Filtrado de imágenes usando hardware dedicado

Acosta Nelson & Tosini Marcelo

INCA/INTIA – Depto. Computación y Sistemas – Fac. Cs. exactas – UNCPBA
(7000) Tandil – Argentina – Email: { *nacosta*, *mtosini* }@exa.unicen.edu.ar

Resumen. Este artículo propone el diseño de una arquitectura dedicada para el procesamiento de imágenes. Se presenta el ciclo de desarrollo, en el cual el punto de partida es la implementación de la solución software, la que es migrada a VHDL para las etapas de simulación y síntesis. El funcionamiento y rendimiento del método fueron verificados usando *Synopsys VHDL* y simulaciones de computadora.

1.- Introducción

El tratamiento de imágenes en medicina y robótica durante la adquisición, transmisión, almacenamiento, interpretación y acceso se realiza comúnmente en forma digital. El manejo de este tipo de datos incluye el filtrado para mejorar las imágenes, proceso de gran impacto negativo sobre el tiempo requerido para la obtención de imágenes de alta calidad. Cualquier tratamiento digital de imágenes utiliza operaciones básicas de filtrado digital como interpolación [1, 2, 3, 4], operaciones de *pixel* (negativo, umbral), enmascaramiento (Sobel, Kirsh, Laplaciano, on-pixel, mediana, morfológicas) y vectorización.

Estas operaciones necesitan gran cantidad de tiempo de cálculo para ser procesadas, más aún en procesadores de propósito general con software específico. El uso de hardware dedicado permite alcanzar los límites de frecuencia necesarios para que el procesamiento se realice en tiempo real. Tal es el caso del uso de procesadores de señales digitales (DSP) o dispositivos dedicados.

Este artículo propone, por lo tanto, el diseño de un sistema de hardware dedicado para el filtrado de imágenes en un HDL (*hardware description language*) para su síntesis e implementación en plataformas FPGA.

2.- Implementación en software

El software de filtrado de imágenes desarrollado en la primera etapa, implementa operaciones comunes tales como zoom con varios métodos de interpolación, operaciones de pixel (negativo, umbral con y sin fondo), operaciones de enmascaramiento (Sobel, Kirsh, Laplaciano, on-pixel, mediana, morfólogicas) y vectorización.

En la etapa de diseño de hardware se utilizó el paquete de desarrollo ACTIVE-HDL v3.6; el cual posee una interfase potente TCL/TK que facilita la comunicación entre un HDL y un HLL (*high level language*). La herramienta de síntesis usada fue el FPGA Express v3.4 (de Synopsys) y el Foundation v3.1i (de Xilinx). Los filtros elegidos para su implementación en hardware fueron variaciones del Laplaciano (con/sin binarización), algunos filtros de "*compass*" y el filtro mediana.

3.- Filtros desarrollados

Se implementaron 2 tipos de filtros de imagen: a) Algoritmos convolucionales (Sobel, Laplaciano) y b) Morfológiccos (filtro mediana) con imágenes de 8 bits de profundidad de color.

El filtro de realce de ejes de Sobel extrae todos los arcos o ejes de una imagen, sin considerar su dirección (omnidireccional). Se implementa como la suma de operaciones de realce de arcos direccionales. La imagen resultante aparece como un contorno omnidireccional de los objetos de la imagen original. Las regiones de brillo constante resultan oscuras, mientras que aquellas con variaciones de brillo son resaltadas [5]. La implementación comprende 5 etapas: a) Definición de la

máscara para el primer espacio de convolución, b) Procesamiento por grupos de pixeles, c) Definición de la máscara para el segundo espacio de convolución, d) procesamiento por grupos de pixeles, y e) Proceso punto a punto de las 2 imágenes (adición). La Fig. 1 muestra las máscaras clásicas de Sobel.

