figura 3.84.

Figura 3.84. Distancia de un punto al centro de masa

Así mismo, le agrego una protección para evitar que los puntos estén muy cerca y así evitar falsos positivos en las quelas de la jaiba.

Teniendo los dos puntos más lejanos se encuentra el punto medio de la recta imaginaria que se encuentra entre ellos, línea roja que se muestra en la figura 3.85.

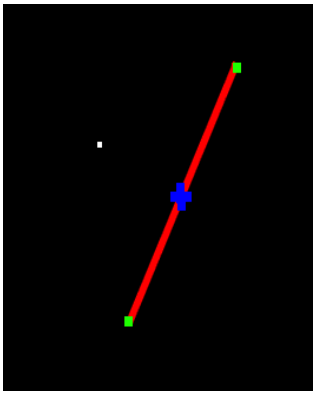


Figura 3.85. Centro de masa (Blanco), Puntos Máximos (Verde), Segmento (Rojo), Punto medio (azul)

En la figura 3.85 se muestra el resultado obtenido del punto medio, una x de color azul, así como el centro de masa un punto blanco. Al tener el punto medio y las coordenadas del centro de masa, estos conforman parte de una recta imaginaria, la cual encontraremos por medio de los dos puntos y la pendiente.

Procedemos a representarla en el intervalo que limitan nuestros puntos. En la figura 3.86 se muestra una recta de color blanco resultante con las coordenadas del centro de masa y el punto medio que es lo que representa el vector de posición de la jaiba.

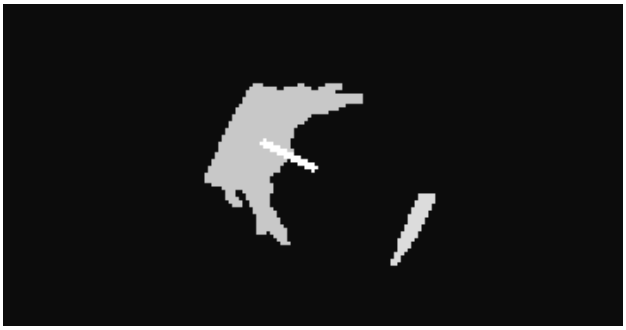


Figura 3.86. Imagen obtenida en blanco se puede observar el vector de la jaiba

3.2.4.2.3 Obtención del vector de posición de la jaiba con el algoritmo propio

El algoritmo propuesto utiliza los resultados de la segmentación (figura 3.82), en primer lugar las coordenadas del centro de masa, que es utilizado como punto de inicio, en este punto se recupera el valor del píxel que será utilizado como semilla ya que es el valor que contiene toda la región de la jaiba.

El siguiente paso será encontrar los bordes de esta región. Debido a que las regiones están bien definidas, por ser animales en estudio, no existe posibilidad de encontrar huecos dentro de las regiones (probado con más de 100 pruebas), por lo que el algoritmo buscará el borde que está por arriba del centro de masa, para posteriormente realizar el recorrido a través de él en sentido anti-horario e ir comparando la distancias entre el centro de masa y los puntos del borde para obtener la mayor distancia. Como lo muestra la figura 3.87.

Figura 3.87. Imagen representativa del recorrido del algoritmo para encontrar el primer punto más lejano

El recorrido para encontrar el punto máximo no terminará hasta que llegue nuevamente al punto del borde donde se inició. De esta manera se tiene el primer punto más lejano.

Es necesario encontrar el otro punto más lejano. Para evitar que algún punto cercano al más lejano interviniera, se discriminó parte de la imagen, para esto se obtiene el perímetro de la región durante el primer recorrido.

Iniciamos el proceso en el punto más lejano, efectuando el recorrido del borde sin realizar cálculo de distancias, hasta que llega a un cuarto del recorrido, que es donde comienza nuevamente el cálculo de las distancias, para encontrar el segundo punto más lejano y terminará un cuarto antes de que llegue al punto donde inició. Esto es siempre para discriminar cualquier punto cercano al primer punto máximo. Esto se muestra en la figura 3.88.

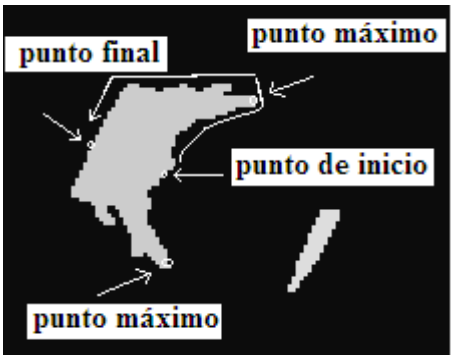


Figura 3.88. Imagen que representa el recorrido para encontrar el segundo punto más lejano

Una vez obtenido los dos puntos más lejanos procedemos al cálculo del vector de posición como se expuso con el método de SUSAN.

CAPITULO 4 – TEST Y RESULTADOS ALCANZADOS

4. Introducción

Este capítulo expone una comparación de los resultados obtenidos entre las diferentes implementaciones del hardware para la recuperación de los datos con dos fabricantes de FPGAS. Se analiza el uso del convertidor de código RGB para las imágenes obtenidas en los procesamientos de resta y umbralización. Se exponen los resultados obtenidos en las diferentes alternativas de implementación de los algoritmos de resta y umbralización, en el microprocesador Microblaze, Máquinas de estado y el procesador de señales digitales. Se muestran los resultados de las comparativas de las implementaciones realizadas para el filtro de pasa bajo en el microprocesador Microblaze y en el procesador de señales digitales. Por último se presentan los diferentes resultados de las implementaciones para la obtención de los puntos más lejanos del camarón y la jaiba en los algoritmos de Susan y el método propuesto.

4.1 Comparativa de la implementación del hardware para la recuperación de los datos

Los avances tecnológicos posibilitan mayores velocidades en la transmisión de datos entre los componentes, y, ofrecen diversas soluciones a los problemas de transmisión de información, por lo que es necesario comparar estas soluciones para decidir la más adecuada a nuestro sistema. Con este fin, en esta Tesis se busca realizar la comparación entre las herramientas y características proporcionadas por los dos principales fabricantes de FPGAs en el mercado (Xillinx y Altera), mediante la implementación del receptor deserializador LVDS propuesto en FPGAs con características similares de cada fabricante. En específico el sistema se implementó en una Virtex 4 LX60 FF668-10 de Xillinx y en una Stratix II EPE2S60F484C3de Altera. Ambas cuentan con características similares de las que podemos resaltar para la Virtex 4, 59.904 celdas lógicas, 26.624 slices y 8 DCMs y para la Stratix II 48.352 celdas lógicas 24.176 módulos de lógica adaptable y 8 PLLs (para mayor información ver Tablas 3.3 y 3.6).

La creciente necesidad de utilizar frecuencias muy elevadas en los sistemas nos lleva a emplear procedimientos de depuración cada vez más exigentes. Un procedimiento que ha tenido excelentes resultados dentro del campo de los FPGAs es el emplazamiento manual y el uso de componentes reductores de skew tales como PLLs y DLLs. Los fabricantes Xilinx y Altera proveen de recursos suficientes para el mejoramiento del sistema en los dos aspectos, tanto en emplazamiento como en la eliminación del skew. El sistema es implementado en dos versiones para los dos fabricantes, una versión sin y otra con emplazamiento manual, ambas con los elementos de eliminación de skew.

4.1.1. Resultados obtenidos de la comparativa

Existen diferencias sustanciales entre los fabricantes Xilinx y Altera. El modulo PLL de Altera tiene una mayor versatilidad que su contraparte, el DCM de Xilinx, debido a sus 6 salidas casi independientes, es importante mencionar que el DCM tiene un mayor número de salidas, estas se encuentran fijas a un mismo desfase y un mismo ciclo de trabajo; esta configuración puede ser una limitante en algunas aplicaciones. La gestión de diversos estándares en los pines de salida y de entrada resulta más fácil de efectuar mediante el Pin Planner de la herramienta Quartus II y podemos eliminar de nuestro código los llamados drivers que son necesarios dentro del entorno Xilinx. La herramienta Plan Ahead es una excelente aplicación que permite realizar diseños integrales e implementar un depuración exhaustiva a nuestro sistema en el plano físico, de forma gráfica podemos realizar emplazamientos manuales, aplicar restricciones a nuestros componentes y además realizar una verificación total dentro de la misma aplicación, su contraparte de Altera, el Chip Planner y el Assigment Editor nos proveen de varias herramientas

de ayuda sin llegar a tener la versatilidad del Plan Ahead. La depuración en general dentro de la herramienta Xilinx resulta de cierta forma más fácil por lo que las velocidades alcanzadas son mayores que en la herramienta Altera. Esto ayuda a disminuir el tiempo de diseño y además permite contemplar mejoras futuras en el sistema. En la tabla 4.1 se muestra el resumen de las implementaciones con los dos fabricantes. En la figura 4.1 se muestra la gráfica del desempeño en frecuencia por los dos fabricantes de FPGA's. Para una implementación sin emplazamiento manual la herramienta de Altera funcionó mejor ya que alcanzó hasta una frecuencia de 30.86 MHz. Por el contrario al realizar la implementación con emplazamiento manual la herramienta de Xilinx lo realizó mejor y alcanzo una frecuencia de 85 MHz

Implementación	Xilinx	Altera
Sin emplazamiento manual	7.14 MHz.	30.86 MHz.
Con emplazamiento manual	85.00 MHz.	70.32 MHz.

Tabla 4.1. Resumen de resultados en Frecuencia en las diferentes implementaciones de los receptores.

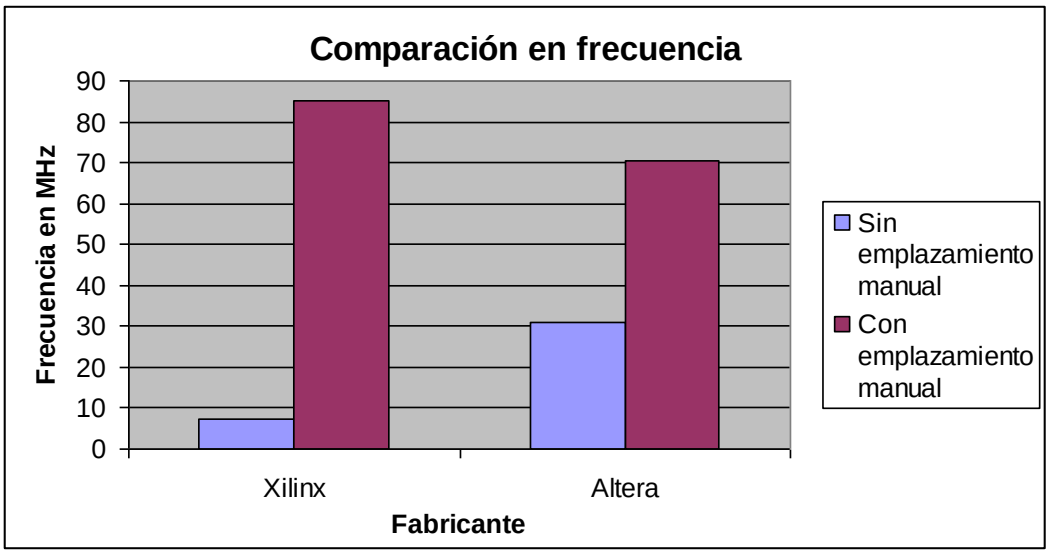


Figura4.1. Gráfica comparativa del desempeño en frecuencia en la implementación de los receptores con los fabricantes de Xilinx y Altera

Otra comparación interesante es el área que utilizó cada implementación; en la tabla 4.2 y en la figura 4.2 se muestra los recursos utilizados en las implementaciones por los dos fabricantes en estudio, en ella se puede apreciar que con respecto a flip-flops y LUT's no existió una diferencia significativa; esto se debió a que el diseño era prácticamente el mismo para los dos fabricantes. Los cambios observados demuestran que se requirieron pequeñas adecuaciones en la máquina de estados que son causadas por las diferencias existentes entre el DCM (Xilinx) y el PLL (Altera). Lo que hay que resaltar es el uso del DCM y PLL ya que como se había mencionado, el DCM de Xilinx estaba limitado a una sola frecuencia de enganche mientras que el PLL de Altera es más independiente por lo que se puede optimizar su uso, prueba de ello es que con uno bastó para todo el diseño.

Resultados de área	Xilinx		Altera	
	Sin emplazamiento manual	Con emplazamiento manual	Sin emplazamiento manual	Con emplazamiento manual
Flip-flops /Registros	228	229	212	213
LUT's/ALUT's	504	505	541	533
DCM / PLL	2	2	1	1

Tabla 4.2. Resumen de resultados en área en las diferentes implementaciones de los receptores

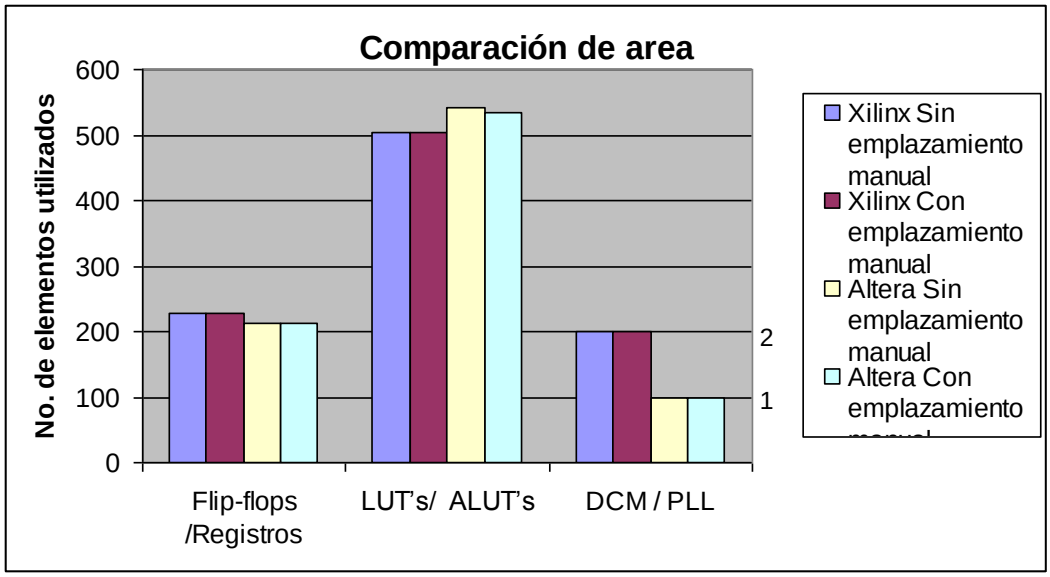


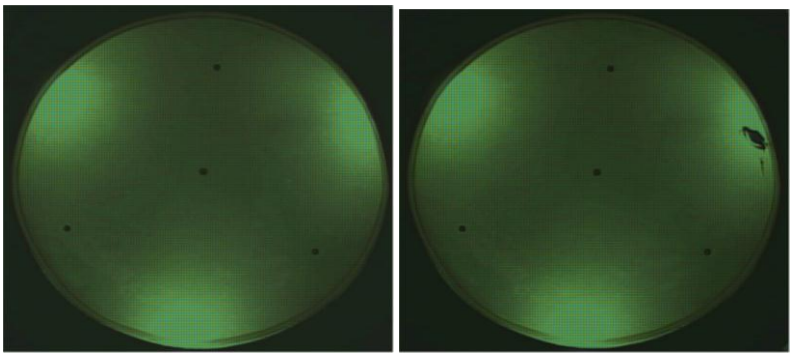
Figura 4.2. Gráfica comparativa del desempeño en área en la implementación de los receptores con los fabricantes de Xilinx y Altera

Con estos resultados y con el análisis de todas las herramientas que se tiene con cada uno de los fabricantes, la tarea de elegir con qué familia trabajar es un proceso difícil ya que cada uno de los fabricantes tiene sus virtudes. La elección estará determinada con la necesidad del proyecto, nosotros optamos trabajar con Xilinx.

4.2 Determinación del uso del código RGB para las imágenes

Después de implementar el convertidor de Bayer a RGB, se planteó la posibilidad de realizar los procesos directamente en formato Bayer, sin aplicar previamente la conversión a RGB. Esto se probó en el primer procesamiento que se requería realizar: la resta de imágenes (ver 2.4.1.2.5).

Se tomaron dos fotos en formato Bayer, la primera sin animales (fondo) y la segunda con los animales, a continuación se efectuó la resta de las dos imágenes, en esta misma función se umbralizó para resaltar las figuras de los animales. En las figuras 4.3 se pueden apreciar la imagen de fondo (a) y la imagen con animales (b).



(a) **(b)**

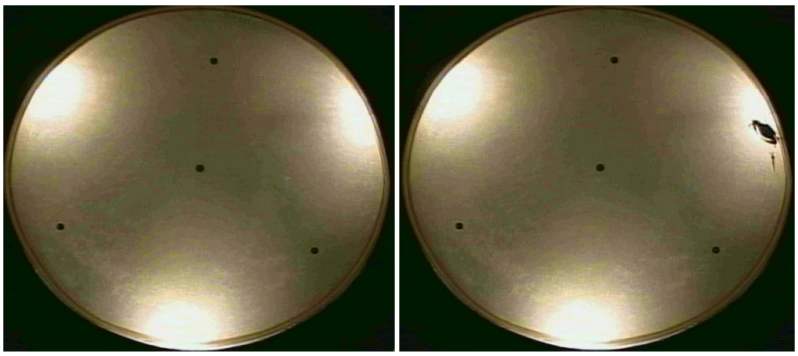
Figura 4.3. (a) Imagen de fondo en formato Bayer, (b) imagen con animales en formato Bayer

Con el resultado de la resta de las imágenes se logró separar a los dos animales obteniéndose la imagen mostrada en la figura 4.4. En la parte (a) de esta figura se recorta la imagen donde están los animales para poderlos apreciar mejor, y en la parte (b) se presenta el resultado obtenido de la resta y de la umbralización.

(a) (b)

Figura 4.4. (a) Ampliación de los animales en estudio, (b) resultados de la resta de imágenes

En el siguiente caso se realizó la conversión de Bayer a RGB para la imagen de fondo y para la imagen donde estaban los animales; las imágenes obtenidas se muestran en la figura 4.5.



(a) **(b)**

Figura 4.5 (a) figura de fondo en formato RGB, (b) figura donde estaban los animales en formato RGB

Una vez convertidas las imágenes se realizo la resta entre ellas y el resultado se umbralizó para resaltar a los animales, esto se presenta en la figura 4.6, además de recortar y agrandar a los animales para una mejor comparación.

(a) (b)

Figura 4.6. (a) Imagen recortada y ampliada en formato RGB, (b) resultado de la resta y umbralización

Observando ambos resultados se aprecian muy poca diferencia, sobre todo para la jaiba, el camarón se ve un poco más afectado en el resultado obtenido, ya que su imagen está casi dividida en el resultado de la resta RGB, esto se puede apreciar en la figura 4.7.

(a) (b)

Figura 4.7 (a) Resultado de la resta en formato Bayer, (b) Resultado de la resta en formato RGB

Por lo que se puede concluir que resultó mejor trabajar las imágenes directamente en formato Bayer que con las imágenes en formato RGB. Esto se debe a que en las operaciones que se realizan en el convertidor van despreciando parte de los resultados por redondeo. Con esto se comprueba que no es necesario realizar la conversión de imágenes en formato Bayer a RGB cuando el primer procesamiento es una resta, con lo que se puede ahorrar memoria de programa, y memoria de datos debido a que la imagen en RGB ocupa el triple de espacio que la imagen en formato Bayer, además existe un ahorro de tiempo por no requerirse la ejecución de las conversiones de imágenes. Lo anterior ratifica los trabajos de Chen [2006].

4.3 Comparativa entre las diferentes alternativas de implementación de los algoritmos de Resta y umbralización

Al hablar de evaluación de desempeño lo que se desea lograr es una estimación cuantitativa y cualitativa del grado de eficacia de una implementación en su tarea realizada.

Los principales puntos que se considerarán para realizar la evaluación del desempeño son:

- Descripción de las funciones realizadas.
- Medición de los resultados.
- Capacidad de desarrollo futuro.

La descripción de las funciones realizadas, se ha expuesto de manera general al inicio de cada sección de implementación. Por consiguiente, en esta sección presentaré los resultados obtenidos en las diferentes versiones de implementación hechas en la investigación, como son los programas realizados en el EDK, en el ISE foundation y en el DSP. También, analizaré las capacidades de desarrollo futuro. Todo esto con la finalidad de evaluar el desempeño de los diversos sistemas considerados.

