Instituto Nacional de Astrofísica, Óptica y Electrónica.

Diseño de un proceso y su implementación parcial en FPGA para reconocimiento morfológico de leucocitos inmaduros como ayuda en el análisis clínico de enfermedades

Por

Gloria Castro Muñoz

Tesis sometida como requisito parcial para obtener el grado de Maestra en Ciencias, en la especialidad de Electrónica en el Instituto Nacional de Astrofísica, Óptica y Electrónica.

Supervisada por:

Dr. Jorge Francisco Martínez Carballido
Investigador Titular de la Coordinación de Electrónica del INAOE

Tonanzintla, Pue. Octubre, 2007.

RESUMEN

Si bien es cierto que el conteo de leucocitos realizado con ayuda de analizadores automatizados ha disminuido en gran medida el error de análisis de los métodos manuales y ha aumentado la exactitud, reproducibilidad y velocidad de análisis por parte del laboratorio clínico; cabe resaltar que estos logros no se han visto reflejados en el caso de un análisis de células que presentan una morfología inmadura o anormal, por lo que su evaluación en forma manual aun sigue siendo necesaria. Ante esta situación y como un primer intento de facilitar la interpretación morfológica de leucocitos inmaduros, en este trabajo se presenta el diseño e implementación de un sistema que determina la razón núcleo/citoplasma (N/C) de leucocitos en la etapa de mieloblastos.

El diseño del sistema se elaboró por medio de procesamiento de imágenes y se implementó en un FPGA Spartan 3 con herramientas de síntesis de la compañía Xilinx. Las acciones de procesamiento del sistema se ejecutaron y evaluaron sobre imágenes digitalizadas de frotis sanguíneos tomadas a una ampliación de 1250x que contienen uno o más leucocitos. Un método de detección de posición del núcleo se utilizó para separar cada uno de los leucocitos contenidos en la imagen original. Posteriormente se realizó una aproximación de la forma de la célula a través de elipses y finalmente se calculó el área del núcleo y del citoplasma para obtener la razón N/C de cada leucocito. Puntos relevantes relacionados a la segmentación de la región del núcleo y el citoplasma en leucocitos son presentados y discutidos.

AGRADECIMIENTOS

A mis padres Flaviano y Gloria por su inmenso cariño y por infundirme la ética y disciplina que hoy guía mi transitar por la vida.

Al M.c. Miguel Ángel Villegas del departamento de análisis clínicos de la Facultad de Ciencias Químicas de la BUAP y a la Ing. Flor Ramos Beltrán del departamento de análisis clínicos del Hospital Universitario de la BUAP por su colaboración en la obtención de las imágenes digitales de frotis sanguíneos.

Al Consejo Nacional de Ciencia y Tecnología (CONACYT) por el apoyo económico otorgado a través de la beca para estudios de maestría no. 201346.

Tabla de contenido

Lista de Figuras

Lista de Tablas

Capítulo 1 Introducción

1.1 Motivación

El conteo diferencial de glóbulos blancos (GB) es un tipo de examen que cuantifica los porcentajes relativos de los glóbulos blancos en la sangre y al mismo tiempo incluye información acerca de la estructura inmadura y anormal de las células. Este examen tuvo su origen aproximadamente un siglo atrás, en la época de Paúl Ehrlich y otros investigadores, cuando el uso del tinte anilina en la preparación de células de la sangre permitió hacerlas visibles en el microscopio para su identificación [1, 2, 3, 4, 5].

La información derivada de un conteo diferencial de glóbulos blancos ha llegado a ser usada y aceptada ampliamente en diversas clínicas de salud debido a que es tomada en cuenta para generar información valiosa de estados de salud y enfermedad de los pacientes; por ejemplo es usada como: (A) un primer diagnóstico que permite redirigir a pruebas más específicas, (B) en diagnósticos de enfermedades infecciosas, virales o de bacterias, (C) evaluación de condiciones alérgicas, (D) monitoreo del progreso de pacientes y su respuesta a tratamientos; entre otras aplicaciones [1, 2, 6, 7].

Actualmente los laboratorios clínicos utilizan una gran variedad de procedimientos para efectuar el recuento diferencial de GB. Sin embargo en general estos procedimientos son englobados en dos grandes grupos: Aquellos donde se realiza un conteo de GB en forma manual y aquellos donde se realiza un conteo de GB con ayuda de analizadores automatizados de hematología [1, 3, 8].

1.2 Descripción del problema

Si bien es cierto el conteo de GB realizado con ayuda de analizadores automatizados en los últimos años ha disminuido el error intrínseco de los métodos manuales y ha aumentado la exactitud, reproducibilidad y velocidad de análisis por parte del laboratorio clínico; cabe resaltar que estos logros no han sido suficientes para incluir un análisis cuantitativo y cualitativo de células que presentan una morfología inmadura o anormal debido a que la mayor parte de los analizadores automatizados solo incluyen un sistema de alerta conocido como “Flagging” en el caso de análisis de especimenes anormales [9, 10, 11, 12, 13].

De ahí que aunque actualmente existe una gran variedad de analizadores hematológicos automatizados, el conteo diferencial manual de glóbulos blancos aun sigue siendo una herramienta de evaluación necesaria en el caso de células inmaduras o anormales.

Ante esta situación y como un primer intento de facilitar la identificación morfológica de leucocitos inmaduros en forma manual, en este trabajo de investigación se establecen bases de un sistema automatizado de reconocimiento morfológico de leucocitos basado en procesamiento de imágenes, que se espera en un futuro sea una herramienta capaz de realizar un conteo diferencial de glóbulos blancos que incluya un análisis cualitativo y cuantitativo de leucocitos inmaduros y linfocitos no típicos.

1.3 Objetivos

Los objetivos generales de este trabajo de investigación son:

- Definir un proceso para reconocimiento morfológico de leucocitos a partir de una imagen digital

- Diseñar e implementar un sistema que determine la razón núcleo-citoplasma de leucocitos inmaduros en la etapa de mieloblastos

1.4 Solución propuesta

El diagrama a bloques del sistema que se diseña en este trabajo de investigación se muestra en la figura 1-1. El sistema parte de una imagen digital de 640x480 píxeles que contiene uno o más leucocitos, eritrocitos y plaquetas. Esta imagen será previamente obtenida de un frotis sanguíneo por medio de un microscopio a una ampliación de 1250x y almacenada en el disco duro de la computadora. Posteriormente el sistema aplicará diversas técnicas de procesamiento de imagen para aislar cada uno de los leucocitos del resto de la imagen. Una vez separados los leucocitos, el sistema calculará el área del núcleo y el área del citoplasma de cada célula y obtendrá el cociente de estas dos áreas. Finalmente, como salida el sistema mostrará la imagen de cada leucocito y el resultado de su respectivo cociente núcleo/citoplasma.

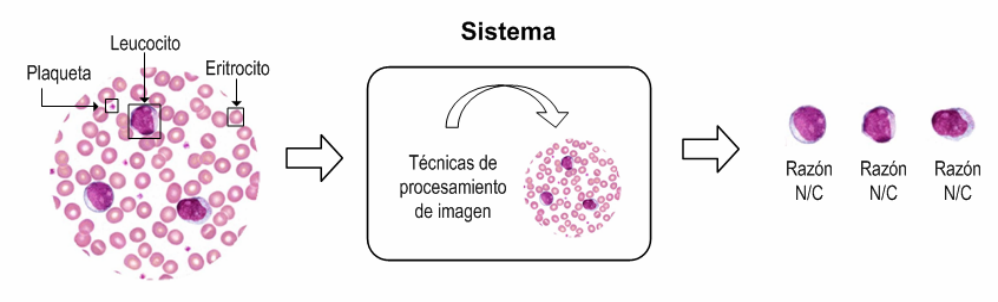


Figura 1-1 Diagrama a bloques del sistema

El sistema se desarrolló en tres etapas (Figura 1-2):

- Etapa de diseño software. En esta etapa se probaron diferentes técnicas de procesamiento de imágenes necesarias para la separación y determinación de la razón núcleo/citoplasma. Posteriormente se eligieron aquellas que mostraron un mejor resultado y se formalizó así el proceso que determinaría la razón núcleo/citoplasma.

- Etapa de diseño hardware. En esta segunda etapa cada una de las acciones de procesamiento que forman parte del proceso que determina la proporción núcleo/citoplasma se trasladó a un lenguaje de descripción de hardware HDL. El diseño total fue estructurado como un diseño “Top Down” secuencial donde cada una de las acciones de procesamiento dio lugar a un módulo funcional y a cada uno de lo módulos funcionales se le asignó un orden de ejecución.

- Etapa de implementación. En esta etapa final, el diseño HDL de la etapa anterior fue sintetizado e implementado en una tarjeta de desarrollo que contiene un FPGA Spartan 3 haciendo uso de las herramientas y protocolos proporcionados por la compañía Xilinx. La imagen resultante después de haber sido procesada fue mostrada en

un monitor VGA conectado a la tarjeta de desarrollo y la razón núcleo/citoplasma fue mostrada en los "displays" de 7 segmentos localizados en la misma tarjeta.

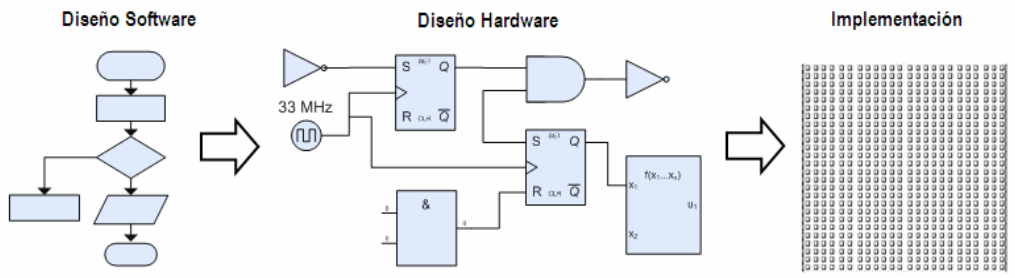


Figura 1-2 Metodología de desarrollo del sistema

1.5 Justificación

El desarrollo del sistema planteado en el presente trabajo de investigación contribuirá a atender la necesidad social de brindar un servicio de salud eficaz debido a que facilitará la identificación de células inmaduras, permitirá ahorrar tiempo y abre la posibilidad de disminuir imprecisiones en cuentas manuales causadas en mayor parte por errores humanos.

Por otro lado en el campo de los servicios de salud clínico-hematológicos se espera que este instrumento sea una fuente de ayuda colateral en el entrenamiento de nuevos elementos; debido a que este sistema les permitirá poner en práctica la “localización y reconocimiento” de células anormales de la sangre.

Por último, en el campo de la educación, este instrumento permitirá a los profesores “en cualquier momento y en cualquier lugar” mostrar de forma fácil a todos sus alumnos la misma imagen y al mismo tiempo resaltar características morfológicas claves en el reconocimiento de leucocitos inmaduros.

Capítulo 2 Revisión de la literatura

2.1 Introducción

El material que se presenta en este capítulo es principalmente información base que será de gran ayuda en el entendimiento de los capítulos subsecuentes. La sección 2.2 proporciona un breve resumen de los trabajos reportados hasta antes del inicio de la presente investigación y que están vinculados a ella. Una idea general de las características, clasificación y conteo de leucocitos desde el punto de vista morfológico es introducida en la sección 2.3. Finalmente, en la sección 2.4 se exponen conceptos relacionados al área de procesamiento digital de imágenes que serán utilizados en los capítulos 3 y 4 como base para el desarrollo del diseño software - hardware del sistema reportado en este trabajo.

2.2 Antecedentes

Los criterios de identificación morfológicos evaluados en la realización del conteo diferencial de leucocitos son encontrados en varios artículos y libros donde se destaca la importancia de la forma, tamaño, color y patrón de cromatina del núcleo y citoplasma [2, 3, 6, 14, 15]. Bacus [16] realizó una clasificación de leucocitos en ocho categorías a través de procesamiento digital de imágenes. En este trabajo se realizó una aproximación del citoplasma mediante los siguientes pasos: se aisló el núcleo con un operador de umbral, se determinó su área y centro de masa, se centró un rectángulo del tamaño y área del núcleo en el centro de masa, se expandió 2 unidades

en cada dirección y finalmente se eliminó el área del núcleo. Sin embargo en su investigación Bacus no incluye las formas más inmaduras de los leucocitos. Su trabajo está restringido a imágenes que contienen una sola célula y la umbralización del núcleo se realiza a través de hardware, truncando la señal de video de televisión después de realizar un filtrado de color amarillo de imagen. Foran [17, 18] diseñó un sistema de soporte de decisión para recuperación de imágenes de leucocitos usando forma, tamaño y textura del núcleo. Esta aproximación consideró 261 imágenes, incluyendo tres clases de desórdenes linfoproliferativos y una clase de leucocitos saludables, Aus [18, 19] calculó varias características de la célula, pero el procesamiento de segmentación de la imagen y análisis omite detalles esenciales para su reproducibilidad. Young [18, 20] desarrolló medidas para describir la cromatina del núcleo, sin embargo consideró imágenes en escala de grises y un reducido conjunto de características. Mayumi [18] diferenció entre los cinco tipos de leucocitos normales y leucemia linfocítica crónica utilizando reconocimiento de patrones e información de textura. En esta investigación; para la segmentación del citoplasma se consideró descripción de textura junto con aspectos de distribución espacial de todos los valores de gris y se propusieron parámetros para sintonización de la función de relación angular entre las células de resolución vecina así como también la distancia entre ellas. Sin embargo la desventaja principal es que sus algoritmos no fueron pensados para su implementación en hardware, por lo que no es una tarea fácil obtener la versión hardware del algoritmo propuesto en este trabajo.

Después de hacer una revisión de los trabajos mencionados anteriormente, se encontró que ninguno de estos trabajos ha incluido el análisis de las formas más inmaduras de los leucocitos ni tampoco ha sido pensado para su implementación en hardware.

2.3 Conceptos de Hematología

La evaluación cuidadosa de los elementos de la sangre es a menudo el primer paso en el análisis de la función y diagnostico hematológico. Esto se debe principalmente a que muchos de los desórdenes hematológicos son definidos por los resultados específicos de análisis de sangre. El examen cuidadoso de la morfología celular, en conjunto con la cuantificación de los elementos de la sangre y de la evaluación de una variedad de parámetros referentes al tamaño y forma celular, son requeridos en este tipo de análisis.

Las células que circulan en el torrente sanguíneo se dividen generalmente en tres tipos [2, 3] (Figura 2-1a) [4]:

- Leucocitos. También conocidos como glóbulos blancos, están relacionados con la función inmune e incluyen una variedad de tipos que tienen funciones específicas y aspectos morfológicos característicos.

- Eritrocitos. También conocidos como glóbulos rojos son las células más numerosas en la sangre y son necesarias para la respiración del tejido fino. A diferencia de los glóbulos blancos carecen de núcleo y contienen hemoglobina, una proteína que contiene hierro y que actúa en el transporte de oxígeno y de bióxido de carbono.

- Plaquetas. También conocidas como trombocitos, son fragmentos citoplásmicos derivados de los megacariocitos en la médula que trabajan en la coagulación y la hemostasis.

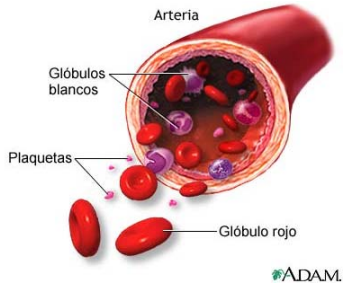


Figura 2-1 Elementos de la sangre

2.3.1 Leucocitos

2.3.1.1 Definición y características

Los leucocitos son un conjunto heterogéneo de células sanguíneas que son los efectores celulares de la respuesta inmune, así intervienen en la defensa del organismo contra sustancias extrañas o agentes infecciosos (antígenos). Se originan en la médula ósea y en el tejido linfático [2, 3, 21].

Los leucocitos son células móviles que se encuentran en la sangre transitoriamente, así, forman una fracción celular de los elementos figurados de la sangre. A diferencia de los eritrocitos (glóbulos rojos), no contienen pigmentos, por lo que se les califica de glóbulos blancos. Son células con núcleo, mitocondrias y otros orgánulos celulares. Su tamaño oscila entre los 8 y 20 µm [2, 3, 21].

2.3.1.2 Clasificación de las formas maduras

La observación a través del microscopio ha permitido clasificar a los leucocitos según sus características de coloración en:

- Granulositos: Este tipo de leucocitos presentan gránulos en su citoplasma, con núcleo redondeado y lobulado, formados en las células madres de la médula ósea: eosinófilos, basófilos y neutrófilos (Figura 2-2) [21].

- Agranulocitos: No presentan gránulos en su citoplasma: linfocitos y monocitos (Figura 2-2) [21].

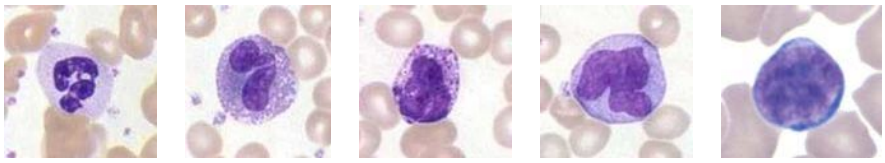


Figura 2-2 Formas maduras de los leucocitos

2.3.1.3 Formas inmaduras (precursores)

Dentro del mismo linaje de los leucocitos pero ubicados en sus diferentes estados de maduración encontramos a 6 estados morfológicos identificables (Figura 2-3) [14]; desde la célula más inmadura identificable, el mieloblasto, a el neutrófilo segmentado maduro [14].

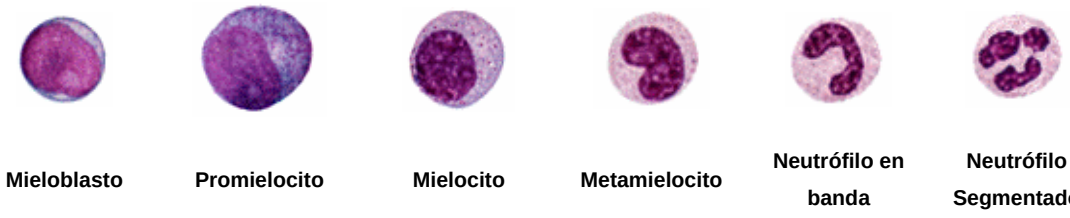


Figura 2-3 Formas inmaduras de los leucocitos

Mieloblasto

Los mieloblastos son los precursores de los granulositos más jóvenes identificables. No es común la existencia de mieloblastos en sangre circulante, excepto en hematología maligna. El mieloblasto es una célula de tamaño medio con una forma más o menos redonda u ovalada, se caracteriza por tener una mayor cantidad de núcleo en comparación con la cantidad de citoplasma. La forma del núcleo es usualmente redonda u ovalada y puede estar ligeramente mellada, presenta un color de morado rojizo a morado verdadero y regularmente contiene uno o más nucléolos bien delineados sobre una cromatina fina. El citoplasma es típicamente de color pálido homogéneo a azul medio y tradicionalmente no presenta gránulos [14].

Promielocito

Los promielocitos rara vez circulan en condiciones no malignas. Cuando los promielocitos son observados en neutrofilia, los precursores más maduros (mielocitos, y metamielocitos) están presentes [14].

El promielocito es de mayor tamaño en comparación con el mieloblasto; generalmente es la célula más grande en la serie de los neutrófilos, es de forma redonda u ovalada con una cantidad de núcleo más o menos igual a la cantidad de citoplasma debido al incremento de éste. Su núcleo es de color morado rojizo medio a morado verdadero con una forma redonda u ovalada y con frecuencia presenta una posición excéntrica; regularmente la cromatina es más condensada que la de los mieloblastos pero aún se observan uno o más nucleolos dentro de ella. El Citoplasma es de color azul medio a azul intenso con una ligera área tintada (debido al complejo Golgi) cerca del núcleo, presenta gránulos prominentes de color

morado-rojo o magenta. Estos pueden ser pocos o muchos en número y algunas veces cubren el núcleo [14].

Mielocito

Los mielocitos son a menudo observados sobre frotis sanguíneos en neutrofilia. Cuando ellos están presentes los metamielocitos están presentes. El mielocito es una célula redonda u ovalada de tamaño medio; más pequeña que el promielocito pero generalmente más grande que el neutrófilo maduro con una cantidad de núcleo más o menos igual a la cantidad de citoplasma. El núcleo es de color morado rojizo a morado verdadero, tiene una forma ovalada, muy ligeramente mellada o aplanada sobre un lado; puede estar localizado en una posición excéntrica. La cromatina es más condensada que la del promielocito y no presenta nucleolos. El citoplasma es de color variable llegando a ser más rosa conforme la célula madura y presenta gránulos neutrófilos específicos o secundarios (pequeños, uniformes, canela-lila-rosa) mezclados con gránulos azurofílicos [14].

Metamielocito

Los metamielocitos circulantes son comunes en reacciones neutrófilas y son usualmente acompañados por un incremento en el número de formas de bandas. El metamielocito es una célula de tamaño medio con coloración morada rojiza a morada verdadera, posee una forma redonda u ovalada y una cantidad de núcleo más o menos igual a la cantidad de citoplasma. Su núcleo generalmente tiene una forma de riñón mellada o aplanada con una cromatina más gruesa y agrupada que la del mielocito. El citoplasma es de color ligeramente rosa y presenta una gran cantidad de gránulos específicos que dan a éste una granulación rosa beige aparente [14].

Banda

Las formas en banda comprenden un pequeño porcentaje de neutrófilos sobre frotis sanguíneos. Un incremento en el número absoluto de bandas es un indicativo de una infección temprana o proceso de inflamación. Por esta razón, algunos laboratorios tratan las bandas separadamente [2, 3, 14].

La banda es una célula de tamaño medio; similar en tamaño a los neutrófilos segmentados, tiene una forma redonda u ovalada y una cantidad de núcleo pequeña en comparación con la cantidad de citoplasma. Su núcleo tiene una forma alongada; curva o en forma de salchicha, con una apreciable longitud sin filamentos, torceduras o dobleces, se encuentra profundamente mellado (dentado) (más de la mitad de la distancia a el margen nuclear opuesto) y a menudo en la forma de una letra C, J o S. La cromatina es gruesa y agrupada bastante distintiva con cortinas de color morado rojo a morado profundo. El citoplasma es de color rosa ligero con gránulos específicos finos de color rosa beige. Unos pocos gránulos azurofílicos están presentes pero a menudo difíciles de distinguir [14].

Neutrófilo segmentado

Sobre un frotis de sangre normal, la mayoría de los neutrófilos se encuentran segmentados. El neutrófilo segmentado es de tamaño medio; usualmente redondo u ovalado en su forma, con una cantidad de núcleo pequeña en comparación con la cantidad de citoplasma. Su núcleo tiene una forma lobulada, con 2 a 5 segmentos irregulares conectados por filamentos o sobrepuesto (torcido o doblado) con cortinas de color morado rojo a morado profundo, con cromatina gruesa y agrupada; algunas veces descrita como un patrón de cromatina “piel de leopardo“. El citoplasma es de color rosa ligero

con gránulos específicos finos que dan al citoplasma una apariencia beige-rosa. Unos pocos gránulos azurofílicos pueden estar presentes [14].

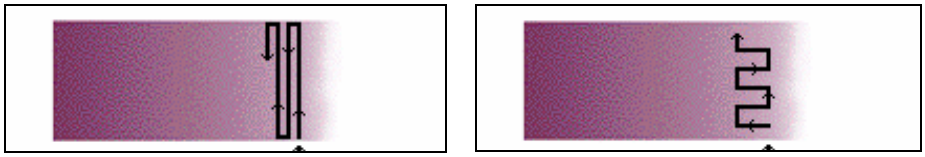
2.3.2 Conteo diferenciado de leucocitos

2.3.2.1 Definición y características

El conteo diferenciado de leucocitos se define como la enumeración de diferentes tipos de glóbulos blancos circulando en la sangre (white blood cell count WBC). Éste a diferencia de otras pruebas realizadas en laboratorio, suministra un reporte patológico de la apariencia morfológica de las células y la frecuencia con la cual ellas aparecen, no en una forma descriptiva pero si en una forma cuantitativa [1, 2, 3, 6].

El conteo diferenciado extendido de leucocitos enumera aparte de las 5 formas maduras presentes normalmente en la sangre (neutrófilos, eosinófilos, basófilos, linfocitos y monocitos), formas inmaduras y anormales de estas células. El número total de células contabilizadas es de 100 y los números relativos de cada tipo de célula observada son reportados en números relativos (porcentaje del total) y en números absolutos por litro (obtenido de multiplicar cada porcentaje -como un decimal- por el total de la cuenta WBC) [1, 14].

La rutina de diferenciado consiste en la identificación de leucocitos localizados consecutivamente mientras se sigue un patrón de trayectoria específico sobre el frotis sanguíneo (figura 2-4) [14]. Esta identificación se lleva acabo de una forma sistemática aplicando criterios morfológicos de identificación bien definidos [14].



Patrón lado a lado Patrón Contrario

Figura 2-4 Patrones de trayectoria más comunes sobre frotis sanguíneos

2.3.2.2 Criterios morfológicos de identificación

Con el objetivo de identificar células de la sangre de forma exacta, éstas son examinadas de una forma sistemática aplicando criterios morfológicos de clasificación relacionados a su estado de madurez y normalidad [14].

La identificación de cada célula se realiza en base a 3 componentes que contribuyen a su apariencia:

- Perfil. Características incluidas en el perfil de la célula son el tamaño, la forma y las cantidades relativas de núcleo y citoplasma (razón N/C [14]).

- Núcleo. En el examen del núcleo se considera la forma, el color, el patrón de cromatina y la presencia de nucleolos.

- Citoplasma. En cuanto al citoplasma se observa su color, la presencia y tipo de gránulos (o ausencia) y si otras inclusiones están presentes.

En este trabajo de investigación solo nos enfocamos a realizar un análisis morfológico del perfil de la célula relacionado al criterio de identificación razón núcleo/citoplasma de leucocitos que presentan una forma

redonda u ovalada (recuadros en rojo figura 2-5), por esta razón el único criterio que se definirá en la siguiente sección es el de “razón núcleo/citoplasma”.

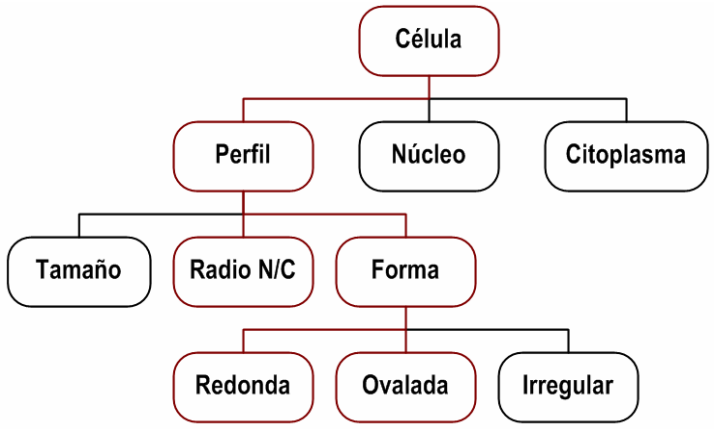


Figura 2-5 Alcance del trabajo de investigación

Razón Núcleo /citoplasma

La cantidad relativa de núcleo en comparación con la cantidad de citoplasma en la célula, es conocida como la razón N/C. Sin embargo no es necesario asignar valores numéricos a la razón N/C, ésta solamente es descrita como alta, moderada o baja. Si el núcleo ocupa una gran porción de la célula con solamente una pequeña cantidad de citoplasma la célula tiene una razón N/C alta. Si la célula tiene aproximadamente igual cantidad de núcleo y citoplasma la razón N/C está en el rango medio a moderado y si la cantidad de núcleo es relativamente pequeña y la del citoplasma es grande la razón es baja [14].