Vertical

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

Horizontal

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

Fig. 1. Máscaras del filtro Sobel

El filtro de realce de ejes Laplaciano extrae todos los ejes de una imagen sin considerar su dirección. La imagen resultante aparece como un contorno omnidirecional de los objetos de la imagen original. Las regiones de brillo constante resultan oscuras mientras que las de brillo variable son enfatizadas. Esta implementación se realiza en 2 pasos: a) Definición de la máscara para el espacio de convolución, y b) Procesamiento por grupos de pixeles. La Fig. 2 muestra tres máscaras comunes para este filtro.

-1	-1	-1
-1	8	-1
-1	-1	-1

0	-1	0
-1	4	-1
0	-1	0

1	-2	1
-2	4	-2
1	-2	1

Fig. 2 – Máscaras usadas para el filtro Laplaciano.

Los filtros de Mediana y *Ranking* permiten obtener el valor promedio de una matriz de NxM pixeles; por ejemplo, en un grupo de 3x3, el píxel con el valor de la mediana está en la mitad entre los 4 más brillantes y los 4 más oscuros. La imagen resultante queda libre de pixeles cuyos valores están en los extremos de cada grupo de pixeles de entrada. Su implementación incluye: a) Definición de las categorías de pixeles, y b) Procesamiento no lineal por grupos de pixeles (Fig. 3).

Fig. 3. Procesamiento por filtro Mediana.

Los algoritmos de convolución operan tomando cada pixel de la imagen original y sus vecinos próximos (creando una matriz de 3 x 3). Sobre este grupo de pixeles se aplica la matriz (Sobel o Laplaciana) apropiada a fin de realizar la detección de arcos verticales, horizontales u oblicuos. Posteriormente se aplica un proceso de binarización a la imagen.

La posición central de esta matriz de 3 x 3 se superpone con cada píxel de la imagen original. La suma de los 9 productos (matriz e imagen) se compara posteriormente con un umbral de nivel. Si el resultado es mayor que el umbral, el píxel forma parte de un eje. Caso contrario, no. La Fig. 4 muestra este proceso.

Para realizar esta operación la imagen entera es leída desde archivo y cargada en una memoria RAM. Todos los píxeles son analizados para determinar si la imagen puede o no ser binarizada. El resultado del proceso (la imagen binarizada) es cargado en otra área de RAM y posteriormente salvado a un archivo de salida. El filtro Mediana funciona con un principio similar [6].

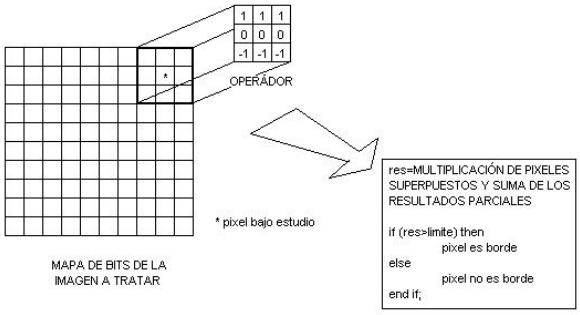

Fig. 4. Proceso de binarización.

4.- Implementación en hardware

El sistema diseñado se divide en tres niveles (Fig. 5):

- El nivel externo actúa de interfaz entre el sistema y los dispositivos externos (memoria, reloj, selección de filtros). Se compone de dos bloques principales: A) El módulo principal que implementa toda la funcionalidad de los filtros y la interacción con la memoria. B) El módulo que implementa la memoria RAM propiamente dicha.
- El nivel de sistema realiza la transferencia entre la memoria externa y el sistema de archivos, y la ejecución de los filtros. Incluye 3 bloques principales: A) Un bloque de proceso de la imagen e interacción con la RAM, B) Un módulo que carga, mantiene y almacena imágenes desde/hacia los archivos externos y C) La unidad de control central basada en un FSM (*Finite State Machine*).
- El nivel de ejecución de filtros, que incluye: A) El contador de pixeles procesados de la imagen, B) El contador de pixeles en la matriz de filtro, C) Un circuito de verificación de límites laterales de la imagen analizada, D) La dirección RAM de cada pixel, E) La memoria de filtro usada para seleccionar el filtro de convolución o morfológico para procesar los pixeles, F) La memoria de almacenamiento de resultados parciales y G) La unidad de control específica.