4.3.1. Implementación con el microprocesador Microblaze

4.3.1.1. Mediciones de tiempo de procesamiento

Una vez descargado el proyecto en la tarjeta ML 401 que contenía el microprocesador "Microblaze" se procedió a medir los tiempos de procesamiento, para las funciones de resta y umbralización para imágenes de 102X 150 píxeles.

Para encontrar los tiempos de procesamiento se le agregó al programa principal una bandera que indica cuando inicia el procesamiento (flanco de subida) y cuando termina (flanco de bajada). Para poder observar los tiempos se requirió de un osciloscopio donde se colocó una punta de prueba a una salida de la tarjeta y se procedió con la medición. Los resultados se observan en la figura 4.8.

Figura 4.8. Tiempo de procesamiento

La duración del ancho de pulso positivo que se observa en la figura 4.8 es el tiempo de procesamiento total, y con una herramienta del osciloscopio se logró medir el periodo del pulso positivo y se obtuvo el tiempo exacto, en la parte inferior izquierda de la pantalla se puede encontrar las leyendas +Width(1)=190.0ms.

4.3.1.2. Recursos Utilizados en la implementación

Un FPGA es un dispositivo semiconductor que contiene bloques lógicos, por tanto, una variable interesante para medir es la cantidad de recursos que ocupa un diseño implementado en función de los bloques lógicos que utiliza. En la tabla 4.3 se puede apreciar los recursos utilizados para esta implementación.

Device Utilization Summary			
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slice Flip Flops	1,217	21,504	5%
Number of 4 input LUTs	1,639	21,504	7%
Logic Distribution			
Number of occupied Slices	1,353	10,752	12%
Number of Slices containing only related logic	1,353	1,353	100%
Number of Slices containing unrelated logic	0	1,353	0%
Total Number 4 input LUTs	1,977	21,504	9%
Number used as logic	1,639		
Number used as a route-thru	15		
Number used for Dual Port RAMs	256		
Number used as Shift registers	67		
Number of bonded IOBs	43	448	9%
Number of BUFG/BUFGCTRLs	3	32	9%
Number used as BUFGs	3		
Number used as BUFGCTRLs	0		
Number of FIFO16/RAMB16s	27	72	37%
Number used as FIFO16s	0		
Number used as RAMB16s	27		
Number of DSP48s	3	48	6%
Number of DCM_ADVs	1	8	12%
Number of RPM macros	1		
Total equivalent gate count for design	41,811		
Additional JTAG gate count for IOBs	2,064		

Tabla 4.3. Se presenta los recursos que utiliza el FPGA

El porcentaje de bloques de flip flops que contiene el FPGA Virtex IV se utilizó en un 5%, tomando en cuenta el microprocesador Microblaze y el resto del sistema electrónico.

4.3.1.3. Capacidad de desarrollo futuro en el EDK(Embedded Development Kit)

La ventaja de desarrollar los algoritmos en el EDK es que se permite el uso del lenguaje C para programar el microprocesador embebido, no obstante tiene unas instrucciones reservadas propias del fabricante Xilinx.

La tarjeta de evaluación que se utilizó en este proyecto tiene un reloj maestro de 100MHz, se pueden mejorar los resultados de los algoritmos planteados utilizando una tarjeta de más prestaciones, que maneje un reloj de mayor frecuencia y como consecuencia se tendría un aumento en la velocidad de ejecución.

Otra manera en que se puede mejorar estos tiempos es utilizando procesadores de núcleo sólido como el Power PC; este tiene la ventaja de que puede manejar mejores velocidades y mejor eficiencia en ciclos de reloj por instrucción.

4.3.2. Implementación con el procesador diseñado en el Ise Foundation (Máquina de estados)

4.3.2.1. Mediciones de tiempo de procesamiento

Una vez realizado el sistema electrónico completo y ver su funcionalidad se procedió a medir el tiempo de procesamiento para la imagen analizada en el caso anterior.

Para encontrar los tiempos de procesamiento se utilizó el mismo procedimiento que el caso anterior de utilizar una bandera para indicar cuándo inicia y cuándo termina el procesamiento. De esta manera se puede saber el tiempo total al hacer la medición entre cada cambio de nivel de voltaje. Los resultados obtenidos se observan en la figura 4.9.

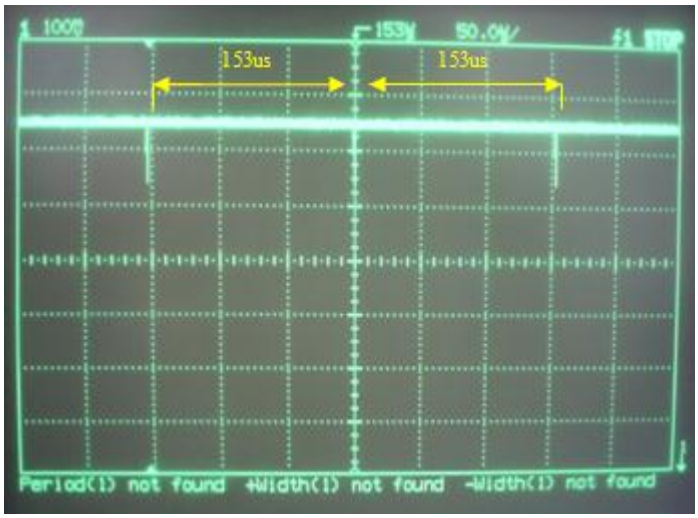


Figura 4.9. Tiempo de procesamiento del hardware

El tiempo medido es de 153µs por cada frame que se procesa y se indica en la figura 4.9 con las flechas.

También se procedió a medir los resultados mediante una herramienta dentro del ISE Foundation que genera un test bench; a través de él se puede simular el sistema electrónico completo.

Para facilidad de visualización solo se le añadieron los pines necesarios como se observa en la figura 4.10. El tiempo de procesamiento que presenta es de 153µs.

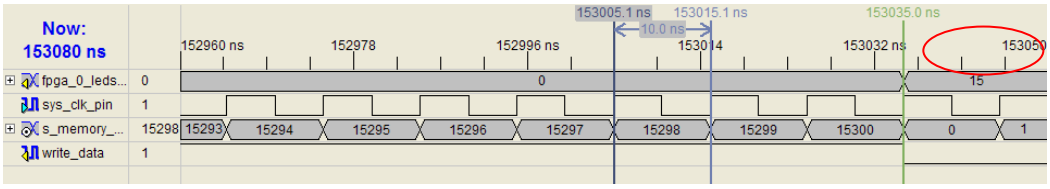


Figura 4.10. Resultados de la simulación

4.3.2.2. Recursos Utilizados en la implementación

En la tabla 4.4 se especifican los recursos utilizados para la implementación de la segmentación.

Device Utilization Summary			
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slice Flip Flops	94	21,504	1%
Number of 4 input LUTs	163	21,504	1%
Logic Distribution			
Number of occupied Slices	100	10,752	1%
Number of Slices containing only related logic	100	100	100%
Number of Slices containing unrelated logic	0	100	0%
Total Number 4 input LUTs	175	21,504	1%
Number used as logic	163		
Number used as a route-thru	12		
Number of bonded IOBs	35	448	7%
Number of BUFG/BUFGCTRLs	2	32	6%
Number used as BUFGs	2		
Number used as BUFGCTRLs	0		
Number of FIFO16/RAMB16s	19	72	26%
Number used as FIFO16s	0		
Number used as RAMB16s	19		
Total equivalent gate count for design	2,193		
Additional JTAG gate count for IOBs	1,680		

Tabla 4.4. Recursos que utiliza el FPGA

Del porcentaje de bloques de flip flops que contiene el FPGA Virtex IV se utilizó solo el 1%, tomando en cuenta que en esta implementación se encuentra contenida la máquina de estados que funciona como procesador y el resto del sistema electrónico necesario para realizar este procesamiento.

4.3.2.3. Capacidad de desarrollo futuro en la máquina de estados

Entre los principales beneficios que se pueden alcanzar al desarrollar los algoritmos de segmentación en el lenguaje de descripción de hardware (por sus siglas en ingles HDL) se puede mencionar la paralelización del procesamiento, con lo cual se alcanzaría mayores tasas de frames. Esto podría aumentar la eficiencia de instrucciones por ciclo de reloj.

Para esta implementación se utiliza un reloj maestro de 100MHz de la tarjeta de evaluación que se expone en el sistema EDK.

También se puede desarrollar un procesador con una arquitectura más adecuada para el procesamiento de imágenes Bayer.

Una desventaja de la implementación de algoritmos a tan bajo nivel es que se vuelve difícil su portabilidad, y requiere de un gran esfuerzo para realizar las adecuaciones necesarias para su uso en otros procesos.

4.3.3. Implementación con el procesador de señales digitales

4.3.3.1. Mediciones de tiempo de procesamiento

Para medir el tiempo de procesado y los recursos que ocupó el DSP se recurrió a una herramienta que trae el Code Composer Studio, el Analysis tool kit.

El recurso utilizado en el DSP se midió con respecto al espacio necesario en la memoria para almacenar el código generado por los algoritmos de resta y umbralización, siendo este de 156 bytes, que al procesar una imagen ejecutó un total de 459,754 instrucciones en ensamblador. El número de ciclos que se llevó el CPU para ejecutar este algoritmo fue de un total de 506258 ciclos, considerando que el procesador funciona a una frecuencia de 1GHz, el tiempo de ejecución estimado fue de 0.506 ms. La imagen que se procesó tenía las mismas dimensiones que los dos casos anteriores.

4.3.3.2. Capacidad de desarrollo futuro del DSP

Las ventajas de desarrollar los algoritmos de segmentación en el compilador del Code Composer Studio es que se pueden modificar de una manera sencilla y al tratarse de un lenguaje de programación en C++ hace más portable el código. También una persona sin muchos conocimientos de desarrollo de hardware podría desarrollar las implementaciones e inclusive mejorarlas ya que esta herramienta está orientada a software y solo se necesita conocimientos elementales de programación.

De igual manera el poder emigrar este código a otras herramientas de programación más avanzadas podría arrojar mejores resultados.

La aplicación se ejecutó en un DSP que tiene un procesador de 1 GHz, y sería factible mejorar el desempeño utilizando DSPs con mejores procesadores.

4.3.4. Comparación entre las implementaciones de segmentación

En esta sección se compararán los resultados obtenidos entre cada una de las implementaciones de los algoritmos de resta y umbralización.

4.3.4.1. Tasa de frames por segundo para las implementaciones que ejecutan los procesos de resta y umbralización

Las tasas de frames por segundo para cada implementación están presentadas en la figura 4-11, fueron calculadas en base a los resultados presentados en las secciones de medición de tiempo de procesamiento. Los resultados de implementación por área de ocupación dentro del FPGA y del DSP, de una forma relativa, quedan de la siguiente manera: un 12% del total de los slices para el Microblaze, un 1 % del total de slices para la máquina de estados y $1x10^{-6}$ **%** del total de localidades disponibles para programas. Comparando cada uno de los procesos, se puede ver de manera clara la gran diferencia entre la tasa de frames alcanzada por las maquinas de estado (hardware), con el DSP y el Microblaze (muy lento en comparación con los otros dos sistemas).

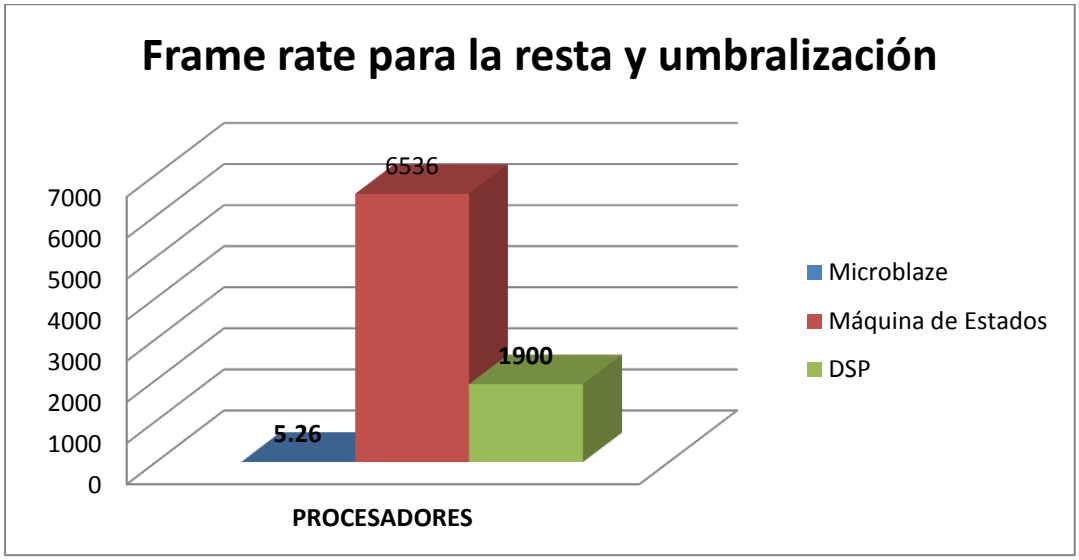


Figura 4.11. Gráfica de comparación de los FPS para la resta y umbralización

Resultados de área	Microblaze	Maquina de Estados	DSP
slices	1353	100	---
Direcciones de memoria en Bytes	---	---	156

Tabla 4.5. Resultados de ocupaciones en área

4.3.4.2. Síntesis de la evaluación del desempeño

En este apartado resumo las implementaciones creadas con varias herramientas de desarrollo para aplicaciones de procesamiento de imágenes, presentando una calificación de los resultados para posteriormente sacar conclusiones más claras.

En la tabla 4-6. se observa las evaluaciones obtenidas para cada procesador en cada rubro. Estas evaluaciones son meramente cualitativas y fueron tomadas con base a las nociones adquiridas durante el desarrollo, ejecución y conclusión de las implementaciones.

Procesadores	FPS	Eficiencia en consumo de recursos del FPGA	Capacidad de desarrollo futuro	Portabilidad de los Códigos.
Microblaze	★☆☆☆☆	★½☆☆☆	★★★★☆	★★☆☆☆
Hdl	★★★★★	★★★★★	★★★★★	½☆☆☆☆
DSP	★★★★☆	--	★★★★★	★★★★★

Tabla 4.6. Evaluaciones de los procesadores

Para poder elegir con que sistema se va a trabajar, se debe considerar el proyecto en su conjunto, ya que en ocasiones la velocidad de procesamiento puede ser una limitante. En este punto es posible recomendar una implementación en HDL, sin embargo es importante considerar el tiempo de implementación, ya que este puede ser muy largo para observar los resultados. En tanto con un sistema como el DSP y el EDK se puede implementar de una manera

más rápida con la que se obtiene resultados inmediatos y determinando con él la viabilidad del proyecto en menor tiempo.

Entre el DSP y EDK para el procesamiento de imágenes, el que tiene mejor desempeño es el DSP, esto se debe a dos razones primordiales, una es la arquitectura de su unidad de procesamiento en la que resalta sus 8 unidades aritméticas que pueden trabajar en paralelo. La segunda es su velocidad de procesamiento, que para el modelo utilizado fue de 1GHz. Estas dos características están por encima de los recursos que tiene el EDK.

4.4 Comparativa de implementación de un filtro pasa bajo

4.4.1. Implementación con el Microblaze

4.4.1.1. Mediciones de tiempo de procesamiento

En este apartado lo más relevante son los tiempos de ejecución de la rutina de difuminación (filtro pasa bajo).

Para medir el tiempo que tarda en ejecutarse la rutina de difuminación se utilizó un osciloscopio digital, tal y como se procedió en la rutina de resta y umbralización. La rutina de difuminación es difícil de medir aisladamente, por lo que la integré con la rutina de resta y umbralización, y procedí de la misma forma como en la rutina de resta y umbralización. El tiempo mostrado por el osciloscopio (figura 4.12) fue de 282 ms. Para calcular el tiempo empleado en difuminar, se realizó la resta del tiempo total menos el tiempo de resta y umbralización medido en el apartado anterior, es decir:

Tiempo de difuminación = 282ms-190ms= 92 ms

Figura 4.12. Tiempo de procesamiento de los dos procesos descritos

4.4.2. Implementación con el procesador digital de señales

4.4.2.1. Mediciones de tiempo de procesamiento

Para medir el tiempo de procesado y los recursos utilizados en el DSP se recurrió nuevamente a la herramienta que trae el Code Composer Studio, el Analysistoolkit.

Los recursos utilizados en esta implementación son:

- El espacio en memoria que ocupó el programa de resta y umbralización; de 217 bytes e implementó un total de 4,345,957 instrucciones en ensamblador.
- El número de ciclos que realizó el CPU para ejecutar este algoritmo; de 5,065,664. Considerando que el procesador va a una frecuencia de 1GHz,el tiempo de ejecución fue de 5.065 ms.

4.4.3. Comparación entre las implementaciones de la difuminación entre el Microblaze y el DSP

4.4.3.1. Tasa de frames por segundo para las implementaciones que ejecutan el proceso de difuminación.

Las tasas de frames por segundo para cada implementación se muestran en la figura 4.13. Se calcularon en base a los tiempos de procesamiento presentados en las secciones de medición de tiempo de procesamiento de cada implementación.

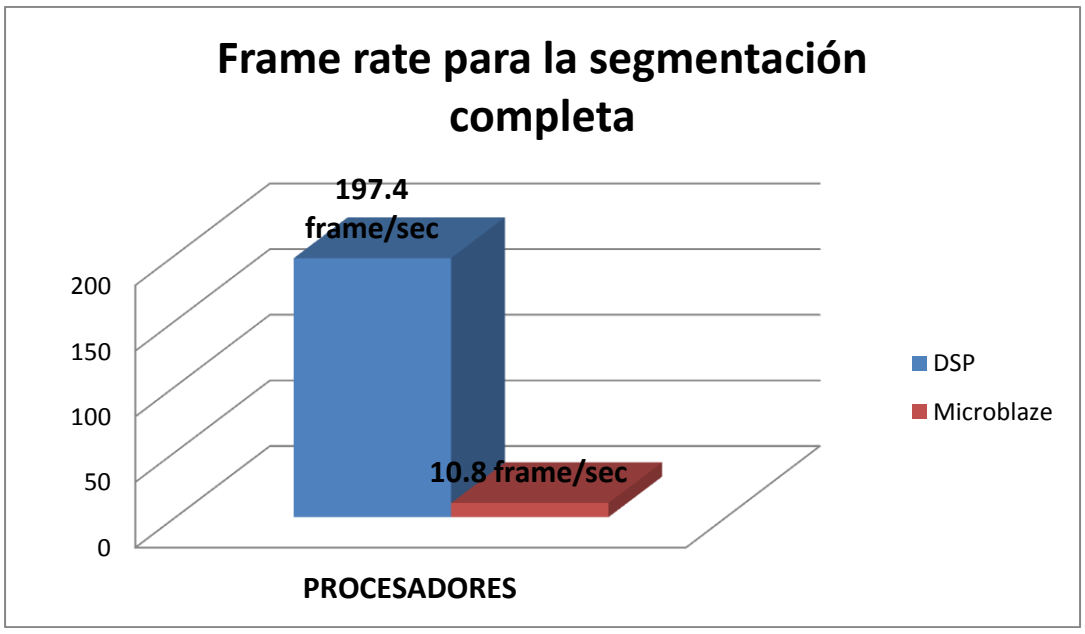


Figura 4.13. Gráfica de comparación de los FPS para la segmentación completa

Con los resultados obtenidos queda desechado trabajar con el Microblaze, ya que en las dos pruebas mostró un rendimiento muy bajo en contra del DSP, teniendo en cuenta que ambos trabajaron con el mismo lenguaje de programación.

4.5 Comparativa de implementación para la obtención de los puntos más lejanos del camarón y la jaiba

Para obtener el vector de posición de los animales en estudio se emplearon algoritmos que obtienen los puntos más alejados del centro de masa de las regiones encontradas por el algoritmo anterior. Estos algoritmos se implementaron en dos versiones: una de ellas utiliza el detector de esquinas de SUSAN y la otra es un método que utiliza una semilla en la región a trabajar y sigue el borde hasta encontrar los puntos de interés (ver apartado **3.2.4.2.3**). Los tiempos obtenidos por cada una de las implementaciones se presentan en la tabla 4.7.

Rutina	Tiempo Estimado
SUSAN	116.480 ms
Método propuesto	0.027 ms

Tabla 4.7. Comparación entre los tiempos de ejecución por los dos métodos

Observando los datos arrojados por el algoritmo de SUSAN, se puede ver que es muy lento, esto se justifica por ser un procesamiento lineal de imágenes, hacer dos barridos y tener un cálculo de promedios que distingue entre sombras aumenta el tiempo del procesado significativamente.

Sin embargo, esto no descarta al algoritmo SUSAN para otras posibles aplicaciones en donde el tiempo no sea de vital importancia y la calidad de resultados que se busque sea sumamente óptima ya que al ser modificable el umbral y el hecho de poder buscar bordes o esquinas dejan abierto el algoritmo para un amplio abanico de posibilidades.