2.4 Procesamiento de imágenes

En términos generales, el procesamiento de imagen se refiere a la manipulación y análisis de información visual. En este caso, información visual significa una imagen en dos dimensiones. Alguna operación que actúa para mejorar, corregir, analizar o en algún sentido cambiar una imagen es llamada procesamiento de imagen [22].

2.4.1 Imagen Digital

Una imagen para su procesamiento computacional es representada de forma digital como un arreglo bidimensional de tamaño $N \times M$ (Figura 2-6). Cada elemento del arreglo es referido como un elemento imagen o píxel $f(x, y)$, donde f es la intensidad del píxel y (x, y) define la posición del píxel a lo largo del arreglo. La intensidad f de la imagen es cuantizada en L+1 niveles de gris [0, L], donde 0 es considerado negro, L es considerado blanco y todos los valores intermedios corresponden a diferentes tonalidades de gris [23]. Así el número, b de bits requeridos para almacenar una imagen digitalizada está dado por

$$b = N \times M \times \log_2(L+1)$$

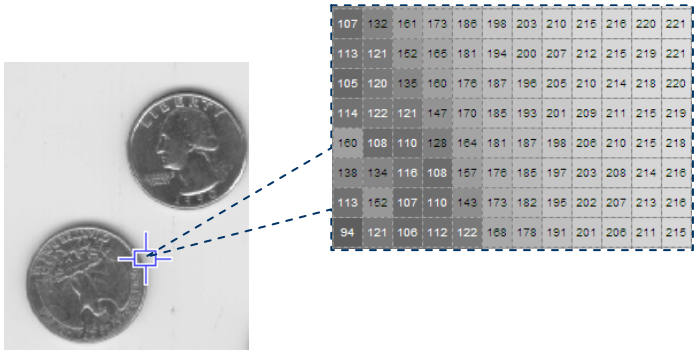


Figura 2-6 Representación digital de una imagen

2.4.2 Multiplicación de dos imágenes

La multiplicación es una operación aritmética que toma dos imágenes y produce una imagen de salida en la cual los valores de los píxeles son justamente aquellos píxeles de la primera imagen multiplicados por los valores de los correspondientes píxeles en la segunda imagen, que en el caso específico de valores binarios corresponde a un AND lógico [24].

$$Q(i, j) = P_1(i, j) \times P_2(i, j)$$

2.4.3 Histograma de Intensidad

El histograma es una representación gráfica de la distribución de frecuencias de las tonalidades de gris en una imagen (Figura 2-7). La abcisa, o eje-x, se refiere al valor de gris cuantizado y la ordenada, o eje-y, se refiere al número de píxeles teniendo tal nivel de gris [25, 26]. Para una imagen en escala de grises de 8 bits hay 256 diferentes intensidades, tal que el histograma desplegará 256 números mostrando la distribución de los píxeles entre tales valores de gris. Para una imagen a color el histograma es calculado de forma individual para cada canal (rojo, verde, azul) [24].

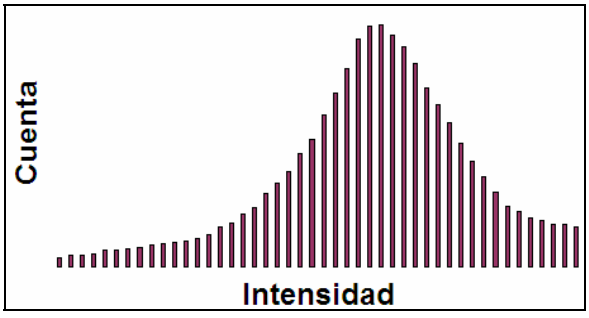


Figura 2-7 Representación de un histograma de intensidad

2.4.4 Dilatación Binaria

La dilatación es uno de los dos operadores básicos en el área de morfología matemática. El efecto básico de este operador sobre una imagen binaria es agrandar gradualmente los límites de regiones de primer plano (es decir píxeles blancos, típicamente) [26]. Así las áreas de los píxeles del primer plano crecen de tamaño mientras las secciones de color negro dentro de estas regiones llegan a ser más pequeños (Figura 2-8) [24]. El operador de dilatación toma dos piezas de datos como entradas. El primero es la imagen que debe ser dilatada, el segundo es un conjunto de puntos conocido como elemento estructural. La definición matemática de la dilatación para imágenes binarias es como sigue:

$$X \oplus K = \left\{ x / Kx \cap X \neq \varnothing \right\}$$

Donde X es el sistema de coordenadas euclidianas que corresponden a la imagen binaria de entrada, K es el sistema de coordenadas para la matriz del elemento estructural y Kx denota la traslación de K tal que su origen está en x. Entonces la dilatación de X por K es simplemente el sistema de todos los puntos x tales que la intersección de Kx con X es no vacía [24].

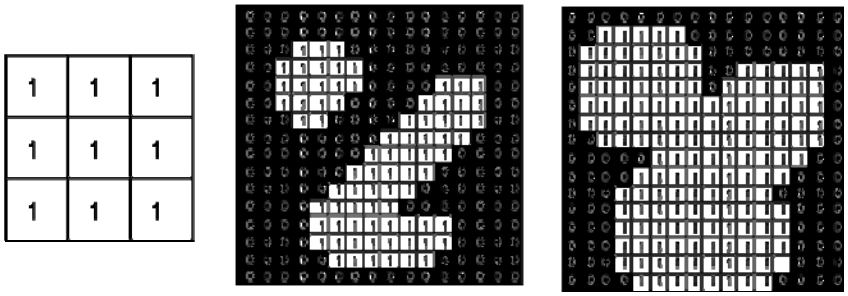


Figura 2-8 Ejemplo de dilatación binaria

2.4.5 Erosión Binaria

La erosión es el otro de los dos operadores básicos en el área de la morfología matemática. El efecto básico de este operador en una imagen binaria es reducir los límites de las regiones correspondientes a los píxeles de primer plano. Así las áreas de los píxeles de primer plano se contraen en tamaño, y los agujeros dentro de esas áreas llegan a ser más grandes (Figura 2-9) [24].

El operador erosión toma dos bloques de datos como entradas. El primero es la imagen que debe ser erosionada y el segundo es el conjunto de puntos coordenados del elemento estructural. La definición matemática de la erosión para las imágenes binarias es como sigue:

$$X \Theta K = \{ x / Kx \subset X \}$$

Donde X es el sistema de coordenadas euclidianas que corresponden a la imagen binaria de entrada, K es el sistema de coordenadas para el elemento estructural y Kx denota la traslación de K tal que su origen esté en x. Entonces la erosión de X por K es simplemente el sistema de todos los puntos x tales que Kx es un subconjunto de X. Esto es si para cada píxel en el elemento estructural, el píxel correspondiente en la imagen es de primer plano, entonces el píxel de entrada se deja como está. Si alguno de los píxeles correspondientes en la imagen es un píxel fondo, entonces el píxel de entrada es puesto al color del fondo [24].

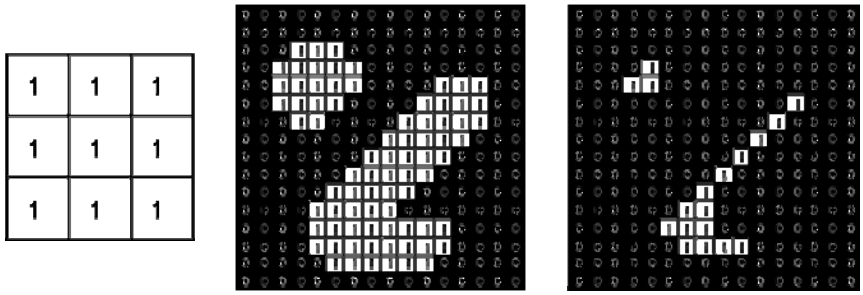


Figura 2-9 Ejemplo de erosión binaria

2.4.6 Cerradura (Llenado)

Cerradura es un operador importante del campo de la morfología matemática. La cerradura es similar en cierto modo a la dilatación en que tiende a agrandar los límites de las regiones del primer plano (brillantes) en una imagen (y contraer los agujeros del color del fondo en tales regiones), pero es menos destructivo sobre la forma original [24]. Como con otros operadores morfológicos, la operación exacta es determinada por un elemento estructural. El efecto del operador es preservar las regiones del fondo que tienen una forma similar a este elemento estructural, o que pueden contener totalmente el elemento estructural, mientras elimina el resto de las regiones de los píxeles del fondo. Este operador es definido simplemente como una operación de dilatación seguida por una erosión usando el mismo elemento estructural para ambas operaciones (Figura 2-10) [24].

$$X^K : (X \oplus K) \Theta K$$

Figura 2-10 Ejemplo de cerradura binaria

2.4.7 Segmentación

La segmentación es el estado inicial de un proceso de reconocimiento, con lo cual la imagen adquirida es separada en regiones significativas o segmentos. Existen dos métodos principales de segmentación: Uno basado en técnicas locales (basado en píxel) y el otro basado en técnicas globales (basado en regiones). El primero de ellos busca detectar y realzar bordes o elementos del borde dentro de una imagen y después unirlos para crear una frontera la cual encierra una región uniforme. El segundo método busca crear regiones agrupando directamente píxeles con características comunes en áreas o regiones uniformes [25, 27].

En este trabajo de investigación se utilizaron ambos métodos de segmentación. El método basado en técnicas locales para separación de las células (sección 3.4.3) y el método basado en técnicas globales para la detección de la región de los núcleos (sección 3.4.2) y refinamiento de la aproximación por rectángulo (sección 3.4.4).

2.4.8 Umbralización

La umbralización es un método para producir regiones de uniformidad dentro de una imagen con base en algún criterio de umbral, T [25].

Una imagen umbralizada g(x, y) es definida como

$$g(x,y) = \begin{cases} 1 \quad si \quad f(x,y) \geq T \\ 0 \quad si \quad f(x,y) < T \end{cases}$$

2.5 Resumen

El propósito principal de este capítulo ha sido brindar un fundamento teórico de las dos áreas incluidas en el presente trabajo de investigación: Hematología y procesamiento digital de imágenes. El conjunto de características claves de leucocitos en sus formas madura e inmadura, proporcionadas en la sección 2.3 son consideradas material base en la sección 3.4.3 para la separación y almacenamiento de leucocitos y en la sección 3.4.5 para el calculo de la proporción núcleo/citoplasma. Las técnicas y operaciones descritas en la sección 2.4 son usadas en las secciones 3.4.2 y 3.4.4 para el desarrollo del diseño software del sistema que determina la proporción N/C.

Capítulo 3 Diseño software del sistema

3.1 Introducción

Este capítulo está enfocado al desarrollo del diseño software del sistema propuesto en este trabajo de investigación. La sección 3.2 da un breve resumen de los pasos seguidos a lo largo del desarrollo del diseño software. La sección 3.3 describe las funciones de mayor relevancia utilizadas en el diseño y finalmente, en la sección 3.4 se incluye una descripción a fondo de cada una de las etapas del diseño.

3.2 Metodología del diseño software del sistema

En esta sección se describe la metodología seguida en el proceso del diseño software del sistema. El primer paso fue adquirir un compendio de imágenes digitales de frotis sanguíneos para después hacer una selección de estas imágenes y formar con ellas un subconjunto que serviría de base para el diseño y prueba del sistema. Posteriormente se aplicaron y evaluaron diferentes acciones de procesamiento de imágenes mediante las herramientas de matlab y halcón y finalmente se determinó y formalizó el diseño software del sistema.

3.2.1 Adquisición de las imágenes digitales

Las imágenes digitales de los frotis de sangre periférica que sirvieron de base en el diseño y prueba del sistema fueron de una resolución de 640x480 píxeles y se obtuvieron en los laboratorios de análisis clínicos del

Hospital Universitario (H.U) y la Facultad de Ciencias Químicas de la BUAP, así como del cd “Hematography Plus” de la Universidad de Minnesota.

El examen microscópico de los frotis sanguíneos realizado fue del tipo examen por inmersión en aceite y se realizó por medio de un microscopio Olympus en el caso del H.U y un microscopio Carl Zeiss en el caso de la Facultad de Ciencias Químicas. En ambos casos para la digitalización de los frotis sanguíneos se utilizó una cámara digital coolpix 990 marca Nikon acoplada manualmente al ocular del microcopio. El proceso de digitalización se realizó de la siguiente forma:

- Primero el frotis de sangre es colocado sobre el portaobjetos del microscopio
- Después se elije un área para examinar y se enfoca usando un objetivo de 10x y oculares estándar de 10x, logrando una ampliación total de 100x
- Posteriormente el objetivo es rotado al espacio entre los objetivos de 10x y 100x y una pequeña gota de aceite de inmersión es colocada directamente sobre el frotis
- Después de esto el objetivo es rotado al objetivo de 100x para lograr una ampliación total de 1000x y el área a examinar enfocada usando un ajuste fino
- Finalmente una vez que el área de examen se encuentra enfocada se procede a su digitalización con la cámara

Todas las imágenes se obtuvieron mediante este proceso de digitalización y se almacenaron en el disco duro de la computadora. Una vez que se obtuvo el compendio de imágenes digitales se eligieron aquellas imágenes que mostraron mayor detalle y calidad de las células y se formó

con éstas el subconjunto que sirvió de base para el diseño y prueba del sistema.

3.2.2 Procesamiento de la imagen

En esta etapa se aplicaron y evaluaron diferentes acciones de procesamiento sobre las imágenes de los leucocitos para obtener la razón núcleo / citoplasma de cada una de las células de interés. El primer paso que se realizó en esta etapa fue tratar de separar cada una de las células de interés del resto de la imagen usando segmentación por umbralización del histograma. Para esto se obtuvieron los histogramas de los tres planos de color (rojo, verde y azul) de la imagen del frotis y después de realizar varias pruebas con diferentes imágenes se determinó que no era posible separar los leucocitos del resto de la imagen debido a que la tonalidad del citoplasma de los leucocitos era muy parecida a la de los eritrocitos (glóbulos rojos) en los tres planos de color. Sin embargo se encontró que usando una segmentación por umbralización sobre el histograma del plano de color verde era posible determinar la región de los núcleos de los leucocitos. Con esto se pensó qué la región de los núcleos de los leucocitos podía ayudar a separar las células de interés si se localizaba la posición de cada núcleo y se determinaba un rectángulo que contuviera al leucocito [18]. Después de realizar esta primera aproximación del leucocito por medio de un rectángulo se observó que aún se encontraban restos de glóbulos rojos, por lo tanto se realizó una refinación de la aproximación por rectángulo usando segmentación por color. Finalmente con el objetivo de lograr un mejor resultado y partiendo del hecho que los mieloblastos poseen una forma redonda u ovalada se realizó una aproximación de la célula a través de una elipse y se calculó la razón núcleo/citoplasma.

3.2.3 Determinación y formalización del diseño software del sistema

El diagrama de flujo del diseño software final del sistema se muestra en la figura 3-1 y consiste de las etapas: Almacenamiento de la imagen RGB del frotis sanguíneo, detección de la región de los núcleos de los leucocitos, separación y almacenamiento de cada leucocito en una nueva imagen, segmentación por cambio de color y aproximación a través de elipses y finalmente la etapa que calcula la razón núcleo-citoplasma.

En la etapa “Almacenamiento de la imagen RGB” el objetivo principal es almacenar por planos de color la imagen digital del frotis sanguíneo, de manera que utilizando el plano de color verde en la etapa “detección de la región de los núcleos de los leucocitos” se puedan separar los píxeles que pertenecen a los núcleos de los leucocitos, de aquellos píxeles que no pertenecen. En la etapa “separación y almacenamiento de cada leucocito en una nueva imagen” se realiza una primera aproximación de la forma de cada leucocito por medio de un rectángulo, con la finalidad de que en la etapa “segmentación por cambio de color y aproximación del leucocito a través de elipses” se haga un refinamiento de la aproximación por rectángulo y se realice una mejor aproximación utilizando elipses. Finalmente en la etapa “cálculo de la razón N/C del leucocito” se calcula el área del núcleo y del citoplasma de cada leucocito y se obtiene el cociente núcleo/citoplasma.

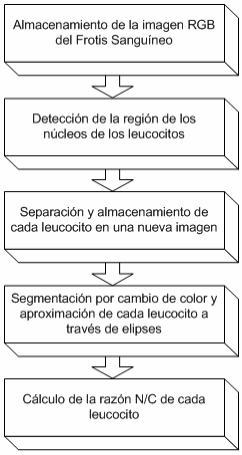


Figura 3-1 Diagrama de flujo del proceso computacional que obtiene la razón núcleo/citoplasma

3.3 Funciones de mayor relevancia implementadas en Matlab

Con el objetivo de facilitar la descripción de las etapas de diseño software, las funciones y operadores utilizados son definidos en esta sección.

La imagen RGB del frotis sanguíneo a lo largo de este capítulo será representada por la matriz tridimensional `I (i,j,:)` con `i=1,...480 j=1,...640 :=R,G,B` en donde el valor de `I(x,y,:)` toma valores enteros positivos en el rango [0,255]. La imagen RGB puede ser transformada en otra imagen vía las siguientes funciones.

3.3.1 Función filtro_histograma

El objetivo de la función `[A] = filtro_histograma(I,umbral)` es obtener la imagen binaria de la región de los núcleos haciendo uso de un umbral (sección 2.4.8). Esta función recibe como entradas el plano de color verde de la imagen RGB `I(i,j,G)`, y un umbral de binarización `umbral` y

regresa como salida la imagen de la región de los núcleos `A(i,j)` en formato binario. La umbralización de la imagen `I` se realiza de tal forma que los píxeles con valor mayor al umbral de decisión (`I(i,j,G)>umbral`) son cambiados a color negro (`A(i,j)=0`) y los píxeles con valor menor o igual al umbral de decisión (`I(i,j,G)≤umbral`) son modificados a color blanco (`A(i,j)=1`). El diagrama de flujo de esta función se muestra en la figura 3-2.

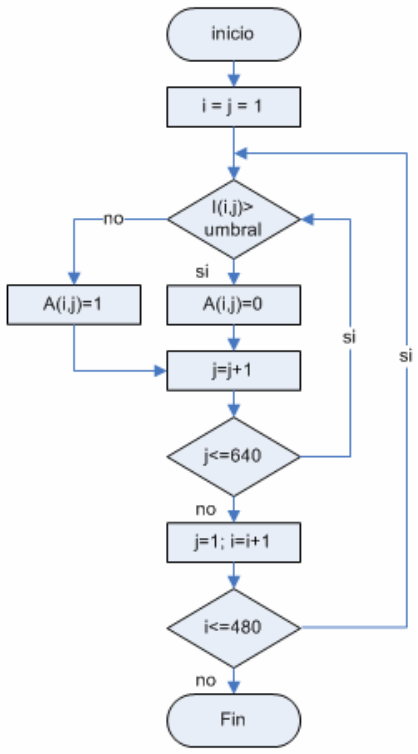


Figura 3-2 Diagrama de flujo de la función filtro_histograma

3.3.2 Función llena_huecos

El objetivo de la función `[H]= llena_huecos(A,se)` es eliminar los huecos de los objetos presentes en la imagen de la región de los núcleos. Esta función recibe como entrada la imagen de la región de los núcleos (`A`) y

regresa como salida la misma imagen con los huecos llenados (`H`). El llenado de los huecos se realiza mediante una operación de cerradura sobre la imagen `A`, que como se describió anteriormente en la sección 2.4.6, una operación de cerradura se define simplemente como una operación de dilatación seguida por una erosión usando el mismo elemento estructura (se). En este caso `se` fue definido como un disco de radio 4. La función `llena_huecos` se definió de la siguiente forma: `H(i,j)= llena_huecos[A(i,j),se]`

```
D(i,j)= dilata [A(i,j),se] con i=1,...480 j=1,...640
H(i,j)= erosiona [D(i,j),se]
```

3.3.3 Función rectangulo

El objetivo de la función `[xmix,xmax,ymin,ymax]= rectangulo(xo,yo,H)` es determinar los límites del rectángulo de menor tamaño que encierra al objeto cuyo punto m forma parte de su borde (Figura 3-3a). Esta función recibe como entradas las coordenadas (`xo,yo`) del punto `m` y la imagen binaria `H(i,j)` donde se encuentra el objeto que se quiere encerrar y regresa como salida los límites del rectángulo `xmix,xmax,ymin,ymax`. Para determinar estos límites, la función realiza los siguientes pasos (figura 3-3b).

- Hace un recorrido del borde del objeto
- Guarda en un arreglo las coordenadas de cada uno de los puntos del borde
- Determina los valores mínimos y máximos de las coordenadas almacenadas

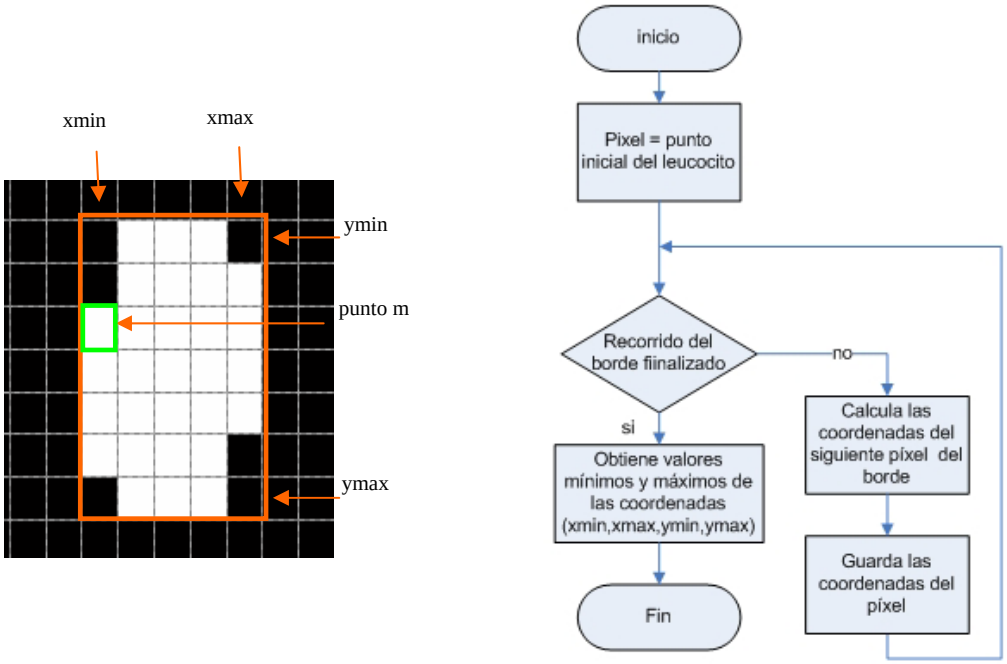


Figura 3-3 a) Ejemplificación de la aproximación por rectángulo, b) Diagrama de flujo de la función `rectangulo`

3.3.4 Función recorta

La tarea de la función `[s_ima] = recorta(H,xmin,xmax,ymin,ymax)` es extraer de la imagen de la región de los núcleos llenada `H`, el objeto encerrado por el rectángulo cuyos límites son `xmin,xmax,ymin,ymax`. Esta función recibe como entradas los límites máximos y mínimos del rectángulo a extraer y la imagen de donde se cortará dicho rectángulo y regresa como salida la imagen del objeto encerrado por el rectángulo mencionado.

La función inicialmente modifica los valores originales de los límites del rectángulo con el objetivo de incluir el citoplasma de la célula en éste. Para el caso particular de las imágenes utilizadas en este trabajo, una expansión de los límites del rectángulo en 14 píxeles fue suficiente. Una vez que se han

definido los nuevos límites, se copian de la imagen de entrada `H` a la imagen de salida `s_ima` los valores de los píxeles con coordenadas (i, j) donde $y_{\min} - 14 \le i \le y_{\max} + 14$ y $x_{\min} - 14 \le j \le x_{\max} + 14$. El diagrama de flujo de esta función se muestra en la figura 3-4a.

3.3.5 Función recorta RGB

La función `[RGB] = recortaRGB(I,xmin,xmax,ymin,ymax)` es muy parecida a la función `recorta`, la única diferencia es que `recortaRGB` recibe como entrada la imagen RGB del frotis sanguíneo `I(i,j,:)` y regresa como salida, la imagen a color del objeto encerrado por los límites `xmin,xmax,ymin,ymax`. Por lo tanto la función `recortaRGB` realiza lo mismo que la función `recorta` pero sobre los tres planos de color de `I`. El diagrama de flujo de esta función se muestra en la figura 3-4b.

3.3.6 Función borra

La tarea de la función `[H] = borra (H, xmin,xmax,ymin,ymax)` es eliminar el objeto encerrado por los límites de entrada. Esta función recibe como entradas la imagen de la región de los núcleos llenada `H(i,j)` y los límites del rectángulo a borrar `xmin, xmax, ymin, ymax` y regresa como salida la misma imagen de entrada `H(i,j)` con la región encerrada por el rectángulo mencionado igualado al color del fondo. Esto es, cambia el color original de los píxeles con coordenadas (i, j) donde $y_{\min} \le i \le y_{\max}$ y $x_{\min} \le j \le x_{\max}$ a un color negro (valor=0). El diagrama de flujo de esta función se muestra en la figura 3-4c.

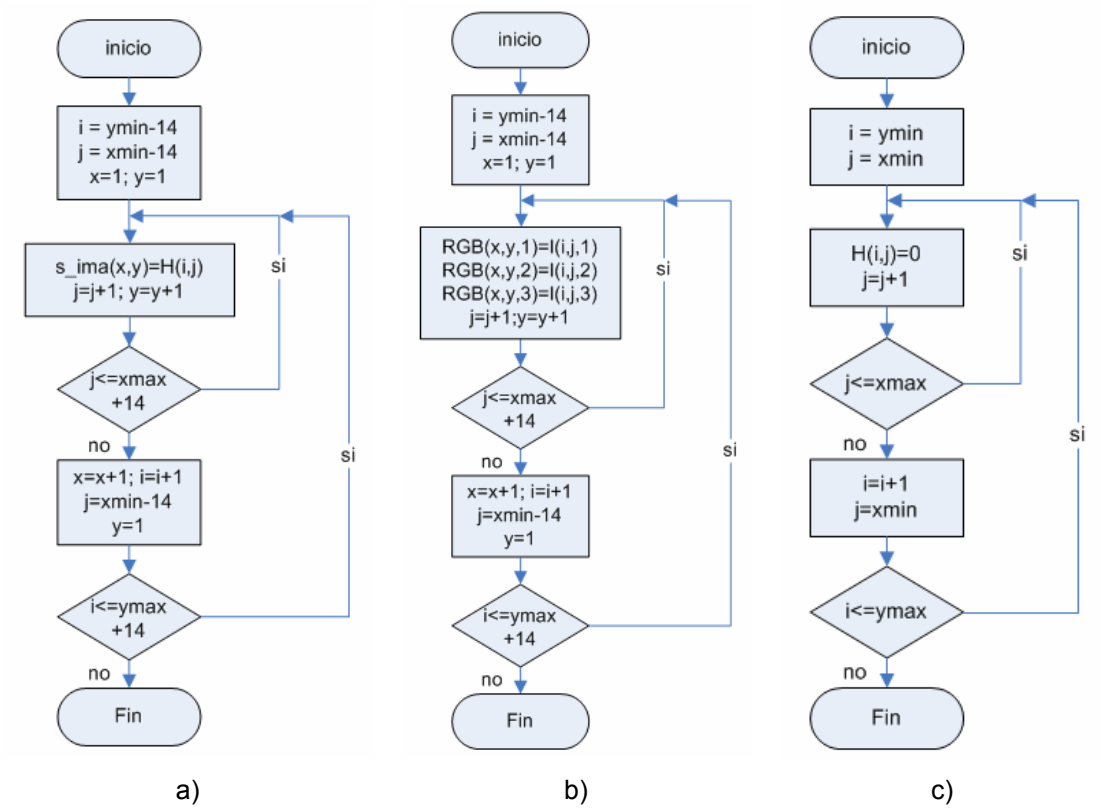


Figura 3-4 a) Diagramas de flujo de las funciones: a) recorta, b) recortaRGB, c) Borra

3.3.7 Función s_color

El objetivo de la función `[ima_n] = s_color (Q,cc,cr)` es determinar los límites donde termina el citoplasma de la célula contenida en `Q` y reducir el tamaño de `Q` al tamaño determinado por estos límites (recuadro continuo Figura 3-5a). La función recibe como entradas la imagen a disminuir `Q`, la columna del centro de la imagen `cc` y el renglón del centro de la imagen `cr` y regresa como salida la imagen de entrada reducida `ima_n`.