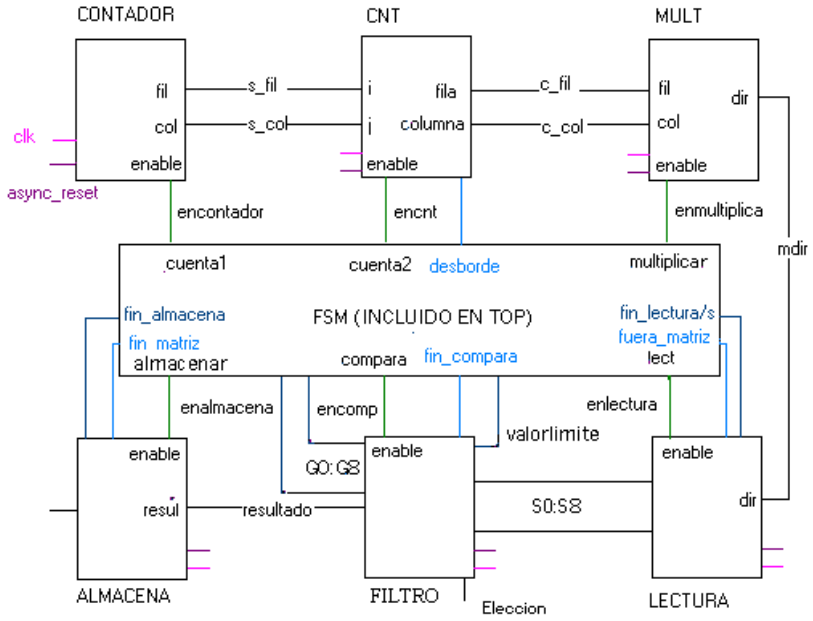

Fig. 5. Diagrama de bloques de la entidad de nivel superior.

Las Figs. 7 a 11 muestran los resultados obtenidos de aplicar los filtros de mediana y de convolución con las máscaras de la Fig. 6. La primera máscara (Fig. 6) es una variación del filtro

Laplaciano. La segunda, un operador que detecta el gradiente para arcos que crecen en sentido norte. La tercera y cuarta se corresponden con las máscaras Sobel horizontal y vertical. La quinta es una variante del operador Kirsch usado para detección de arcos verticales. Todas ellas son usadas como operadores "*compass*", dado que detectan arcos orientados. La Fig. 7 muestra 2 imágenes usadas como ejemplo y los resultados de la aplicación del filtro mediana sobre ellas.

-1	-1	-1
-1	8	-1
-1	-1	-1

1	1	1
0	0	0
-1	-1	-1

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

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

5	5	5
-3	0	-3
-3	-3	-3

1	-2	1
-2	4	-2
1	-2	1

Fig. 6. Máscaras usadas en los ejemplos de las Figs. 7 a 11.

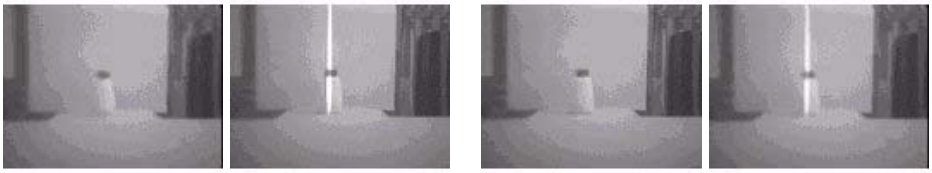

Fig. 7. Imágenes de entrada (las 2 izquierdas) – Resultados de la aplicación del filtro mediana (las 2 derechas).

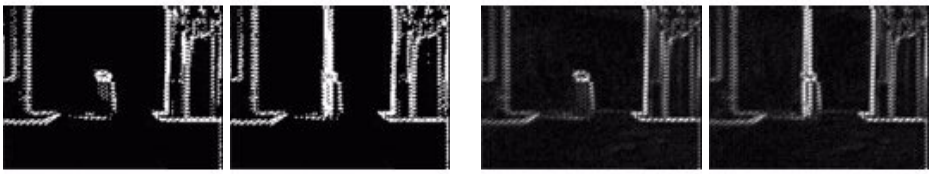

Fig. 8. Resultados de la aplicación de la máscara 1. Umbral = 0 en las 2 izquierdas y umbral = 60 en las derechas.