El algoritmo propuesto tiene la ventaja de ser muy rápido, aprovecha el tener los datos necesarios para realizar el recorrido del borde del objeto, sin embargo, está limitado a objetos que no tengan huecos dentro del objeto que se va a analizar ya que al momento realizar la caminata del centro de masa para buscar el borde, en caso de toparse con un hueco recorrería el borde del hueco en lujar del borde exterior del segmento (figura 3.87).

Con las pruebas realizadas y con el fin de evaluar ambos resultados se realizaron test con cerca de 200 imágenes para el seguimiento de los animales, y en ninguna de las imágenes se encontraron huecos ni el centro de masa quedó fuera de la jaiba, ya que en ambos casos el algoritmo hubiera fallado.

Otra observación es que de los 200 casos analizados el error de convergencia de los puntos fue de dos pixeles de diferencia como máximo. Llevando a cabo los cálculos del punto medio prácticamente no se cometía error alguno entre los dos métodos; de esta manera se validó el método propuesto. Un ejemplo de los datos obtenidos de los puntos más lejanos están representados en la tabla 4.8; se pueden ver dos columnas, una con el nombre loop_susan que son los puntos obtenidos con el método de susan y la otra columna con el nombre de loop_intr7 con los puntos obtenidos con el método propuesto. Una observación importante es que en ambos métodos el centro de masa es el mismo.

	loop_susan				loop_intr7			
	Puntos Máximos				Puntos Máximos			
Imagen	1	2	Cx	C/I	1	2	Cx	C/I
CCS_foto5	335, 237	312, 280	306, 250	C	334, 236	313, 279	306, 250	C
CCS_foto13	334, 236	313, 281	307, 251	C	335, 237	314, 280	307, 251	C
CCS_foto22	336, 238	313, 281	307, 252	C	335, 237	314, 281	307, 252	C
CCS_foto37	335, 238	316, 283	308, 254	C	333, 239	317, 282	308, 254	C
CCS_foto42	335, 241	318, 285	309, 255	C	334, 240	319, 284	309, 255	C
CCS_foto65	371, 229	377, 274	356, 253	C	370, 228	379, 274	356, 253	C
CCS_foto77	380, 220	390, 269	366, 246	C	381, 221	390, 268	366, 246	C
CCS_foto95	382, 219	354, 231	372, 244	I	384, 219	393, 266	372, 244	C
CCS_foto136	382, 219	393, 266	372, 244	C	384, 219	395, 266	372, 244	C
CCS_foto193	392, 211	400, 261	379, 238	C	393, 211	402, 261	379, 238	C
CCS_foto199	549,170	578,202	566,184	C	549,171	577,201	566,184	C

Tabla 4.8. Puntos máximos obtenidos por los dos algoritmos

En la foto 95, en donde la jaiba y el camarón estaban solapados, el algoritmo de SUSAN perdió el punto 2 ya que no correspondía el punto encontrado a la jaiba sino que pertenecía al camarón.

De acuerdo con los resultados obtenidos (tabla 4.7) el segundo método propuesto se ejecuta más de 4000 veces más rápido en comparación con el tiempo requerido al utilizar métodos convencionales.

Gracias a este nuevo algoritmo se acorta el tiempo de procesado de una imagen lográndose analizar más imágenes dentro de el lapso de tiempo determinado por la transmisión de imágenes, con ello se pueden procesar las imágenes en tiempo real evitando la necesidad de contar con un costoso espacio de almacenamiento y reduciendo la latencia entre la realización del experimento y la obtención de resultados, al evitar la necesidad de un postprocesamiento de los videos de los experimentos.

Por último se muestran los resultados que se observaron en la obtención de los vectores de posición de la jaiba y del camarón. El algoritmo propuesto es lo bastante robusto, ya que puede procesar hasta un total de diez camarones y una jaiba sin cometer ningún error en el procesamiento de las imágenes. En las pruebas nos limitamos a tener dos camarones como máximo. En la figura 4.14 y 4.15 se observan los resultados de la ubicación de los vectores de posición utilizando diferente número de animales.

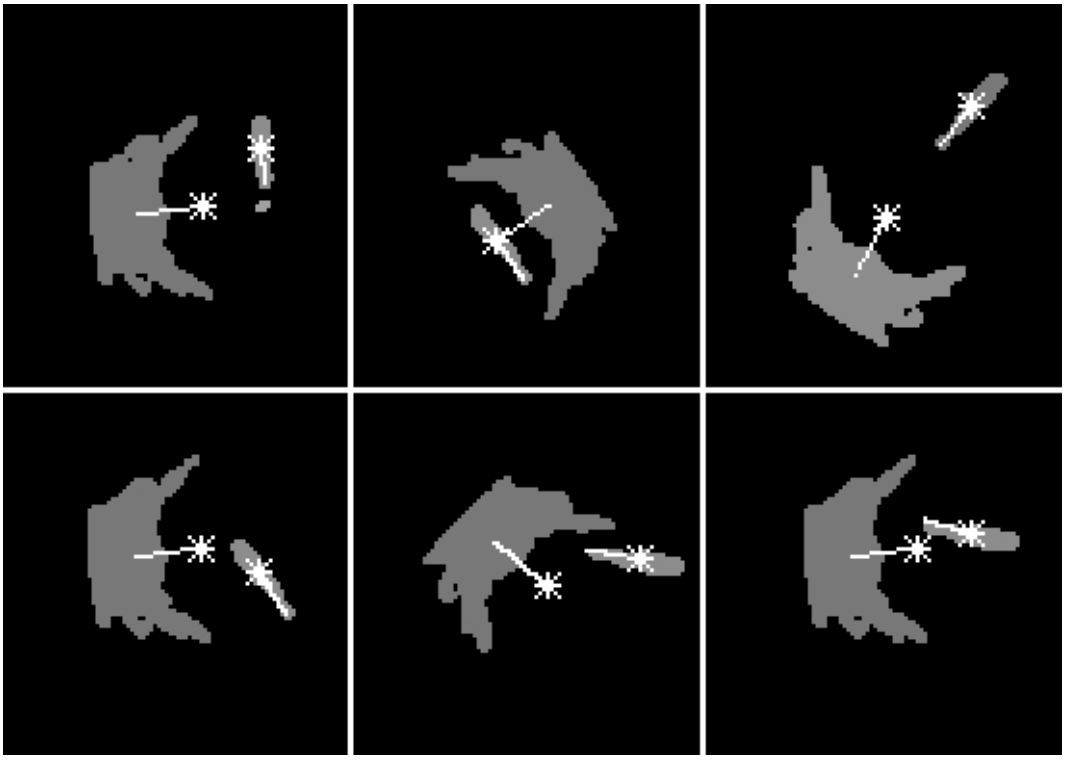


Figura 4.14. Seguimiento de los vectores de posición para un camarón y una jaiba

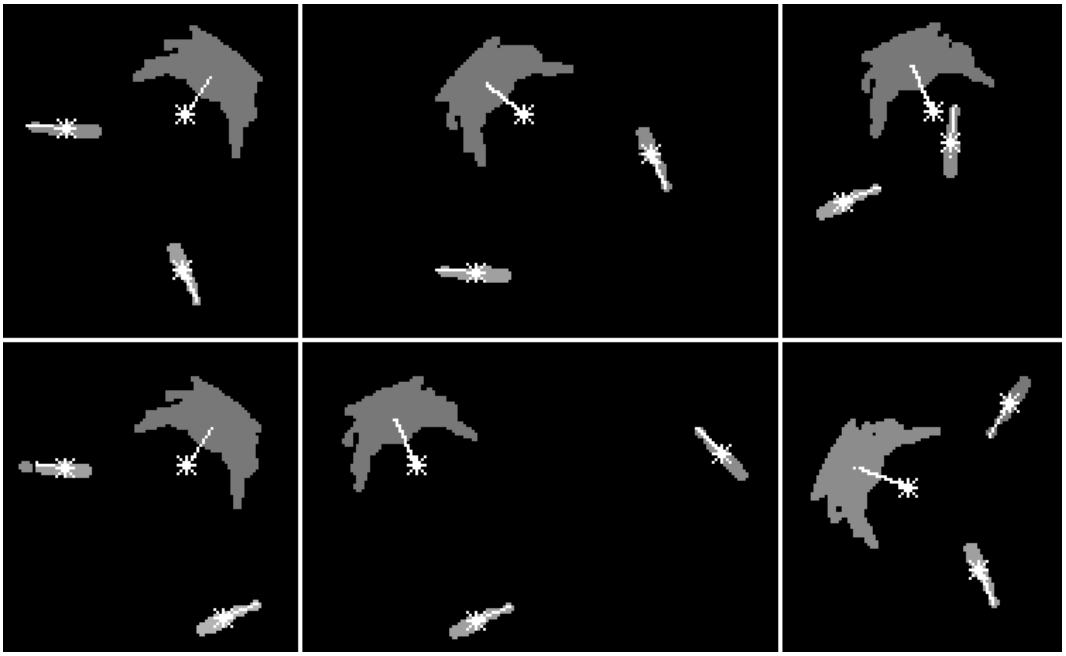


Figura 4.15. Seguimiento de los vectores de posición para dos camarones y una jaiba

En la figura 4.16 se creó una imagen con diez camarones para demostrar que el algoritmo funciona correctamente y en la figura 4.17 se muestra los resultados obtenidos en la ubicación de los vectores de posición de los animales.

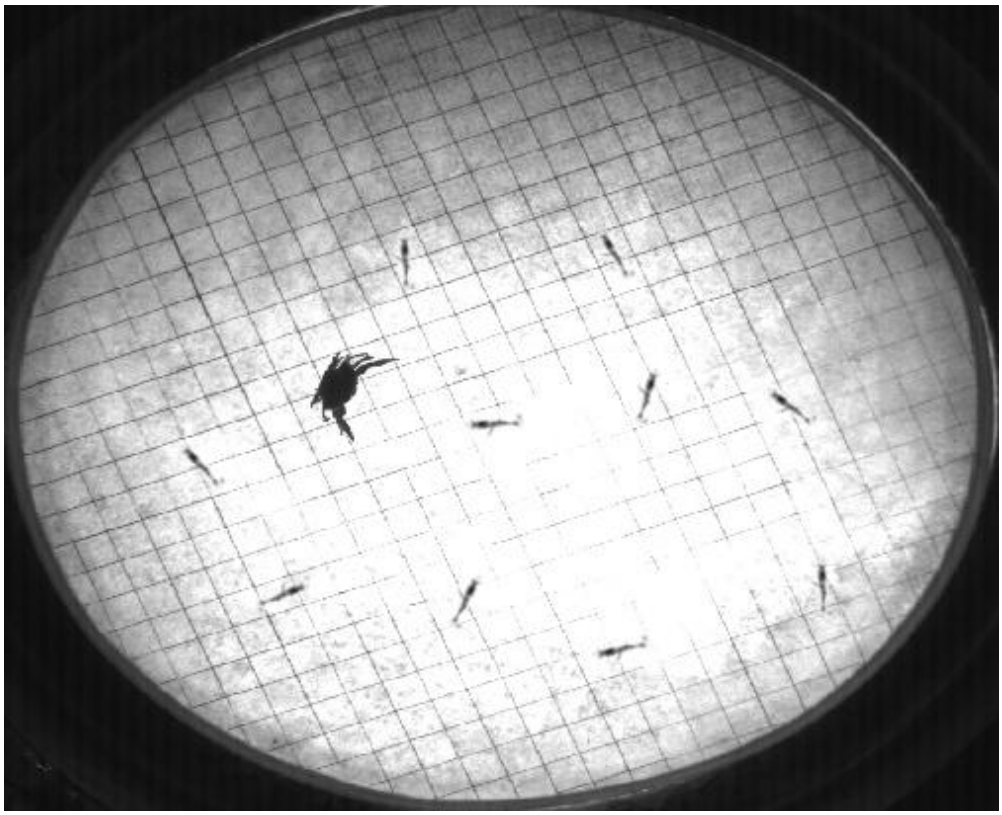


Figura 4.16. Imagen creada

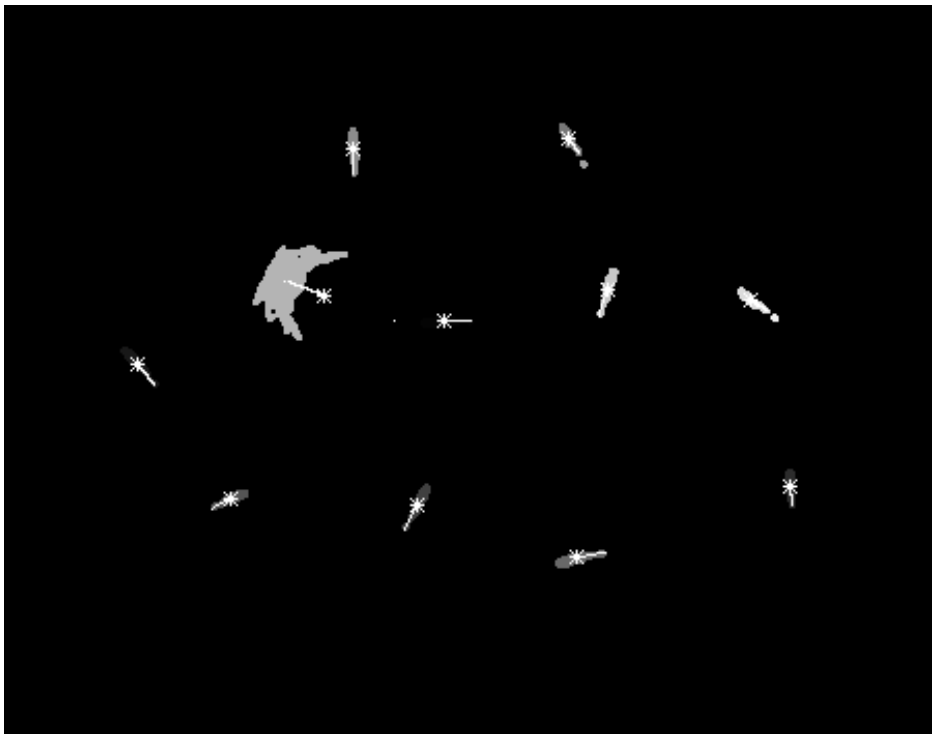


Figura 4.17. Resultados obtenidos con el algoritmo para la ubicación de los vectores de posición

Otro experimento interesante implica reconocer cuando va a realizar el tail-flip, y predecir la ruta de escape del camarón en el momento en el que se siente atacado por el depredador, de acuerdo a lo expuesto en el apartado 2.7 del marco teórico.

Para dar continuidad al trabajo desarrollado en esta Tesis, se presenta una pequeña modificación en el algoritmo con la que se muestra el vector de posición del camarón, así como el cálculo de la distancia entre la cola (realmente la joroba que se forma cuando el camarón se encoje, ver figura 2.29) y el centro de masa en las diferentes fases para la realización del tail-flip. En la figura 4.18 se muestra a través de una serie de imágenes, en donde se empieza a preparar para realizar el tail-flip. En las tres imágenes inferiores se puede apreciar el movimiento del agua cuando está en el tail-flip, se pensó que podría afectar al algoritmo sin embargo esto no sucedió.

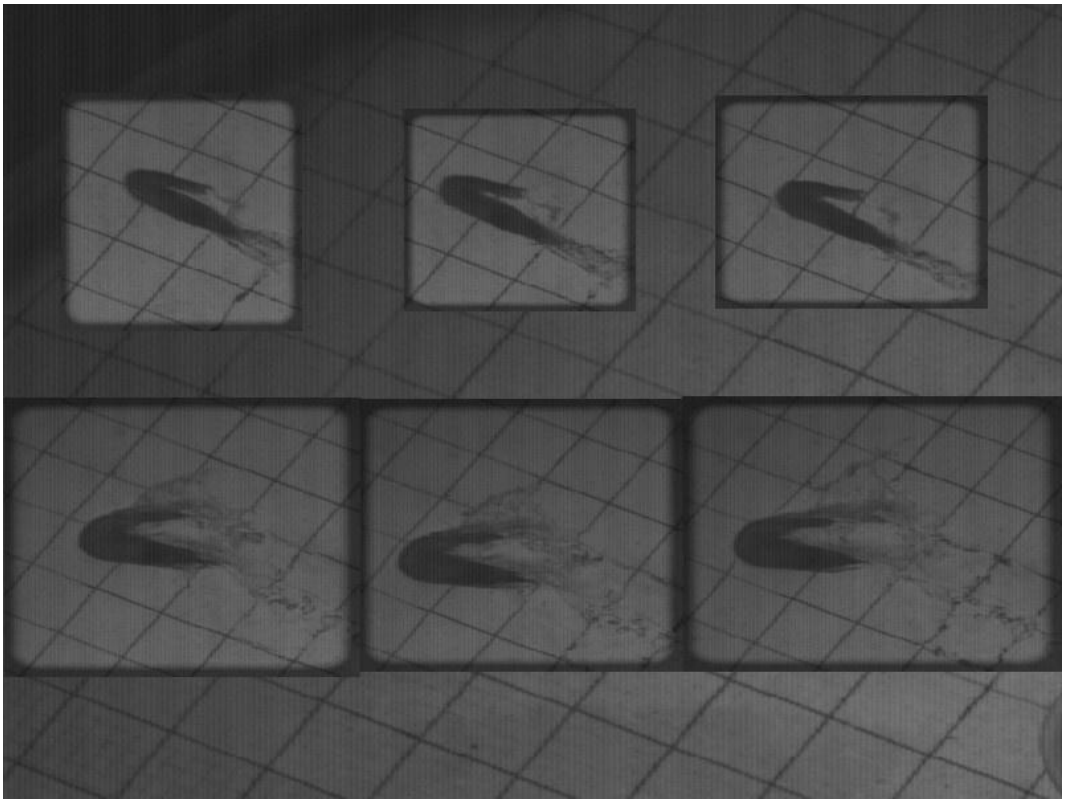


Figura 4.18.Imágenes con acercamiento de la cámara para el cálculo de la distancia entre la cola y el centro de masa

En la figura 4.19, se muestran los resultados del vector de posición con un acercamiento de la cámara, con la finalidad de observar de una forma más precisa las longitudes entre la cola y el centro de masa, ya que esta longitud es la que determina el inicio del tail-flip. En la tabla 4.9 se presentan las coordenadas del centro de masa y del punto de medición para el tail-flip, así como la distancia entre ellos medida en pixeles, la lectura de las imágenes son de arriba hacia abajo y de izquierda a derecha.

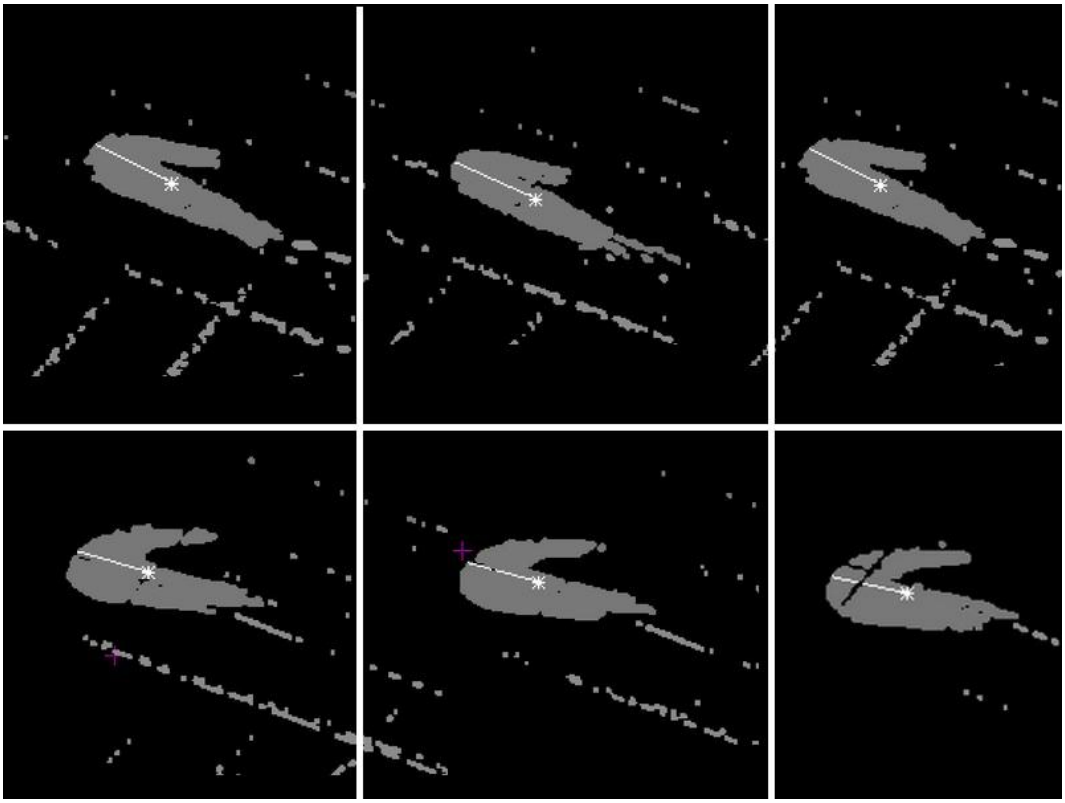


Figura 4.19. Resultados obtenidos con el acercamiento de la cámara

Imagen	Centro de masa	Punto de medición	Distancia en píxeles
CCS_foto31	631, 280	584,257	52.32
CCS_foto32	570,252	618,273	52.39
CCS_foto33	557,243	600,264	47.85
CCS_foto35	470,222	512,234	43.68
CCS_foto36	458,222	500,233	43,41
CCS_foto37	438,222	479,231	41.97

Tabla 4.9. Coordenadas obtenidas del centro de masa y del punto de medición con el cálculo de su distancia entre ellas

4.6 Publicaciones.

4.6.1. Capítulo de Libro.

Carlos Lujan Ramírez, Ramón AtocheEnseñat, Francisco José Mora Más. "*Acquisition and digital images processing, comparative analysis of FPGA, DSP, PC for the subtraction and thresholding.",*publicación en ellibro Image Processing con ISSBN 978-953-307-026-1, impreso en India, diciembre 2009.