Como se puede observar en el diagrama de flujo de la figura 3-5b, la función se implementó por medio de un ciclo cuya variable de control “p” va de 14 a 1. Dentro de este ciclo la función realiza un recorrido de búsqueda de

los límites de segmentación (`Lsup, Lizq, Linf, Lder`) iniciando en el rectángulo que encierra al núcleo $(p,cc),(cr,c\max-p),(r\max-p,cc),(cr,p)$ y terminando en los bordes finales de la imagen. Durante este recorrido la función determina si el píxel de la imagen de entrada `Q (,)` es o no un límite de segmentación, probando los siguientes casos:

- ¿En `Q (,)` termina citoplasma del leucocito e inicia un glóbulo rojo?
- ¿En `Q (,)` termina citoplasma del leucocito e inicia el color del fondo?

Una vez que se tienen los límites de segmentación, esto es b1=b2=b3=b4=1, entonces se reduce el tamaño de la imagen `Q` invocando a la función `recorta_RGBf`.

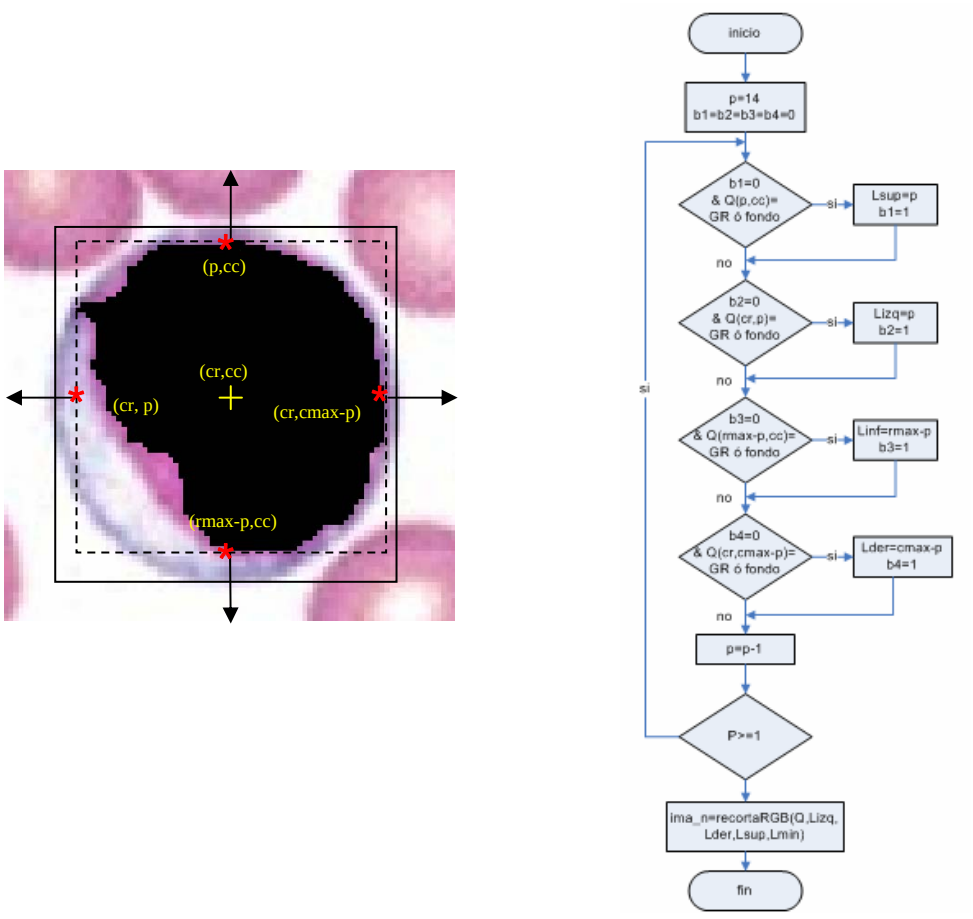


Figura 3-5 Función s_color a) Representación gráfica, b) Diagrama de flujo

3.3.8 Aproximación del leucocito a través de una elipse

Después de realizar un refinamiento de la aproximación de la imagen del leucocito por medio de un rectángulo, se realiza una mejor aproximación de la célula a través de una elipse. Como consecuencia en esta etapa se calculan los puntos de la elipse delimitada por el rectángulo obtenido con la función `s_color`.

Los puntos (x, y) de la elipse fueron calculados en matlab de la siguiente forma: Los valores de la coordenada "x" toman el valor de cada una de las columnas de la imagen y los valores de la coordenada "y"fueron calculados por medio de un ciclo `for` cuya variable de control es la coordenada "x". Dentro de este ciclo `for` el valor de "y" se obtuvo por medio de la ecuación $y = b * \left(\pm \frac{\sqrt{a^2 - (x-h)^2}}{a} \right) + k$, que corresponde al despeje de "y" en la ecuación característica de la elipse [28].

3.4 Descripción de las etapas del diseño software

3.4.1 Almacenamiento de la imagen RGB del frotis sanguíneo

Como se mencionó en la sección 3.2.3 el objetivo de esta primera etapa es almacenar los valores de los píxeles de la imagen a procesar en una matriz de datos por planos de color. Para lograr esto se abre desde Matlab el archivo JPG de la imagen a procesar, después de esto, la imagen se divide en sus tres planos de color (Red, Green, Blue RGB) y finalmente los píxeles de cada plano se almacenan como números enteros sin signo de 8 bits (píxeles en el rango de [0,255]).

3.4.2 Detección de la región de los núcleos de los leucocitos

Con el objetivo de lograr la detección de la región de los núcleos descrita en la sección 3.2.3, en esta etapa primero se determina con ayuda del histograma del plano de color verde un umbral de segmentación. Para calcular el umbral de segmentación se obtiene el plano de color verde de la imagen RGB a procesar y se calcula su histograma, después se determinan los mínimos de la gráfica del histograma en el rango [0,100], posteriormente se calcula el punto de inflexión del mínimo más cercano a 100 (umbral) y se umbraliza el plano de color verde de la imagen RGB ejecutando la función `filtro_histograma` descrita en la sección 3.3.1. Finalmente, después de que la umbralización se realiza, se ejecuta la función `llena_huecos` descrita en la sección 3.3.2 para eliminar los huecos presentes en la región de los núcleos. La figura 3-6 muestra el diagrama de flujo de la etapa “Detección de la región de los núcleos de los leucocitos”.

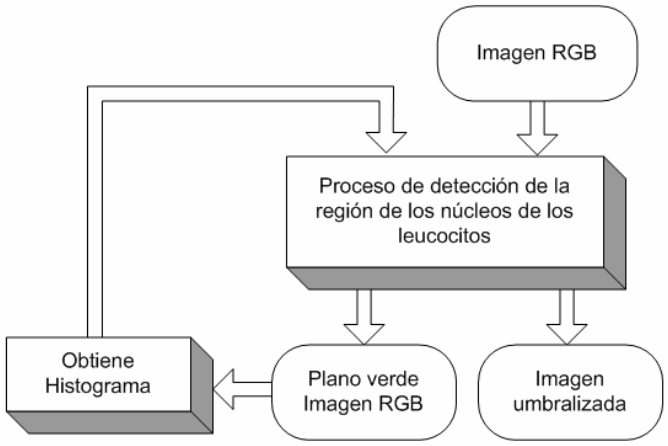


Figura 3-6 Diagrama de flujo “etapa detección de la región de los núcleos de los leucocitos”

3.4.3 Separación y almacenamiento de cada leucocito en una nueva imagen.

En esta etapa se separan cada uno de los leucocitos localizados sobre la imagen RGB del frotis sanguíneo a `n` imágenes de dimensión $l_{x_n} * l_{y_n}$ donde `n` es el número de leucocitos en la imagen y su dimensión depende del tamaño de su núcleo. La separación de cada uno de los leucocitos se realiza usando un método de detección de posición de su núcleo sobre la imagen umbralizada y realizando una aproximación de éste por medio de un rectángulo.

Para determinar la posición del primer núcleo sobre la imagen umbralizada, primero se realiza un barrido de la imagen (sentido arriba-abajo e iniciando en la esquina superior izquierda) buscando el primer punto (x_0,y_0) donde se dé un cambio de intensidad de 0 a 1 (Figura 3-7) ya que esto nos indicará la presencia del primer núcleo. Una vez que se localiza la posición del primer núcleo se genera un rectángulo del tamaño y área de éste ejecutando la función `rectangulo` descrita en la sección 3.3.3. Posteriormente la posición y tamaño de este rectángulo son utilizados para generar dos imágenes. La primera imagen corresponde a una máscara del núcleo del leucocito localizado y es generada ejecutando la función `recorta` (sección 3.3.4) sobre la imagen de la región de los núcleos. La segunda imagen corresponde a la imagen completa del leucocito y es generada ejecutando la función `recortaRGB` (sección 3.3.5) sobre la imagen RGB del frotis sanguíneo. Finalmente, estas dos imágenes son multiplicadas para crear una imagen final del leucocito que se caracteriza por presentar su núcleo en color negro. El objetivo del color negro del núcleo es facilitar el cálculo de su área en la etapa “cálculo de la proporción N/C”. Para separar cada uno de los leucocitos restantes localizados en la imagen RGB, se

ejecuta el mismo procedimiento mencionado anteriormente. La figura 3-8 muestra el diagrama de flujo de la etapa “Separación y almacenamiento de cada uno de los leucocitos en una nueva imagen”. Cabe señalar que el algoritmo extrae sólo aquellos leucocitos cuyo núcleo se encuentra totalmente dentro de la imagen, ya que de presentarse un leucocito parcialmente fuera de la imagen equivale a tener un borde no cerrado y por lo tanto éste es desechado.

Figura 3-7 Ejemplo de localización de la posición de los núcleos

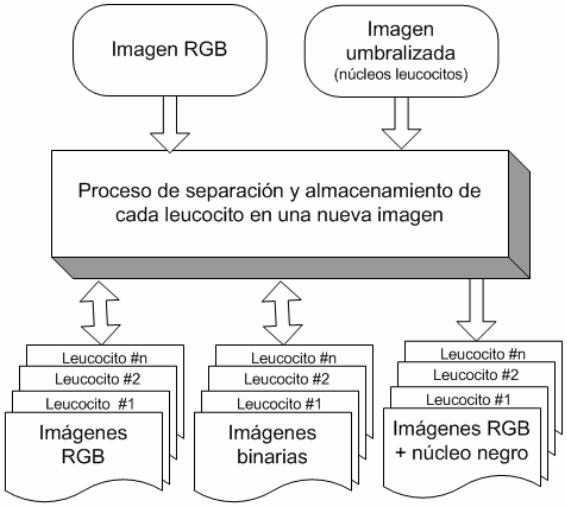


Figura 3-8 Diagrama de flujo “etapa de separación y almacenamiento de cada leucocito”

3.4.4 Segmentación por cambio de color y aproximación del leucocito a través de una elipse

En esta etapa la segmentación por color permite reducir la dimensión de cada una de las imágenes obtenidas en la etapa anterior a una dimensión menor a ($l_{x_n} * l_{y_n}$) y así obtener una mejor aproximación del tamaño del leucocito. La segmentación por color se obtiene ejecutando la función `s_color` (sección 3.3.7) para cada una de las imágenes de los leucocitos obtenidas en la etapa separación y almacenamiento de cada leucocito en una nueva imagen. Una vez que se tiene una mejor aproximación del tamaño de cada leucocito por medio de un rectángulo, se determina la elipse de mayor tamaño contenida por este rectángulo. Los ejes mayor y menor de la elipse quedan determinados por el largo y ancho del rectángulo que contiene al leucocito y los puntos se calculan de la forma descrita en la sección 3.3.8. Una vez que se tienen los puntos de la elipse todos los píxeles fuera de la elipse calculada son puestos a color blanco con la finalidad de distinguir los píxeles del fondo, de los píxeles del núcleo y de los píxeles del citoplasma. La figura 3.9 muestra el diagrama de flujo de la etapa de segmentación por cambio de color y aproximación del leucocito a través de una elipse.

3.4.5 Cálculo de la razón Núcleo/Citoplasma del leucocito.

Como se mencionó anteriormente en esta etapa se calcula el cociente del área del núcleo con respecto al área del citoplasma. Por lo tanto en esta etapa se calcula el área total de la imagen, el área del núcleo, el área del fondo y el área del citoplasma. El área total de la imagen es obtenida multiplicando el largo por el ancho de la imagen, las áreas del núcleo y el fondo son calculadas realizando un barrido por filas de la imagen, sumando

los píxeles de color negro para el caso del núcleo y sumando los píxeles de color blanco para el caso del fondo. El área del citoplasma se calcula restando del área total de la imagen el área del núcleo y el área del fondo. Finalmente una vez que se tienen el área del núcleo y del citoplasma se realiza la división de éstas y se muestra el resultado obtenido.

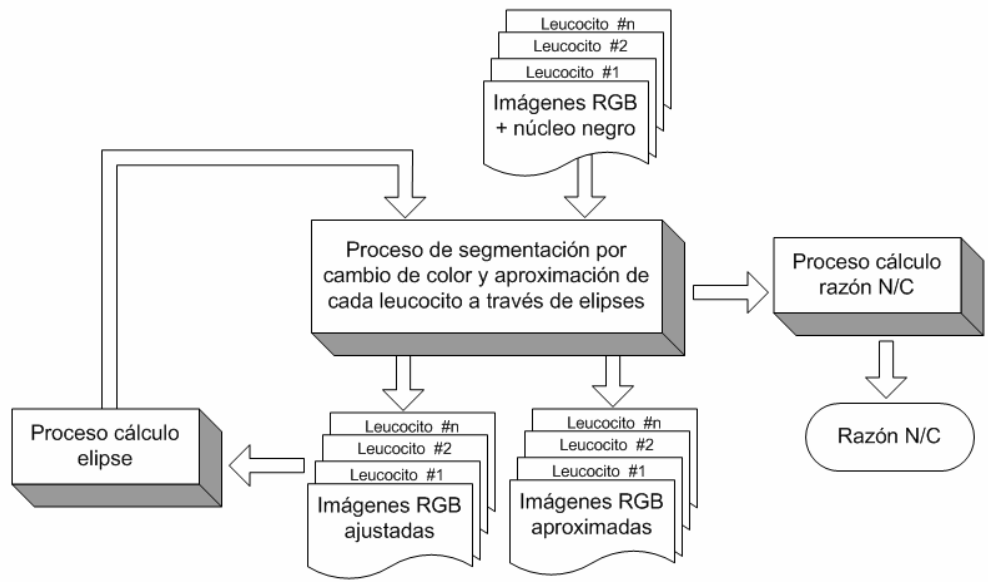


Figura 3-9 Diagrama de flujo "etapa segmentación por cambio de color y aproximación por elipse"

3.5 Resumen

El material en este capítulo estuvo enfocado al desarrollo del diseño software del sistema reportado en este trabajo. Dentro de este contexto, los puntos esenciales para un entendimiento básico del diseño han sido desarrollados e ilustrados. Se dio un énfasis especial a lo difícil que es la separación del citoplasma del leucocito, debido a que fue uno de los mayores problemas enfrentados en este trabajo.

Capítulo 4 Diseño e implementación hardware del sistema

4.1 Introducción

En el siguiente capítulo se expone el diseño e implementación hardware del diseño software discutido en el capítulo 3. En la sección 4.2 se incluyen las especificaciones de los materiales usados en la implementación del sistema. Una descripción general del sistema es proporcionada en la sección 4.3. En la sección 4.4 se da una explicación a fondo de cómo se genera el archivo binario de la imagen que es almacenada en la memoria RAM. En la sección 4.5 se reporta el diseño HDL del sistema. Finalmente en la sección 4.6 se proporciona la síntesis e implementación del diseño HDL.

4.2 Materiales

Los materiales utilizados en el sistema descrito en esta sección son del tipo software y hardware. En cuanto a la parte software los materiales utilizados fueron: El software Digilent Adept Suite utilizado para la configuración del FPGA [29, 30], el software MenUtil requerido en la configuración del módulo de memoria RAM [31], el software ISE de Xilinx versión 8.2 empleado para la síntesis del diseño HDL [32, 33] y finalmente el software Matlab versión 7.1 requerido en el desarrollo de diferentes programas de apoyo [34].

Por otro lado los materiales hardware utilizados fueron: Una tarjeta de desarrollo Spartan 3 starter kit que contiene un FPGA XC3S200FT256 [35] donde se implementó el diseño hardware y un módulo de “displays” de 7 segmentos donde se mostró la proporción núcleo-citoplasma, un módulo de memoria RAM MEM1C1 [36] externo a la tarjeta de desarrollo con capacidad de 512 Kbytes donde se almacenó la imagen del leucocito a procesar, un módulo de programación externo USB2 utilizado para programar la memoria RAM [37], un convertidor digital analógico (Digital to Analog Converter DAC) de 24 bits externo utilizado como interfaz al monitor VGA[δ] y un monitor VGA donde se mostró la imagen procesada.

4.3 Descripción del sistema hardware

El sistema hardware que se desarrolla en esta sección, se muestra en la figura 4-1. El sistema parte de la primera aproximación de la imagen de un leucocito realizada por medio de un rectángulo[β]. Esta imagen posteriormente se almacena en el bloque de memoria RAM. Una vez que se tiene la imagen almacenada en la memoria RAM se programa el FPGA con el diseño hardware del sistema. Éste a su vez ejecuta las acciones de procesamiento sobre la imagen almacenada y finalmente muestra la razón núcleo-citoplasma de la célula en los “displays” y la imagen modificada en el monitor VGA.

[δ] El convertidor D/A de 24 bits utilizado fue diseñado en base al DAC de 12 bits de la referencia [38]
[β] Imagen resultado de la etapa separación y almacenamiento de cada leucocito en una nueva imagen.

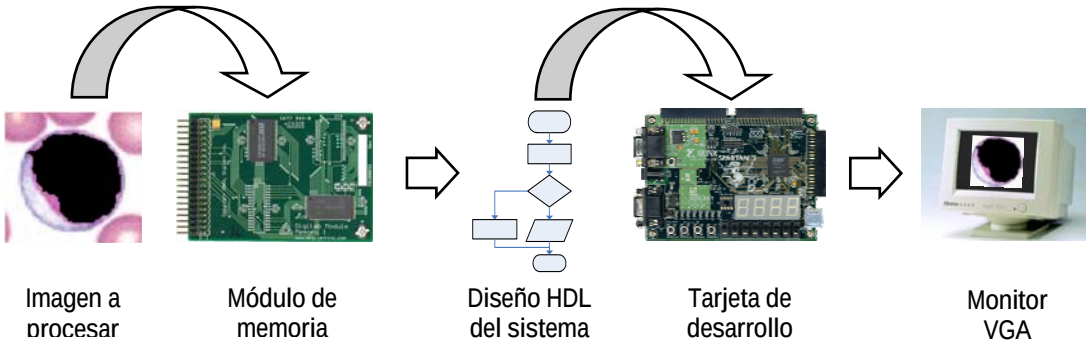


Figura 4-1 Diagrama a bloques del sistema hardware

4.4 Generación del archivo binario de la imagen

El archivo binario de la imagen se genera mediante un programa realizado en matlab[χ]. En este programa se asume que la imagen del leucocito es almacena por filas en el módulo de memoria RAM, de tal forma que cada píxel de la imagen ocupa una localidad de memoria equivalente a un byte (8 bits).

Al inicio del programa la imagen RGB del leucocito con dimensiones (n x m) es separada en sus tres planos de color (rojo, verde, azul), después estos tres planos de color son almacenados en forma numérica en una matriz de dimensión (n x m x 3). Posteriormente la dimensión de esta matriz es transformada de (n x m x 3) a (100 x 100 x 3) debido a que el sistema hardware está diseñado para trabajar con imágenes de dimensión menor o igual a 100 x 100. Una vez que el tamaño de la matriz ha sido completado con ceros, cada una de sus partes es transpuesta y almacenada por columnas en un archivo binario, donde cada elemento de la matriz es almacenado como un entero sin signo de 8 bits.

[χ] Para mayor detalle consultar el apéndice B, sección B.1

4.5 Diseño HDL del sistema

La arquitectura a bloques del diseño HDL de mayor jerarquía se muestra en la figura 4-2. En este diseño el módulo etiquetado como `Dis_ima` lee los datos de la imagen almacenada en la memoria RAM, calcula el área del núcleo, determina las dimensiones del rectángulo que mejor se aproxima a la forma del leucocito, modifica el tamaño de la imagen almacenada en RAM al tamaño del rectángulo calculado y le pasa al módulo `Puntos_elipse` las dimensiones de esta imagen.

Después de que el módulo `Dis_ima` termina su tarea, habilita al módulo `Puntos_elipse`, el cual recibe las dimensiones de la imagen modificada, calcula los puntos de la elipse que aproxima la forma del leucocito, modifica en RAM los valores de los píxeles correspondientes a un cuarto de la elipse y calcula el área fuera de ella (área del fondo).

Una vez que se tiene el valor del área del fondo, en la arquitectura del módulo `sub_top` se calcula el área del citoplasma y la proporción núcleo-citoplasma. Después de realizar estos cálculos el controlador de VGA (`módulo VGA`) muestra la imagen modificada del leucocito y finalmente el controlador de “displays” (`módulo decoder`) muestra el valor de la proporción N/C. El diseño del `módulo VGA` fue obtenido de la referencia [39] y ligeramente modificado a nuestras necesidades por esta razón, no será expuesto en este trabajo.

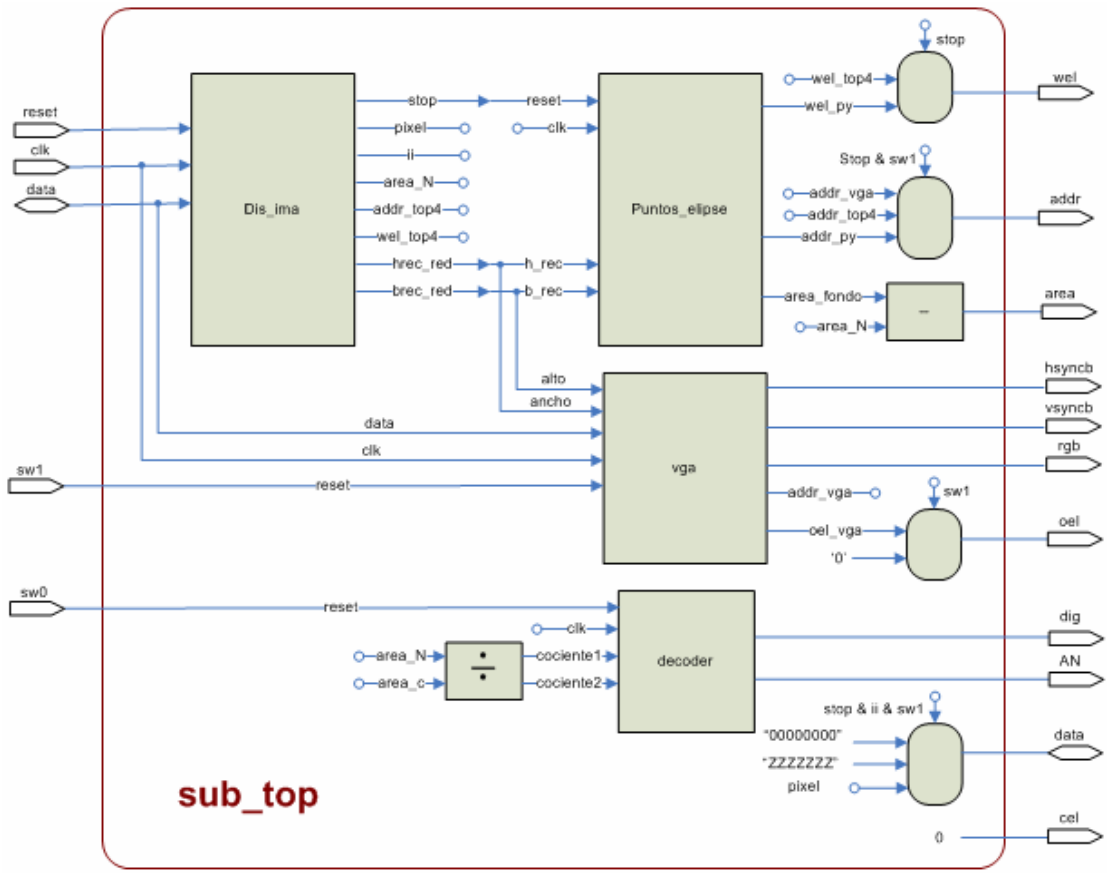


Figura 4-2 Arquitectura hardware del diseño HDL

4.5.1 Módulo Disminuye imagen (Dis_ima)

El módulo `Dis_ima` es la versión hardware de la función `s_color` de matlab y al igual que ésta distribuye su tarea en dos partes. La primera, encargada de determinar los límites del rectángulo que aproxima mejor al leucocito es realizada por el módulo `lim_cell` y la segunda encargada de reducir el tamaño de la imagen al tamaño determinado por los límites calculados es realizada por el módulo `f_corta`.

4.5.1.1 Módulo límites de la célula (lim_cell)

El módulo `lim_cell` realiza un barrido de 14 píxeles en las direcciones superior, inferior, izquierda y derecha, donde busca aquellos píxeles cuyo color sea igual al de un glóbulo rojo o al del fondo de la imagen y una vez que encuentra esos píxeles determina los límites del rectángulo.

Así el diseño HDL del módulo `lim_cell` consta de los siguientes procesos:

- El proceso clk2 determina el orden de ejecución de cada uno de los procesos del módulo Lim_cell.
- El proceso cont_13_0 el cual genera un contador de 13-0 que modifica las direcciones de los píxeles de la imagen para de este modo hacer un barrido de los 14 píxeles dentro del área del citoplasma.
- Los procesos adder1 a adder12 encargados de generar las direcciones RAM de los píxeles de los tres planos de color (RGB) en las cuatro direcciones (superior, inferior, izquierda y derecha).
- El proceso det_lim el cual determina los píxeles de color igual al color de un glóbulo rojo o al color del fondo de la imagen.
- Los procesos RegR, RegG, RegB encargados de respaldar los datos de los píxeles RGB leídos de la RAM.

4.5.1.2 Módulo función corta (f_corta)

El módulo `f_corta` es la versión hardware de la función `recortaRGBf` y de forma similar, su objetivo es reducir el tamaño de la imagen al rectángulo cuyos límites fueron calculados por el módulo `lim_cell`. Como consecuencia el módulo `f_corta` copia los píxeles de la imagen que se

encuentran dentro de los límites calculados por `lim_cell` y genera con ellos la imagen reducida.