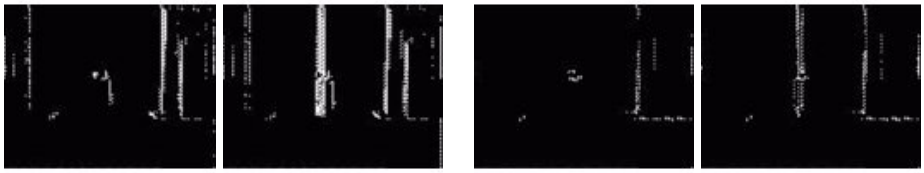

Fig. 9. Resultados de la aplicación de la máscara 2 (las 2 izquierdas) y 3 (las 2 derechas).

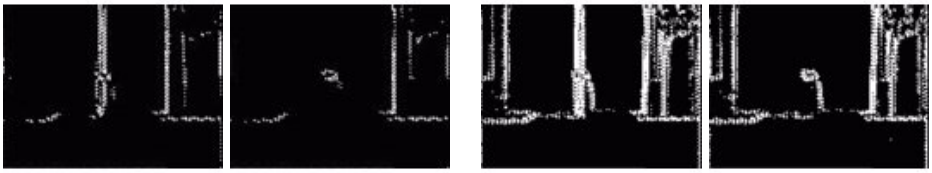

Fig. 10. Resultados de la aplicación de la máscara 4 (las 2 izquierdas) y 5 (las 2 derechas).

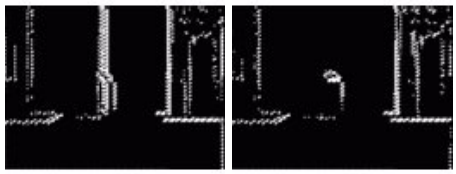

Fig. 11. Resultados de la aplicación de la máscara 6.

6.- Conclusiones

Las estrategias de diseño de las etapas de software y hardware muestran claramente la diferencia entre un desarrollo HLL y HDL (Pascal y VHDL, respectivamente). Por otro lado, la puesta a punto de la aplicación hardware se ve bastante favorecida por el uso de herramientas de simulación potentes. Esta facilidad resultó interesante por la facilidad que brindó a la hora de realizar pruebas extensivas de diversas máscaras sobre diferentes imágenes.

Se logró desarrollar una arquitectura simple para el procesamiento de imágenes digitales. Se probaron diversos casos experimentales y se llevó adelante el diseño algorítmico y arquitectural.

Los posibles trabajos futuros son: Un análisis más profundo de los costos comparativos en velocidad y área de distintos filtros; La extensión de los operadores de convolución con la incorporación de operadores que usen 2 matrices ortogonales de convolución; La adaptación de la arquitectura a fin de soportar paradigmas alternativos como múltiples unidades de proceso, ruta de datos segmentada, etc.; El desarrollo físico del circuito final basado en una FPGA para proceso de imágenes en tiempo real.

Referencias

[1] R. C. Gonzalez and R.E. Woods: "*Digital Image Processing*". Addison Wesley, Reading, 1992.

[2] Gregory A. Baxes, "*Digital Image Processing*", John Wiley & Sons, Inc.,1994.

[3] Toshiro Kubota, "*CSCI770 Digital Image Processing*", 1999, http://www.cs.sc.edu/~kubota/csci770/syl.html.

[4] Anil K. Jain, "*Fundamentals of Digital Image Processing*", Prentice Hall, Inc.,1989.

[5] R. Klette, y P. Zamperoni, "*Handbook of Image Processing Operators*", J. Wiley & Sons, 1995.

[6] John L. Smith, "*Implementing Median Filters in XC4000E FPGAs*", http://www.xilinx.com.