4.6.2. Revistas.

Carlos Lujan Ramírez, Ramón Atoche Enseñat, Francisco José Mora Más. "*Obtención en tiempo real de un vector de posición en secuencias de imágenes utilizando técnicas de visión computacional",* Revista Académica de la Facultad de Ingeniería, Universidad Autónoma de Yucatán, Volumen 15, numero 2, ISSN: 1665-529X. Pp. 129-139. Mérida, Yucatán, México, Mayo-Agosto 2011.

José R. Atoche E., Héctor J. Pinto A., Carlos A. Luján R. "Software Para Medición Y Análisis Microscópico", Revista del Centro de Graduados e Investigación, Instituto Tecnológico de Mérida, Ano XXIV Núm. 47, ISSN 0185-6294, pp 178-208. Mérida, Yucatán, México, 12 Noviembre de 2008.

C. A. Luján, J. R. Atoche, F. Mora-Más. "Uso de las Imágenes en Formatos Bayer en el Procesamiento Digital de Imágenes", Revista del Centro de Graduados e Investigación, Instituto Tecnológico de Mérida, Ano XXIV Núm. 46, ISSN 0185-6294. Pp 213-226. Mérida, Yucatán, México, 20 Mayo de 2008.

4.6.3. Publicaciones en Congreso internacional.

C. A. Acosta Ramos, J. R. Atoche Enseñat, C.A. Lujan Ramírez, "*Convolucionador genérico implementado en FPGA para procesamiento de imágenes 2D en tiempo real*", 5o Congreso Internacional de Ingeniería Electromecánica yde Sistemas (CIIES 5), pp ELO030 1-5.México, 10 al 14 de Noviembre de 2008.

C. A. Lujan, F. J. Mora, J. R. Atoche *"Comparative Analysis in the Implementation of Subtraction and Thresholding for Digital Image Processing"*, 5th International Conference on Electrical Engineering, Computing Science and Automatic Control (CCE 2008), IEEE, **ISBN** 978-1-4244-2498-6, pp 465-469 México, México, November 12-14, 2008.

Carlos Alberto Lujan Ramírez, Francisco José Mora Más, José Ramón Atoche Enseñat, "*Property of images in Bayer formats in the Digital Processing of Images*". Electronics, Robotics and Automotive Mechanics Conference (CERMA 2008), IEEE-Computer Society, **ISBN** 978-0-7695-3320-9, pp 267-271, Cuernavaca, Morelos, Mexico, September 30 - October 3, México, 2008.

C. A. Lujan, F. J. Mora, J. D. Martínez."*Comparative analysis between the Stratix II (Altera) and Virtex 4 (Xilinx) for implementing a LVDS bus receiver",*p.p 372-376, 2007 4th International Conference on Electrical and Electronics Engineering (ICEEE 2007), en Mexico D.F., Mex, Septiembre, 2007.

4.6.4. Publicaciones en Congresos Nacionales.

J. R. Atoche Enseñat, C. A. Acosta Ramos, C. A Luján Ramírez, "*Arreglo Sitólico Reconfigurable Para Aplicaciones 2d En Procesamiento De Imágenes En Tiempo Real"*, Congreso de Instrumentación SOMI XXIII, **ISBN**: 970-32-2673-6, Xalapa, Veracruz, México, 1 al 3 de Octubre de 2008.

Josué Moisés Aguilar Garrido, Carlos Alberto Luján Ramírez, José Ramón Atoche Enseñat. "*BAYER TOOLBOX"*, VIII Congreso Nacional de Ingeniería Eléctrica y Electrónica del Mayab. **ISSN** 1665-0271. Pp 435-442. Mérida, Yucatán, México, 21 al 25 de Abril de 2008.

Josué Moisés Aguilar Garrido, Carlos Alberto Luján Ramírez, José Ramón Atoche Enseñat. *"Implementación Del Algoritmo De Conversión De Bayer A RGB"*, VIII Congreso Nacional De Ingeniería Eléctrica y Electrónica del Mayab. **ISSN** 1665-0271. Pp 443-447. Mérida, Yucatán, México, 21 al 25 de Abril de 2008.

Josué Moisés Aguilar Garrido, Carlos Alberto Luján Ramírez, José Ramón Atoche Enseñat. *"Detección De Objetos A Partir De Procesamiento De Imágenes Con Formato Bayer Con Un Microprocesador Embebido Microblaze",* VIII Congreso Nacional De Ingeniería Eléctrica y Electrónica del Mayab. **ISSN** 1665-0271. Pp 448-452. Mérida, Yucatán, México, 21 al 25 de Abril de 2008.

Josué Moisés Aguilar Garrido, Carlos Alberto Luján Ramírez, José Ramón Atoche Enseñat, "*Detección De Objetos A Partir De Procesamiento De Imágenes Con Formato Bayer Con C++ Builder6",* VIII Congreso Nacional De Ingeniería Eléctrica y Electrónica del Mayab. **ISSN** 1665-0271. Pp 453-460. Mérida, Yucatán, México, 21 al 25 de Abril de 2008.

C. A. Lujan, F. J. Mora, J. D. Martínez."*Implementación de un receptor para el bus LVDS utilizando el DCM de un FPGA en la reducción del skew.",*CM 11,P-24, Decimosétima Reunión de Otoño de comunicaciones, Computación, Electrónica y Exposición Industrial, IEEE sección México, en Acapulco Guerrero, México, Diciembre, 2006

CAPITULO 5 - CONCLUSIONES Y TRABAJOS FUTUROS

En este capítulo, se plantea una recapitulación del propósito de esta investigación en la que se muestran las aportaciones y se extraen las conclusiones más importantes del trabajo desarrollado. De igual manera se presentan los resultados que directa o indirectamente ha dado lugar la presente Tesis. Por último se abordan las futuras líneas de investigación por las que se podrían continuar.

5.1 CONCLUSIONES.

Al iniciar esta Tesis doctoral, y tratando de cumplir con el objetivo planteado de generar una instrumentación que permita la monitorización y extracción de datos de experimentos biológicos con jaibas y camarones, y almacenar los vectores de sus posiciones, fue necesario llevar a cabo un estudio de las diferentes tecnologías a utilizar y determinar la mejor de ellas; para ello, fue esencial diseñar el hardware que cumpliera con las necesidades de estas tecnologías así como de los algoritmos requeridos para el funcionamiento óptimo.

El contar con el diseño de una plataforma propia que pueda lograr el procesamiento en tiempo real mediante una plataforma embebida en lugar de un paquete computacional con el Ethovision. Que depende un equipo robusto de cómputo. Adicionalmente, el autor considera que el sistema embebido tiene la posibilidad de ser escalable en cuanto a las funcionalidades del mismo y que éstas se pueden ir modificando de acuerdo a las necesidades de la investigación.

5.1.1 Adquisición de las imágenes.

El estudio da inicio examinando las características de la cámara fotográfica; es importante resaltar que esta se utilizaba de manera manual por parte de los investigadores que analizan el comportamiento de las especies marinas. El primer paso consistió en la obtención de las imágenes que se transmiten en serie con el protocolo CameraLink; para ello se evaluaron dos alternativas, la primera utilizando una tarjeta de evaluación que incluye un FPGA e implementando el deserializador del protocolo CameraLink así como el hardware necesario para almacenar a las imágenes, y la segunda, empleando un circuito integrado DS90CR286 como deserializador del protocolo CameraLink. En este punto se puede concluir que ambas alternativas pueden recuperar las imágenes, sin embargo se descartó la segunda opción ya que el circuito integrado solo se utiliza para deserializar los datos, y además se necesita la conexión de una memoria de doble puerto y el hardware de control de la memoria que está incluido en la primera opción, así como de una tarjeta de circuito impreso adicional para el uso del circuito integrado.

Con la determinación del uso del FPGA se realizó el análisis para decidir qué fabricante, entre Xilinx y Altera, cumplía con el mejor desempeño. Para esto se compararon dos FPGA´s con características semejantes; con los resultados obtenidos se observó que bajo ciertas condiciones Xilinx tenia ventajas sobre Altera y al modificar estas se invierten los desempeños, es decir Altera tenia ventajas sobre Xilinx, sin embargo lo importante fue que ambas tecnologías superaban el requerimiento de 66MHz. En Altera se logró una frecuencia de 70.32 MHz y en Xilinx de 85 MHz. Con estos resultados se determinó el uso del FPGA fabricado por Xilinx y muy particularmente una Virtex 4.

Para lograr conectar la cámara con la tarjeta de evaluación fue necesario realizar el diseño de una tarjeta que acondicionara las señales entre la cámara y el FPGA. Para ello se tomaron una serie de recomendaciones indicadas para el diseño de circuitos impresos en los que se utilicen

señales diferenciales y señales en LVDS, entre estas recomendaciones están el cuidado de las longitudes de las pistas entre pares diferenciales, la separación entre ellas, planos diferentes para las pistas del mismo par diferencial. Con esto se logró obtener una correcta comunicación entre la cámara y el FPGA, estos detalles que a veces parecen insignificantes pueden ocasionar perdida de muchas horas buscando el error en la adquisición de los datos debido a que no se pueden recuperar correctamente a pesar de que todo está conectado correctamente en apariencia. El uso de las herramientas como HyperLynx fue de gran ayuda para la realización de las medidas de las longitudes de las pistas y la realización de los diagramas de ojo para cada par diferencial. Contando con los resultados obtenidos con esta herramienta se diseñó la tarjeta.

Una vez conectada la tarjeta de evaluación del FPGA LX60 de Avnet a la cámara y a la computadora, se procedió a realizar los algoritmos necesarios para la configuración de la cámara, considerando principalmente la velocidad de la adquisición y el tamaño de la imagen. El primer algoritmo fue implementado para leer la configuración actual de la cámara, con ello se logro verificar que todas las conexiones estuvieran correctas. Con la seguridad de tener el control de la cámara se procedió a la adquisición de las imágenes; esto requirió de otra tarjeta para acondicionar la conexión entre la tarjeta de evaluación del FPGA con otra tarjeta de evaluación DSP STARTER KIT TMS320C6416. La tarjeta del DSP procesa a los algoritmos en una frecuencia de 1 GHz. La tarjeta encargada de conectar al FPGA y el DSP no requiere de tanto cuidado de las señales por no ser pares diferenciales, sin embargo, por la frecuencia del bus que es de 125 MHz, es recomendable cuidarlas, para ello utilizamos la herramienta HyperLynx para la medición de las longitudes de las pistas.

Una vez interconectadas las tarjetas y la computadora, se dio inicio a la obtención de las imágenes en el formato Bayer, que es el formato en el que tanto esta, como la mayoría de las cámaras digitales entregan la imagen. Para poder apreciar las imágenes obtenidas se requiere un algoritmo cuya función es convertir la imagen en formato Bayer a formato RGB. En este momento surge la duda si es necesario convertir las imágenes de Bayer a RGB para su análisis. Para decidir la mejor opción se implementaron las dos primeras funciones del algoritmo (la resta y la umbralización), tanto en formato RGB como directamente en formato Bayer, con objeto de tener una comparativa que pudiera indicarnos la conveniencia o inconveniencia de utilizar directamente las imágenes en formato Bayer eliminando la necesidad de realizar la conversión a RGB. Primeramente se realizaron la resta y la umbralización directamente sobre las imágenes adquiridas en formato Bayer, para la segunda etapa se realizó primero la conversión de Bayer a RGB y posteriormente se realizaron la resta y la umbralización, en ambos procedimientos se utilizó las mismas imágenes adquiridas como se aprecian en la figura 4.3. Los resultados fueron muy similares, pero en algunos aspectos el procedimiento que evita el convertidor fue mejor, basándonos en el análisis de los resultados observados en las figuras 4.4 y 4.6. Habiendo disipado la duda, se determinó que no era necesario el convertidor y con ello se ahorró el tiempo de ejecución del mismo, así como el espacio para almacenar la imagen RGB que ocupa tres veces más que el de formato Bayer.

5.1.2 Procesamiento en diferentes tecnologías: Microblazer, Máquina de estados y DSP.

Con el fin de evaluar la tecnología que mejore el desempeño, se procedió a implementar los mismos algoritmos en 3 diferentes tecnologías utilizando las mismas imágenes de 102 X 150 pixeles. La primera tecnología implementa los algoritmos en un microprocesador embebido en un FPGA (Microblazer), la segunda utiliza maquinas de estado y procesamientos en paralelo para implementar dichos algoritmos en un FPGA, y por último se utiliza un DSP de la familia Texas

Instrument para implementar los mismos procesos. Los algoritmos evaluados fueron la resta y la umbralización. Durante este análisis se observo lo siguiente: la mejor tecnología es el FPGA basada en una maquina de estados, la segunda mejor tecnología es la basada en un DSP y la menos efectiva es la implementada en el microprocesador; esto lo determino en base a la cantidad de frames que ejecuta en un segundo cada tecnología y está mostrada en la figura 4.11. Entre estas dos últimas FPGA (Microblazer) y el DSP, el algoritmo fue casi el mismo ya que en ambas se utiliza el lenguaje C, sin embargo, la diferencia fue muy significativa. A pesar de que el FPGA basado en máquina de estados tiene el mejor desempeño, lo realmente difícil fue realizar el programa, y cualquier modificación por insignificante que sea, requiere repetir el proceso de diseño-simulación-implementación, lo cual hace muy complicado su uso, ya que cuando se lleva a cabo alguna modificación del algoritmo, el tiempo de desarrollo es un factor importante a considerar, en tanto que con el DSP cualquier cambio en el algoritmo puede realizarse con rapidez pues se trabaja a alto nivel con compiladores de C.

Otro análisis comparativo que se llevó a efecto entre las tecnologías del microprocesador embebido en el FPGA y el DSP está relacionado con el tiempo de ejecución de la difuminación, y nuevamente el DSP tuvo el mejor desempeño, con lo que se determino claramente que el microprocesador embebido tiene las menores prestaciones frente al DSP.

5.1.3 Comparativa entre el algoritmo de SUSAN y el diseñado en este trabajo, para encontrar los puntos más lejanos en la jaiba.

Con los animales libres del fondo, se procedió a segmentar la imagen, en ella se identificó cada una de las regiones encontradas pintándolas de un tono de gris diferente, y se guardó la siguiente información: el valor del área para saber si es un camarón, jaiba, ruido o si es un cuerpo extraño, y las coordenadas X y Y del centro de masa para cada una de las áreas. Con el valor del área se aplicó una rutina de discriminación para dejar únicamente al camarón y a la jaiba. El algoritmo tiene capacidad para identificar a una jaiba y hasta 10 camarones, para esta investigación se trabajó en su mayoría con un solo camarón y en algunos casos con dos.

Para localizar el vector de posición de la jaiba, partiendo de la certeza de que solo se encuentran en el campo de interés dos tipos de animales (la jaiba y uno o más camarones), y de que la jaiba tiene un comportamiento bien definido cuando ataca a alguna víctima extendiendo completamente ambas quelas (tenazas), se estudiaron varios algoritmos para la identificación de los puntos más lejanos al centro de masa de la jaiba, ya que estos puntos están ubicados en el extremo de la quela extendida. El primero de los algoritmos evaluados requiere la extracción de contornos de la imagen; para esto se estudiaron los métodos de SUSAN, CANNY, SOBEL, operador LAPLACIANO entre otros. Se seleccionó el algoritmo de SUSAN por ser el más rápido y cuya convergencia es la más precisa. Sin embargo, y a pesar de ser el más rápido, no cumplía con las exigencias de nuestros requerimientos, por lo que fue necesario implementar un algoritmo que no requiera la extracción previa de contornos de la imagen, sino que basado en la imagen ya segmentada por el paso previo, pudiera seguir el contorno de la región de interés, con este nuevo algoritmo se aumentó significativamente la velocidad de procesamiento. Para encontrar el vector de posición de la jaiba son necesarias las coordenadas del centro de masa y el punto medio entre los puntos más lejanos (los extremos de las quelas), las coordenadas de estos dos puntos son los que se envían al ordenador.

Para encontrar el vector de posición del camarón es necesario el centro de masa y el punto más lejano. El algoritmo desarrollado para encontrar los puntos en la jaiba también se le aplicó

al camarón para encontrar el punto más lejano (la cola). Estos dos datos junto con el vector de posición de la jaiba, son los que enviamos al ordenador para futuros procesamientos por los investigadores.

El trabajo de investigación cumplió los siguientes objetivos: encontrar los puntos que determinan los vectores de posición de la jaiba y del camarón en las imágenes obtenidas, para después enviarlas por el puerto USB al ordenador.

Una de las características analizadas y de mayor importancia es el tiempo de ejecución del algoritmo para la obtención de los vectores de los animales. Durante el desarrollo de esta Tesis y al aplicarle el algoritmo a diferentes imágenes, se obtuvieron como mejor tiempo de procesamiento 61.52 milisegundos con lo que se lograron procesar 16.25 frames. El tiempo máximo de procesamiento es de 61.66 milisegundo para procesar 16.21 frames, para imágenes de 480 x 720 píxeles. Esta diferencia de tiempos se debe a que las regiones obtenidas en las imágenes son diferentes en tamaño ya que la posición de los animales puede afectar su tamaño y por eso requiere de mayor o menor tiempo en el procesamiento. Tomando en cuenta que la cámara puede mandar fotos a una velocidad de 118 frames por segundo a su máxima frecuencia para imágenes de 480 X 720, podemos decir que estamos a un 14% de la frecuencia máxima.

A través de este estudio se pudo identificar la mejor tecnología a utilizar; en este sentido, podemos aportar que utilizando una mezcla de tecnologías entre el FPGA y el DSP se puede lograr mejorar el desempeño del algoritmo diseñado durante el desarrollo de esta Tesis. Esto considerando lo siguiente, para llevar a cabo la resta y la difuminación se puede utilizar el FPGA y el resto del análisis de la imagen lo realizará el DSP; durante este procesamiento requerimos de un total de 12.99 milisegundos, y podríamos procesar 76.9 frames en un segundo, quedando a un 66% de la frecuencia máxima.

5.2 Trabajos futuros.

Aún cuando los objetivos que se establecieron muestran un avance para el logro del estudio del comportamiento entre el camarón y la jaiba, es importante señalar que todavía queda trabajo por realizar; lo primero a considerar es el estudio de la mezcla de tecnologías utilizadas en este trabajo, es decir, entre el FPGA y el DSP con la finalidad de aumentar la máxima velocidad a la que va la cámara en la captura de las imágenes, esto es, lograr procesar al 70% del tiempo, y lo segundo es identificar cuando el camarón está cerca de la jaiba y realizar de manera automática un acercamiento de la cámara en el lugar donde están los animales para capturar la secuencia de imágenes de menor dimensión en el momento del tail-flip. Al estar cerca no se requiere de una imagen muy grande por lo que se podrían analizar el mismo número imágenes en el frame, para que el requerimiento en velocidad sea menor de de tal forma que no afectaría el área en estudio. Con esto se puede equilibrar el número de imágenes en un frame con la velocidad de la cámara.

BIBLIOGRAFÍA

[Altera Corporation 2007] Stratix II Device Handbook, Volume 1, Altera Corporation, 2007.