De manera que el diseño HDL del módulo `f_corta` consta de los siguientes procesos:

- El proceso origen encargado de determinar el orden de ejecución de cada uno de los procesos del módulo `f_corta`.
- El proceso dir_lect el cual genera las direcciones de los píxeles a leer de la RAM
- El proceso dir_escr encargado de generar las direcciones a escribir en RAM
- Un proceso encargado de generar la señal de finalización de módulo
- Un proceso cuya tarea es respaldar el píxel leído de la RAM
- Un proceso encargado de generar la señal de habilitación de escritura de la RAM
- El proceso cont_areas encargado de calcular el área del núcleo

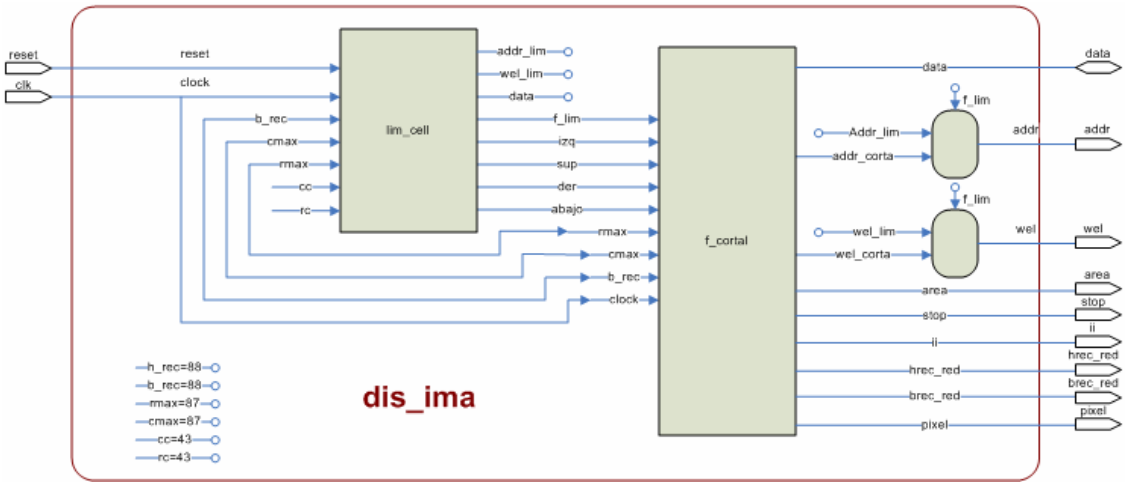


Figura 4-3 Arquitectura a bloques del módulo Dis_ima

4.5.2 Módulo puntos de la elipse (Puntos_elipse)

El módulo `puntos_elipse` es la implementación hardware de la parte "Aproximación del leucocito a través de una elipse". Este módulo calcula los puntos de la elipse encerrada por el rectángulo obtenido con el módulo `lim_cell` y dibuja algunos de estos puntos. Al igual que en su contraparte implementada en matlab, para el cálculo de los puntos (x, y) de la elipse se tomaron los valores de cada una de las columnas de la imagen como valores de la coordenada "x" y los valores de la coordenada "y" fueron calculados por medio de la ecuación $y = b * \left(\pm \frac{\sqrt{a^2 - (x-h)^2}}{a} \right) + k$, que corresponde al despeje de "y" en la ecuación característica de la elipse. La parte de la raíz cuadrada fue implementada como una tabla de valores que corresponden a la raíz cuadrada de los números enteros en el rango [0, 2500] debido a que el término $a^2 - (x-h)^2 \leq 2500$ y los valores de las coordenadas de los píxeles no pueden ser números flotantes. En cuanto a la operación de la división, ésta fue implementada mediante un algoritmo basado en la referencia [40] y ligeramente modificada para nuestros fines.

El diseño HDL del módulo `puntos_elipse` consta de los siguientes procesos:

- El proceso clk2 que determina el orden de ejecución de cada uno de los procesos del módulo `puntos_elipse`.
- El proceso cont_1_A encargado de generar los valores de la coordenada "x" para la evaluación de los puntos de la elipse.
- El proceso contador encargado de generar los valores de "x" para el cálculo de la dirección de los puntos de la elipse a dibujar.

- El proceso escritura cuya función es generar la señal de habilitación de escritura de la RAM.
- El proceso fondo encargado de calcular el área del fondo de la imagen (suma los píxeles que se encuentran fuera de la elipse calculada).
- Un proceso encargado de calcular los valores de la coordenada “y” por medio de la ecuación característica de la elipse.

4.6 Implementación HDL

En las siguientes secciones se describe la metodología del diseño estándar seguida para el diseño e implementación hardware del sistema basado en FPGA según la compañía Xilinx [13].

4.6.1 Diseño de entrada y síntesis

En esta etapa se creó un proyecto usando un lenguaje de descripción de hardware (HDL) basado en texto soportado por Xilinx (ISE 8.2). En este proyecto se añadieron las descripciones de alto nivel de los módulos `dis_ima`, `puntos_elipse`, `decoder`, `vga`, `lim_cell` y `f_corta` descritos en la sección 4.5. Posteriormente el diseño HDL de mayor jerarquía se sintetizó con la aplicación XST (Xilinx Síntesis Tool).

En la etapa de síntesis la descripción del sistema realizada en un lenguaje de alto nivel se transformó a un listado de conexiones (netlist) entre dispositivos convencionales como son: compuertas AND, OR, registros, latches, multiplexores, contadores, acumuladores y decodificadores entre otros [9]. El diagrama esquemático RTL del diseño se muestra en la figura 4-4 y los resultados de la síntesis para cada unidad de diseño se muestran en la tabla 1. La primera fila de la tabla hace referencia a los nombres de cada

unidad de diseño y la primera columna hace referencia a los componentes inferidos por la síntesis.

Módulo / Componente	puntos_elipse	decoder	vga	lim_cell	f_corta	dis_ima	sub_top	Top
Contadores	3	1	6	2	1			
Acumuladores	1							
flip-flops tipo D	45	9	50	40	36			1
Sumador/restador	17		1	14	21	6	34	
Multiplicadores	4		1	2	2		2	
Comparadores	108	1	12	11	8			
ROM		1						
Multiplexores			43		19		19	
Buffer de tres estados							8	

Tabla 1 Resultados de la síntesis para cada unidad de diseño

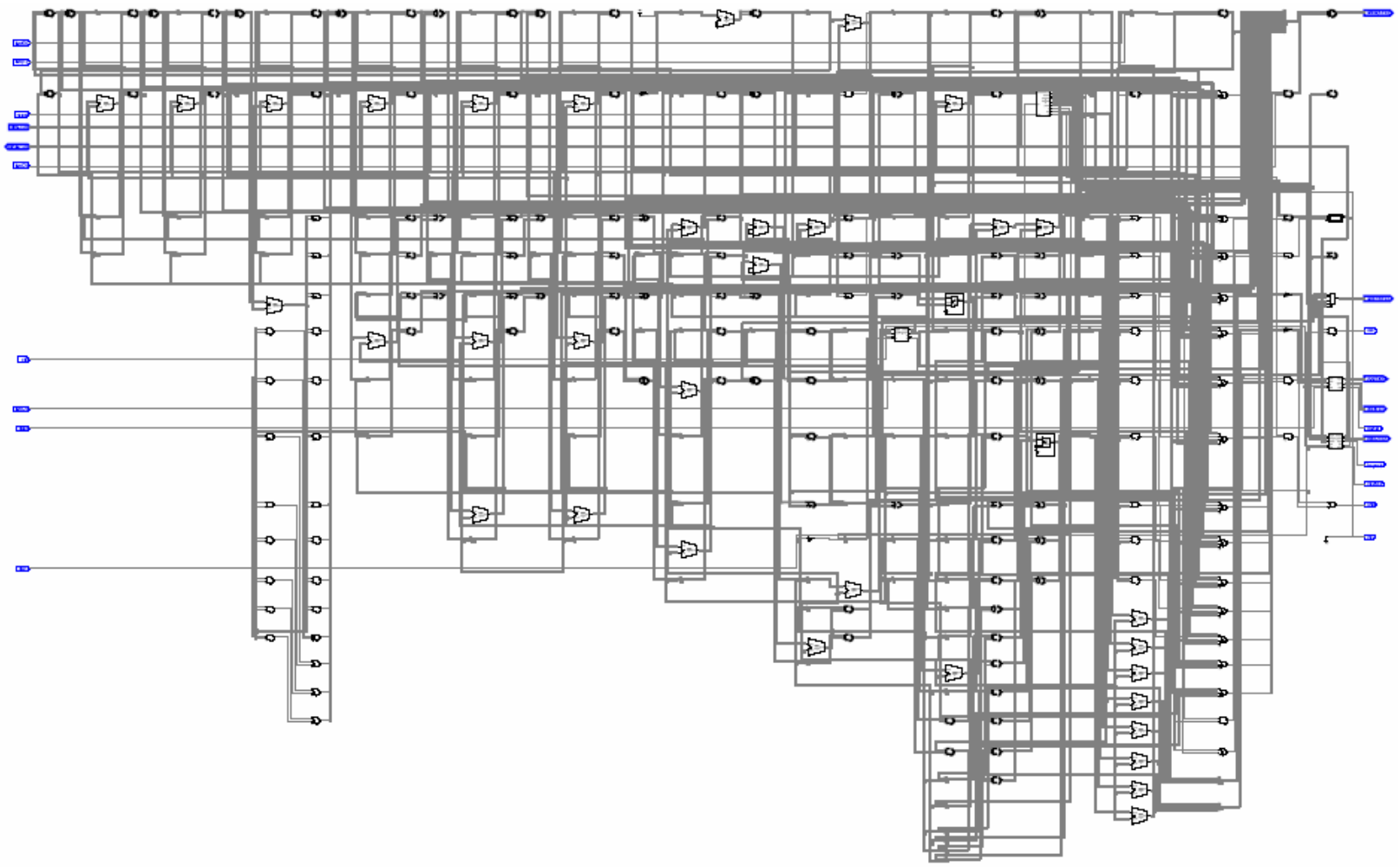


Figura 4-4 Esquemático RTL del diseño HDL

4.6.2 Mapeo

En esta etapa la descripción lógica del diseño obtenido en la etapa anterior es mapeada a componentes específicos del FPGA XC3S200FT256 a través del programa MAP; como son celdas lógicas (logic cells), celdas entrada/salida (I/O cells) y otros componentes. Los resultados de la etapa de mapeo se muestran en la tabla 2.

Device Utilization Summary				
Logic Utilization	**Used**	**Available**	**Utilization**	**Note(s)**
Number of Slice Flip Flops	306	3,840	7%	
Number of 4 input LUTs	2,614	3,840	68%	
Logic Distribution				
Number of occupied Slices	1,632	1,920	85%	
Number of Slices containing only related logic	1,632	1,632	100%	
Number of Slices containing unrelated logic	0	1,632	0%	
Total Number 4 input LUTs	3,023	3,840	78%	
Number used as logic	2,614			
Number used as a route-thru	409			
Number of bonded IOBs	89	173	51%	
IOB Flip Flops	34			
Number of MULT18X18s	10	12	83%	
Number of GCLKs	2	8	25%	
Total equivalent gate count for design	67,067			
Additional JTAG gate count for IOBs	4,272			

Tabla 2 Resultados de la etapa de Mapeo del diseño HDL

4.6.3 Colocado y ruteo

En esta etapa se colocaron y rutearon los componentes físicos obtenidos en la etapa de mapeo sobre el FPGA con ayuda del programa PAR (Place And Route). El colocado y ruteado se ejecutó en el modo “Evaluación de desempeño” para mejorar automáticamente el desempeño de los relojes internos del diseño. En la figura 4-5 se muestra la forma en que quedó ruteado el diseño VHDL sobre el FPGA XC3S200FT256 y un resumen de los retardos máximos es suministrado en la tabla 3.

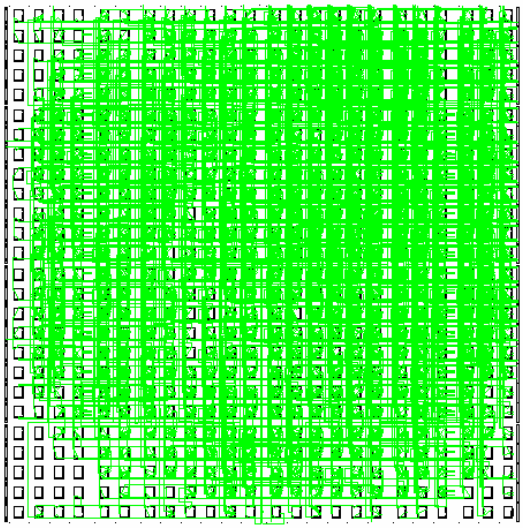


Figura 4-5 Ruteo del diseño HDL sobre el FPGA XC3S200FT256

Resumen de retardos	
Retardo promedio de conexión	1.272ns
Retardo máximo de pin	7.423ns
Retardo promedio de conexión sobre las 10 peores redes	5.453

Tabla 3 Resultados de la etapa de colocado y ruteo del diseño HDL

4.6.4 Generación del archivo de configuración

En esta etapa se generó el archivo de configuración del dispositivo el cual es un conjunto de bits de datos (bitstream) que contienen información para la colocación física de CLBs, IOBs, TBUFs (3-state buffers), pines y elementos de ruteo. Este archivo se generó con el programa BitGen y posteriormente se cargó al dispositivo FPGA con el programa Digilent TransPort.

4.7 Resumen

El material en este capítulo estuvo enfocado a la descripción del diseño e implementación hardware del sistema reportado en este trabajo. Con la información expuesta se buscó proporcionar los puntos esenciales del diseño así como proveer detalles suficientes para la reproducción del sistema desarrollado en este trabajo.

Capítulo 5 Resultados y pruebas

5.1 Introducción

En este capítulo se presentan los resultados de esta investigación. En la sección 5.2 se ilustra el diseño software aplicado a un caso de estudio. En la sección 5.3 se presenta el diseño hardware aplicado al mismo caso de estudio de la sección 5.2 y finalmente en la sección 5.4 se evalúa el desempeño del sistema expuesto.

5.2 Prueba del diseño software mediante un caso de estudio

En el transcurso de esta sección se ejemplifican las etapas del diseño software con ayuda de una imagen de prueba. La imagen utilizada fue una imagen jpg de 640x480 píxeles que contiene tres mieloblastos y varios glóbulos rojos (Figura 5-1a).

Como se mencionó en el capítulo 3, el flujo del diseño software consta de 5 etapas principales: Apertura de la imagen RGB del frotis sanguíneo, detección de la región de los núcleos de los leucocitos, separación y almacenamiento cada leucocito en una nueva imagen, segmentación por cambio de color y aproximación del leucocito a través de elipses y finalmente el cálculo de la proporción núcleo/citoplasma del leucocito. El desarrollo de cada una de las etapas se menciona en las siguientes secciones.

5.2.1 Almacenamiento de la imagen RGB del frotis sanguíneo

En esta etapa la primera función que se ejecuta es `imread`, la cual recibe como parámetro de entrada el archivo de la imagen "*mieloblasto_640x480.jpg*" y proporciona como salida una matriz `I`. Después se ejecuta la función `imshow` que se encarga de mostrar la matriz resultante `I` (figura 5-1a) y finalmente se ejecuta la función `I(:,:,2)` la cual recibe como parámetro de entrada la matriz de la imagen RGB, obtiene el plano de color verde y lo almacena en una nueva matriz (Figura 5-1b).

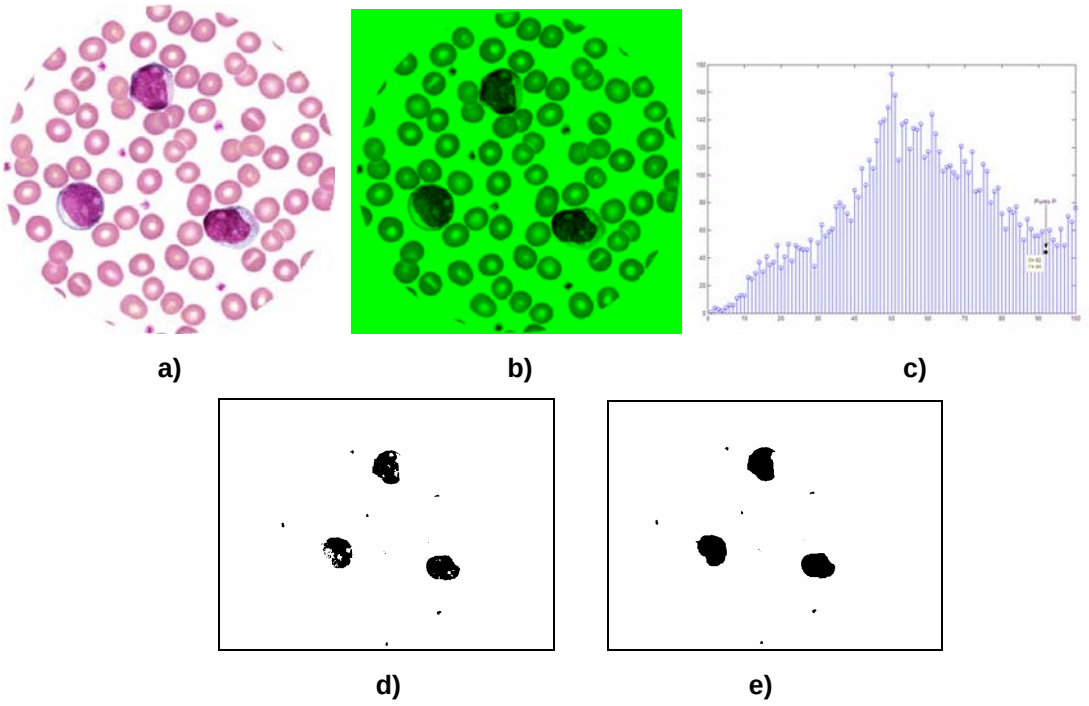


Figura 5-1 a) Ejemplo de la imagen RGB de un frotis sanguíneo a procesar, b) Plano de color verde de la imagen RGB, c) Histograma del plano verde, d) Región de los núcleos de los leucocitos, e) Región de los núcleos de los leucocitos llenada

5.2.2 Detección de la región de los núcleos de los leucocitos

En esta etapa la primera función que se ejecuta es `imhist` la cual obtiene el histograma del plano de color verde. Después de esto se determinan los mínimos de la gráfica y el umbral de segmentación `P` (Figura 5-1c) a través de un ciclo for. Posteriormente se ejecuta la función `filtro_histograma` la cual regresa la imagen binaria del plano de color verde de la imagen RGB haciendo uso del umbral de segmentación `P` (Figura 5-1d) y finalmente se ejecuta la función `llena_huecos` la cual llena los huecos de los núcleos de la imagen aplicando operaciones de dilatación y erosión (Figura 5-1e).

5.2.3 Separación y almacenamiento de cada leucocito en una nueva imagen

En esta etapa primero se separan cada uno de los leucocitos en una nueva imagen. Para esto la primera función que se ejecuta es `rectangulo` la cual determina la posición y dimensión del rectángulo más pequeño que encierra al núcleo del primer leucocito (Figura 5-2a). La segunda función que se ejecuta es `recortaRGB` la cual extrae de la imagen RGB (Figura 5-2c) la imagen encerrada por el rectángulo calculado. La tercera función que se ejecuta es `recorta` la cual extrae de la imagen binaria (Figura 5-2b) la imagen encerrada por el rectángulo calculado, después de esto se realiza una multiplicación de las imágenes separadas (Figuras 5-2b y 5-2c) y se obtiene una imagen a color con el núcleo en color negro (Figura 5-2d). Posteriormente el núcleo del primer leucocito se elimina de la imagen binaria ejecutando la función `borra` y después se ejecutan las mismas funciones para los demás leucocitos (Figuras 5-2e, 5-2f, 5-2g, 5-2h, 5-2i, 5-2j, 5-2k, 5-2l).

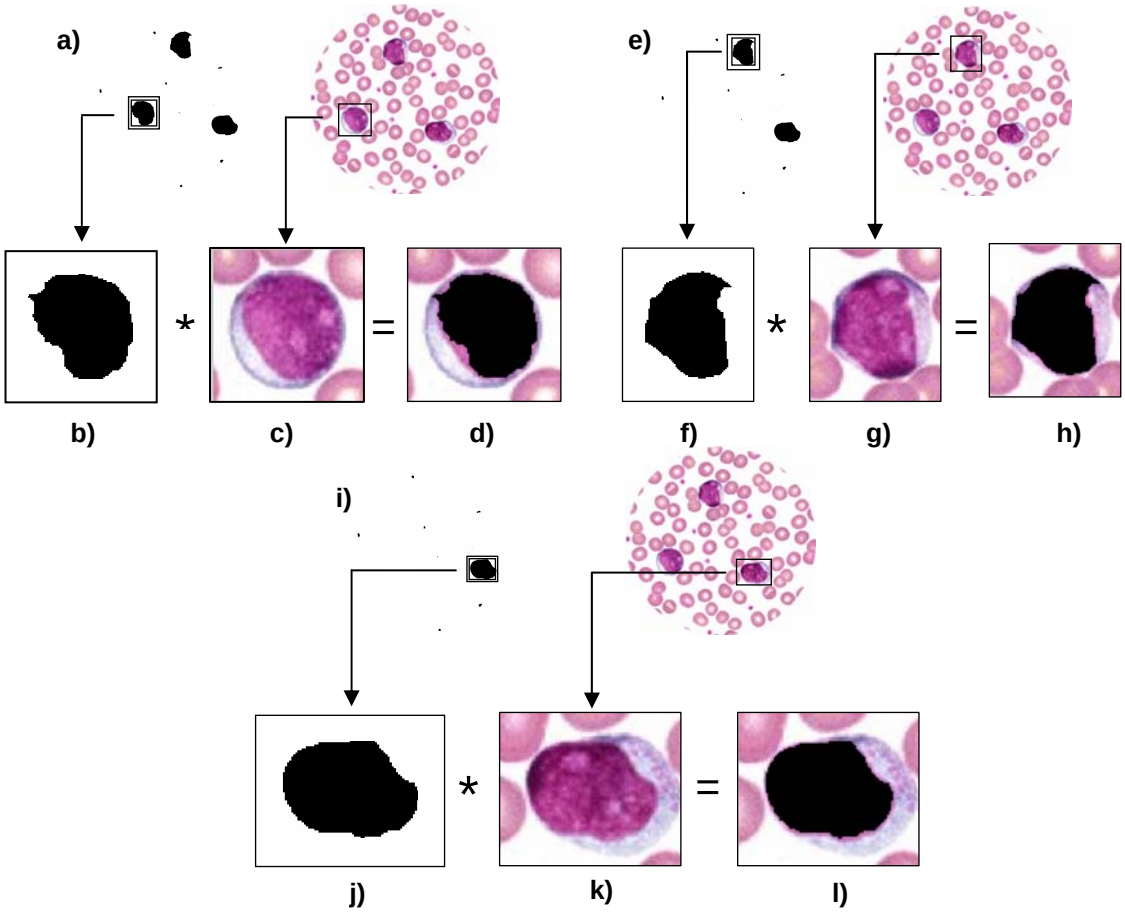


Figura 5-2 Ejemplificación de la etapa separación de cada leucocito en una nueva imagen

5.2.4 Segmentación por cambio de color y aproximación del leucocito a través de elipses

En esta etapa se reduce cada una de las imágenes obtenidas en la etapa anterior y se calculan los puntos de las elipses que aproximarán a cada uno de los leucocitos.

La primera función que se ejecuta es `size`, esta función se encarga de obtener las dimensiones de la imagen que serán usadas para calcular su

centro. Una vez que se tiene el centro de la imagen se ejecuta la función `s_color` cuyo objetivo es determinar los puntos de segmentación y reducir el tamaño de la imagen al tamaño determinado por estos puntos (Figura 5-3a). En las figuras 5-3b y 5-3c se muestran las imágenes reducidas de las figuras 5-2h y 5-2l.

Figura 5-3 a) Imagen reducida del primer leucocito, b) Imagen reducida del segundo leucocito, c) Imagen reducida del tercer leucocito

Después de reducir la imagen al tamaño del leucocito, se realiza una mejor aproximación de la célula a través de una elipse (Figuras 5-4a, 5-4b y 5-4c) y finalmente todos los píxeles fuera de la elipse calculada son borrados (color blanco) con la finalidad de distinguir en la siguiente etapa los píxeles del fondo, de los píxeles del núcleo y de los píxeles del citoplasma (Figuras 5-5a, 5-5b, 5-5c).

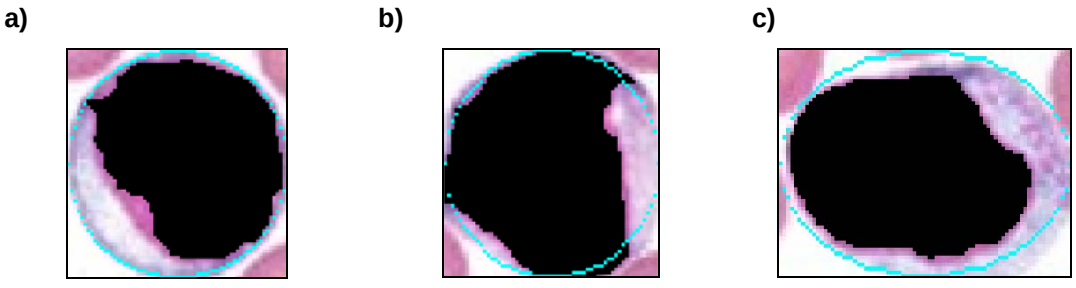


Figura 5-4 Aproximación de los glóbulos blancos por medio de elipses

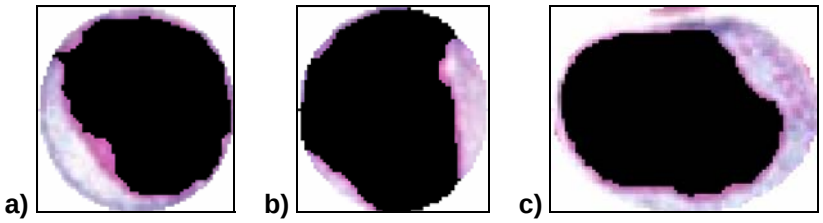


Figura 5-5 Eliminación de los objetos localizados fuera de la elipse

5.2.5 Cálculo de la razón N/C del leucocito

Como se comentó anteriormente, en la etapa final el objetivo es calcular el área del núcleo y citoplasma de cada leucocito y obtener la razón del área del núcleo con respecto a la del citoplasma.

El área total de la imagen es obtenida multiplicando el largo x ancho de la imagen, las áreas del núcleo y el fondo son calculadas mediante un ciclo `for` que barre toda la imagen y suma los píxeles de color negro para el caso del núcleo y los píxeles de color blanco para el caso del fondo. El área del citoplasma se calcula restando del área total de la imagen el área del núcleo y el área del fondo. Finalmente una vez que se tienen el área del núcleo y del citoplasma se realiza la división de éstas y se muestra el resultado obtenido (Tabla 4).

Glóbulo Blanco	**Figura 5-5a**	**Figura 5-5b**	**Figura 5-5c**
Área total =largo x ancho	4623 píxeles	3965 píxeles	4819 píxeles
Área del núcleo	2536 píxeles	2572 píxeles	2417 píxeles
Área del fondo	1028 píxeles	888 píxeles	1076 píxeles
Área del citoplasma	1059 píxeles	505 píxeles	1326 píxeles
Proporción N/C	2.3	5.09	1.82

Tabla 4 Áreas y razón núcleo/citoplasma de los leucocitos

5.3 Prueba del diseño hardware del sistema mediante un caso de estudio

En el transcurso de esta sección se ilustra el diseño hardware aplicado a la imagen de prueba de la figura 5-1a). Como se mencionó en las secciones anteriores, el diseño hardware está limitado debido a la memoria RAM, por esta razón la parte de implementación se realiza para cada una de las figuras 5-2d, 5-2h y 5-2l de forma individual.