[Arnott, 1998] Stephen A. Arnott, Douglas M. Neil and Alan D. Ansell "Tail-Flip mechanism and size-dependent kinematics of escape swimming in the brown shrimp crangon crangon" The Journal of Experimental Biology 201, pp 1771-1784. Año 1998.

[Arnott, 1999] Stephen A. Arnott, Douglas M. Neil and Alan D. Ansell "Escape trajectories of the brown shrimp crangon crangon, and a theoretical consideration of initial escape angles from predators " The Journal of Experimental Biology 202, pp 193-209. Año 1999.

[Barrero, 2005] Federico Barrero Garcia, Sergio Toral Marin y Mariano Ruiz González "Procesadores Digitales de Señal de altas prestaciones de Texas Instruments" Mc Graw Hill, 2005.

[Benezeth, 2008] Y. Benezeth, P. M. Jodoin, B. Emile, H. Laurent, and C. Rosenberger, "Review and evaluation of commonly-implemented background subtraction algorithms," in *Proc. IEEE Int. Conf. Patt. Recog.*, Dec. 2008, pp.1–4.

[Bayer, 1976] Bayer, B. E.; (1976) Color imaging array, U.S. Patent, 3971065.

[Bloomingdale, 2002] Bloomingdale Cindy "LVDS Data Outputs for High-Speed Analog-to-Digital Converters", AN-586, Application Note, Analog Device, 2002.

[Bravo, 1996] Antonio J., Bravo V. "Procesamiento Digital de Imágenes". Tutorial: http://www.ing.ula.ve/~abravo/document/tutorial/imagenes/indice.html #c22. Universidad de Los Andes, Facultad de Ingeniería, Grupo de Ingeniería Biomédica. Mérida, Venezuela, 1996.

[Canny 1986] J. F. Canny. "A computational approach to edge detection". IEEE Transaction on Pattern Análysis and Machine Inteligence. Pp 679-698 1986.

[Chen, 2005] Yu -Jen Chen, Yan-Chay Li, Ke-Nung Huang and Ming-Shing Young" *The Implementation of a Stand-alone Video Tracking and Analysis System for Animal Behavior Measurement in Morris Water Maze* " Engineering in Medicine and Biology Society, 2005. 27th Annual International Conference of the IEEE.

[Chen, 2006] Chia-Hsiung Chen, Sao-Jie Chen and Pei-Yung Hsiao "*Edge Detection on the Bayer Pattern*" Asia Pacific Conference on Circuits and Systems, 2006. IEEE 2006.

[Cole, 2002] E. Cole, "Performance of LVDS with Different Cables", SLLA053B, Application report, Texas Instruments, February 2002.

[Dinamarca, 2002] José Antonio Dinamarca Ossa, "Análisis de diagramas de ojo". Universidad Técnica Federico Santa María. 6 de noviembre del 2002.

[Egrid, 2006] John Egri, Imperx, Ethernet vs Camera Link, Machine Vision, Test & Measurement World, p.p,41-47, may 2006

[Ertürk, 2003] Ertürk Sarp; Digital Image Processing, IMAQTM, LabVIEWTM, 2003.

[González, 1992] R. C. González y R. E. Woods, "Digital Image Processing". Addison-Wesley, 1992.

[Grant, 2002] Mick Grant, "Signal Integrity Considerations for High Speed Digital Hardware Design". White Paper. November 1, 2002.

[Gribbon, 2005] K. T. Gribbon, D. G. Bailey and C. T. Johnston "Using Design Patterns to Overcome Image Processing Constraints on FPGAs" Proceedings of the Third IEEE International Workshop on Electronic Design, Test and Applications (DELTA'06).

[Hansen, 2002] Christopher L. Hansen, "Digital image processing for clinicians, part I: Basics of image formation" Journal of Nuclear Cardiology, 2002; Pp. 343– 349.

[Hua, 2010] Lei Hua, Lei Xie1 and Huifang Chen, "A Color Interpolation Algorithm for Bayer Pattern Digital Cameras Based on Green Components and Color Difference Space" Progress in Informatics and Computing (PIC), 2010 IEEE International Conference on. Pp. 791-795.

[Hovard, 1993] R. Horavd, O. Monga, "Visión par ordinateur", Editions Herme Paris, 1993.

[INAOEP, 2005] ¿Qué es la visión por computadora?; http://ccc.inaoep.mx/~labvision/vcomp.htm.

[Kumar, 2010] K Susheel Kumar, Shitala Prasad, Pradeep K. Saroj and R.C. Tripathi "Multiple cameras using real time object tracking for surveillance and security system". Third International Conference on Emerging Trends in Engineering and Technology, ICETET2010.

[Knighten, 2000] J. Knighten, N. Smith, J. DiBene and L. Hoeft "EMI Common Mode Current Dependence on Delay Skew Imbalance in High Speed Differential Transmission Lines Operating at 1 Gigabit/Second Data Rates," Quality Eectronic Design, 2000. ISQED 2000. Proceedings. IEEE 2000 First International Symposium on volume, Issue, 2000 , pp. 309-313.

[Lei, 2010] Wang Lei and Shen Yuming "A Study on Comparisons of Four Corner Detection Algorithms In Palmprint Identification System" The Chinese Sympsium on Information Science and Technology 20010. CSIST 2010.

[Lania, 1999] Aplicaciones de la visión por computadora; http://www.lania.mx/biblioteca/newsletters/1999-primavera-verano/aplicaciones.html.

[Maxwell, 1998] Maxwell Bruce A., Teaching Computer Vision to Computer Scientists:Issues and a Comparative Textbook Review; University of North Dakota.

[Mingliang, 2010] Hou Mingliang and Xing Shubin "Study of Improving the Stability of SUSAN Corner Detection Algorithm" 2010 International Conference on Computer Application and System Modeling (ICCASM 2010).

[National Semiconductor A, 2004] LVDS Owner´s Manual. National semiconductor. 3rd Edition Spring 2004.

[National Semiconductor, 2006] Channel Link Design Guide, National semiconductor, June 2006

[National Semiconductor B, 2004] National Semiconductor, DS90CR285/DS90CR286 +3.3V Rising Edge Data Strobe LVDS 28-Bit Channel Link-66 MHz, Julio 2004.

[Pajares, 2008] Gonzalo Pajares Martinsanz y Jesús M. de la Cruz García, "Visión por Computador". Addison-Wesley, 1992.

[Rezai, 2006] Gholamali Rezai-Rad and Majid Aghababaie "Comparison of SUSAN and Sobel Edge Detection in MRI Images for Feature Extraction". Information and Communication Technologies, 2006. ICTTA 06. 2nd.

[Rivoallon,2006] Rivoallon Frédéric "Achieving Breakthrough Performance in Virtex-4 FPGAs" WP218 (v1.4) , Xilinx, May 19, 2006

[Sakamoto, 1998] Sakamoto, Nakanishi and Hase, "Software Pixel Interpolation for Digital Still Camaras Suitable for a 32-bit MCU", IEEE Transactions on Consumer Electronics, Vol. 44, No.4, November 1998.

[Sawyer, 2008] Sawyer N., "1:7 Deserialization in Spartan-3E/3ª FPGAs at Speeds Up to 666 Mbps", XAPP485 (v1.2), Application notes, Xilinx, 27 May 2008.

[Schoenemann, 2008] Thomas Schoenemann and Daniel Cremers "Globally Optimal Shape-based Tracking in Real-time" IEEE Computer Society Conference on Computer Vision and Pattern Recognition 2008, Anchorage, Alaska.

[Shapiro, 2001] Shapiro Linda and Stockman George; Computer Vision, First Edition; Prentice Hall, 2001.

[Smith, 1995] S. M. Smith and J. M. Brady, "SUSAN – A New Approach to Low Level Image Processing", Technical Report, FMRIB, 1995.

[Takaya, 2006] Kunio Takaya, "Detection and Segmentation of Moving Objects in Video", Electrical and Computer Engineering, 2006. CCECE '06. Canadian Conference on. Pp. 2069-2073.

[Texas Instruments, 2002] "Interface Circuit for TIA/EIA-644 (LVDS)", SLLA038B, Application notes, Texas Instruments, September 2002.

BIBLIOGRAFÍA

[Torres, 2001] F. Torres, "Síntesis y Descripción de Circuitos Digitales Utilizando VHDL", Universidad Autónoma de Guadalajara, 2001.

[Wang, 2004] Wenxin Wang and Robert D. Dony "Evaluation of image corner detectors for hardware implementation". Electrical and Computer Engineering, 2004. Canadian Conference on, vol. 3, pp. 1285–1288, May 2004.

[Wang, 2008] Qiang Wang and Zhanhong Gao "Study on a Real-time Image Object Tracking System", International Symposium on Computer Science and Computational Technology, ISCSCT.2008. P.p. 788-791.

A.1. Descripción Técnica del LVDS.

La Señalización Diferencial de Bajo Voltaje (Low Voltage Differential Signaling por sus siglas en ingles LVDS) es un estándar de interface de alta velocidad, baja potencia y propósito general[Texas Instrumen, 2002][Cole, 2002]. El estándar, conocido como ANSI/TIA/EIA-644, fue aprobado en marzo de 1996. LVDS usa señalización diferencial, con una variación de voltaje nominal de 350mV diferencial. LVDS usa transmisores o drivers en modo corriente, el cual limita el consumo de potencia. Las señales diferenciales son inmunes a ruido de ±1 V común.

Hoy en día existen dos formas de operación para los circuitos eléctricos de interfaz, la llamada de terminal única o desbalanceada (single-ended) y la diferencial o balanceada. Para la transmisión balanceada o diferencial, un par de líneas para las señales son necesarias por cada canal. En una línea la señal de información es transmitida mientras que en la otra línea se transmite el inverso de la señal de información. El receptor detecta la diferencia de voltaje entre las dos señales y conmuta la salida dependiendo del valor de la diferencia. El par diferencial para un canal se muestra en la figura A.1.

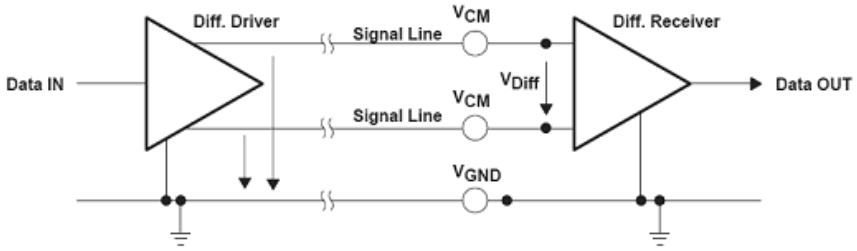


Figura A.1. Transmisión diferencial

El cable de par trenzado usado en estas interfaces, en combinación con una correcta terminación de las líneas para evitar las posibles reflexiones, permite una enorme velocidad de transmisión de los datos. Velocidades de transmisión por arriba de 10Gbps son posibles con la transmisión de datos diferenciales.

Los circuitos de transmisión de datos diferencial son menos susceptibles al ruido de modo común que en los circuitos de terminal única [Knighten, 2000]. Puesto que este tipo de transmisión usa dos cables con corriente y oscilación de voltaje opuestos, comparadas con los circuitos de terminal única con una sola línea, cualquier ruido externo se acopla a las dos líneas como voltaje de modo común y es rechazado por el receptor.

La especificación ANSI/EIA/TIA-644 estipula que una resistencia en la terminal diferencial LVDS es necesaria en la entrada del receptor para generar el voltaje de salida diferencial (V_{OD}) con valor de 90Ω a 132Ω. La terminal diferencial LVDS es más sencilla que en otras tecnologías como se puede observar el la figura A.2.

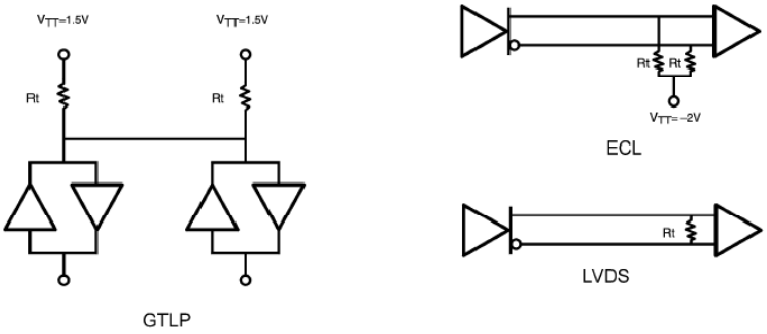


Figura A.2. Terminales características de las diferentes tecnologías

La velocidad de procesamiento es un parámetro importante cuando se necesita determinar los requerimientos de ancho de banda de un circuito de intercambio de datos de una línea del controlador, medios de interconexión o de una línea del receptor. La transmisión de dos cambios de estado sucesivos forma el inicio y el final de un pulso de voltaje y corriente que pasa a través del circuito. Como se puede esperar, existe un límite de que tan corto puede ser la duración del pulso para lograr una buena recepción en el otro extremo de la línea.

Existe un criterio de calidad de señal que especifica o describe la duración mínima del pulso o la máxima velocidad de procesamiento de un circuito. Muchas especificaciones describen la calidad de un pulso de datos binario como la cantidad de tiempo gastado entre las transiciones de los estados en relación con la duración del pulso. TIA/EIA-644 y TIA/EIA-899 y otras especificaciones definen este criterio. Estos estándares definen la duración del pulso como una unidad de intervalo t_{ui} y el máximo tiempo de transición como una fracción de t_{ui}. La figura A.3 muestra los criterios de calidad de la señal del estándar TIA/EIA-644

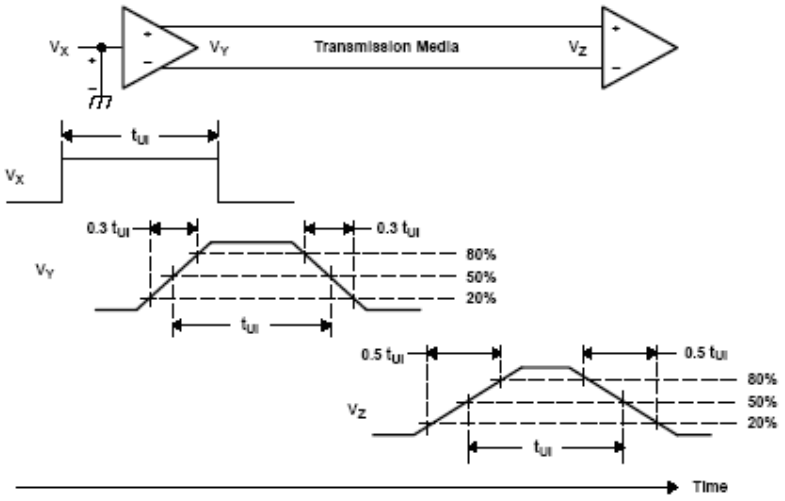


Figura A.3. Recomendaciones de calidad de señales del estándar EIA-644

Cuando el tiempo de transición de la señal sobrepasa los $0.5t_{ui}$ al final de la línea de transmisión, se habrá alcanzado la duración mínima de pulso y la máxima velocidad de procesamiento para el circuito de interfaz. Sin embargo cuando se evalúa la capacidad de velocidad de procesamiento de la línea de algún circuito, se asume una línea sin perdidas y aplicamos los limites de $0.3t_{ui}$ para tanto el controlador como el receptor (esto es por la

degradación del tiempo de transición de $0.5t_{ui}$ permite perdidas en los medios de transmisión y no en las líneas del circuito). Esto implica que la capacidad de velocidad de procesamiento de una línea del circuito pueda ser determinada por su aumento en la salida y sus tiempos de caída.

En base a la velocidad de procesamiento en tiempo real de un pulso de información ignora cuando el estado del bus se convierte en valido en relación al instante de muestreo y asumiendo que el tiempo de muestreo es estable y se repite, cualquier variación del tiempo de duración de cambio de estado del bus suma o resta del tiempo de hold o set-up del muestreo. Se llama genéricamente a esta variación "jitter" y esta proviene de muchas fuentes dentro de nuestro circuito.

Idealmente, la línea del controlador o del receptor reproduce un cambio de estado en la salida un cierto tiempo luego de que ocurra en la entrada. En la realidad, el tiempo de retraso de propagación varía con la temperatura, voltaje, proceso de manufacturación, la secuencia de bits anterior y el ruido. Todos estos factores afectan cuando el estado es válido para el muestreo o para un futuro procesamiento. El diseñador de la interfaz debe tomar en cuenta para la sincronización de un sistema de comunicaciones.

Los circuitos de interfaz de grandes velocidades mayormente transmiten la información de sincronización junto con la información, integrado a la información transmitida en un circuito paralelo separado. El método de transmisión de la información de sincronización (reloj) determina que tipo de jitter en la línea del circuito debemos de planificar. En general, solo las componentes jitter de alta frecuencia son concernientes al reloj codificado con la información, y todas las demás son planificadas mediante un reloj paralelo.

La variación del retraso de tiempo en la propagación de las señales, con la temperatura y los procesos de manufacturación pueden afectar con forme aumente la frecuencia por lo que estas afectaciones son registrados en las hojas de datos del componente. Dado que la mayoría de los circuitos paralelos operan en una misma temperatura y con un mismo voltaje de alimentación, las hojas de datos también especifican un parámetro adicional llamado skew o t_{SKPP} para caracterizar la variabilidad por el proceso de manufacturación. Este parámetro es el retraso del tiempo de propagación de diferentes circuitos integrados que trabajan con una misma fuente de alimentación o en una temperatura dentro del rango de operación recomendado. Un parámetro relativo es el skew de salida o $t_{sk(o)}$ y es la diferencia en el retraso del tiempo de propagación entre dos circuitos paralelos que están en el mismo integrado. En un dispositivo monolítico, los dos circuitos están manufacturados bajo el mismo procedimiento y los retrasos deben de ser casi iguales.

B.1. Introducción a los FPGAs

Los dispositivos lógicos programables fueron introducidos en los años 70´s. La idea fue la construcción de circuitos lógicos combinacionales que fueran programables. Sin embargo, contrario a los microprocesadores que pueden correr un programa pero se contempla como un hardware arreglado, la programación de un PLD se intento que sea en un nivel de hardware. En otras palabras un PLD es un chip de propósito general que puede ser reconfigurado para que se acople a cualquier característica que uno desee.

El primer PLD fue llamado PAL (arreglo lógico programable) o PLA dependiendo del esquema de programación utilizado. Estos solo utilizan compuertas lógicas (no flip-flops) permitiendo así solo la implementación de circuitos combinacionales. Para solventar este problema, PLD's que incluían un flip-flop en cada salida del circuito fueron lanzados poco tiempo después. Con esta mejora, se podían implementar simples circuitos secuenciales. Al principio de los 80´s, circuitería lógica adicional fue adherida a cada salida del circuito del PLD. La nueva célula de salida fue llamada macro célula, la cual contenía compuertas lógicas, multiplexores y flip-flops. Además la célula era completamente programable permitiendo con esto un sin número de operaciones. Adicionalmente a esto se incluyo una retroalimentación de la salida del circuito hacia el arreglo programable, lo cual daba una gran flexibilidad al PLD. Esta nueva estructura del PLD fue llamada PAL genérico o GAL. Una estructura similar fue conocida como PALCE (PAL CMOS erasable/ programable).

Todos estos dispositivos (PAL, PLA, PLD con registros, GAL / PALCE) fueron conocidos como SPLD (PLD simple). Tan solo el GAL/PALCE sigue siendo fabricado en un encapsulado único.

Tiempo después, un cierto número de dispositivos GAL fueron fabricados dentro de un mismo encapsulado usando un modo sofisticado de ruteado, una tecnología de manipulación del silicón mas moderna y un numero de atributos especiales (como el soporte JTAG y la interfaz sobre muchos estándares lógicos). Estos dispositivos se conocen como CPLDs. Los CPLDs son muy famosos hoy en día debido a su alta densidad, alto desempeño y bajo costo.

Finalmente a mediados de los 80s los FPGAs fueron introducidos. Estos dispositivos difieren de los CPLDs en arquitectura, tecnología, características de construcción, y costo. Estos se utilizan mayormente en la implementación de grandes circuitos y sistemas de desempeño superior.

La estructura de un FPGA depende de cada fabricante ya que cada uno tiene su propia arquitectura, pero en general todas son una variación del prototipo general. La arquitectura consiste de bloques lógicos configurables (por su definición en inglés CLB´s), bloques configurables de entradas y salidas (por su definición en inglés I/OBs), e interconexiones programables (Ver Figura B.1). Además, cuentan con circuitos y conexiones especiales para manejar y transportar las señales de reloj a cada bloque lógico, recursos lógicos adicionales como ALU´s, memorias, y procesadores también pueden estar disponibles.

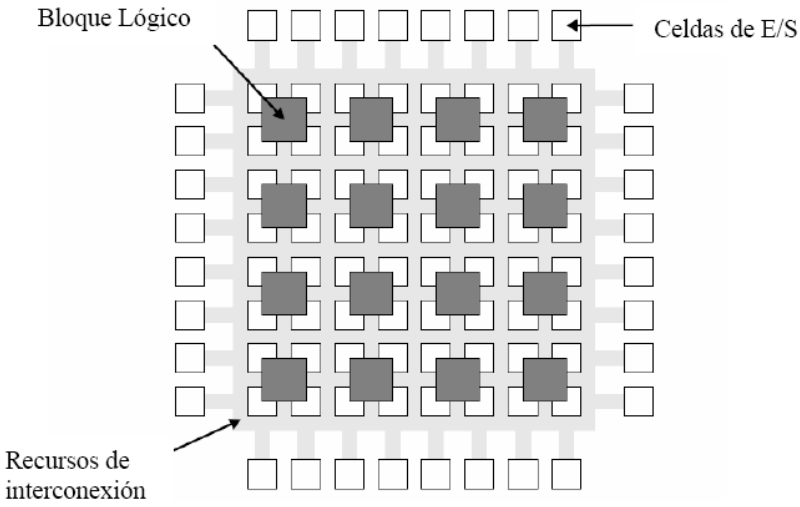


Figura B.1. Arquitectura interna de un FPGA [Torres, 2001].

Cada bloque lógico configurable es funcionalmente similar a los bloques lógicos de un CPLD. La diferencia está en que un FPGA normalmente utiliza generadores de funciones en vez de compuertas. Cada uno de estos generadores es como una memoria en donde en vez de implementar la función lógica mediante compuertas, se **precalcula** el resultado y se almacena en el generador. Las entradas al generador funcionan como un bus de direcciones, y mediante las diferentes combinaciones de las entradas al generador se selecciona el resultado correcto. Esto le da una gran densidad al dispositivo ya que se maneja un gran número de generadores, pero el tiempo de propagación al implementar una función lógica en estos generadores es menor al que se necesitaría si utilizáramos compuertas. La estructura de los bloques lógicos y las formas en que estos pueden ser interconectados, tanto salidas como entradas del bloque, varían de acuerdo al fabricante. En general un bloque lógico tiene menos funcionalidad que la combinación de sumas de productos y macroceldas de un CPLD, pero como cada FPGA tienen una gran cantidad de bloques lógicos es posible implementar grandes funciones utilizando varios bloques lógicos en cascada. Además de los bloques lógicos también es importante la tecnología utilizada para crear las conexiones entre los canales, las más importantes son las siguientes.

• **ANTIFUSE:** Al igual que la tecnología PROM, un FPGA que utiliza este tipo de tecnología sólo se puede programar una vez y utilizan algo similar a un fusible para realizar las conexiones. Una vez que éste es programado ya no se puede recuperar. La diferencia radica en que un fusible normal se desactiva deshabilitando la conexión, en tanto que estos *anti - fusibles* cuando son programados se producen una conexión por lo que normalmente se encuentran abiertos. La desventaja obvia es que no son reutilizables, pero por el contrario disminuyen considerablemente el tamaño y costo de los dispositivos.

SRAM: Los bloques SRAM son implementados como generadores de funciones para remplazar la lógica combinacional y, además, son usadas para controlar multiplexores e interconectar los bloques lógicos entre si. En estas el contenido se almacena mediante un proceso de configuración en el momento de encendido del circuito que contiene al FPGA. Ya que al ser SRAM, el contenido de estos bloques de memoria se pierde cuando se deja de suministrar la energía. La información binaria de los bloques SRAM generalmente se almacena en memorias seriales EEPROM conocidas como memorias de configuración. En el momento de encendido del

circuito toda la información binaria es transferida a los bloques del FPGA mediante el proceso de configuración el cual es generalmente automático y el propio FPGA contiene un circuito interno que se encarga de hacer todo el proceso.

Un FPGA que tiene una gran cantidad de canales de interconexión tiende a tener pequeños bloques lógicos con muchas entradas y salidas en comparación con el número de compuertas que tiene el bloque, este tipo de FPGAs generalmente utilizan tecnología ANTIFUSE. Un FPGA que tiene una estructura pequeña en canales de interconexión tiende a tener grandes bloques lógicos con pocas entradas y salidas en comparación con el número de compuertas que hay en el bloque. Este tipo de FPGA generalmente está fabricado con tecnología SRAM. Una arquitectura con bloques lógicos pequeños permite utilizar todos los recursos del dispositivo. Sin embargo, si los bloques lógicos son muy pequeños entonces tendremos que utilizar un gran número de estos para poder implementar funciones lógicas de varios términos, lo cual agrega un tiempo de retardo por cada bloque lógico implementado. Cuando el tamaño del bloque lógico es grande sucede lo contrario, en este tipo de bloques es posible utilizar un gran número de compuertas por lo que podemos implementar funciones lógicas de varios términos con pocos bloques lógicos. El que el tamaño de la celda sea grande no afecta la frecuencia máxima de trabajo porque estamos hablando de que existe un gran número de compuertas que pueden ser usadas en la función paralelamente, siendo el mismo tiempo de retardo para todas. Sin embargo, cuando las funciones son pequeñas en comparación con el tamaño del bloque no es necesario utilizar todas las compuertas del mismo, por lo que este tipo de bloques no son precisamente los más indicados para desempeñar pequeñas funciones. La tecnología SRAM es utilizada por Altera, Lucent Technologies, Atmel, Xilinx y otros. La tecnología ANTIFUSE es utilizada por Cypress, Actel, QuickLogic, y Xilinx.

Una diferencia fundamental entre un FPGA y un CPLD es la forma de almacenamiento de las interconexiones. Mientras que un CPLD es de forma no volátil, en un FPGA se utiliza una memoria RAM y por lo tanto es volátil. Esto resulta en una disminución de recursos y de espacio y por lo tanto un mayor número de interconexiones programables, aunque para esto requiere de una memoria de tipo ROM. Sin embargo, existen FPGAs no volátiles, los cuales presentan la ventaja de que la reprogramación no es necesaria.

Los FPGAs pueden ser bastante sofisticados. Existen chips fabricados con tecnología CMOS de 90mu, con 9 capas de cobre y más de 1000 pines de entrada/salida.

B.1.1 Estructura interna de una Virtex 4 fabricada por Xilinx. [Rivoallon, 2006]

La Virtex 4 usa internamente una unidad lógica llamada Slice. Esta unidad consiste en dos LUTs , cada una cuenta con cuatro entradas, dos multiplexores embebidos y la lógica de acarreo y dos registros, los Slices pueden subdividirse en dos medios Slice y contendrá solo la mitad de un Slice. En la figura B.2 se muestra esta estructura.

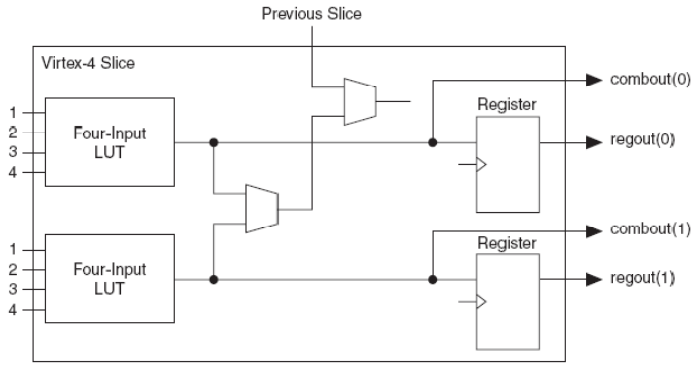


Figura B.2. Estructura interna de una Virtex 4

B.1.1.1 Microblaze, Procesador de núcleo Suave. [Moreno, 2006]

MicroBlaze es un microcontrolador de 32 bits con un conjunto de instrucciones reducidas, está disponible para las series de FPGA´s Spartan II, Spartan IIE, Spartan 3, Virtex, Virtex E, Virtex II, Virtex II PRO, Virtex 4 y Virtex V. Tiene un desempeño de 102 millones de instrucciones por segundo a 150MHz en un FPGA Virtex II PRO, requiere un mínimo de 900 celdas lógicas, cuenta con 32 registros de propósito general de 32 bits, soporta buses de memoria local (por sus siglas en inglés LMB) para acceso rápido a la memoria RAM del circuito, soporta el núcleo de conexión IBM para colocar periféricos en el bus, es compatible con el bus de periféricos del procesador PowerPC, y cuenta con herramientas completas para el diseño, desarrollo y depuración de sistemas [MicroBlaze PB, 2002].

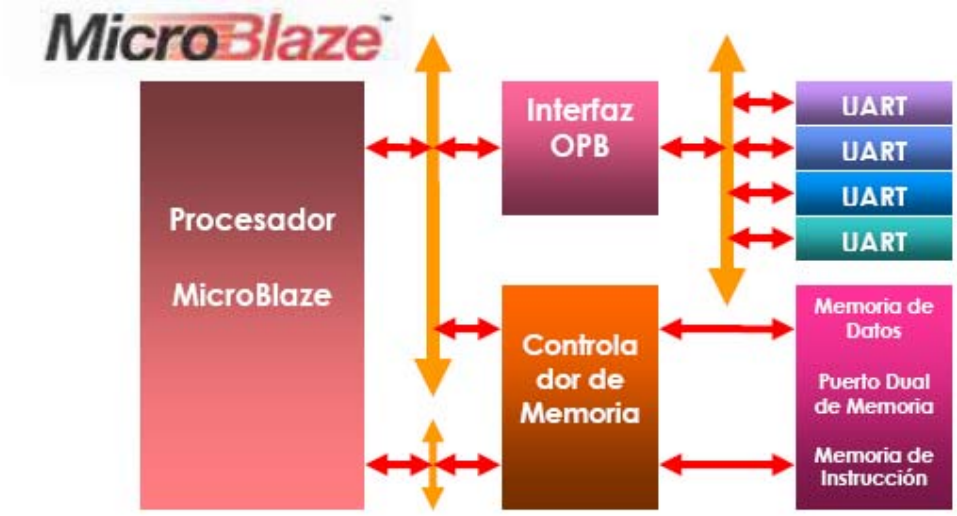


Figura B.3. Diagrama a Bloques de un Sistema Típico Basado en el Microcontrolador MicroBlaze

Con un alto desempeño de desarrollo el sistema de éste microcontrolador puede ser un reto hoy en día en los ambientes competitivos. No sólo se requiere un procesador para facilitar el uso de un sistema de un área, sino también optimizarlo en costo y características. El núcleo MicroBlaze ofrece un gran desempeño para crear sistemas potentes, con una completa serie de periféricos, memorias, puertos de entrada-salida de alta velocidad, compatibilidad con toda la lógica digital programable. Además es posible interconectar varios MicroBlaze con PowerPC en un mismo sistema embebido para crear un multisistema de procesamiento.

La construcción de un sistema embebido alrededor del núcleo MicroBlaze se compromete a un gran número de partes: el núcleo suave del procesador MicroBlaze, bloque de memoria RAM, bus de interconexiones estandarizado, y el bus de periféricos en circuito (por sus siglas en inglés OPB). El sistema MicroBlaze puede ser pequeño como el núcleo del procesador y un mínimo de memoria local hasta un largo sistema de varios MicroBlaze interconectados, grandes bloques de memoria externa, y un gran número de periféricos OPB. Las aplicaciones pueden ir desde una simple máquina de estados, hasta controladores complejos para aplicaciones de Internet u otras aplicaciones embebidas.

El núcleo suave del procesador MicroBlaze es el corazón del sistema embebido MicroBlaze. La memoria de instrucción y la memoria de datos, cada una tiene una interfaz conectada al bus de memoria local (por sus siglas en inglés LMB). El sistema puede ser construido de manera estricta a una arquitectura Hardvard, o, compartir recursos con un único bus OPB utilizado en conjunto con el árbitro del bus (dado como un periférico de MicroBlaze). Ya que los requerimientos del sistema son variados, el núcleo MicroBlaze brinda 6 combinaciones para organizar el bus de memoria local (LMB) y el bus de periféricos en circuito (OPB). El bus de memoria local (LMB) accesa al bloque de memoria RAM en un solo ciclo de reloj. Este es un eficiente y único protocolo de bus maestro, ideal para interconectar interfaces rápidas a la memoria local. El bus de periféricos en circuito (OPB) cuenta con 32 bits y es un bus multimaestro, el cual es ideal para conectar periféricos y memoria externa al núcleo del MicroBlaze [MicroBlaze PB, 2002].

El núcleo del procesador MicroBlaze esta incluido en el conjunto de desarrollo embebido (por su definición en inglés Embedded Development Kit) de Xilinx. El EDK de un conjunto estandarizado de infraestructura, interfaces, núcleos IP y periféricos [MicroBlaze FS, 2003].

Arquitectura de MicroBlaze

El núcleo suave MicroBlaze es un conjunto de instrucciones de cómputo reducidas (RISC) que ha sido optimizado para su implementación en FPGA´s de Xilinx. Cuenta con 32 registros de propósito general de 32 bits cada uno, palabras de instrucciones de 32 bits con tres *operandos* y dos modos de direccionamiento, buses separados de instrucción y datos conforme a la especificación de OPB´s de IBM (Ver Figura B.4), soporte de multiplicadores de hardware en los FPGA´s Virtex II en adelante [MicroBlaze HW, 2002].

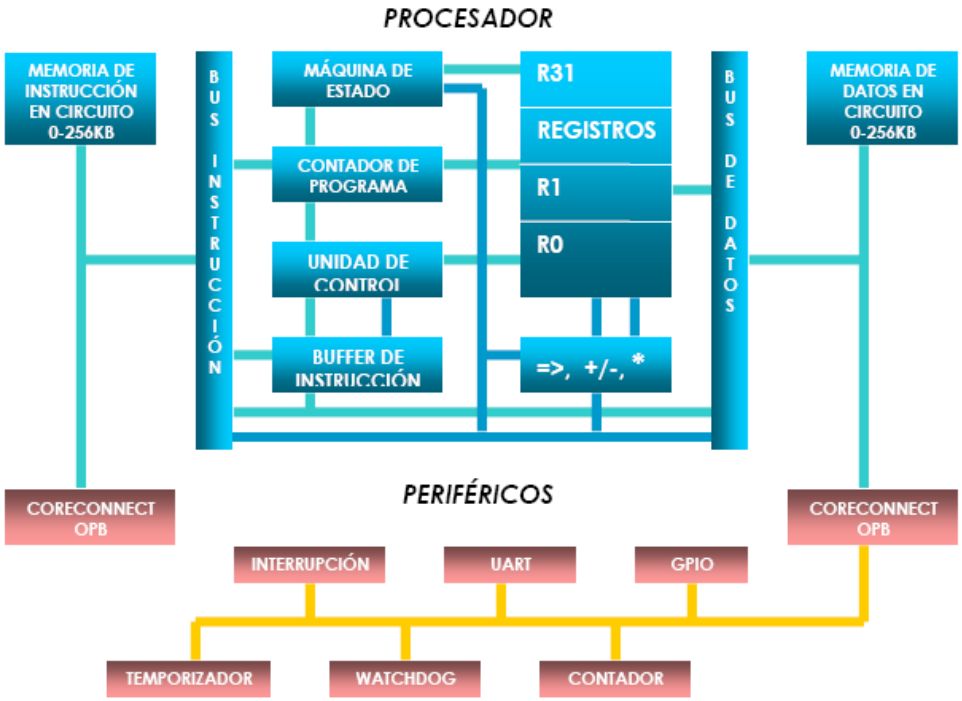


Figura B.4. Diagrama a Bloques de la Arquitectura del Microcontrolador MicroBlaze

B.1.2. Estructura interna de una Stratix II fabricada por Altera [Altera Corporation 2007].

La unidad lógica para la Stratix II son llamados modulo de lógica adaptiva (ALM) y esta formada por dos registros programables, dos bloques de sumadores lógicos, multiplexores embebidos y una LUT (Look-table) que consiste en la lógica combinacional de ocho entradas. Dadas las características de la arquitectura de la Stratix, esta LUT puede ser dividida en dos ALUTs (Adaptive Look-Up Table), con el propósito de que las ocho entradas puedan ser divididas en varias combinaciones posibles. De este modo, si una función requiere seis entradas y otra requiere dos entradas, ambas funciones pueden compartir el mismo ALM de tal forma que usen pocos recursos. En la Figura B.5, se describe esta estructura.

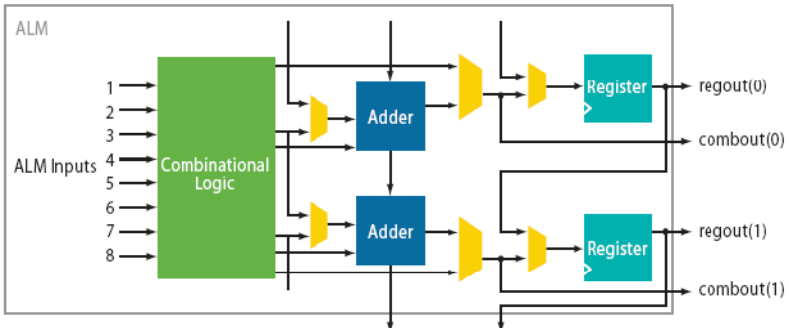


Figura B.5. Estructura interna de una Stratix II

C.1. Introducción a los DSP's

Uno de los elementos más importantes para el procesamiento de datos es el DSP, del cual se dará una definición de acuerdo a Barrero [Barrero 2005].

Un procesador digital de señales es un dispositivo con capacidad de procesamiento en línea, o tiempo real, de información que presenta, a la vez, de tener características de microprocesador y microcontrolador. Posee una CPU de gran potencia de cálculo preparada para el tratamiento digital de señales en tiempo real y para la realización del mayor número de operaciones aritméticas en el menor tiempo posible. Por lo tanto, su circuiteria interna ha sido optimizada para la implementación de funciones tales como el filtrado, la correlación, el análisis espectral etc., de una señal digital de entrada al sistema.

Con respecto a su estructura interna corresponde a una arquitectura Harvard ya que con ella se logra acelerar la ejecución de instrucciones y la realización de las operaciones aritméticas.

De la gran oferta que existe en el mercado se seleccionó un DSP fabricado por Texas Instrument. Este fabricante tiene un gran repertorio de dispositivo del cual por sus características se selecciono la serie TMS320C6000 y más particular el C6416 ya que ofrece mayor potencia de cálculo, posibilidad de funcionar con relojes de hasta 1 GHz Además de trabajar con 32 bits en punto flotante.

En la figura C.1 se resume la arquitectura y los periféricos de la generación TMS320C64x. y en la siguiente lista se describen brevemente cada uno de ellos.

- Tres temporizadores de propósito general de 32 bits
- Interfaz externa EMIF, que soporta conexión directa con una amplia variedad de dispositivos, como memorias asíncronas, memorias DRAM síncronas, memorias SBSRAM, ETC. Además cuenta hasta dos interfaces externas, EMIFA y EMIFB, esta última con un bus de datos de 16 bits.
- Hasta tres McBSP (Multichannel Buffered Serial Port)) para la comunicación serial con otros dispositivos.
- Interfaz PCI (Peripheral Component Interface), que permite la conexión del DSP a otro dispositivo que disponga de este tipo de interfaz, funcionando como maestro o esclavo.
- Interfaz HPI (host port interface). Puede ser de 32 bits, de 16 o ambos.

- Entradas y salidas de propósito general, GPIO.
- Modulo I2C, que permite transmitir y recibir hasta 8 bits a dispositivos compatibles I2c.
- Modulo selector de interruptores. Del cual el CPU solo cuenta con 12 entradas de interrupción.
- UTOPIA (Universal Test and Operations Physical layer Interface for ATM). Se trata de un controlador de ATM esclavo que permite la conexión a un controlador ATM MAESTRO.