Los resultados de la razón núcleo/citoplasma obtenidos para cada una de las figuras se muestran en la tabla 5 y las imágenes resultantes del sistema se muestran en las imágenes de las Figuras 5-6, 5-7 y 5-8.

Figura	Razón N/C
5-2d	2.3
5-2h	5.1
5-2l	1.8

Tabla 5 Resultados de la razón N/C de la implementación hardware

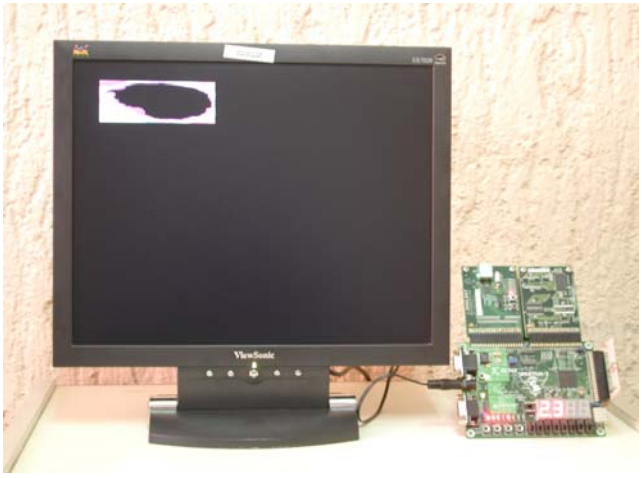


Figura 5-6 Imagen obtenida de la implementación hardware de la figura 5-2d

Figura 5-7 Imagen obtenida de la implementación hardware para la imagen de la figura 5-2h

Figura 5-8 Imagen obtenida de la implementación hardware para la imagen de la figura 5-2l

5.4 Resultados

Como se mencionó en la sección 2.3.2.2, cuando se realiza la identificación de leucocitos no se asigna un valor numérico a la razón N/C, ésta solamente es descrita como alta, moderada o baja. Por esta razón y con

ayuda de hematólogos especializados, los valores numéricos obtenidos del cálculo del cociente N/C con la fórmula mostrada a continuación, fueron clasificados en los rangos alto, medio y bajo según se muestra en la tabla 6.

$$N/C = \frac{\textit{área del núcleo}}{\textit{área del citoplasma}}$$

Valor Numérico N/C	Clasificación Relativa
$N/C \geq 1.7$	Alto
$0.95 < N/C < 1.7$	Medio a moderado
$N/C \leq 0.95$	Bajo

Tabla 6 Clasificación de los valores numéricos N/C

Una vez que se establecieron los valores numéricos para los rangos alto, medio y bajo, se probó el sistema con diferentes imágenes. Después estas mismas imágenes fueron evaluadas por hematólogos del Hospital Universitario y finalmente se realizó una evaluación comparando los resultados obtenidos del sistema con los resultados obtenidos de la clasificación de los hematólogos.

En base a los resultados obtenidos de la comparación de los resultados del sistema con los resultados de la clasificación de los hematólogos se obtuvo un porcentaje de acierto del sistema del 90.9% en la clasificación razón núcleo/citoplasma alto, medio y bajo. Este es un resultado importante ya que esto significa que de cada 100 leucocitos evaluados 91 de ellos son evaluados correctamente.

Después de realizar las pruebas del sistema, se hizo un análisis de las imágenes de los leucocitos que fueron clasificados correcta e incorrectamente, de este análisis se obtuvieron las siguientes observaciones:

- Se obtiene una buena aproximación por rectángulo en el caso de células menos maduras como es el caso de las figuras 5-9a, 5-9b, 5-9c, 5-9d, 5-9e, 5-9f, 5-9g, 5-9h y 5-9i.

- Se logra una mala aproximación por rectángulo en el caso de que el citoplasma posea un color muy parecido al de los eritrocitos como es el caso de la figura 5-10.

- Se logra una buena aproximación por elipse de la célula en el caso de que ésta tenga una forma circular como en el caso de la figura 5-9 inciso a) ó cuando la célula se aproxime por una elipse que tenga su eje mayor paralelo a alguno de los ejes cartesianos x, y como es el caso de las figuras c), d), f) e i) mostradas en la figura 5-9.

- Se logra una aproximación por elipse de la célula no tan buena en el caso de que ésta deba ser aproximada por una elipse cuyo eje mayor posea cierta rotación, como es el caso de las células mostradas en las figuras b), e), g), h, j y k de la figura 5-9.

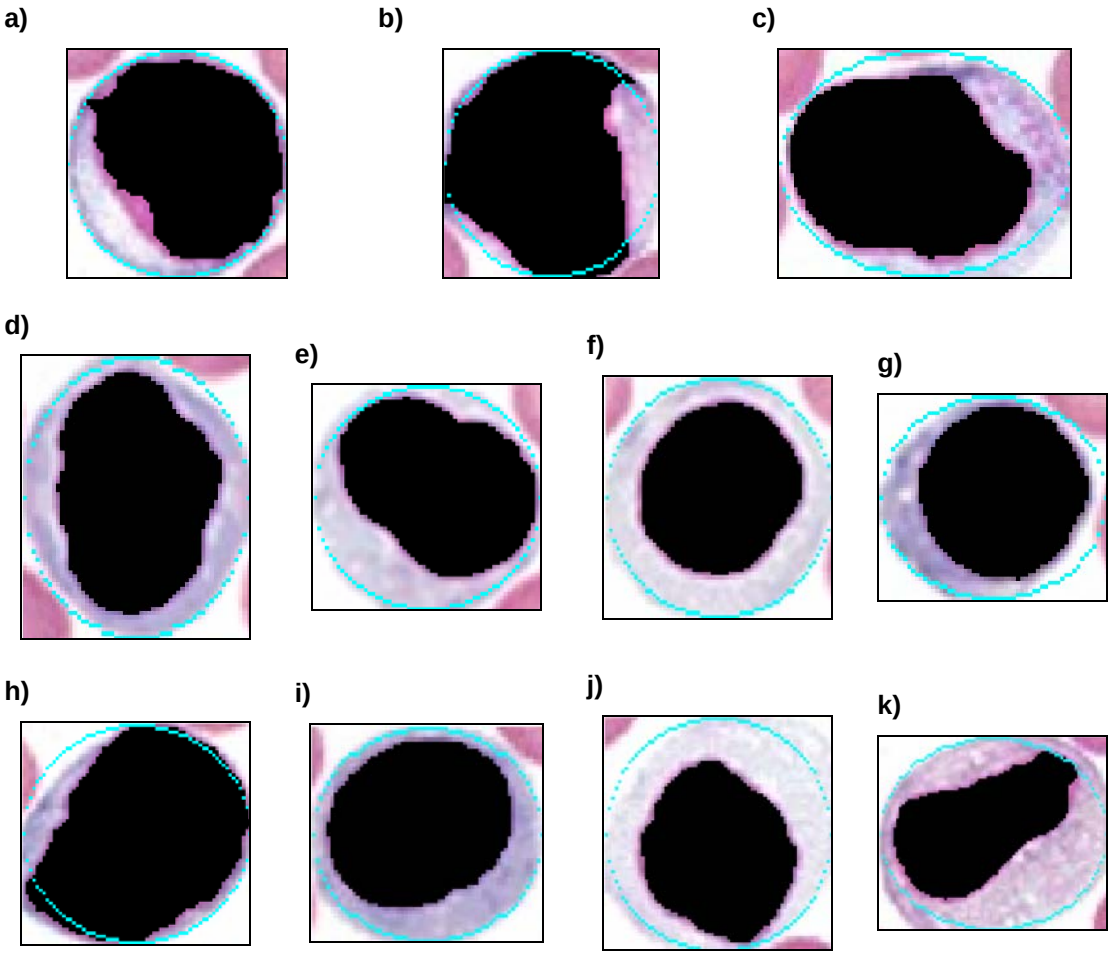


Figura 5-9 Resultados de las aproximaciones por elipses del diseño software

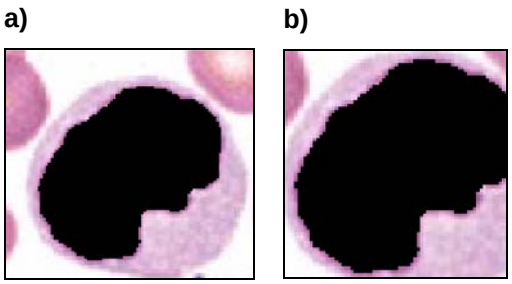


Figura 5-10 Ejemplo de la mala aproximación del leucocito a través de un rectángulo

En cuanto a la falla en la aproximación por rectángulo se pudo concluir que esto se debe principalmente a que la segmentación por color es puntual y a que la orientación de los ejes de simetría del núcleo del leucocito no es paralela a los ejes cartesianos. La posible solución a este problema es descrita en la siguiente sección.

La falla en la aproximación por elipse se debió principalmente a que la ecuación que se utilizó para calcular la elipse que aproxima al leucocito está restringida a cuando el eje mayor es paralelo a cualquiera de los ejes cartesianos. Sin embargo esto se puede mejorar de una forma sencilla incluyendo el parámetro de orientación del eje en la ecuación.

5.5 Resumen

Los propósitos principales de este capítulo han sido mostrar los resultados de las pruebas realizadas al sistema y exponer los alcances y limitaciones del diseño en general.

Capítulo 6 Conclusiones y Trabajo Futuro

6.1 Conclusiones

Con base en las pruebas y resultados obtenidos en este trabajo de investigación, se pueden hacer las siguientes conclusiones:

- Se definió un proceso para reconocimiento de leucocitos a partir de una imagen digital que contiene uno o más leucocitos, eritrocitos y plaquetas utilizando localización basada en núcleo

- Se diseñó un proceso para reconocimiento de las regiones del núcleo y citoplasma de mieloblastos usando segmentación por umbralización para la región del núcleo y segmentación por color para la región del citoplasma

- Se diseñó un proceso software que determina la razón núcleo/citoplasma de mileoblastos a través del cálculo de áreas de las regiones del núcleo y el citoplasma de la célula

- Se diseñó e implementó un sistema hardware que determina la razón núcleo/citoplasma de leucocitos inmaduros en la etapa de mieloblastos

- Se propuso un método automático que usa segmentación por color junto con información de posición espacial para la detección de la región del citoplasma

- ❖ Se establecieron bases del diseño de un sistema automatizado que permitirá incluir análisis completos de células inmaduras en el conteo diferencial de leucocitos.

6.2 Trabajo Futuro

Como se estableció al inicio de este reporte, el trabajo de investigación realizado, establece las bases de un sistema automatizado que ambiciona incluir análisis completos de células inmaduras por medio de análisis de imagen. Con la finalidad de dar seguimiento al desarrollo de este sistema, se plantean tres nuevos trabajos de investigación.

1. El primer trabajo pretende mejorar la aproximación de la forma del leucocito realizada en la presente investigación. Para ello se pretende determinar los ejes de simetría de la forma del núcleo, calcular la elipse más pequeña que lo encierra y hacer crecer la elipse hasta encontrar ya sea el primer eritrocito o el color del fondo.

2. El segundo trabajo busca añadir al sistema diseñado en la presente investigación, la determinación de la razón N/C de los leucocitos en las etapas de maduración restantes (promielocito, mielocito, metamielo, neutrófilo en banda y neutrófilo segmentado).

3. El tercer trabajo ambiciona incluir al sistema diseñado todos los criterios morfológicos utilizados en la identificación de leucocitos como son: color del núcleo, patrón de cromatina del núcleo, color del citoplasma, tipo de gránulos incluidos en el citoplasma y presencia de otras inclusiones.

Como trabajo final se pretende que los tres proyectos del trabajo a futuro sean integrados en una sola herramienta facilite la identificación de células inmaduras, permita ahorrar tiempo, disminuya imprecisiones en cuentas manuales causadas en su mayor parte por errores humanos y sea una fuente de ayuda colateral en el entrenamiento de nuevos técnicos hematólogos.

Apéndice A. Guía de usuario de la implementación del sistema

Este manual suministra una guía rápida del proceso de configuración y uso del sistema diseñado en este trabajo.

Esta guía contiene las siguientes secciones:

- "Requerimientos Software"
- "Requerimientos Hardware"
- "Conexión Hardware"
- "Programando la memoria RAM"
- "Configurando la funcionalidad del sistema en el FPGA"

A.1 Requerimientos Software

Para usar esta guía, se debe instalar el siguiente software:

- ISE 8.2i
- Matlab 7.1
- Digilent Adept Suite1.6
- Digilent MemUtil 2.0

A.2 Requerimientos Hardware

Para usar esta guía, se debe tener el siguiente hardware:

- Un kit de desarrollo Spartan 3
- Un módulo de memoria RAM MEM1C1
- Un módulo de programación externo USB2
- Un convertidor digital analógico (Digital to Analog Converter DAC) de 24 bits
- Un monitor VGA de 15''

A.3 Conexión del Hardware

La conexión del hardware es un paso importante para obtener un buen funcionamiento del sistema. La conexión del hardware se realiza como se muestra en la figura A-1 y consta de los siguientes pasos:

1. Primero se conecta el módulo USB2 al puerto de expansión A1 del kit de desarrollo de la tarjeta Spartan 3
2. Después se conecta el módulo de memoria MEM1 C1 al puerto de expansión A2 del kit de desarrollo de la tarjeta Spartan 3
3. Posteriormente se conecta el DAC de 24bits al puerto de expansión A3 de la tarjeta Spartan 3
4. Finalmente se conecta el monitor VGA al conector DB25 hembra del DAC

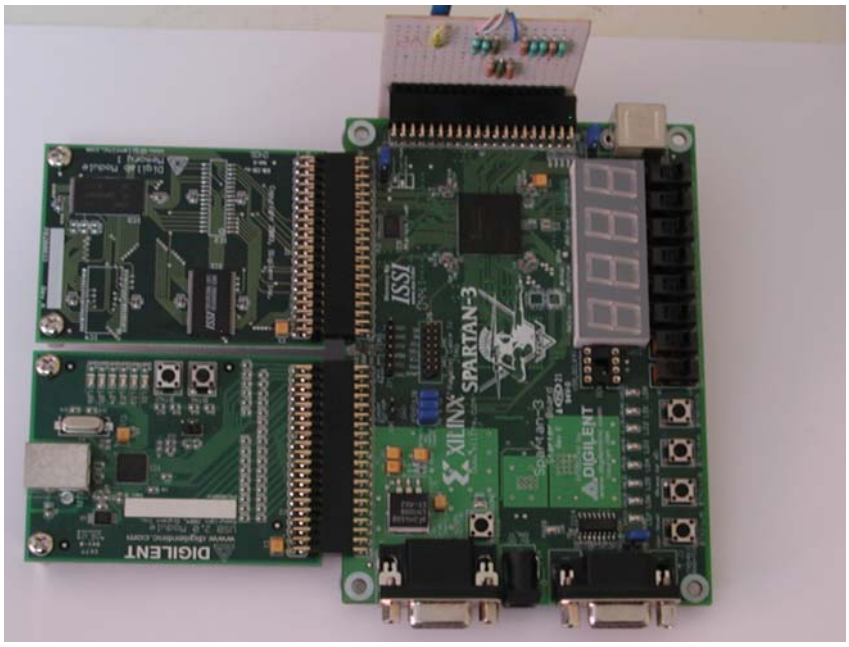


Figura A-1 Diagrama de conexión hardware del sistema

A.4 Programando la memoria RAM

El archivo binario de la imagen del leucocito a procesar se genera ejecutando el programa mostrado en la sección B.1. En este programa se debe introducir la ruta del archivo de la imagen y el nombre del archivo binario de salida. Una vez que se cuenta con el archivo binario de la imagen del leucocito, este será descargado a la memoria RAM ubicada en el módulo externo MEM C1 con ayuda del módulo USB2.

Para descargar la imagen a la memoria RAM primero se configura el FPGA con el archivo main.bit y después se programa la RAM.

A.4.1 Configurando el FPGA para la programación de la memoria RAM

Antes de programar la memoria RAM, primero debe configurarse el FPGA con el archivo main.bit debido a que el FPGA sirve como interfaz de programación entre la PC y el módulo de memoria RAM [22]. La configuración del FPGA se realiza de la siguiente forma:

1. Primero se abre el programa *ExPort* localizado en *menú inicio-> Todos los programas-> Digilent->Adept->ExPort.*
2. Después se selecciona el botón "*Add File*" y se anexa el archivo *main.bit* (Figura A2).
3. Posteriormente se selecciona *Initialize Chain* y en la caja de asignación de archivo de configuración del FPGA se elije main.bit.
4. Una vez asignado main.bit se programa el FPGA dando clic en *Program Chain.*
5. Finalmente se cierra la ventana de Digilent ExPort y se prosigue a la etapa "Programando la memoria RAM"

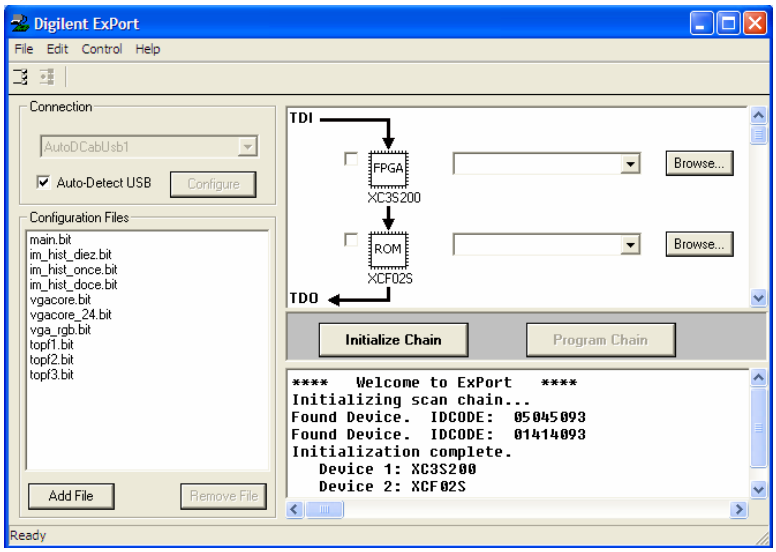


Figura A-2 Ventana del programa Digilent ExPort

A.4.2 Descargando el archivo binario

La programación de la memoria RAM consiste en descargar el archivo binario de la imagen del leucocito al módulo Mem1C1 y se realiza de la siguiente forma:

1. Primero se abre la aplicación *MemUtil.exe* localizado en el escritorio de la PC
2. Después en la pestaña *Properties* se selecciona el módulo *C1 (512K Flash, 512 RAM)* y en la caja *"Connection Properties"* se elije el cable *"AutoDModUSB1"* (Figura A-3a).
3. Una vez realizado el paso anterior se da clic en la pestaña *Load RAM* y en *File To Load* se proporciona la ruta del archivo binario de la imagen a descargar (Figura A-3b).
4. Después en "*File Start location"* y en *start Address* se introduce el número 0, en *lenght* se escribe 30000 y se da clic en el botón *Load*.
5. Finalmente se cierra la ventana de Digilent MemUtil y se prosigue a la configuración del FPGA

a) **b)**

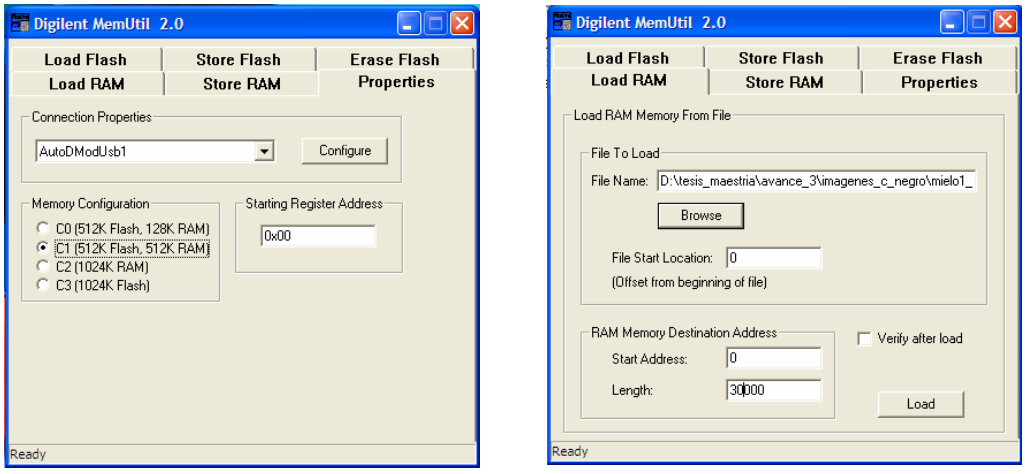


Figura A-3 Ventanas Digilent MemUtil a) Pestaña "Properties" b) Pestaña "Load RAM"

A.5 Configurando la funcionalidad del sistema en el FPGA

En esta etapa se configura el FPGA con el archivo de configuración del sistema obtenido en la sección 4.6.4. La configuración se lleva a cabo de la siguiente forma:

1. Primero se ponen en alto (Hi) los interruptores sw6 y sw7
2. Después se abre el programa ExPort localizado en *menú inicio-> Todos los programas-> Digilent->Adept->ExPort.*
3. Posteriormente se elije *Initialize Chain* y asigna el archivo de configuración del sistema.
4. Una vez que se ha asignado el archivo de configuración, se programa el FPGA dando clic en *Program Chain*
5. Finalmente de ponen en bajo(Low) los interruptores sw6 y sw7

.

Apéndice B. Programas

B.1 Matlab 7.1

B.1.1 Generando el archivo binario de la imagen

```
%&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
% Este programa genera el archivo binario de una imagen
% Autor: Gloria Castro
% Lugar:INAOE
% fecha: 9/01/07
%&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
clc
% clear al
Ancho =93;
largo=77;
T_ima= ancho*largo;
T_zeros=10000-T_ima;
% Plano de color rojo de la imagen
A=A2{3}(:,:,1); % define el plano de color a procesar 1=R 2=G 3=B
bw = A'; % transpone la matriz debido a que en el archivo la matriz se almacena conforme a las columnas
figure, imshow(bw) % desplega la imagen contenida en la matriz bw
fid = fopen('figura3.bin', 'wb'); % asigna un identificador al archivo que se va a crear
fwrite(fid,bw,'uint8') % escribe la matriz bw al archivo cuyo identificador es fid donde c/e de la matriz
status = fclose('all') % es almacenado como un entero sin signo de 8 bits
B=zeros(1,T_zeros);
fidB = fopen('figura3.bin', 'a'); % asigna un identificador al archivo que se va a crear
fwrite(fidB,B,'uint8') % escribe la matriz bw al archivo cuyo identificador es fid donde c/e de la matriz
status = fclose('all')
% Plano de color verde de la imagen
A=A2{3}(:,:,2); % define el plano de color a procesar 1=R 2=G 3=B
bw = A'; % transpone la matriz debido a que en el archivo la matriz se almacena conforme a las columnas
figure, imshow(bw) % desplega la imagen contenida en la matriz bw
fid = fopen('figura3.bin', 'a'); % asigna un identificador al archivo que se va a crear
fwrite(fid,bw,'uint8') % escribe la matriz bw al archivo cuyo identificador es fid donde c/e de la matriz
status = fclose('all') % es almacenado como un entero sin signo de 8 bits
%B=zeros(1,T_zeros);
fidB = fopen('figura3.bin', 'a'); % asigna un identificador al archivo que se va a crear
fwrite(fidB,B,'uint8') % escribe la matriz bw al archivo cuyo identificador es fid donde c/e de la matriz
status = fclose('all')
% Plano de color azul de la imagen
A=A2{3}(:,:,3); % define el plano de color a procesar 1=R 2=G 3=B
bw = A'; % transpone la matriz debido a que en el archivo la matriz se almacena conforme a las columnas
figure, imshow(bw) % desplega la imagen contenida en la matriz bw
fid = fopen('figura3.bin', 'a'); % asigna un identificador al archivo que se va a crear
fwrite(fid,bw,'uint8') % escribe la matriz bw al archivo cuyo identificador es fid donde c/e de la matriz
status = fclose('all') % es almacenado como un entero sin signo de 8 bits
%B=zeros(1,T_zeros);
fidB = fopen('figura3.bin', 'a'); % asigna un identificador al archivo que se va a crear
fwrite(fidB,B,'uint8') % escribe la matriz bw al archivo cuyo identificador es fid donde c/e de la matriz
status = fclose('all')
```