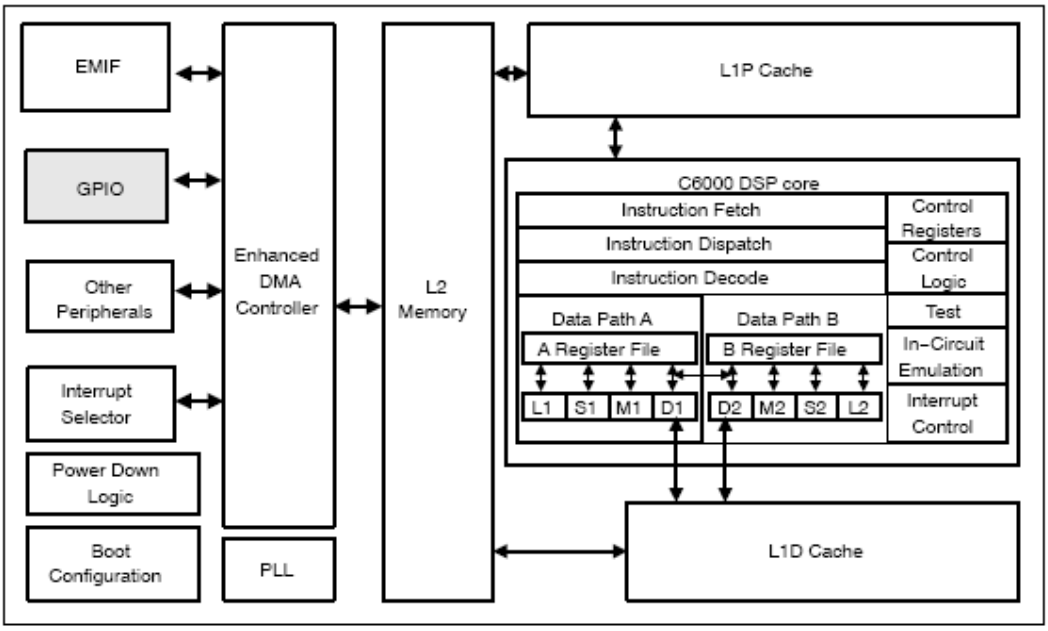


Figura C.1. Diagrama a bloques del TMS320C64x

La CPU posee una arquitectura de tipo VLIW (Very Long Instruction Word), que contiene muchas unidades funcionales que operan en paralelo, permitiendo de esta forma ejecutar varias instrucciones en un mismo ciclo de reloj. En la figura B.6 se pueden apreciar un total de ocho unidades funcionales, dos bloques de registros, A y B, de 32 registros de 32 bits cada uno, y dos data-path. Dos de las unidades funcionales, .M1 y .M2, contienen cada una un multiplicador, con capacidad para realizar dos productos de 16 x 16 bits en un ciclo maquina. O bien, cada multiplicador puede realizar cuatro multiplicaciones de 8 x 8 bits cuando se este utilizando imágenes.

La estructura de ejecución de instrucciones es supersegmentada con un pipeline de once fases: las tres primeras corresponden a la recogida de la instrucción, las dos siguientes a la decodificación y las últimas cinco a la ejecución.

D.1 Consideraciones de los circuitos impresos para el manejo de las señales LVDS

Algo importante por mencionar es el diseño del circuito impreso. Existen muchas reglas y consideraciones para lograr un buen funcionamiento del circuito impreso por lo que en este apartado se mencionarán ciertas reglas que se tienen que seguir al momento de diseñar el circuito impreso. Todas o la mayoría de estas reglas están basadas en la buena manipulación de las señales LVDS, las cuales necesitan un trato especial al momento de implementarlas en un circuito impreso.

Consideraciones para diseño de circuitos impresos de acuerdo con [National, 2004, A].

Una lista detallada para el diseño de circuitos impresos se muestra a continuación

Use al menos 4 capas de circuito impreso (de la tapa al fondo): señales LVDS, tierra, alimentación y señales TTL. Planos dedicados a Vcc y a GND son requeridos para diseños de alta velocidad. El plano de tierra es requerido para estabilizar una impedancia controlada para las interconexiones de las líneas de transmisión. Un espacio estrecho entre Vcc y tierra creara de igual forma una excelente capacitancia de desacoplo de alta frecuencia.

Aislar las señales CMOS/TTL de alta velocidad de las LVDS, debido al ruido de la terminación única de las señales CMOS/TTL se acoplara en las líneas LVDS. Es mejor colocar las señales TTL y las LVDS en diferentes capas, las cuales estén aisladas por capas de alimentación o de tierra.

Mantener los controladores y los receptores LVDS lo mas cerca posible de los conectores. Esto ayuda a asegurarnos que el ruido de la tarjeta no se acopla en las líneas diferenciales o no escapara de la tarjeta como EMI mediante el cable interfaz. Esta recomendación también ayuda a minimizar el skew entre líneas. El skew tiende a ser proporcional a la longitud de las líneas; por lo tanto longitudes pequeñas resulta en niveles de skew pequeños.

Un desacoplo de cada dispositivo LVDS ayuda a distribuir la capacitancia parasita. Capacitores de montaje superficial colocados cerca de los pines de alimentación o tierra son los mejores.

Otros aspectos a considerar dentro del diseño de un circuito impreso se cuentan los siguientes:

Alimentación.- un capacitor de tantalio de 4.7uF o de 10uF de 35 volts trabajan de buena forma al conectarlo entre la alimentación y tierra. Se escoge este tipo de valores ya que son los que mejor filtran las componentes más grandes (mayormente 100Mhz a 300Mhz) de la frecuencia alimentación/tierra. Esto puede ser verificado mediante el espectro de ruido de Vcc a través del capacitor de desacoplo. El nivel de voltaje del capacitor no debe de ser menos de 5xVcc. Algunos capacitores electrolíticos trabajan de igual forma.

Pines de Vcc.- uno o dos capacitores cerámicos de montaje superficial multicapa (MLC) de valores 0.1uf o 0.01uf deben de conectarse en paralelo entre el pin de Vcc y tierra. Para mejores resultados los capacitores deben de estar lo mas cerca posible de los pines de Vcc para minimizar los efectos parásitos que puedan afectar la respuesta en frecuencia de la capacitancia.

Dispositivos LVDS con un gran número de bits (>4 bits) o con alguna implementación de un PLL (por ejemplo channel link o fpd-link) deben de tener al menos dos capacitores por tipo de alimentación, mientras que otros tipos de dispositivos están bien con un capacitor de 0.1uF.

Los problemas por EMI mayormente empiezan con la distribución de alimentación y tierra. El EMI puede ser reducido de gran manera manteniendo los planos de alimentación y tierra de manera separada. Como regla se debe mantener un ruido menor a 100mV en las líneas de alimentación en cuando al desacoplo de estas, sin embargo existen dispositivos con requerimientos mas rigurosos por lo que se deben de consultar las hojas de datos.

Para el trazado de las Pistas según [National, 2004, A] se deben tomar en cuenta las siguientes recomendaciones:

Las pistas de alimentación y tierra deben de ser anchas (de baja impedancia), su trabajo es mantener un punto de baja impedancia. No use las reglas de diseño de 50Ω para estas pistas.

Mantenga las trayectorias de regreso de los circuitos impresos cortas y anchas. Proveer de una trayectoria de regreso que logre un lazo para el regreso de las imágenes de las corrientes más pequeñas.

Los cables deben de emplear un conductor conectado a la tierra de los dos sistemas interconectados. Esto provee para las corrientes de modo común una trayectoria de regreso conocida lo cual es importante para aplicaciones box-to-box donde las trayectorias de regreso a tierra pueden ayudar a eliminar oscilaciones en el potencial de tierra.

Utilizar dos vías para conectar la alimentación y la tierra mediante capacitores de desacoplo para minimizar los efectos inductivos. Capacitores de montaje superficial son buenos, dado a su tamaño compacto y la posibilidad de colocarse más cerca de los pines.

Microstrip con los bordes acoplados, stripline con los bordes acoplados o striplines de lados amplios trabajan bien para líneas diferenciales.

Las pistas para señales LVDS deben de estar estrechamente acopladas y diseñadas para una impedancia diferencial de 100Ω

Las líneas de microstrip de bordes acoplados ofrecen la ventaja que la impedancia diferencial Zo puede ser más grande (de 100Ω a 150Ω). También es posible rutear del pin del conector hacia el pin del dispositivo sin ninguna vía. Esto provee un ruteo “mas limpio”. Una limitación de las líneas de microstrip es que solamente se pueden rutear en las dos capas de afuera del PCB, lo cual impone un límite en la densidad de ruteo por canal.

El stripline puede ser tanto con bordes acoplados como de lados amplios. Puesto que están embebidos en el stack de la tarjeta y se encuentran entre capas de tierra, tienen una característica de un blindado mayor. Esto limita la radiación y el acople de ruido en ambas líneas. Sin embargo se requiere del uso de vías para conectarse entre si.

Para el trazado de las Pistas diferenciales de acuerdo con [National, 2004, A] se debe tener las siguientes consideraciones:

1.- Use pistas de PCB con impedancia controlada que coincidan con la impedancia del medio de transmisión y de la terminal resistiva. Rute las pistas de pares diferenciales tan cerca como sea posible y tan pronto como salgan del integrado. Esto ayuda a eliminar reflexiones y asegura que el ruido de modo común este acoplado en el par. Señales que están apartadas 1 mm radian menos que las que están apartadas 3 mm, puesto que la cancelación de campos magnéticos es mayor cuando el espacio entre pistas es mas estrecho. Además el ruido inducido en las líneas diferenciales es tomado mayormente como ruido de modo común lo cual es rechazado en el receptor.

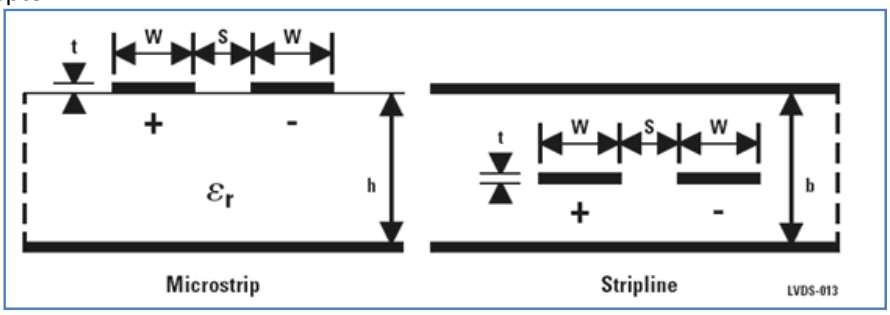


Figura D.1. Pares diferenciales en Microstrip y Stripline

Cuando se diseña una impedancia diferencial específica Zo (Z_{diff}) para líneas de bordes acoplados es mejor variar el ancho de pista W para poder alterar Z_{diff}. No es recomendable ajustar S puesto que es el mínimo espacio especificado por el proveedor de PCB´s para el espacio entre líneas. Uno puede obtener ciertas especificaciones de la impedancia para un circuito con las siguientes formulas.

Microstrip

$$Z_{DIFF} \approx 2 \times Z_0 \left(1 - 0.48 e^{-0.96 \frac{S}{h}}\right) \Omega$$

Stripline

$$Z_{DIFF} \approx 2 \times Z_0 \left(1 - 0.374 e^{-0.29 \frac{S}{h}}\right) \Omega$$

Microstrip

$$Z_0 = \frac{60}{\sqrt{0.457\varepsilon_r + 0.67}} \ell n \left(\frac{4h}{0.67\left(0.8W + t\right)}\right) \Omega$$

Stripline

$$Z_0 = \frac{60}{\sqrt{\varepsilon_r}} \ell n \left(\frac{4b}{0.67\pi\left(0.8W + t\right)}\right) \Omega$$

Note: For edge-coupled striplines, the term "0.374" may be replaced with "0.748" for lines which are closely coupled (S < 12 mils).

Las estructuras de líneas de acoplo de costado también se pueden utilizar. Las dimensiones para este tipo de estructura se muestran en la figura D.2. Las striplines de acoplo de costado pueden ser de mucha utilidad para diseños de blackplanes mientras que se utilicen un solo canal de ruteo lo cual resulta en una facilidad de conexión con los pines del conector.

No existe una ecuación absoluta para calcular la impedancia de striplines de acoplo de costado. En vez de esto una solución de campo puede ser utilizada.

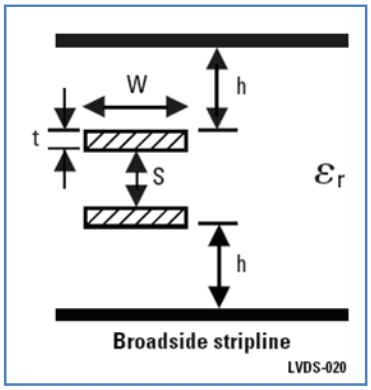


Figura D.2. Stripline de acoplo de costado (Broadside couple)

Utilice siempre dimensiones consistentes cuando se realicen los cálculos de s, w, h y t. Valores comunes de ε_r (constante dieléctrica) para varios tipos de materiales de tarjetas de circuito impreso se muestran en la tabla D.1. Para un buen diseño de la tarjeta de circuito impreso es necesario consultar las especificaciones del proveedor para mayor seguridad

PCB material	Dielectric constant (´r)	Loss tangent
Air	1.0	0
PTFE (Teflon)	2.1 to 2.5	0.0002 to 0.002
BT resin	2.9 to 3.9	0.003 to 0.012
Polyimide	2.8 to 3.5	0.004 to 0.02
Silica (quartz)	3.8 to 4.2	0.0006 to 0.005
Polyimide/glass	3.8 to 4.5	0.003 to 0.01
Epoxy/glass (FR-4)	4.1 to 5.3	0.002 to 0.02
GETEK	3.8 to 3.9	0.010 to 0.015 (1 MHz)
ROGERS4350 core	3.48 ± 0.05	0.004 @ 10G, 23°C
ROGERS4403 prepreg	3.17 ± 0.05	0.005 @ 10G, 23°C

Tabla D.1 – Propiedades de materiales para PCB

2.- Iguale las longitudes entre pistas diferenciales para un menor skew. El skew entre los pares diferenciales puede implicar en una diferencia de fase entre las señales. La diferencia de fase puede destruir la característica de la cancelación de campos magnéticos de los pares diferenciales y esto puede resultar en una producción de EMI. Una regla general sería mantener una diferencia entre longitudes de máximo 100mils.

3.- No confíe en el auto ruteo de los programas, revise los resultados cuidadosamente para asegurar que las longitudes entre pares diferenciales coincidan y tengan el aislamiento necesario.

4.- Minimice el número de vías y de discontinuidades de las líneas.

5.- Evite vueltas de 90˚ (esto causa discontinuidad en la impedancia). Use arcos de 45 para sustituir.

6.- Dentro de un par de pistas, la distancia entre ellas debe de ser minimizada para mantener el rechazo de modo común en los receptores. En la tarjeta de circuito impreso, esta distancia debe de mantenerse constante para evitar discontinuidades en la impedancia diferencial. Son permitidas pequeñas violaciones en los puntos de conexión. La clave para el desequilibrio es mantenerlo lo menos y lo más pequeño posible. Las transmisiones diferenciales trabajan mejor en interconexiones balanceadas. Para mejores resultados cada pista del par debe de ser completamente igual.

Un pequeño ejemplo grafico de errores en el circuito impreso que pueden resultar en skew, producción de EMI, etc., se muestra en la figura D.3.

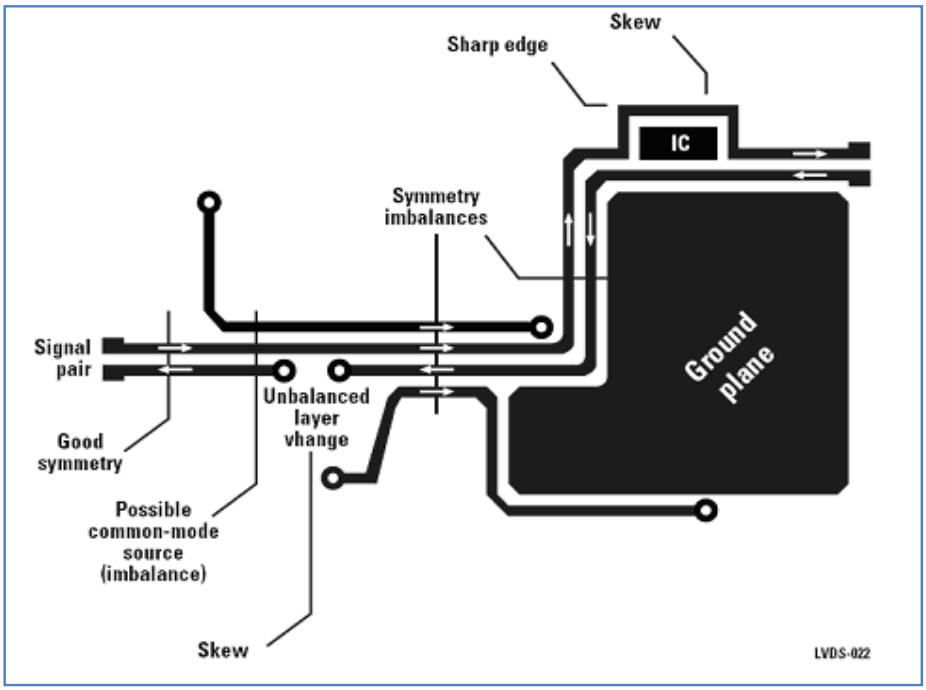


Figura D.3. Ejemplos de errores en el circuito impreso para pares diferenciales

De acuerdo con [National, 2004, A] las consideraciones para las terminaciones con pares diferenciales o single-ended que se deben de tomar en cuenta es lo siguiente:

Use terminales resistivas que coincidan con la impedancia diferencial de la línea de transmisión. Este debe de estar entre 90 y 130 para cables de aplicaciones punto a punto. Recuerde que la corriente de salida de los controladores necesita de una terminal resistiva para generar el voltaje diferencial apropiado. El LVDS no podrá trabajar sin esta resistencia.

Típicamente, una simple resistencia pasiva a través del par diferencial en el receptor es suficiente.

Resistencias de montaje superficial son mejores. Los stubs del PCB, los componentes principales y la distancia entre la terminal resistiva y el receptor deben de ser minimizados. La distancia entre la terminal y las entradas del receptor debe de ser menor a 7mm (12mm es el máximo).

Una tolerancia de la resistencia de 1% o de 2% es el recomendado. Con respecto al punto de vista de reflexión una tolerancia de 10% se traduce en una reflexión del 5%. Mientras más cerca del valor nominal sea la resistencia mejor. Iguale la impedancia diferencial nominal de la

interconexión. Para aplicaciones multicaida o multipunto, iguale la impedancia diferencial de todo el sistema.

Un capacitor entre dos resistencias de 50 puede ser utilizado como terminal (ver figura D.4.), esto es para filtrar el ruido de modo común, si el aumento de componentes no interesa. Este tipo de terminación es la más usada y requerida.

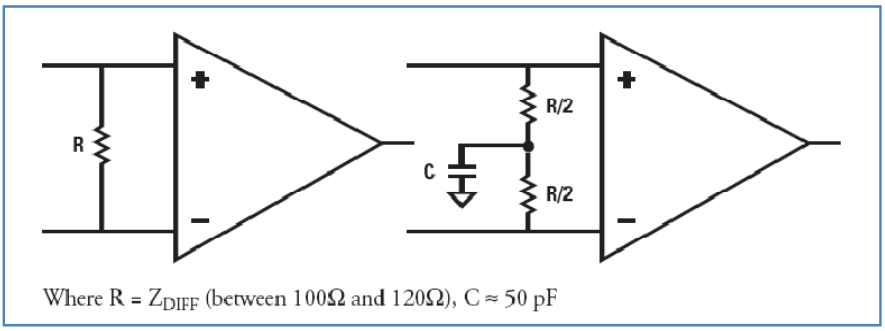


Figura D.4. Ejemplo de terminación en pares diferenciales

La regla de la S de a cuerdo con [National, 2004, A].

La utilización de la distancia entre límites de las pistas del par diferencial, s esta definida por:
La distancia de separación entre dos pares diferenciales debe de ser mayor a 2s
La distancia entre pares diferenciales y señales TTL/CMOS deben de ser mayor de 3s como mínimo. Sería más recomendable utilizar una capa diferente para estas señales junto con un plano de tierra entre la capa TTL/CMOS y la LVDS para un mejor aislamiento.
Si una pista de tierra es utilizada, esta debe de estar a una distancia mayor de 2s.

D.1.1. Integridad de señal

A pesar de que la integridad de señal es de lo más importante dentro de un diseño, la comunidad de diseñadores la ha ignorado durante mucho tiempo. A través de la época de los diseños de lógica de baja velocidad, el estudio de integridad de señal resultaba una pérdida de esfuerzo, debido a que las probabilidades de defectos en la transmisión resultaban casi nulas. Sin embargo al tiempo que las frecuencias de reloj fueron incrementadas y los tiempos de puesta en marcha fueron decrementandose, la necesidad de un análisis de integridad de señal de igual forma se incrementó. Desgraciadamente muchos diseñadores siguen sin realizar el estudio de integridad de señal a pesar de los problemas existentes.

Circuitos modernos pueden operar en frecuencias de Gigaherz con tiempos en el orden de 50 picosegundos. A estas velocidades, un diseño de circuito impreso sin cuidados solamente necesita de una pulgada o menos para radiar energía. Las pistas con radiación producen voltajes y tiempos erróneos, interferencia y otros problemas que no solo afectan a esa línea sino que abarcan toda la tarjeta y puede llegar hasta las tarjetas adyacentes. El problema resulta más crítico en el momento de que se manejan diferentes señales dentro de una tarjeta.

El diseño para tener una integridad de señal no debe de ser un arduo trabajo. La clave para la integridad de señal es el reconocimiento de los problemas tan pronto surjan o mejor aun antes.

D.1.2. Aislamiento de acuerdo con[Grant, 2002].

Los componentes dentro de un circuito impreso tienen diversas características tanto de frecuencia como de tolerancia al ruido entre otras. El método más directo para mejorar la integridad de señal es aislar los componentes de un circuito de acuerdo a sus características. Un ejemplo se muestra en la figura D.5, en ella se puede apreciar una buena o mala ubicación de todos los módulos que se pueden integrar en una tarjeta de circuito impreso.

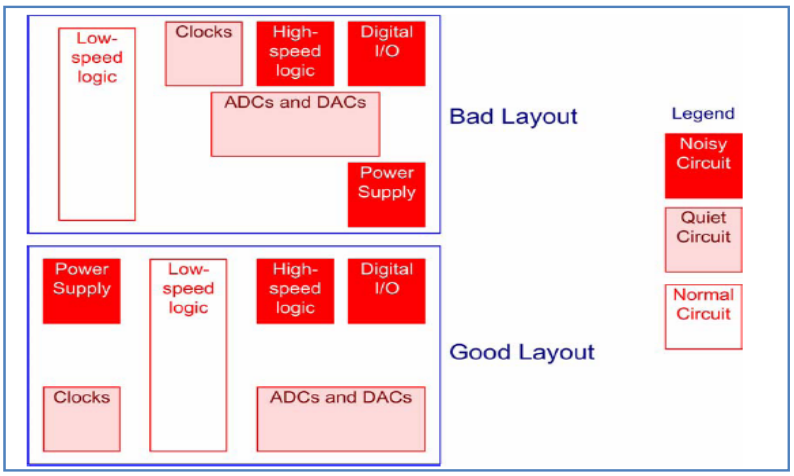


Figura D.5. Ejemplo de la distribución de los módulos dentro de un circuito impreso

En el primer circuito impreso se muestra a los módulos con alta emisión de ruido adyacentes a circuitos con baja tolerancia al ruido (reloj y convertidores). El segundo circuito impreso es mejor debido a que los módulos de alta emisión de ruido son aislados mediante el modulo de lógica de baja velocidad el cual no es tan susceptible al ruido como los módulos de reloj y de los convertidores.

D.1.3. Impedancia, reflexiones y terminación de acuerdo con [Grant, 2002].

El control de la impedancia y las terminaciones en sus líneas es fundamental en el diseño de circuitos impresos, que manejan altas frecuencias. Sin embargo muchos diseñadores de circuitos impresos que manejan frecuencias mayores a las frecuencias RF se niegan a considerar la impedancia y las terminaciones en sus diseños.

Las desigualdades en las impedancias producen un severo daño a la integridad de las señales en los circuitos digitales tal como se muestra en la figura D.6.

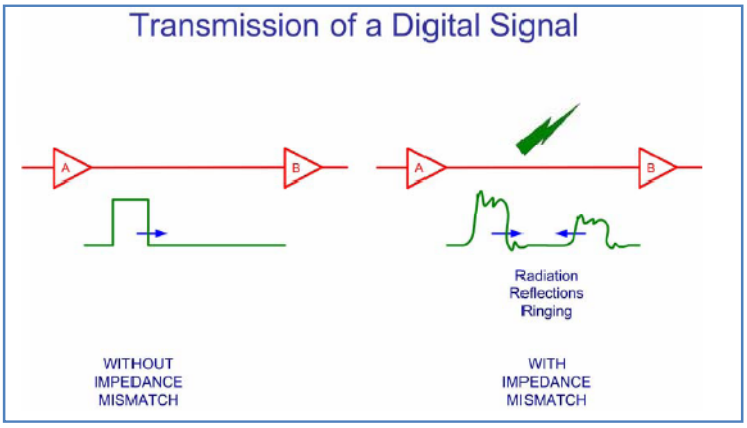


Figura D.6. Ejemplo de la trasmisión de una señal digital

Las señales digitales son reflejadas entre la entrada del dispositivo receptor y las salidas del dispositivo transmisor. Las señales reflejadas van de un lado a otro en la línea de transmisión hasta que son absorbidas por elementos resistivos. Las señales reflejadas producen una oscilación en las señales que son enviadas a través de las líneas de transmisión. Estas oscilaciones producen una afectación en el voltaje y en los tiempos de la señal causando efectos negativos en las pistas.

Los problemas que aparecen por causa de la desigualdad de impedancias pueden ser resueltas mediante las terminaciones en la línea. Las terminaciones son usualmente dos o tres elementos pasivos colocados en la entrada del modulo receptor.

Las resistencias limitan el tiempo del flanco de subida de las señales y absorben parcialmente la energía radiada. Es importante notar que el resistor no elimina totalmente los efectos destructores de la desigualdad de impedancias. Sin embargo una buena selección de la configuración y los valores en los componentes pueden absorben gran cantidad de los efectos nocivos en la integridad de señal

Nótese que no todas las pistas necesitan un control en sus impedancias. Algunos estándares como el PCI compacto requieren una impedancia de pista y terminales específicas. Cumpliendo con estos requerimientos, estos estándares son bastante efectivos al momento de minimizar reflexiones, oscilaciones y emisiones de las líneas de transmisión.

Otros estándares no especifican estos parámetros por lo que se deja al criterio del diseñador cuando se debe implementar terminales.

D.1.4. Planos y su división de acuerdo con [Grant, 2002].

Un hecho que muchas veces ignoran los diseñadores es que la corriente viaja a través de lazos. Por ejemplo considérese una señal de simple terminal (single ended) que es transmitida a través de dos compuertas como se muestra en la figura D.7. La corriente viaja a través de la compuerta "A" hacia la "B" y viaja de retorno de "B" hacia la "A" a través de la conexión a tierra.

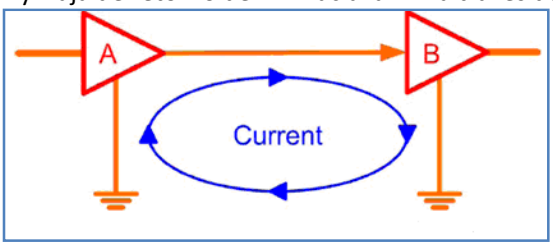


Figura D.7. Flujo de la corriente de punto a punto

Existen dos problemas potenciales:

Las tierras necesitan ser conectadas a través de una trayectoria de baja impedancia.

Si las tierras son conectadas a través de una trayectoria de alta impedancia, existirá una caída de voltaje entre los pines de tierra, lo cual causara interrupciones en todos los dispositivos referenciados a ese plano de tierra lo cual degradara los márgenes de ruido de entrada.

El área del lazo formada por el lazo de la corriente debe de ser la mínima posible. Los lazos actúan como antenas. Un área de lazo mayor incrementara las posibilidades de que el lazo radie y conduzca. Esto debe de ser una meta para cualquier diseñador de circuitos impresos que la corriente de retorno pueda viajar por debajo de la pista de la señal, en el lazo mínimo.

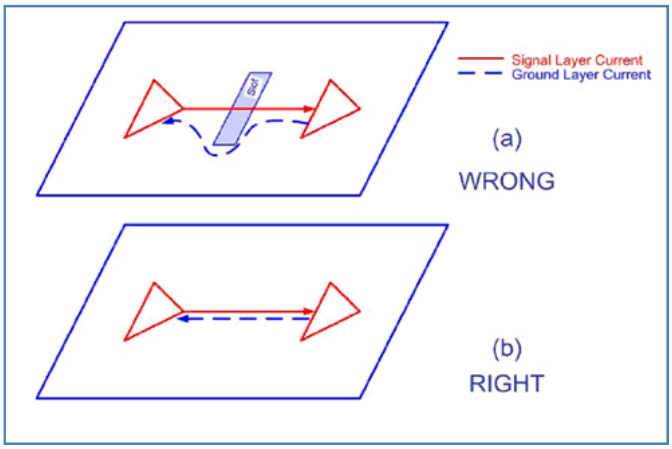


Figura D.8. Ejemplo del flujo de la corriente cuando se tiene un slots dentro de un circuito impreso

Usando un plano para la tierra se solucionan muchos problemas de este tipo. Un plano de tierra nos da una trayectoria de baja impedancia para todos los puntos de tierra proviniéndonos de una trayectoria corta de la corriente de retorno.

Un error común de los diseñadores es la utilización de huecos y ranuras en el plano de tierra. La figura D.8a muestra como la corriente de retorno viaja a través de una ranura en el plano de tierra. La corriente es forzada a bordear la ranura lo cual conlleva a la creación de un lazo mucho mayor. En la figura D.8b no existe ninguna ranura lo cual disminuye el lazo de la corriente.

Las ranuras no deben de utilizarse en los planos de tierra. Sin embargo existen ocasiones donde las ranuras no pueden ser evitadas. Cuando esto pase el diseñador del circuito impreso debe de asegurarse que ninguna señal se ruten a través de la ranura. Las mismas reglas aplican a circuitos impresos que contienen diferentes señales excepto cuando se utilicen múltiples planos de tierra. Esto es típico para un convertidor de alto desempeño que puede utilizar tierras separadas tanto para la parte analógica como para la parte digital. Cuando se utilizan diferentes tipos de tierra las ranuras no pueden ser evitadas aunque los diseñadores del circuito impreso deben de asegurarse que no se ruten las señales sobre estas ranuras

Por la misma razón, se aplica a los planos de alimentación, con una regla adicional: ocasionalmente un circuito impreso diseñado con planos de alimentación y de tierra pueden radiar a través de los límites de la placa. La energía electromagnética emitida en estos límites puede interrumpir el funcionamiento de las tarjetas adyacentes. Esta situación se muestra en la figura D.9. La solución mostrada es disminuir el plano de alimentación para que el plano de tierra lo sobrepase cierta distancia. Esto disminuye la cantidad de energía electromagnética emita hacia las tarjetas que están en un área inmediata y reducirá el impacto sobre estas tarjetas adyacentes.

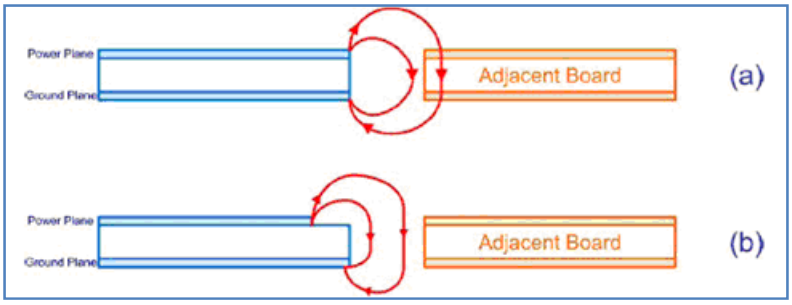


Figura D.9. Ejemplos de emisiones de energía en los límites del circuito impreso

D.1.5. Crosstalk o interferencia de acuerdo con [Grant, 2002].

La interferencia es otra de las mayores preocupaciones para un diseñador de circuitos impresos. La figura D.10 muestra un corte a un circuito impreso indicando tres pistas paralelas y sus respectivos campos electromagnéticos. Cuando el espacio entre ellos es demasiado corto el campo electromagnético de las pistas influyen entre ellos y posiblemente causen problemas en su información. Esto es la llamada interferencia.

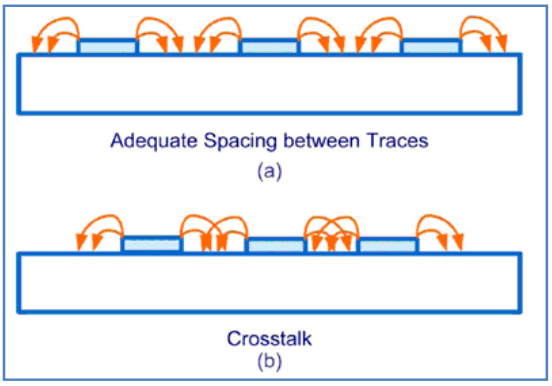


Figura D.10. Ejemplos de interferencia entre pistas de un circuito impreso

La interferencia puede reducirse aumentando las distancias entre pistas. Sin embargo los diseñadores de circuitos impresos están constantemente presionados para reducir el tamaño de los circuitos impresos, por lo tanto reducir el espacio entre pistas. También, existen casos en los que los diseñadores no pueden de ninguna forma reducir el crosstalk dentro de su circuito impreso, por lo que es necesaria una estrategia para poder lidiar con este tipo de ruido. Muchas reglas han sido publicadas durante años sobre el espacio aceptable entre pistas. Una regla común es la de 3W donde el espacio entre pistas debe de ser al menos tres veces mayor que el ancho de pista. Sin embargo el espacio realmente aceptable entre pistas varía dependiendo de la aplicación, el medio y los márgenes del diseño. Los espacios entre pistas varían de situación a situación y debe de ser calculada para cada una. Además existen casos donde la interferencia no puede ser evitada y por lo tanto se debe cuantificar los daños que resultan de ella. En estos casos no existen sustitutos para la simulación por computadora. Un buen ejemplo de estos temas es en los conectores de alta densidad y de alta velocidad. En estos casos el diseñador conoce la cantidad de interferencia resultante y puede calcular los efectos nocivos que pueden producir esta interferencia.

D.1.6. Métodos de verificación utilizando el diagrama de ojo de acuerdo con [Dinamarca, 2002].

El diagrama de ojo es una eficiente herramienta para la medición de la calidad total de las señales digitales al final de la línea de transmisión. Esto incluye todos los efectos de la distorsión sistemática y variable y muestra el tiempo durante el cual la señal se considera válida. La figura D.11 muestra los atributos significativos del patrón del ojo.

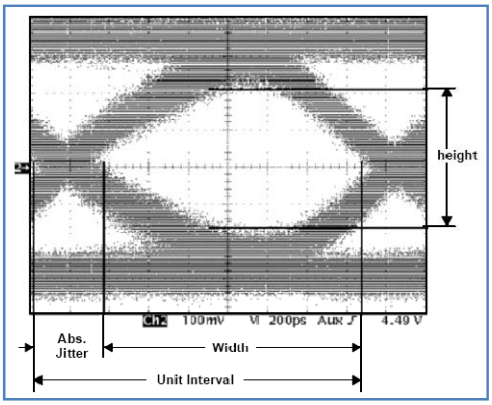


Figura D.11. Diagrama de ojo típico

Cuando no se pueda determinar el origen de un problema en un enlace digital los diseñadores de circuitos impresos recurren al diagrama de ojo para el análisis de las formas de onda de los pulsos que se propagan en un enlace de comunicaciones, para lograr observar sus formas, desfases (jitter), niveles de ruido, potencias de las señales, etc.

El diagrama de Ojo, muy utilizado en el análisis de formas de ondas en telecomunicaciones digitales, corresponde esencialmente, a un diagrama que muestra la superposición de las distintas combinaciones posibles de unos y ceros en un rango de tiempo o cantidad de bits determinados. Dichas señales transmitidas por el enlace, permiten obtener las características de los pulsos que se propagan por el medio de comunicación, sean estos por medio de fibra óptica, coaxial, par trenzado, enlaces satelitales, etc.

Por ejemplo en una secuencia de 3 bits tenemos una cantidad total de 8 combinaciones posibles, las que pueden ser observadas en la figura D.12. Debido a la capacidad de los diagramas de ojo de representar la superposición de varias señales simultáneamente es que son conocidos como patrones multi-valores, ya que a diferencia de las señales medidas normalmente en un osciloscopio, cada punto en el eje del tiempo tiene asociado múltiples niveles de voltaje.

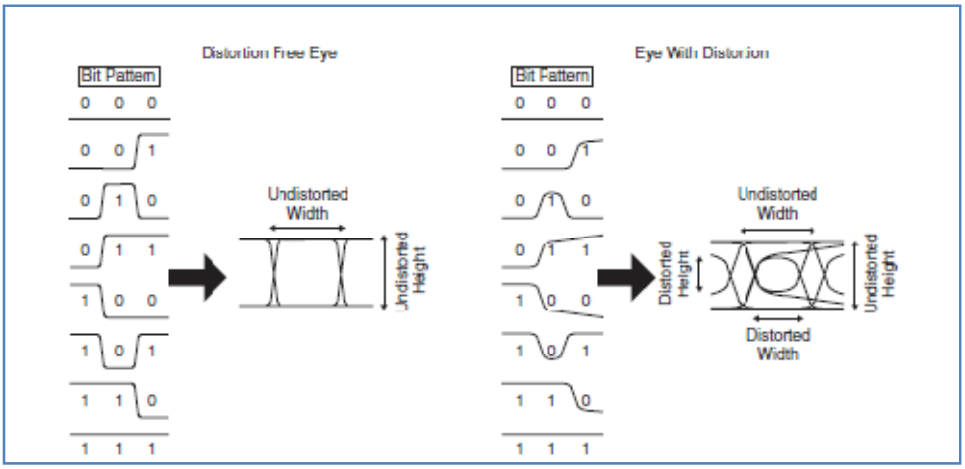


Figura D.12. Ejemplo de la formación del diagrama de ojo

Principalmente existen dos tipos de análisis de los diagramas de ojo. El primero se refiere fundamentalmente al análisis de las distintas características de la forma de onda del pulso como son el Risetime, Falltime, overshoot, undershoot y el jitter, que están referidas a cuatro propiedades fundamentales del Ojo, el nivel cero, nivel uno, cruce de amplitud y cruce en el tiempo. En la figura D.13 se observa los diferentes parámetros que constituyen un pulso en un diagrama de ojo.

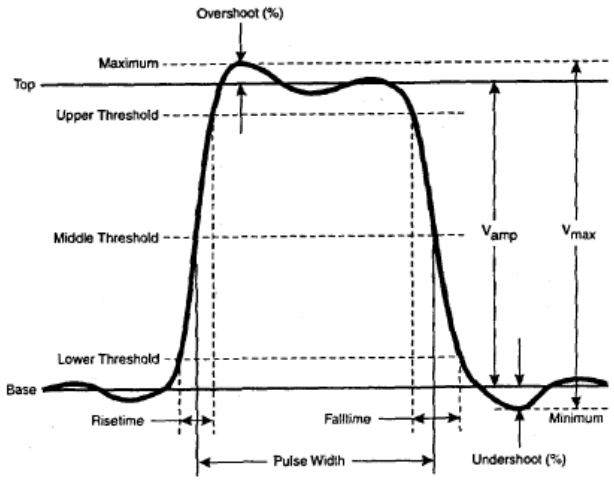


Figura D.13. El segundo método consiste en la comparación de la máscara medida directamente en el diagrama de ojo con una máscara preestablecida

Técnicamente, las máscaras preestablecidas definen regiones específicas en el diagrama de ojo, dentro de las cuales los pulsos u ondas no deben introducirse. Dichas máscaras son muy útiles, ya que se utilizan en el diseño de canales de transmisión, especificando por medio de ellas zonas no permitidas para las señales. Con ello se logra preestablecer un diseño óptimo de enlaces que cumplan ciertas características, ya que si la señal digital que se propaga por el canal se introduce en dichas regiones, se observan claramente problemas y errores en la transmisión. En la figura D.14 se presenta un esquemático de una máscara.

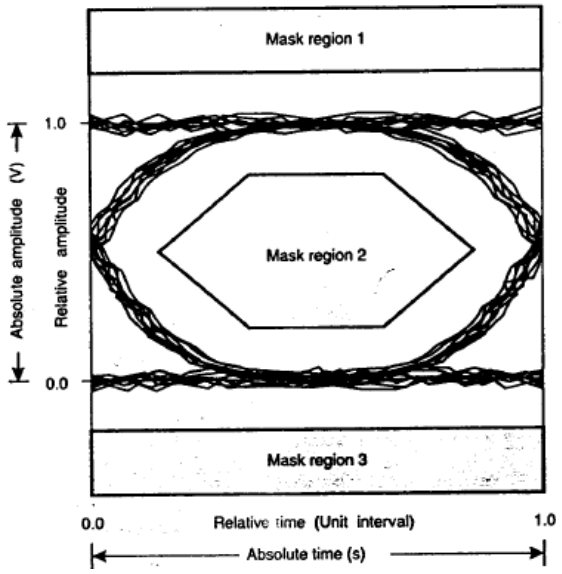


Figura D.14. Ejemplo de Máscara para el diagrama de ojo. Las formas de onda del diseño no deben inmiscuirse en las regiones sombreadas de la máscara