B.1.2 Programa del diseño Software

```
%>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
```

```
% Resumen : Este programa obtiene la proporción Núcleo/Citoplasma de c/u de los leucocitos de la imagen de un frotis sanguíneo
% Autor : Gloria castro Muñoz
% Tesis : Maestría
% Fecha : INAOE 7/05/2007
%>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
clc;
clear all;
%-----------------------------------------------------------------------
% Almacenamiento de la imagen RGB del frotis sanguíneo
%-----------------------------------------------------------------------
% Abre el archivo de la imagen del frotis sanguíneo a procesar
I= imread ('D:\tesis_maestria\ avance_2\ reporte_av_2\ detec_cell_Matlab\cromatina_640x480.jpg');
% figure,imshow(I) % muestra la imagen contenida en I
%-----------------------------------------------------------------------
% Detección de la región de los núcleos de los leucocitos
%-----------------------------------------------------------------------
A=I(:,:,2);              % Obtiene el plano de color verde a procesar 2=Green
%figure,imhist(A) % muestra el histograma de la imagen A
his = imhist(A); % Obtiene histograma de A
% Determina los mínimos de la gráfica del histograma en el rango [0,100] y calcula el umbral P
for i=7:1:100 % los mínimos se determinan usando una vecindad de 6
if his(i)<his(i-1) & his(i)<his(i-2) & his(i)<his(i-3) & his(i)<his(i-4) & his(i)<his(i-5) & his(i)<his(i-6) & his(i)<his(i+1) & his(i)<his(i+2) & his(i)<his(i+3) & his(i)<his(i+4) & his(i)<his(i+5) & his(i)<his(i+6)
umbral=i;   % umbral más cercano a 100
end
end
bw=filtro_histograma(A,umbral); % obtiene la imagen resultante en el rango [0 - P]
%figure,imshow(bw) % Imagen que contiene la región de los núcleos
e = llena_huecos(bw); % Llena los huecos de los núcleos
figure, imshow(e); % Muestra la imagen sin huecos
%-----------------------------------------------------------------------
% Separación y almacenamiento cada leucocito en una nueva imagen
%-----------------------------------------------------------------------
i=1;
for y = 1:1:639          % y = columna
   for x = 1:479         % x = renglon
      if e(x,y)==1
% determina la posición y dimensión del rectángulo más pequeño que encierra al núcleo del leucocito
[lmax,lmin]=rectángulo(x,y,e);
if (lmax(1)-lmin(1)) > 40
% Extrae de la imagen RGB la imagen encerrada por el rectángulo calculado
A2{i}=recortaRGB(I,lmax,lmin);
%figure, imshow(A2{i}); % muestra la imagen encerrada x el rectánguo
% Extrae de la imagen binaria la imagen encerrada por el rectángulo calculado
An{i}=uint8(~(recorta(e,lmax,lmin)));% se paso a uint8 para que A2 y An fueran compatibles
%figure,imshow(An{i}); % se observan negras x q el rango se amplia de 0-1 a 0-255
A2{i}(:,:,1)= An{i} .* A2{i}(:,:,1); % imagen RGB del leucocito con núcleo negro plano rojo
A2{i}(:,:,2)= An{i} .* A2{i}(:,:,2); % imagen RGB del leucocito con núcleo negro plano verde
A2{i}(:,:,3)= An{i} .* A2{i}(:,:,3); % imagen RGB del leucocito con núcleo negro plano azul
%figure,imshow(A2{i}); % muestra la imagen RGB del leucocito con núcleo negro
i=i+1; % obtiene siguiente leucocito
end
e=borra(e,lmax,lmin); % borra el núcleo extraído de la imagen binaria
%figure,imshow(e)     % muestra secuencia de imágenes de como se borran c/u de los núcleos
end
end
end
%-----------------------------------------------------------------------
% Segmentación por cambio de color y aproximación del leucocito a través de elipses
%-----------------------------------------------------------------------
% inicia parte de segmentación de color
f=size(A2);
f=f(2);                  % determina cuantos leucocitos se extrajeron de la imagen del frotis
```

```
for j=1:1:f
Q=A2{j};                % carga un leucocito extraído a la vez
[rmax,cmax,x]=size(Q);  % calcula el tamaño de la imagen que contiene al leucocito
cr=rmax/2;              % coordenada y del centro de la imagen
cc=cmax/2;              % coordenada x del centro de la imagen
% si las coordenadas del centro son flotantes, las convierte a enteros de 8 bits
if (isfloat(cr))
cr=int8(cr);
end
if (isfloat(cc))
cc=int8(cc);
end
Qs=s_color(Q,cc,cr); % Reduce la imagen usando segmentación por color
A3{j}=Qs;   % Crea un arreglo con las imágenes reducidas
figure,imshow(A3{j});   % Muestra la secuencia de imágenes creada
end
% Calcula el área del núcleo
for g=1:1:f  % número total de leucocitos extraídos
V=A3{g}(:,:,1); % carga leucocito
%figure,imshow(V)  % muestra leucocito cargado
ta=size(V);  % obtiene el tamaño del leucocito
area_n(g)=0;  % inicializa área del núcleo a cero
for a=1:1:ta(1)
for a2=1:1:ta(2)
if (V(a,a2)==0) % cuenta los píxeles negros para
area_n(g)=area_n(g)+1; % calcular área del núcleo
end
end
end
end
% Inicia aproximación por elipse
for t=1:1:f          % número total de leucocitos extraídos
Q=A3{t};         % carga leucocito
Q2=A3{t};
[k1,h1,x]=size(Q);  % obtiene el tamaño de la imagen del leucocito
area_figura(t)=k1*h1; % calcula el área total de la figura en píxeles
% obtiene coordenadas del centro de la imagen
h=(h1/2);
k=(k1/2);
if (isfloat(h)) % si la coordenada x del centroide es flotante
h=int8(h);  % la convierte a entera de 8 bits
h=double(h);% la pasa a double x q raíz no acepta int8 en matlab
end
if (isfloat(k)) % igual para la coordenada y del centroide
k=int16(k);
k=double(k);
end
a=h-1;        % eje < elipse
b=k-1;        % eje > elipse
area_fondo(t)=0;
% cálculo de la coordenada "y" despejando de la ec. de la elipse
% y=(+-b*raiz(1-((x-h)^2/a^2)))+k
for x=1:1:h1
wa=a^2-((x-h)^2);
ws=int16(sqrt(wa));
wo=b*real(ws);
we=wo/a;
y =(we+k);          % parte de abajo de la elipse
y2 =(-we+k);        % parte de arriva
% El área del fondo se obtiene sumando los píxeles que están fuera de la elipse
area_fondo(t)= area_fondo(t)+(k1-y)+(y2-1);
if (isfloat(y))     % si los puntos de la elipse son flotantes
y=int16(y)      % los pasa a enteros
end
if (isfloat(y2))    % si los puntos de la elipse son flotantes
y2=int16(y2)    % los pasa a enteros
end
for i=1:1:y2-1      % todo lo que este fuera de la elipse no incluyendo los puntos de la elipse
Q(i,x,1)=255;  % es puesto a color blanco
Q(i,x,2)=255;
Q(i,x,3)=255;
end
Q2(y2,x,1)=0;  % es puesto a color blanco
Q2(y2,x,2)=255;
Q2(y2,x,3)=255;
for j=y+1:1:k1       % todo lo que este fuera de la elipse incluir los puntos de la elipse
Q(j,x,1)=255;  % es puesto a blanco
Q(j,x,2)=255;
Q(j,x,3)=255;
```

```
end
Q2(y,x,1)=0;  % es puesto a blanco
Q2(y,x,2)=255;
Q2(y,x,3)=255;
end
A4{t}=Q; % almacena c/u de los leucocitos en un arreglo
de imágenes
A5{t}=Q2;
area_elipse(t)=area_figura(t)-area_fondo(t);
%figure, imshow(A4{t})    % muestra las imágenes
%figure,imshow(A5{t})    % muestra las imágenes
end
%--------------------------------------------------------------------
% Cálculo de la proporción N/C del leucocito
%--------------------------------------------------------------------
for t=1:1:f
area_citoplasma(t)=area_elipse(t)-area_n(t);
N_C(t)= area_n(t)/area_citoplasma(t);
end
```

B.1.2.1 Función Filtro_histograma

```
function [A] = filtro_histograma(A,umbral)
% Esta función convierte una imagen en escala de grises
a % binaria a través de un umbral de decisión
for i = 1:480
        for j = 1:640
           if (A(i,j)> umbral)
              A(i,j)=logical(0); %negro
           else
              A(i,j)=logical(1); %blanco
           end
        end
end
```

B.1.2.2 Función llena_huecos

```
function [H]= llena_huecos(bw)
 se = [0 0 1 1 1 1 1 0 0;
     0 1 1 1 1 1 1 1 0;
     1 1 1 1 1 1 1 1 1;
     1 1 1 1 1 1 1 1 1;
     1 1 1 1 1 1 1 1 1;
     1 1 1 1 1 1 1 1 1;
     1 1 1 1 1 1 1 1 1;
     0 1 1 1 1 1 1 1 0;
     0 0 1 1 1 1 1 0 0];
se= logical(se);
for i=5:1:480
   for j=5:1:640
      if (i<475 & j<635)
         I = subima(i,j,bw);
         bw2(i,j)=dilata(I,se);
       else
         bw2(i,j)=0;
       end
   end
end
bw2=logical(bw2);
%figure,imshow(bw2)
 for i2=5:1:480
   for j2=5:1:640
      if (i2<470 & j2<630)
         I = subima(i2,j2,bw2);
         H(i2,j2)=erociona(I,se);
      else
         H(i2,j2)=0;
      end
   end
end
H=logical(H);
%figure,imshow(H)
```

Función sub_ima

```
function [I]= subima(r,c,bw)
ro=r-5;
co=c-5;
for i=1:1:9
   for j=1:1:9
      I(i,j)=bw(ro+i,co+j);
   end
end
```

Función dilata

```
function [x]= dilata(I,se)
 x=0;
   for r=1:1:9
      for c=1:1:9
         %if ((I(r,c) & se(r,c))==1)
         if ((I(r,c) & se(r,c))==1)
            x=1;
            c=9;
            r=9;
```

```
        end
      end
    end
```

Función erosiona

```
function [x]= erociona(I,se)
 x=1;
   for r=1:1:9
      for c=1:1:9
         %if ((I(r,c) & se(r,c))==0)
         if (I(r,c)==0 & se(r,c)==1)
            x=0;
            c=9;
            r=9;
         end
      end
    end
```

Función rectángulo

```
function [bmax,bmin] = rectángulo(xo,yo,e)
B(1,1)=xo;  %renglón del punto inicial
B(1,2)=yo;  %columna del punto inicial
 m=1;
 p=1;
 paro=0;
while (paro == 0)
 p=m+1;
 [B(p,1),B(p,2)] = vecino(B(m,1),B(m,2),e,B,m);
   if (B(p,1)==B(1,1) & B(p,2)==B(1,2))| (B(p,1)==B(m,1)
& B(p,2)==B(m,2))
      paro=1;
   end
 m=m+1;
end
%-------------------------------------------------
% limites del rectángulo que encierra el núcleo
%-------------------------------------------------
bmax= max (B);
bmin= min (B);
```

Función vecino

```
function [p,q] = vecino(x,y,e,B,m)
%x renglón
%y columna
%[8 1 2]
%[2 * 3]
%[6 5 4]
if (e(x-1,y)==1 & (vecino_4(x-1,y,e))& (visitado(x-
1,y,B,m)))         % vecino 1
   p=x-1;
   q=y;
elseif (e(x-1,y+1)==1 & vecino_4(x-1,y+1,e)& (visitado(x-
1,y+1,B,m)))  % vecino 2
   p=x-1;
   q=y+1;
elseif (e(x,y+1)==1 & vecino_4(x,y+1,e)&
(visitado(x,y+1,B,m)))   % vecino 3
   p=x;
   q=y+1;
elseif (e(x+1,y+1)==1 & vecino_4(x+1,y+1,e)&
(visitado(x+1,y+1,B,m)))  % vecino 4
   p=x+1;
   q=y+1;
elseif (e(x+1,y)==1 & vecino_4(x+1,y,e)&
(visitado(x+1,y,B,m)))  % vecino 5
   p=x+1;
   q=y;
elseif (e(x+1,y-1)==1 & vecino_4(x+1,y-1,e)&
(visitado(x+1,y-1,B,m)))  % vecino 6
   p=x+1;
   q=y-1;
elseif (e(x,y-1)==1  & vecino_4(x,y-1,e)& (visitado(x,y-
1,B,m)))  % vecino 7
   p=x;
   q=y-1;
elseif (e(x-1,y-1)==1 & vecino_4(x-1,y-1,e)& (visitado(x-
1,y-1,B,m)))  % vecino 8
   p=x-1;
   q=y-1;
else
   p=x;
   q=y;
end
```

Función vecino_4

```
function [a] = vecino_4(x,y,e)
%x renglón
%y columna
%[  1  ]
%[4 * 2]
%[  3  ]
if e(x-1,y)==0 & e(x,y+1)==0 & e(x+1,y)==0 & e(x,y-1)==0
   a=0;
elseif e(x-1,y)==0 & e(x+1,y)==0 & e(x,y+1)==1 & e(x,y-
1)==1  & e(x-1,y+1)==0   & e(x+1,y+1)==0  & e(x+1,y-
1)==0 & e(x-1,y-1)==0 % vecinos 1 y 3
   a=0;
```

```
elseif e(x,y+1)==0 & e(x,y-1)==0 & e(x-1,y)==1 & e(x+1,y)==1 & e(x-1,y+1)==0 & e(x+1,y+1)==0 & e(x+1,y-1)==0 & e(x-1,y-1)==0 % vecinos 2 y 4
   a=0;
elseif e(x-1,y)==0     % vecino 1
   a=1;
elseif e(x,y+1)==0     % vecino 2
   a=1;
elseif e(x+1,y)==0     % vecino 3
   a=1;
elseif e(x,y-1)==0     % vecino 4
   a=1;
else
   a=0;
end
```

Función visitado

```
function [a] = visitado(x,y,B,m)
band=0;
for i=1:1:m-1
   if x==B(i,1) & y==B(i,2)
      band=1;
   end
end

if x==B(1,1) & y==B(1,2)
   a=1;
elseif band==1
   a=0;
else
   a=1;
end
```

B.1.2.3 Función recorta

```
Function [s_ima] = recorta(ima,lmax,lmin)

% Esta función borra el contenido del rectángulo definido
x los limites
rmin=lmin(1)-14;
cmin=lmin(2)-14;
rmax=lmax(1)+14;
cmax=lmax(2)+14;
 x=1;
for i=rmin:1:rmax
   y=1;
   for j=cmin:1:cmax
      s_ima(x,y)=ima(i,j);
      y=y+1;
   end
   x=x+1;
end
```

B.1.2.4 Función recortaRGB

```
Function [RGB] = recortaRGB(ima,lmax,lmin)
% Esta función borra el contenido del rectángulo definido
x los limites
rmin=lmin(1)-14;
cmin=lmin(2)-14;
rmax=lmax(1)+14;
cmax=lmax(2)+14;
 x=1;
for i=rmin:1:rmax
   y=1;
   for j=cmin:1:cmax
      RGB(x,y,1)=ima(i,j,1);
      RGB(x,y,2)=ima(i,j,2);
      RGB(x,y,3)=ima(i,j,3);
      y=y+1;
   end
   x=x+1;
end
```

B.1.2.5 Función recortaRGBf

```
function [RGB] = recortaRGBf(ima,lmax,lmin)

% Esta función borra el contenido del rectángulo definido
x los limites
rmin=lmin(1)+2;
cmin=lmin(2)+2;
rmax=lmax(1)-2;
cmax=lmax(2)-2;
 alto = rmax-rmin+1;
base = cmax-cmin+1;
 if mod(alto,2)== 0
      rmax = rmax - 1;
   end
if mod(base,2)== 0
      cmax = cmax - 1;
end
```

```
 x=1;
for i=rmin:1:rmax
   y=1;
   for j=cmin:1:cmax
      RGB(x,y,1)=ima(i,j,1);
      RGB(x,y,2)=ima(i,j,2);
      RGB(x,y,3)=ima(i,j,3);
      y=y+1;
   end
   x=x+1;
end
```

B.1.2.6 Función borra

```
function [ima] = borra(ima,lmax,lmin)

% Esta función borra el contenido del rectángulo definido
x los limites
rmin=lmin(1);
cmin=lmin(2);
rmax=lmax(1);
cmax=lmax(2);
for i=rmin:1:rmax
   for j=cmin:1:cmax
      ima(i,j)=0;
   end
end
```

B.1.2.7 Función s_color

```
function [s_ima] = s_color(Q,cc,cr)
% El objetivo de esta función es hacer una mejor
aproximación del leucocito, reduciendo la imagen al
tamaño justo del leucocito contenido
b1=0; % bandera que indica si se ha encontrado el punto
de segmentación superior
b2=0; % bandera que indica si se ha encontrado el punto
de segmentación izquierdo
b3=0; % bandera que indica si se ha encontrado el punto
de segmentación inferior
b4=0; % bandera que indica si se ha encontrado el punto
de segmentación derecho
[rmax,cmax,x]=size(Q); % Obtiene el tamaño de la
imagen de entrada
lmin(1)=0;          % se usan en el caso de no encontrar
punto de segmentación
lmin(2)=0;          % se usan en el caso de no encontrar
punto de segmentación
lmax(1)=rmax;       % se usan en el caso de no
encontrar punto de segmentación
lmax(2)=cmax;       % se usan en el caso de no
encontrar punto de segmentación
% Un punto de segmentación se da cuando se
encuentra un glóbulo rojo o bien el color del fondo
% Un glóbulo rojo tiene la siguiente combinación de
colores: 203<rojo<222 124<verde<172 171<azul<207
% El fondo tiene un color parecido al blanco por lo que
rojo=verde=azul>243
% Recordar que en matlab Q(p,cc) p=renglón
cc=columna a diferencia de los pares coordenados (x,y)
x=columna y=renglón
for p=14:-1:1
%límite superior
if ((b1==0) & (((Q(p,cc,1)>203 & Q(p,cc,1)<222) &
(Q(p,cc,2)>124 & Q(p,cc,2)<172) & (Q(p,cc,3)>171 &
Q(p,cc,3)<207))|(Q(p,cc,1)>243 & Q(p,cc,2)>243 &
Q(p,cc,3)>243)))
lmin(1)=p; % Si encuentra un GR o el fondo de la imagen
entonces almacena punto de segmentación superior
b1=1;
end
%límite izquierdo
if b2==0 & (((Q(cr,p,1)>203 & Q(cr,p,1)<222) &
(Q(cr,p,2)>124 & Q(cr,p,2)<172) & (Q(cr,p,3)>171 &
Q(cr,p,3)<207))|(Q(cr,p,1)>243 & Q(cr,p,2)>243 &
Q(cr,p,3)>243))
lmin(2)=p;% Si encuentra un GR o el fondo de la imagen
entonces almacena punto de segmentación izquierdo
b2=1;
end
% límite inferior
p2=rmax-p;
if b3==0 & (((Q(p2,cc,1)>203 & Q(p2,cc,1)<222) &
(Q(p2,cc,2)>124 & Q(p2,cc,2)<172) & (Q(p2,cc,3)>171 &
Q(p2,cc,3)<207))|(Q(p2,cc,1)>243 & Q(p2,cc,2)>243 &
Q(p2,cc,3)>243))
lmax(1)=p2; % Si encuentra un GR o el fondo de la
imagen entonces almacena punto de segmentación
inferior
b3=1;
end
```

```
%límite derecho
p3=cmax-p;
if b4==0 & (((Q(cr,p3,1)>203 & Q(cr,p3,1)<222) & (Q(cr,p3,2)>124 & Q(cr,p3,2)<172) & (Q(cr,p3,3)>171 & Q(cr,p3,3)<207))|(Q(cr,p3,1)>243 & Q(cr,p3,2)>243 & Q(cr,p3,3)>243))
lmax(2)=p3; % Si encuentra un GR o el fondo de la imagen entonces almacena punto de segmentación derecho
b4=1;
end
end
 s_ima=recortaRGBf(Q,lmax,lmin);
```

B.2 ISE 8.2i

```
----------------------------------------------------------------------------
-- Company:          INAOE
-- Engineer:         Gloria Castro
-- Create Date:      26/07/07
-- Module Name:      Modulo Top
-- FPGA:             Spartan 3 xc3s200-4ft256
----------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
use ieee.numeric_std.all;
use work.PKG.all; -- definición de las funciones raíz y división
--
entity Top is
Port
(reset : in  std_logic;-- reset gral del FPGA
clk    : in  std_logic;-- reloj de 50MHz
sw0,sw1: in std_logic; --sw0 habilita displays, sw1 habilita vga
data: inout std_logic_vector (7 downto 0);-- datos de la Ram
addr: out std_logic_vector (18 downto 0);-- dirección de la Ram
oel,wel,cel : out std_logic;  -- pines de ctrl de la Ram
lim : in std_logic_vector(2 downto 0);-- seleccionadores para ver los 4 límites calculados x lim_cell
area:out std_logic_vector(13 downto 0);-- visualiza las areas  en los leds
dig: out std_logic_vector(6 downto 0);-- 7 segmentos del display
AN: out std_logic_vector(3 downto 0); -- control de los ánodos del display
punto: out std_logic;-- punto decimal del display
hsyncb : buffer  std_logic;-- señal de sincronía horizontal del VGA
vsyncb : out  std_logic;-- señal de sincronía vertical del VGA
 rgb         : out  std_logic_vector (23 downto 0));
-- señal RGB del VGA
end Top;

architecture Behavioral of Top is
component sub_top is
 Port (
reset: in  std_logic;-- reset gral del FPGA
clk: in  std_logic;-- reloj de 50MHz
sw0,sw1: in std_logic;-- sw0 habilita displays,sw1 habilita vga
data: inout std_logic_vector (7 downto 0);--  datos de la Ram
addr: out std_logic_vector (18 downto 0);-- dirección de la Ram
oel,wel,cel : out std_logic;-- pines de ctrl de la Ram
lim : in std_logic_vector(2 downto 0);-- seleccionadores para ver los 4 límites calculados x lim_cell
area:out std_logic_vector(13 downto 0);-- visualiza las areas en los leds
dig: out std_logic_vector(6 downto 0);-- 7 segmentos del display
AN: out std_logic_vector(3 downto 0);-- control de los ánodos del display
punto: out std_logic;--punto decimal del display
hsyncb:buffer  std_logic;-- señal de sincronía horizontal del VGA
vsyncb:out  std_logic;-- señal de sincronía vertical del VGA
rgb: out  std_logic_vector (23 downto 0));-- señal RGB del VGA
end component;
----------señales----------
signal clock:std_logic;
-----------------------------------------------------------
begin
U5:sub_top portmap
```

```
(reset, clock, sw0, sw1, data, addr, oel, wel, cel, lim, area,
dig, AN, punto, hsyncb, vsyncb, rgb);
clk_25Mhz: process(reset,clk)
begin
if reset='1' then
clock<='1';
elsif clk 'event and clk='1' then
clock <= not clock;
end if;
end process;
end Behavioral;
```

B.2.1 Módulo SubTop

```
----------------------------------------------------------------------------
-- Company:                     INAOE
-- Engineer:                    Gloria Castro
-- Create Date:       26/07/07
--Module Name: SubTop: incluye Módulos calculo_elipse,
decoder, encuentra limites, recorta imagen y VGA
-- FPGA: Spartan 3 xc3s200-4ft256
----------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
use ieee.numeric_std.all;
use work.PKG.all;
entity sub_top is
Port (
reset       : in  std_logic;-- reset gral del FPGA
clk         : in  std_logic;-- reloj de 50MHz
sw0,sw1: in std_logic;-- sw0 habilita displays,sw1 habilita
vga
data        : inout std_logic_vector (7 downto 0);--  datos
de la Ram
addr        :  out  std_logic_vector  (18  downto  0);--
dirección de la Ram
oel,wel,cel : out std_logic;-- pines de ctrl de la Ram
lim : in std_logic_vector(2 downto 0);-- seleccionadores
para ver los 4 límites calculados x lim_cell
area:out  std_logic_vector(13  downto  0);--  visualiza  las
áreas en los leds
dig: out std_logic_vector(6 downto 0);-- 7 segmentos del
display
AN: out std_logic_vector(3 downto 0);-- control de los
ánodos del display
punto: out std_logic;-- punto decimal del display
hsyncb : buffer  std_logic;-- señal de sincronía horizontal
del VGA
vsyncb : out  std_logic;-- señal de sincronía vertical del
VGA
rgb         : out  std_logic_vector (23 downto 0));-- señal
RGB del VGA
end sub_top;

architecture Behavioral of sub_top is
component dis_ima is
Port         (
reset       : in  std_logic;-- reset gral del FPGA
clk: in  std_logic;-- reloj de 25MHz
 data       : inout std_logic_vector (7 downto 0);--  datos
de la Ram
addr        :  out  std_logic_vector  (18  downto  0);--
dirección de la Ram
wel : out std_logic;-- pines de ctrl de la Ram
area:out std_logic_vector(13 downto 0);-- área del núcleo
del GB
stop:out  std_logic;--  señal  de  finalización  del  módulo
dis_ima
ii:out std_logic;-- señal de control de addr
hrec_red:out  std_logic_vector(6  downto  0);--  alto  de  la
imagen reducida
brec_red: out std_logic_vector(6 downto 0);-- ancho de la
imagen reducida
píxel: out std_logic_vector(7 downto 0));-- píxeles de la
imagen reducida
end component;
component puntos_elipse is
Port(
reset: in  std_logic;-- reset del módulo puntos elipse
clk: in  std_logic;-- reloj de 25MHz
wel:out std_logic;-- señal de escritura de la RAM
b_rec:  in  std_logic_vector(6  downto  0);--  ancho  de  la
imagen reducida
h_rec:  in  std_logic_vector(6  downto  0);--  alto  de  la
imagen reducida
addr: out std_logic_vector(18 downto 0);-- dirección de la
RAM
area_fondo:  out  std_logic_vector(11  downto  0);--  area
del background de la imagen
fin:out std_logic-- señal de finalización del módulo puntos
elipse);
end component;
```

```
component decoder is
Port (
reset       : in  std_logic;-- reset del módulo decoder
clk         : in  std_logic;-- reloj de 25Mhz
A: in std_logic_vector(3 downto 0);-- dígito 1 a mostrar
B: in std_logic_vector(3 downto 0);-- dígito 2 a mostrar
dig: out std_logic_vector(6 downto 0);-- 7 segmentos del display
AN: out std_logic_vector(3 downto 0);-- ctrl de ánodos del display
punto: out std_logic);-- punto decimal del display
end component;
component vga is
Port (
reset       : in  std_logic;-- reset del módulo VGA
clock       : in  std_logic;-- reloj d 25MHz
hsyncb : buffer  STD_LOGIC;-- señal del sincronía horizontal
vsyncb : out  std_logic;-- señal de sincronía vertical
rgb         : out  std_logic_vector (23 downto 0);-- señal RGB del VGA
data        : inout std_logic_vector (7 downto 0);-- datos de la RAM
addr        : out std_logic_vector (18 downto 0);-- dirección de la RAM
oel : out std_logic;-- señal de lectura de la RAM
alto: std_logic_vector(6 downto 0);-- largo de la imagen a mostrar
ancho: std_logic_vector(6 downto 0));-- ancho de la imagen a mostrar
end component;
---------- señales ----------
Signal  addr_top4,addr_py,  addr_vga  :  std_logic_vector (18   downto 0 );
signal wel_top4,wel_py,oel_vga:std_logic;
signal stop:std_logic;
signal ii:std_logic;
signal hrec_red, h_ima:std_logic_vector(6 downto 0);
signal brec_red, b_ima:std_logic_vector(6 downto 0);
signal pixel: std_logic_vector(7 downto 0);
signal area_N,area_c: std_logic_vector(13 downto 0);
signal area_fondo: std_logic_vector(11 downto 0);
signal  RD_REG2,cociente2: std_logic_vector(28 downto 0);
signal cociente1: std_logic_vector (14 downto 0);
signal residuo1: std_logic_vector(13 downto 0);
signal fin:std_logic;
begin
--------------------------------------------------------------------------
--                    Connexion de componentes
--------------------------------------------------------------------------
U2: dis_ima port map (reset, clk, data, addr_top4, wel_top4, area_N, stop, ii, hrec_red, brec_red, pixel);
U3:puntos_elipse port map (stop, clk, wel_py, brec_red, hrec_red, addr_py, area_fondo, fin);
U4:decoder port map (sw0, clk, cociente1 (3 downto 0), cociente2 (17 downto 14), dig, AN, punto);
U7:vga port map (sw1, clk, hsyncb, vsyncb, rgb, data, addr_vga, oel_vga, hrec_red, brec_red);
-- asegura que la imagen sea de un tamaño impar
b_ima <=
brec_red - 1 when brec_red(0)='0' else
brec_red;
h_ima <=
hrec_red - 1 when hrec_red(0)='0' else
hrec_red;
-- Calculo del area del citoplasma
area_c <= (b_ima * h_ima)-("00" & area_fondo)-area_N;
-- División para obtener el primer dígito de la razón area núcleo/area citoplasma
RD_REG2 <=DIVISION("00000000000000" & area_N, area_c);
cociente1 <= RD_REG2(28 downto 14);
residuo1 <= RD_REG2(13 downto 0);
-- División para obtener el segundo dígito de la razón N/C (primera décima)
cociente2 <= DIVISION("0000000000" & residuo1 * "1010", area_c);
-- Multiplexor para visualización de datos en los leds
area <=
area_N when lim="000" else
area_c when lim="001" else
"00" & area_fondo when lim="010" else
cociente1(13 downto 0) when lim="011" else
cociente2(27 downto 14);
--Multiplexor de la señal de habilitación de escritura de la RAM
wel<=
wel_top4 when stop='1' else
wel_py;
-- Multiplexor que determina la dirección de la RAM
addr<=
addr_top4 when stop='1' else
addr_vga  when sw1='0' else
```

```
addr_py;
-- Multiplexor para los datos de la RAM
data<=
pixel when (stop='1' and ii='1') else
"00000000" when (stop='0' and sw1='1') else
"ZZZZZZZZ";
-- La memoria RAM siempre está habilitada (cel=0)
cel<='0';
-- Señal de lectura de la memoria de video
oel<= '0' when sw1='1' else
oel_vga;
end Behavioral;
```

B.2.2 Módulo Dis_ima

```
----------------------------------------------------------------------------
-- Company: INAOE
-- Engineer: Gloria Castro
-- Create Date: 26/07/07
-- Module Name: unión de los módulos: calcula límites y reduce imagen
-- FPGA:  Spartan 3 xc3s200-4ft256
----------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
--
entity dis_ima is
generic(
h_rec: std_logic_vector(6 downto 0):= "1001101";--altura imagen=#1=88,#2=94 #3=77
b_rec: std_logic_vector(6 downto 0):= "1011101";--base imagen #1=88 #2=82 #3=93
rmax:std_logic_vector(6 downto 0):= "1001100";--renglón max de la imagen #1=87 #2=93 #3=76
cmax:std_logic_vector(6 downto 0):= "1011100"; --columna max de la imagen #1=87 #2=81 #=92
cc:std_logic_vector(5 downto 0):= "101101"; --columna del centro de la imagen #1=43 #2=40 #3=45
rc:std_logic_vector(5 downto 0):= "100101" --renglón del centro de laimagen #1=43 #2=46 #3=37);
 Port (
reset: in  std_logic;-- reset gral del FPGA
clk : in  std_logic;-- reloj de 50MHz
data: inout std_logic_vector (7 downto 0);--  datos de la Ram
addr: out std_logic_vector (18 downto 0);-- dirección de la Ram
wel : out std_logic;-- pines de ctrl de la Ram
area:out std_logic_vector(13 downto 0);-- area del núcleo
stop:out std_logic;-- señal de finalización del módulo dis_ima
ii:out std_logic;-- ctrl de las direcciones de la RAM
hrec_red:out std_logic_vector(6 downto 0);-- altura de la imagen reducida
brec_red: out std_logic_vector(6 downto 0);-- ancho de la imagen reducida
pixel: out std_logic_vector(7 downto 0));-- datos de los píxeles de la imagen reducida
end dis_ima;

architecture Behavioral of dis_ima is
component lim_cell is
Port (
reset       : in std_logic; --reset gral FPGA
clock: in std_logic; -- reloj de 25 MHz
data       : inout std_logic_vector (7 downto 0);-- datos de/hacia la RAM
addr       : out std_logic_vector (18 downto 0);-- dirección de la RAM
izq : out std_logic_vector(3 downto 0);-- límite izquierdo imagen
sup : out std_logic_vector(3 downto 0);-- límite superior imagen
der : out std_logic_vector(3 downto 0);-- límite derecho imagen
abajo: out std_logic_vector(3 downto 0);-- límite inferior imagen
wel : out std_logic; --pin de habilitación de escritura Ram
f_lim : out std_logic; --señal de término del módulo lim_cell
b_rec: in std_logic_vector(6 downto 0);--largo de la imagen original
rmax: in std_logic_vector(6 downto 0);-- renglón máximo de la Ima original
cmax: in std_logic_vector(6 downto 0);-- columna máxima de la Ima original
cc: in std_logic_vector(5 downto 0);-- columna correspondiente al centro de la Ima
rc: in std_logic_vector(5 downto 0));-- renglón correspondiente al centro de la Ima
end component;
component f_corta is
```

```
Port (
reset      : in std_logic; -- rst particular del módulo f_corta
clock      : in std_logic; -- reloj de 25MHz
data       : inout std_logic_vector (7 downto 0);-- datos de la RAM
izq : in std_logic_vector(3 downto 0);-- límite izq imagen reducida
sup : in std_logic_vector(3 downto 0);-- límite sup imagen reducida
der : in std_logic_vector(3 downto 0);-- límite der imagen reducida
abajo: in std_logic_vector(3 downto 0);-- límite inferior imagen reducida
b_rec: in std_logic_vector(6 downto 0);-- largode la imagen original
cmax: in std_logic_vector(6 downto 0);-- máxima columna de la imagen original
rmax: in std_logic_vector(6 downto 0);-- máximo renglón de la imagen original
addr: out std_logic_vector (18 downto 0);-- dir de la Ram
wel : out std_logic;-- pin de escritura de la RAM
area: out std_logic_vector(13 downto 0);-- area del fondo
stop:out std_logic;-- señal de finalización de módulo
ii:out std_logic;---- ctrl de las direcciones de la RAM
pixel: out std_logic_vector(7 downto 0));-- datos de píxel de la imagen reducida
end component;
------- declaración de señales ----------------------------------
-
signal Nf_lim: std_logic; -- señal de reset para f_corta
signal wel_corta,wel_lim: std_logic;-- señales de control de escritura para la Ram
signal addr_corta,addr_lim: std_logic_vector(18 downto 0);-- señales de dirección para Ram
signal f_lim: std_logic;--señal de termino de lim_cell
signal izq,sup,der,abajo: std_logic_vector(3 downto 0);-- señales de los 4 limites genradas x lim_cell
--------------------------------------------------------------------------
begin
-- Mapeo de los puertos entre componentes
-- lim_cell calcula los límites del rectángulo + pequeño que contiene a la célula
U0: lim_cell port map (reset, clk, data, addr_lim, izq, sup, der, abajo, wel_lim, f_lim, b_rec, rmax, cmax, cc, rc);
-- f_corta disminuye el tamaño de la imagen original a el tamaño del rectángulo calculado
-- por lim_cell U1: f_corta port map (Nf_lim, clk, data, izq, sup, der, abajo, b_rec, cmax, rmax, addr_corta, wel_corta, area, stop, ii, pixel);
-- calculo del taaño de la imagen reducida
brec_red <= b_rec-izq-der-4;
hrec_red <= h_rec-sup-abajo-4;
-- Una vez que se tienen los 4 limites calculados por lim_cell se procede a disminuir
-- la imagen con f_corta. f_lim es invertido porque f_corta trabaja con reset en alto
Nf_lim<= not f_lim;
-- Multiplexor determina si se lee o escribe la memoria RAM
wel<=
wel_corta when f_lim='1' else
wel_lim;
-- Multiplexor que determina el origen de la dirección hacia la RAM
addr<= addr_corta when f_lim='1' else
addr_lim;
end Behavioral;
```

B.2.3 Módulo lim_cell

```
----------------------------------------------------------------------------
-- Company:  INAOE
-- Engineer:  Gloria Castro
-- Create Date: 26/07/07
-- Module Name: calculo de los limites del rectángulo + pequeño que envuelve al GB
-- FPGA:  Spartan 3 xc3s200-4ft256
-------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity lim_cell is
Port (
reset : in std_logic; --rst gral FPGA
clock: in std_logic; -- reloj de 25 MHz
data: inout std_logic_vector (7 downto 0);-- datos de/hacia la RAM
addr: out std_logic_vector (18 downto 0);-- dirección de la RAM
```

```
izq: out std_logic_vector(3 downto 0);-- límite izquierdo imagen
sup : out std_logic_vector(3 downto 0);-- límite superior imagen
der : out std_logic_vector(3 downto 0);-- límite derecho imagen
abajo: out std_logic_vector(3 downto 0);-- límite inferior imagen
wel : out std_logic; --pin de habilitación de escritura Ram
f_lim : out std_logic; --señal de termino del módulo lim_cell
b_rec: in std_logic_vector(6 downto 0);--largo de la imagen original
rmax: in std_logic_vector(6 downto 0);-- renglón máximo de la Ima original
cmax: in std_logic_vector(6 downto 0);-- columna máxima de la Imagen original
cc: in std_logic_vector(5 downto 0);-- columna correspondiente al centro de la Imagen
rc: in std_logic_vector(5 downto 0));-- renglón correspondiente al centro de la Imagen
end lim_cell;

architecture Behavioral of lim_cell is
-- declaración de señales
Signal addr_Rleft, addr_Rup, addr_Rder, addr_Rdown : std_logic_vector (18 downto 0);
signal addr_Gleft, addr_Gup, addr_Gder, addr_Gdown : std_logic_vector (18 downto 0);
signal addr_Bleft, addr_Bup, addr_Bder, addr_Bdown : std_logic_vector (18 downto 0);
signal i:std_logic_vector(3 downto 0):="0000";
signal cont: std_logic_vector (18 downto 0) := "00000000000000001101"; --13
signal pixr,pixg,pixb: std_logic_vector(7 downto 0);
signal f_izq,f_sup,f_der,f_down:std_logic;
begin
--- Este proceso es un contador de 15 ciclos de reloj debido a que necesitamos
-- 12 ciclos para leer los píxeles RGB de la parte izquierda,superor,dercha y baja de la imagen
-- 4 ciclos para decidir si se trata de un límite (izq,sup,der,baja) o no
clk2: process (reset,clock)
begin
if reset='1' then
i<="0000";
elsif  (clock 'event and clock = '1')  then
if (i<15)then
i<=i+1;
else i<="0000";
end if;
end if;
end process;
-- Este proceso genera un contador descendente de 13->0 que es la orgura q se le dio a la
-- imagen para encontrar el citoplasma. Cabe resaltar que el decremento es cada 16 ciclos de reloj
-- debido a que es el tiempo para leer los píxeles de los 4 limites y determinar si es limite o no
cont_13_0: process(clock,reset)
begin
if reset ='1' then
cont <= "00000000000000001101";  --  carga inicial 13
elsif (clock 'event and clock= '1') then
if(i=15 and cont>0) then
cont <= cont - 1;
end if;
end if;
end process;
-- Genera la dirección del lím izquierdo
adder1:    addr_Rleft <= rc*b_rec + cont;    --3784
adder2:    addr_Gleft <= rc*b_rec + cont + 10000;
adder3:    addr_Bleft <= rc*b_rec + cont + 20000;
-- Genera la dirección del lím superior
adder4:addr_Rup <= cont(11 downto 0) * b_rec + cc;
adder5:addr_Gup <= cont(11 downto 0) * b_rec + cc + 10000;
adder6:addr_Bup <= cont(11 downto 0) * b_rec + cc + 20000;
-- Genera la dirección del lím derecho
adder7:addr_Rder <= (cmax - cont) + rc*b_rec;
adder8:addr_Gder <= (cmax - cont) + rc*b_rec + 10000;
adder9:addr_Bder <= (cmax - cont) + rc*b_rec + 20000;
-- Genera la dirección del lím inferior
adder10:addr_Rdown <= (rmax-cont(11 downto 0))* b_rec + cc;
adder11:addr_Gdown <= (rmax-cont(11 downto 0))* b_rec + cc + 10000;
adder12:addr_Bdown <= (rmax-cont(11 downto 0))* b_rec + cc + 20000;
---Multiplexor que determina el origen de la dirección de la Ram
addr<=
```

```
            addr_Rleft when i="0000" else
            addr_Gleft when i="0001" else
            addr_Bleft when i="0010" else
            addr_Rup  when i="0100" else
            addr_Gup   when i="0101" else
            addr_Bup   when i="0110" else
            addr_Rder  when i="1000" else
            addr_Gder  when i="1001" else
            addr_Bder  when i="1010" else
            addr_Rdown when i="1100" else
            addr_Gdown when i="1101" else
            addr_Bdown when i="1110" else
            "XXXXXXXXXXXXXXXXXXX";
-- pin de control de escritura de la Ram
wel<='1';
-- Registro que almacena el píxel del plano rojo,el registro se actualiza cuando
-- se lee un píxel rojo de cualquiera de los 4 límites(izq,sup,der,abajo)
RegR: process (clock, reset)
begin
if reset ='1' then
pixr <= (others => '0');
elsif (clock 'event and clock = '1')then
if (i=0 or i=4 or i=8 or i=12) then
pixr <= data;
else
pixr <= pixr;
end if;
end if;
end process;
-- Registro que almacena el píxel del plano verde, el registro se actualiza cuando
-- se lee un píxel verde de cualquiera de los 4 límites(izq,sup,der,abajo)
RegG: process (clock, reset)
begin
if reset ='1' then
pixg <= (others => '0');
elsif (clock 'event and clock = '1')then
if (i=1 or i=5 or i=9 or i=13) then
pixg <= data;
else
pixg <= pixg;
end if;
end if;
end process;
-- Registro que almacena el píxel del plano azul,el registro se actualiza cuando
-- se lee un píxel azul de cualquiera de los 4 límites(izq,sup,der,abajo)
RegB: process (clock, reset)
begin
if reset ='1' then
pixb <= (others => '0');
elsif (clock 'event and clock = '1')then
if (i=2 or i=6 or i=10 or i=14) then
pixb <= data;
else
pixb <= pixb;
end if;
end if;
end process;
-- Este proceso determina si se ha encontrado uno de los bordes de la célula (izq,sup,der,abajo)
-- tomando en cuanta que un borde es cuando se encuentra un GR o el fondo de la imagen(blanco)
-- esta decisión toma lugar una vez que han leído y guardado los píxeles de los cuatro limites
det_lim: process (clock,reset,f_izq,f_sup,f_der,f_down)
begin
if reset ='1' then
f_izq<='0';
f_sup<='0';
f_der<='0';
f_down<='0';
izq<=(others=>'0');---inicializar en 1
sup<=(others=>'0');
der<="0001";
abajo<=(others=>'0');
elsif (clock 'event and clock = '1')then
if (i=3) then
if ((f_izq='0')and(((pixr>203 and pixr<222)and (pixg>124 and pixg<172)and (pixb>171 and pixb<207))or (pixr>243 and pixg>243 and pixb>243))) then
izq<=cont(3 downto 0);
f_izq<='1';
elsif cont=0 then
f_izq<='1';
end if;
elsif (i=7) then
if ((f_sup='0')and(((pixr>203 and pixr<222)and (pixg>124 and pixg<172)and (pixb>171 and pixb<207))or (pixr>243 and pixg>243 and pixb>243))) then
```

```
sup<=cont(3 downto 0);
f_sup<='1';
elsif cont=0 then
f_sup<='1';
end if;
elsif (i=11) then
if ((f_der='0')and(((pixr>203 and pixr<222)and (pixg>124 and pixg<172)and (pixb>171 and pixb<207))or (pixr>243 and pixg>243 and pixb>243))) then
der<=cont(3 downto 0);
f_der<='1';
elsif cont=0 then
f_der<='1';
end if;
elsif (i=15) then
if ((f_down='0')and(((pixr>203 and pixr<222)and (pixg>124 and pixg<172)and (pixb>171 and pixb<207))or (pixr>243 and pixg>243 and pixb>243))) then
abajo<=cont(3 downto 0);
f_down<='1';
elsif cont=0 then
f_down<='1';
end if;
end if;
end if;
f_lim <= f_izq and f_sup and f_der and f_down;
end process;
end Behavioral;
```

B.2.4 Módulo f_corta

```
----------------------------------------------------------------------------
-- Company: INAOE
-- Engineer: Gloria Castro
-- Create Date: 26/07/07
-- Module Name:   reducción de la imagen al tamaño del GB
-- FPGA:  Spartan 3 xc3s200-4ft256
----------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity f_corta is
Port (
reset: in  std_logic; -- rst particular del módulo f_corta
clock: in  std_logic; -- reloj de 25MHz
data: inout std_logic_vector (7 downto 0);-- datos de la RAM
izq:in std_logic_vector(3 downto 0);-- límite izq imagen reducida
sup : in std_logic_vector(3 downto 0);-- límite sup imagen reducida
der : in std_logic_vector(3 downto 0);-- límite der imagen reducida
abajo: in std_logic_vector(3 downto 0);-- límite inferior imagen reducida
b_rec: in std_logic_vector(6 downto 0);-- largode la imagen original
cmax: in std_logic_vector(6 downto 0);-- máxima columna de la imagen original
rmax: in std_logic_vector(6 downto 0);-- máximo renglón de la imagen original
addr: out std_logic_vector (18 downto 0);-- dir de la Ram
wel : out std_logic;--pin de escritura de la RAM
area: out std_logic_vector(13 downto 0);-- area del núcleo
stop:out std_logic;-- señal de finalización de modulo
ii:out std_logic;-- ctrl de las direcciones de la RAM
pixel: out std_logic_vector(7 downto 0));-- datos de la imagen reducida
end f_corta;

architecture Behavioral of f_corta is
-- declaración de señales
Signal cont2: std_logic_vector(18 downto 0): = "00000000000000000"; --13
signal cont: std_logic_vector(18 downto 0);
signal i:std_logic;
signal temp:std_logic_vector(6 downto 0);
signal pix:std_logic_vector(7 downto 0);
signal fin:std_logic_vector (13 downto 0);
signal ini_dos:std_logic_vector (13 downto 0);
signal l_dos,l_tres, ini_tres:std_logic_vector (14 downto 0);
signal ini: std_logic_vector (10 downto 0);
signal b_ima:std_logic_vector(6 downto 0);
signal area_N: std_logic_vector(13 downto 0);
begin
----- calculo de inicializacion de contadores y límites
ini <=((sup+2)*b_rec)+izq+"10";
```

```
ini_dos <= ((sup+2)*b_rec)+ izq + "10011100010010"; --10002
ini_tres <= ((sup+2)*b_rec)+ izq + "100111000100010"; --20002
fin <= ((rmax - abajo - 2)*b_rec)+(cmax-der-2);
l_dos <= ((rmax - abajo - 2)*b_rec)+(cmax-der) + "010011100001110"; --10000-2 = 9998
l_tres<= ((rmax - abajo - 2)*b_rec)+(cmax-der) + "100111000011110"; --20000-2 = 19998
b_ima <= cmax-izq-der-4;
-- Este contador controla el multiplexor que determina que plano de
-- de la imagen se lee y cuando se latchean los píxeles
origen: process (reset,clock)
begin
if reset='1' then
i<='0';
elsif  (clock 'event and clock = '1')  then
if (i='0' and cont < l_tres)then  --27206
i<='1';
else i<='0';
end if;
end if;
end process;
-- Este proceso genera las direcciones a leer de la RAM (rectángulo + pequeño que encierra a la
-- célula de interés)
dir_lect: process(clock,reset)
begin
if reset ='1' then
cont <= "00000000" & ini;
temp<=(others=>'0');
elsif (clock 'event and clock= '1') then
if i='1' then
if cont=fin then
cont<= "00000" & ini_dos;
temp<=(others=>'0');
elsif cont= l_dos then
cont<= "0000" & ini_tres;
temp<=(others=>'0');
elsif (temp=b_ima and cont< l_tres) then
cont<= cont + der + izq + 5;--izq+der+1+4
temp<=(others=>'0');
elsif (cont< l_tres) then
cont <= cont + 1;
temp<=temp+1;
end if;
end if;
end if;
end process;
-- Este proceso genera las direcciones a escribir dela Ram (0 a el tamaño de la imagen
-- reducida)
dir_escr:process(clock,reset)
begin
if reset ='1' then
cont2 <= (others=>'0');-- 0 plano rojo
elsif (clock 'event and clock= '1') then
if i='1' then
if cont=fin then
cont2<="0000010011100010000"; --10000 plano verde
elsif cont=l_dos then
cont2<="0000100111000100000"; --20000 plano azul
elsif (cont<l_tres) then
cont2 <= cont2 + 1;
end if;
end if;
end if;
end process;
-- genera la señal de finalización de módulo
process (clock,reset)
begin
if reset ='1' then
stop<='1';
elsif (clock 'event and clock= '1') then
if cont >= l_tres then
stop<='0';
end if;
end if;
end process;
--Mux que define el origen de la dirección de la Ram
A: addr<=
cont when i='0' else
cont2 when i='1' else
"XXXXXXXXXXXXXXXXXXX";
-- Este proceso genera el registro donde se almacena el píxel leído de la Ram

Reg: process (clock, reset)
begin
if reset ='1' then
pix <= (others => '0');
elsif (clock 'event and clock = '1')then
if (i='0') then
```

```
pix <= data;
else
pix <= pix;
end if;
end if;
end process;
--Mux que define si los datos van hacia o de la Ram (escritura o lectura)
pixel<=pix;
ii<=i;
-- Pin de habilitación de la Ram
wel<= not i;
-- Area del nucleon
cont_areas: process (clock, reset)
begin
if reset ='1' then
area_N<=(others=>'0');
elsif (clock 'event and clock = '1')then
if (i='1' and cont<=fin ) then
if pix=0 then
area_N <= area_N+1;
end if;
end if;
end if;
end process;
area<=area_N;
end Behavioral;
```

B.2.5 Módulo Puntos_elipse

```
----------------------------------------------------------------------------
-- Company:INAOE
-- Engineer: Gloria Castro
-- Create Date:26/07/07
-- Module Name:  calculo_de_elipse
-- FPGA:  Spartan 3 xc3s200-4ft256
----------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
use ieee.numeric_std.all;
use work.PKG.all;

entity puntos_elipse is
generic( data_width:integer:=6);
Port (
reset: in  std_logic;
clk : in  std_logic;
wel:out std_logic;
b_rec: in std_logic_vector(6 downto 0);
h_rec: in std_logic_vector(6 downto 0);
addr: out std_logic_vector(18 downto 0);
area_fondo: out std_logic_vector(11 downto 0);
 fin:out std_logic );
end puntos_elipse;

architecture Behavioral of puntos_elipse is
----------------------------------------------------------------------------
-- señales usadas en la arquitectura de punto_y
---------------------------------------------------------------------------
signal i: std_logic_vector(1 downto 0);
signal y,h,k,a,b: std_logic_vector(6 downto 0);
signal x: std_logic_vector(5 downto 0):="000001";
signal xx: std_logic_vector(5 downto 0):="000000";
signal wa: std_logic_vector(13 downto 0);
signal ws:std_logic_vector(5 downto 0);
signal wo: std_logic_vector (12 downto 0);
signal wel_in: std_logic:='1';
signal DIVIDEND:std_logic_vector (data_width*2 - 1 downto 0);
signal DIVISOR: std_logic_vector (data_width - 1 downto 0);
signal DIV_RESULT: std_logic_vector (data_width downto 0);
signal te:std_logic_vector(18 downto 0);
signal con_f:std_logic_vector(8 downto 0);
----------------------------------------------------------------------------
--- declaracion de señales usadas en la arquitectura de div_unit
-------------------------------------------------------------------------
signal A_REG: std_logic_vector(data_width*2 - 1 downto 0);
signal B_REG: std_logic_vector(data_width - 1 downto 0);
signal RD_REG: std_logic_vector(data_width*2 downto 0);
signal temp: std_logic_vector(data_width +1 downto 0);
----------------------------------------------------------------------------
-- descripción de la arquitectura de div_unit
-----------------------------------------------------------------
begin
aaa: process(CLK)
begin
```

```
if CLK'event and CLK = '1' then
A_REG <= DIVIDEND;
B_REG <= DIVISOR;
RD_REG <= DIVISION(A_REG, B_REG);
end if;
end process;
temp <= RD_REG(data_width - 1 downto 0)* "10";
DIV_RESULT <= RD_REG(data_width*2 downto
data_width)+1 when temp > DIVISOR else
RD_REG(data_width*2 downto data_width);
h <= std_logic_vector(unsigned(b_rec+1)/2);
k <= std_logic_vector(unsigned(h_rec+1)/2);
a <= h-1;
b <= k-1;
--------------------------------------------------------------------------
---          contador de sincronización
--------------------------------------------------------------------------
clk2: process (reset,clk)
begin
if reset='1' then
i<="00";
elsif  (clk 'event and clk = '1')  then
if (i<2 and xx<a)then
i<=i+1;
else
i<="00";
end if;
end if;
end process;
--------------------------------------------------------------------------
-- generador de valores de x (abcisa)para evaluación de
la elipse
--------------------------------------------------------------------------
cont_1_A: process(clk,reset)
begin
if reset ='1' then
x <= "000001";  --  carga inicial 1
elsif (clk 'event and clk= '1') then
if(i=2 and x<a) then
x <= x + 1;
end if;
end if;
end process;
--------------------------------------------------------------------------
-- generador de valores de x (abcisa)para evaluación de
la dirección de y en ram
--------------------------------------------------------------------------
contador: process(clk,reset)
begin
if reset ='1' then
xx <= "000000";  --  carga inicial 13
elsif (clk 'event and clk= '1') then
if(i=1 and x>1 and xx<a) then
xx <= xx + 1;
end if;
end if;
end process;
--------------------------------------------------------------------------
-- señal de habilitación de escritura de la ram
--------------------------------------------------------------------------
esritura: process(clk,reset)
begin
if reset ='1' then
wel_in<='1';  --  carga inicial 13
elsif (clk 'event and clk= '1') then
if(i=2 and xx<a) then
wel_in<='0';
elsif (i=0 and xx<=a) then
wel_in<='1';
end if;
end if;
end process;
--------------------------------------------------------------------------
-- contador del fondo
--------------------------------------------------------------------------
fondo: process(clk,reset)
begin
if reset ='1' then
con_f<=(others=>'0');  --  carga inicial 13
elsif (clk 'event and clk= '1') then
if(i=2 and xx<=a) then
con_f<=con_f+y;
elsif xx=a then
area_fondo <= con_f * "100";
end if;
end if;
end process;
process (clk,reset)
begin
if reset ='1' then
fin <= '1';
elsif (clk 'event and clk= '1') then
if xx=a then
fin<='0';
```

```
end if;
end if;
end process;
----------------------------------------------------------------------
-- operaciones de la ecuación de la elipse
----------------------------------------------------------------------
wel<=wel_in;
wa <= (a*a)-((h-x)*(h-x));
ws <= raiz(wa(11 downto 0));
wo <= b*ws;
DIVIDEND <=wo(11 downto 0);
DIVISOR <=a(5 downto 0);
te<="00000"&(((DIV_RESULT                                    +
b)*std_logic_vector(b_rec))+ xx);--k-1=b
y <= b - DIV_RESULT;
----------------------------------------------------------------------
addr <= te;
end Behavioral;
```

B.2.6 Módulo Decoder

```
----------------------------------------------------------------------
-- Company:                    INAOE
-- Engineer:                   Gloria Castro
-- Create Date:       26/07/07
-- Module Name:   decodificador binario a decimal
-- FPGA:                                   Spartan          3
xc3s200-4ft256
----------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity decoder is
Port (
reset : in  STD_LOGIC; --reset
clk : in  STD_LOGIC;-- reloj para retardo de visualización
A: in std_logic_vector(3 downto 0);-- dígito #1
B: in std_logic_vector(3 downto 0);-- dígito #2
dig: out std_logic_vector(6 downto 0);-- 7 segmentos
display
AN: out std_logic_vector(3 downto 0);-- ctrl ánodos del
display
punto: out std_logic);-- punto decimal display
end decoder;

architecture Behavioral of decoder is
----------------------------------------------------------------------
-- señales usadas
----------------------------------------------------------------------
signal clkdiv  : std_logic_vector(16 downto 0);
signal MUXOUT : std_logic_vector(3 downto 0);
----------------------------------------------------------------------
begin
-- Divide the master clock (50Mhz) a una frecuencia mas
baja.
reloj:       process (reset,clk)
begin
if reset='1' then
clkdiv<=(others=>'0');
elsif clk = '1' and clk'event then
clkdiv <= clkdiv + 1;
end if;
end process;
--Multiplexor para los dos dígitos del display y el pd
multiplexor:process(reset,clk)
begin
if reset='1' then
AN <= "1111";
MUXOUT <= "1111";
punto <='1';
elsif clk='1' and clk'event then
if clkdiv < 65535 then
AN <= "0111";
MUXOUT <= A;
punto <= '0';
else
AN <="1011";
MUXOUT <= B;
punto <= '1';
end if;
end if;
end process;
---decodificador al display de 7 segmentos
dig <="1000000" when MUXOUT = "0000" else
"1111001" when MUXOUT = "0001" else
"0100100" when MUXOUT = "0010" else
"0110000" when MUXOUT = "0011" else
"0011001" when MUXOUT = "0100" else
"0010010" when MUXOUT = "0101" else
"0000010" when MUXOUT = "0110" else
"1111000" when MUXOUT = "0111" else
"0000000" when MUXOUT = "1000" else
"0010000" when MUXOUT = "1001" else
```

```
"1111111";
end Behavioral;
```

B.2.7 Módulo VGA

```
----------------------------------------------------------------------------
-- Company: INAOE
-- Engineer: Gloria Castro
-- Create Date:15:40:09 02/05/2006
-- Design Name:      Controlador VGA
-- Module Name:   vgacore - Behavioral
-- Project Name:      Controlador VGA
-- FPGA:                                    Spartan          3
xc3s200-4ft256
----------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity vga is
Port (
reset       : in  STD_LOGIC;-- resetmodulo vga
clock       : in  STD_LOGIC;-- reloj modulo vga
hsyncb : buffer   STD_LOGIC;-- señal de sincronía
horizontal
vsyncb : out  STD_LOGIC;-- señal de sincronía vertical
rgb         : out   STD_LOGIC_VECTOR (23 downto 0);--
senales canales rgb
data        : inout std_logic_vector (7 downto 0);--datos
memoria de video
addr        : out std_logic_vector (18 downto 0);--
direcciones memoria video
oel : out std_logic;-- señal de lectura memoria de video
alto: std_logic_vector(6 downto 0);--altura imagen a
desplegar
ancho: std_logic_vector(6 downto 0));-- ancho de la
imagen a desplegar
end vga;

architecture Behavioral of vga is
--- Señales del módulo
signal hcnt             : std_logic_vector (9 downto 0);
signal vcnt             : std_logic_vector (9 downto 0);
signal lim_h : std_logic_vector(9 downto 0);
signal pixr,pixg,pixb: std_logic_vector (7 downto 0);
signal blank: std_logic;
signal addr_B: std_logic_vector (18 downto 0);
signal addr_G: std_logic_vector (18 downto 0);
signal addr_R: std_logic_vector (18 downto 0);
signal pix: std_logic_vector (23 downto 0);
signal i:std_logic_vector(1 downto 0);
begin
--Este proceso describe la operación del contador de
píeseles horizontal. Este es puesto síncronamente a
cero cuando el reset va a 1. El contador se incrementa
con el flanco ascendente de cada reloj de píxel. El rango
del contador de píxel horizontal es (0,1583). Cuando el
contador rebasa los 1583 este regresa a 0 en el
siguiente ciclo. debido a que el contador tiene un periodo
de reloj de 1584 píxeles con un reloj de 50MHz este
representa un periodo de 31.66 microsegundos.
A: process(clock, reset)
begin
if reset ='1' then
hcnt <= "0000000000";
elsif (clock 'event and clock= '1') then
if hcnt<792 then
hcnt <= hcnt + 1;
else
hcnt <= "0000000000";
end if;
end if;
end process;
-- Este proceso describe la operación del contador de
líneas vertical. El contador es asíncronamente puesto  a
cero cuando el reset va a 1. El contador se incrementa
en el flanco ascendente del pulso de sincronía horizontal,
después de que una línea de píxeles se ha completado.
El rango para este contador es de 0 a 527 Cuando el
contador rebasa los 527 este regresa a cero en el
siguiente ciclo. Debido a que el contador tiene un
periodo de 528 líneas y debido a que la duración de
cada línea es de 31.75 microsegundos esto hace un
intervalo de figuras a 16.76ms.
B: process (hsyncb, reset)
begin
if reset ='1' then
vcnt <= "0000000000";
elsif (hsyncb 'event and hsyncb='1')then
if vcnt<527 then
vcnt<= vcnt + 1;
else
vcnt <= "0000000000";
end if;
```

```
end if;
end process;
```
-- Este proceso describe la operación del generador de pulsos de sincronía horizontal. La sincronía horizontal es puesta a su nivel inactivo alto cuando el reset es activado. Durante su operación normal, la salida de sincronía horizontal es actualizada con cada pulso de reloj de píxel. la señal de sincronía va a bajo después de que el contador de píxeles alcanza 1212 y continúa un ciclo después de que el contador rebasa 1404. Esto da un pulso de sincronía horizontal bajo de (1404-1212)= 192 pulsos de reloj de píxeles Con un reloj de píxel de 50MHz, este representa 3.84us. El pulso de sincronía horizontal empieza 1213 ciclos después de que la línea de píxeles comienza, la cual representa 24.26us.

```
C: process (clock, reset)
begin
if reset ='1' then
hsyncb <='1';
elsif (clock'event and clock ='1') then
if (hcnt >= 606 and hcnt <702)then
hsyncb <= '0';
else
hsyncb <= '1';
end if;
end if;
end process;
```
-- Este proceso describe la operación del pulso de sincronía vertical. La sincronía vertical es inactiva al ponerla a su nivel alto cuando el reset es activado. Durante su operación normal la salida de sincronía vertical es actualizada después de que cada línea de píxeles es completada. La señal de sincronía va a bajo después de que el contador de líneas rebasa 493 y continua asta un ciclo después de que el contador rebasa 495. Esto da un pulso de sincronía de 495-493=2 líneas. Con un intervalo de línea de 31.75us, esto representa un pulso de sincronía en bajo de 63.5 us. El pulso de sincronía empieza en 494x31.75 us =15.68ms después comenzar la primera línea de video

```
D: process (hsyncb, reset)
begin
if reset='1' then
vsyncb <='1';
elsif (hsyncb 'event and hsyncb ='1')then
if (vcnt >= 490 and vcnt < 492)then
vsyncb <= '0';
else
vsyncb <= '1';
end if;
end if;
end process;
lim_h<=ancho * "011";
```
-- acota la señal de video fuera de la región visible (0,0)->(299,99)

```
E: blank <= '1' when (hcnt >= lim_h or vcnt >= alto )else '0';-- 480
G:oel <= blank;
```
-- Este proceso genera las direcciones de los píxeles del plano rojo de la imagen la dirección se incrementa cada 3 ciclos de reloj y se reinicia cuando vcnt<480

```
dirR: process(clock, reset,blank,vcnt)
begin
if reset ='1' then
addr_R <= (others => '0');
elsif (clock 'event and clock= '1') then
if(blank = '0' and i=2) then
addr_R <= addr_R + 1;
elsif vcnt>480 then
addr_R <= (others => '0');
end if;
end if;
end process;
```
-- Este proceso genera las direcciones de los píxeles del plano verde de la imagen la dirección se incrementa cada 3 ciclos de reloj y se reinicia (dir 10000 porque a.C. inicia el plano rojo)cuando vcnt<480

```
dirG: process(clock, reset,blank,vcnt)
begin
if reset ='1' then
addr_G <= "00000100111000011111";--9999
elsif (clock 'event and clock= '1') then
if(blank = '0' and i=0) then
addr_G <= addr_G + 1;
elsif vcnt>480 then
addr_G <= "00000100111000011111";--9999
end if;
end if;
end process;
```
-- Este proceso genera las direcciones de los píxeles del plano azul de la imagen la dirección se incrementa cada 3 ciclos de reloj y se reinicia (dir 20000 x q a.C. inicia el plano azul)cuando vcnt<480

```
dirB: process(clock, reset,blank,vcnt)
```

```
begin
if reset ='1' then
addr_B <= "0000100111000011111";--19999
elsif (clock 'event and clock= '1') then
if(blank = '0' and i=1) then
addr_B <= addr_B + 1;
elsif vcnt>480 then
addr_B <= "0000100111000011111";--19999
end if;
end if;
end process;
-- Este proceso es un contador de 0-2 para determinar cuando se realiza el incremento de las direcciones de la Ram
origen: process (reset,clock)
begin
if reset='1' then
i<="00";
elsif  (clock 'event and clock = '1')  then
if  (i<=1  and  hcnt<lim_h  -  1  and  vcnt<alto)then  -- hcnt<299 vcnt<100
i<=i+1;
else
i<="00";
end if;
end if;
end process;
---Decodificador que define el origen de la dirección de la Ram
addr<=addr_R when i="00" else
addr_G when i="01" else
addr_B when i="10" else
“XXXXXXXXXXXXXXXXXXX";
-----------------------------------------------------------
RegR: process (clock, reset)
begin
if reset ='1' then
pixr <= (others => '0');
elsif (clock 'event and clock = '1')then
if (blank='0' and i=0) then
pixr <= data;
else
pixr <= pixr;
end if;
end if;
end process;
---------------------------------
RegG: process (clock, reset)
begin
if reset ='1' then
pixg <= (others => '0');
elsif (clock 'event and clock = '1')then
if (blank='0' and i=1) then
pixg <= data;
else
pixg <= pixg;
end if;
end if;
end process;
-----------------------------
RegB: process (clock, reset)
begin
if reset ='1' then
pixb <= (others => '0');
elsif (clock 'event and clock = '1')then
if (blank='0' and i=2) then
pixb <= data;
else
pixb <= pixb;
end if;
end if;
end process;
--- Este proceso describe el proceso por el cual el píxel es mapeado dentro de los seis bits lo cuales manejan el rojo, verde y azul. El registro es cero cuando el reset va a alto.
J: process (clock, reset,vcnt)
begin
if reset ='1' then
pix <= (others => '0');
elsif (clock 'event and clock ='1')then
if blank='0' and i=0 then    --pblank
if hcnt=0 then
pix<=(others=>'0');
else pix <= pixb & pixg & pixr;
end if;
elsif (blank='1')   then
pix<=(others=>'0');
else
pix <= pix;
end if;
end if;
end process;
rgb<=pix;
```

```
end Behavioral;
```

B.2.8 Funciones: Raíz cuadrada, división

```
--------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
package PKG is
function raiz (P:std_logic_vector(11 downto 0) )
return std_logic_vector;
function division (DIVIDEND: std_logic_vector;DIVISOR:
std_logic_vector)
return STD_LOGIC_VECTOR;
function div(A: std_logic_vector; B: std_logic_vector; Q:
std_logic_vector; EXT: std_logic)
return STD_LOGIC_VECTOR;
function div1(A: std_logic_vector; B: std_logic_vector; Q:
std_logic_vector; EXT: std_logic)
return STD_LOGIC_VECTOR;
end PKG;

package body PKG is
-- Raiz cuadrada 0-2500
function raiz (P:std_logic_vector(11 downto 0) )
return std_logic_vector is
variable U : std_logic_vector(5 downto 0);
begin
if (P>=1 and P<=2) then
                U:= "000001";
        elsif (P>=3 and P<=6) then
                U:= "000010";
        elsif (P>=7 and P<=12) then
                U:= "000011";
        elsif (P>=13 and P<=20) then
                U:= "000100";
        elsif (P>=21 and P<=30) then
                U:= "000101";
        elsif (P>=31 and P<=42) then
                U:= "000110";
        elsif (P>=43 and P<=56) then
                U:= "000111";
        elsif (P>=57 and P<=72) then
                U:= "001000";
elsif (P>=73 and P<=90) then
                U:= "001001";
elsif (P>=91 and P<=110) then
                U:= "001010";
elsif (P>=111 and P<=132) then
                U:= "001011";
elsif (P>=133 and P<=156) then
                U:= "001100";
elsif (P>=157 and P<=182) then
                U:= "001101";
elsif (P>=183 and P<=210) then
                U:= "001110";
elsif (P>=211 and P<=240) then
                U:= "001111";
elsif (P>=241 and P<=272) then
                U:= "010000";
elsif (P>=273 and P<=306) then
                U:= "010001";
elsif (P>=307 and P<=342) then
                U:= "010010";
elsif (P>=343 and P<=380) then
                U:= "010011";
elsif (P>=381 and P<=420) then
                U:= "010100";
elsif (P>=421 and P<=462) then
                U:= "010101";
elsif (P>=463 and P<=506) then
                U:= "010110";
elsif (P>=507 and P<=552) then
                U:= "010111";
elsif (P>=553 and P<=600) then
                U:= "011000";
elsif (P>=601 and P<=650) then
                U:= "011001";
elsif (P>=651 and P<=702) then
                U:= "011010";
elsif (P>=703 and P<=756) then
                U:= "011011";
elsif (P>=757 and P<=812) then
                U:= "011100";
elsif (P>=813 and P<=870) then
                U:= "011101";
elsif (P>=871 and P<=930) then
                U:= "011110";
elsif (P>=931 and P<=992) then
                U:= "011111";
elsif (P>=993 and P<=1056) then
```

```
                U:= "100000";
        elsif (P>=1057 and P<=1122) then
                U:= "100001";
        elsif (P>=1123 and P<=1190) then
                U:= "100010";
        elsif (P>=1191 and P<=1260) then
                U:= "100011";
        elsif (P>=1261 and P<=1332) then
                U:= "100100";
        elsif (P>=1333 and P<=1406) then
                U:= "100101";
        elsif (P>=1407 and P<=1482) then
                U:= "100110";
        elsif (P>=1483 and P<=1560) then
                U:= "100111";
        elsif (P>=1561 and P<=1640) then
                U:= "101000";
        elsif (P>=1641 and P<=1722) then
                U:= "101001";
        elsif (P>=1723 and P<=1806) then
                U:= "101010";
        elsif (P>=1807 and P<=1892) then
                U:= "101011";
        elsif (P>=1893 and P<=1980) then
                U:= "101100";
        elsif (P>=1981 and P<=2070) then
                U:= "101101";
        elsif (P>=2071 and P<=2162) then
                U:= "101110";
        elsif (P>=2163 and P<=2256) then
                U:= "101111";
        elsif (P>=2257 and P<=2352) then
                U:= "110000";
        elsif (P>=2353 and P<=2450) then
                U:= "110001";
        elsif (P>=2451 and P<=2500) then
                U:= "110010";
        else
                U:= "000000";
        end if;
return U;
end raiz;
--- Función división de dos enteros positivos
function division(DIVIDEND: std_logic_vector; DIVISOR:
std_logic_vector)
return std_logic_vector is
variable B : std_logic_vector(DIVISOR'length - 1 downto
0);
variable A : std_logic_vector(DIVIDEND'length - 1
downto 0);
variable QUOTIENT, REMAINDER : std_logic_vector
(DIVISOR'length - 1 downto 0);
variable VECT : std_logic_vector(DIVIDEND'length
downto 0);
variable QI : std_logic_vector(0 downto 0);
variable OVFL : std_logic;
begin
A := DIVIDEND;
B := DIVISOR;
QI := (others =>'0');
VECT := div(A, B, QI, '0');
QUOTIENT := VECT(VECT'high - 1 downto B'high + 1);
REMAINDER := VECT(B'high downto 0);
OVFL := VECT(VECT'high );
return OVFL & QUOTIENT & REMAINDER;
-- return VECT;
end division;
function div(A: std_logic_vector; B: std_logic_vector;
Q: std_logic_vector; EXT: std_logic)
 return std_logic_vector is
variable R : std_logic_vector(A'length - 2 downto 0);
variable RESIDUAL : std_logic_vector(A'length - 1
downto 0);
variable QN : std_logic_vector(Q'length downto 0);
variable S : std_logic_vector(B'length + Q'length downto
0);
begin
S := div1(A(A'high downto A'high - B'high), B, Q, EXT);
QN := S(S'high downto B'high + 1);
 if A'length > B'length then
   R := S(B'high - 1 downto 0) & A(A'high - B'high - 1
downto 0);
return DIV(R, B, QN, S(B'high));    -- save MSB '1' in the
rest for future sum
else
RESIDUAL := S(B'high downto 0);
return QN(QN'high - 1 downto 0) & RESIDUAL;  -- delete
initial '0'
end if;
end div;
function div1(A: std_logic_vector; B: std_logic_vector; Q:
std_logic_vector; EXT: std_logic)
return std_logic_vector is
```

```
variable S : std_logic_vector(A'length downto 0);
variable REST : std_logic_vector(A'length - 1 downto 0);
variable QN : std_logic_vector(Q'length downto 0);
begin
S := EXT & A - B;
QN := Q & (not S(S'high));
if S(S'high) = '1' then
REST := A;
else
REST := S(S'high - 1 downto 0);
end if;
return QN & REST;
end div1;
end PKG;
```

B.2.9 Archivo de restricciones

```
--------------------------------------------------------------------------
-- Company: INAOE
-- Engineer: Gloria Castro
-- Create Date:15:40:09 02/05/2006
-- Module Name: Archivo UCF
-- FPGA:Spartan 3 xc3s200-4ft256
--------------------------------------------------------------------------
# direcciones de la memoria RAM
NET "addr<0>"  LOC = "E6"  ;
NET "addr<10>"  LOC = "A10"  ;
NET "addr<11>"  LOC = "B10"  ;
NET "addr<12>"  LOC = "B12"  ;
NET "addr<13>"  LOC = "B11"  ;
NET "addr<14>"  LOC = "B13"  ;
NET "addr<15>"  LOC = "A12"  ;
NET "addr<16>"  LOC = "B14"  ;
NET "addr<17>"  LOC = "A13"  ;
NET "addr<18>"  LOC = "D9"  ;
NET "addr<1>"  LOC = "C5"  ;
NET "addr<2>"  LOC = "C6"  ;
NET "addr<3>"  LOC = "C7"  ;
NET "addr<4>"  LOC = "C8"  ;
NET "addr<5>"  LOC = "C9"  ;
NET "addr<6>"  LOC = "B8"  ;
NET "addr<7>"  LOC = "A7"  ;
NET "addr<8>"  LOC = "A9"  ;
NET "addr<9>"  LOC = "A8"  ;
# datos de la memoria RAM
NET "data<0>"  LOC = "D5"  ;
NET "data<1>"  LOC = "D6"  ;
NET "data<2>"  LOC = "E7"  ;
NET "data<3>"  LOC = "D7"  ;
NET "data<4>"  LOC = "D8"  ;
NET "data<5>"  LOC = "D10"  ;
NET "data<6>"  LOC = "B4"  ;
NET "data<7>"  LOC = "B5"  ;
# leds del FGPA
#NET "area<13>"  LOC = "F13"  ;
#NET "area<12>"  LOC = "R16"  ;
#NET "area<11>"  LOC = "P15"  ;
#NET "area<10>"  LOC = "N15"  ;
#NET "area<9>"  LOC = "G13";
#NET "area<8>"  LOC = "E14";
NET "area<7>"  LOC = "P11"  ;
NET "area<6>"  LOC = "P12"  ;
NET "area<5>"  LOC = "N12"  ;
NET "area<4>"  LOC = "P13"  ;
NET "area<3>"  LOC = "N14"  ;
NET "area<2>"  LOC = "L12"  ;
NET "area<1>"  LOC = "P14";
NET "area<0>"  LOC = "K12";
# pines de habilitación de la memoria RAM
NET "wel"  LOC = "A3"  ;
NET "oel"  LOC = "A4"  ;
NET "cel"  LOC = "A5"  ;
NET "clk"  LOC = "T9"  ;
NET "reset"  LOC = "L14"  ;
NET "sw0" LOC = "F12";
NET "sw1" LOC = "G12";
# push buttons 4,3,2
NET "lim<0>" LOC = "M13";
NET "lim<1>" LOC = "M14";
NET "lim<2>" LOC = "L13";
# ánodos de los displays
NET "AN<0>"  LOC = "D14"  ;
NET "AN<1>"  LOC = "G14"  ;
NET "AN<2>"  LOC = "F14"  ;
NET "AN<3>"  LOC = "E13"  ;
# 8 segmentos del display
NET "dig<0>"  LOC = "E14"  ;
NET "dig<1>"  LOC = "G13"  ;
NET "dig<2>"  LOC = "N15"  ;
NET "dig<3>"  LOC = "P15"  ;
NET "dig<4>"  LOC = "R16"  ;
NET "dig<5>"  LOC = "F13"  ;
NET "dig<6>"  LOC = "N16"  ;
```

```
NET "punto"  LOC = "P16"  ;
# señales de sincronía del VGA
NET "hsyncb"  LOC = "R3"  ;
NET "vsyncb"  LOC = "C16";
# Canal rojo del VGA
NET "rgb<7>"  LOC = "T3"  ;
NET "rgb<6>"  LOC = "N11"  ;
NET "rgb<5>"  LOC = "P10";
NET "rgb<4>"  LOC = "R10";
NET "rgb<3>"  LOC = "T7"  ;
NET "rgb<2>"  LOC = "R7"  ;
NET "rgb<1>"  LOC = "N6";
NET "rgb<0>"  LOC = "M6";
# canal verde del VGA
NET "rgb<15>"  LOC = "C15";
NET "rgb<14>"  LOC = "D15";
NET "rgb<13>"  LOC = "E15";
NET "rgb<12>"  LOC = "F15";
NET "rgb<11>"  LOC = "G16"  ;
NET "rgb<10>"  LOC = "H16"  ;
NET "rgb<9>"  LOC = "K16";
NET "rgb<8>"  LOC = "L15";
# canal azul del VGA
NET "rgb<23>"  LOC = "C10";
NET "rgb<22>"  LOC = "E10";
NET "rgb<21>"  LOC = "C11";
NET "rgb<20>"  LOC = "D11";
NET "rgb<19>"  LOC = "C12"  ;
NET "rgb<18>"  LOC = "D12"  ;
NET "rgb<17>"  LOC = "E11";
NET "rgb<16>"  LOC = "B16";
```

REFERENCIAS

[1] Berend Houwen*."The differential Cell Count"* in XVIIth International Symposium on Technological Innovations in Laboratory Hematology, May 13-15, 2004 in Barcelona, Spain, pp. 89-100.

[2] Themi H., Diem H, Haferlach T, "*Color atlas of Hematology: Practical Microscopic and Clinical Diagnosis"*, Thieme 2th edition, New York 2004, p. 209.

[3] John P. Greer, John Foerster, John N. Lukens. "*Wintrobe's Clinical Hematology"*. Lippincott Williams & Wilkins Publishers; 11th edition, December 2003.

[4] "Fórmula leucocitaria." *Medline Plus Enciclopedia Médica.* 6 de septiembre 2007, 01:15. http://www.nlm.nih.gov/medlineplus/spanish/ency/article/003657.htm

[5] Novis David A, Walsh Molly. *" Laboratory Productivity and the rate of manual periferical Blood Smear Review",* May 2006, Arch Pathol LabMed, vol 130.

[6] Williams J William, Beutler Ernest, Erslev J. Allan and Rundles R Wayne. *Hematology*. Mc Graw Hill 1972 United States of America.

[7] Vlachopoulos C, Aznaouridis K, Pietri P, "*White blood cell count is a maker of wave reflections and arterial stiffness in healthy individuals"*. Atherosclerosis. 2006 Jul; 187(1):218-219.Epub 2006 May 3.

[8] Green James E. "*A practical Application of Computer Pattern Recognition Research,The Abbott ADC-500 Differential Classifier"*. The journal of Histochemistry and Cytochemistry. Vol. 27, 1979, pp. 160-173.

[9] K. Langford, L. Luchman, Jhones, R. Miller, D. Walck. "*Performance evaluation of the sysmex XT-2000i automated haematology analyzer"*. Laboratory Hematology, volume 9, number 1, March 2003, pp 29-37.

[10] O'Neil Patick, Vital Esther, Betancourt Noemi, "*Performance Evaluation of the complete blood count and white blood cell diferential parameters on the Act 5diff hematology analyzer"*. Laboratory Hematology, volume 7, number 3, September 2001, pp 116-124.

[11] Harris Neil, Jou Josep Maria, Devoto Gianluigi. "*Performance Evaluation of the ADVIA 2110 Hematology Analizer: An International Multicenter Clinical Trial*". Laboratory Hematology volume 11, March 2005 p. 62-70.

[12] Brambila Eduardo, Castillo Guerra Rosalba y Zarain Lozano Patricia. "*Comparación entre tres métodos manuales empleados en la cuenta diferencial de leucocitos respecto a un equipo automatizado*". Bioquimia volumen 28 número 3, septiembre 2003.

[13] Ruzicka Katharina, Veitl Mario, Thalhammer Renate, "*The new hematology Analyzer Sysmex XE-2100 PerformanceEvaluation of Novel white Blood Cell DifferentialTechnology*", Phatol Laboratory Medicine Vol 125, March 2001.

[14] Lofsness Karen. *Hematography Plus "An Instruccional Program and Atlas od Blood and Bone Marrow Morphology*": University of Minnesota, 2000.

[15] Grignaschi. J. V. "*Diagnóstico citológico de las hemopatías*". Ed panamericana, España 1991.

[16] W. Bacus James and Gose Earl, *"Leukocyte Pattern Recognition"*, IEEE Transactions on Systems, Man and Cybernetics, vol. 4, September 1972.

[17] Diaz G, Cappai C, Setzu MD, Diana A. "*Nuclear pattern recognition by two-parameter texture analisis".* Computer Methods and Programs in Biomedicine 1996;49:1-9.

[18] Ushizima Sabino Daniela Mayumi, Fontoura Costa Luciano da, Gil Rizzatti Edgar, Zago Marco António, *"A texture approach to leukocyte recognition"*, Real-Time Imaging: special issue on imaging bioinformatics: Part III, vol.10 issue 4, pp. 205-216, August 2004.

[19] Aus H, Harms H, ter Meulen V,Gunzer U. "*Statistical evaluation of computer extracted blood cell features for screening populations to detect leukemias"*. Pattern Recognition,Theory and Applications 1987;87:509-18.

[20] Young IT, Verbeek P, Mayall BH. "*Characterization of chromatin distribution in cell nuclei*". Cytometry 1986;(7)5:467-74.

[21] "Leucocito." *Wikipedia, La enciclopedia libre*. 8 jul 2007, 08:12 UTC. 5 sep 2007,15:02. http://es.wikipedia.org/w/index.php?title=Leucocito&oldid=9925145

[22] Baxes Gregory. "*Digital Image Processing principles and applications"*. Ed John Wiley. United States of America 1994.p.452.

[23] Gonzalez Rafael C., Wintz Paul. "*Digital Image Processing"*. Addison Wesley. United States 1977, p. 431,

[24] "Worksheets", *Image Processing Learning Resources HIPR2*. Noviembre 2006, 6:08. http://homepages.inf.ed.ac.uk/rbf/HIPR2/histgram.htm

[25] Awcock G.W. & Thomas R. "*Applied image processing"*. McGraw-Hill. United States of America 1996. p. 300.

[26] Russ John C. "*The image processing handbook"*. Thrid edition, ed. CRC Press, Springer and IEEE Press. United States of America 1998. p. 984.

[27] Bankman Isaac N. "*Handbook of medical imaging processing and analysis"*. Ed. Academic Press. San Diego, CA USA 2000. p. 910.

[28] B. Hasser, Norman; P. LaSalle, Joseph;A. Sullivan, Joseph. "*Análisis Matemático 1"*, 1a edición, Ed. Trillas, 1970, p.810.

[29] "Digilent Adept User Manual", Digilent Corporation Website, 2005, p 10.

[30] "Reference Project Library User Manual", Digilent Corporation Website, 2005, p.5

[31] "Digilent MenUtil User Manual", Digilent Corporation Website,2005, p.3

[32] "ISE 8.2i Quick Start Tutorial". Xilinx Website.p.30

[33] "Development System Reference Guide", Xilinx Corporation Website, 1994-2002 p. 584.

[34] "Image processing Toolbox 6 User's Guide", MathWorks Corporation Website,2007, p. 1254.

[35] "Spartan-3 Starter Kit Board User Guide V1.1", Xilinx Corporation Website May 13,2005, p.64.

[36] "Digilent Memory Module 1 Reference Manual", Digilent Corporation Website, 2005, p. 3.

[37] "Digilent PmodUSB2 Module Board Reference Manual". Digilent Corporation Website, 2005,p.3.

[38] "Digilent NXVGA Reference Manual". Digilent Corporation website, November 9,2006.

[39] Vanden Bout D. Application Note *"VGA Generator for the XSA Boards"*, XESS Corporation Website, October 2004 p. 10.

[40] "Single Clock Unsigned Division Algorithm: Overview". *OPENCORES.ORG*. 27 de junio de 2007, 16:10:23. http://www.opencores.org/projects.cgi/web/single_clock_divider/overview