Implementación de un Algoritmo para procesamiento de imágenes en una FPGA

Implementación de un Filtro de Mediana para reducción de ruido

Jorge Osio*; Jose Rapallini; Antonio Adrián Quijano; Jesús Ocampo
Centro de Técnicas Analógico – Digitales (CeTAD)
Facultad de Ingeniería – Universidad Nacional de La Plata
La Plata, Argentina
* Becario de Estudios CIC – Comisión de investigaciones Científicas de la prov. de Bs. As.
josio@gioia.ing.unlp.edu.ar; josrap@ing.unlp.edu.ar; Quijano@ing.unlp.edu.ar; jmfocampo@ing.unlp.edu.ar

Resumen— Este trabajo contempla la corrección de defectos en imágenes mediante el filtrado espacial no lineal. En general los filtros espaciales no lineales más conocidos son los filtros de orden estadístico, cuya respuesta está basada en el ordenamiento (ranking) de los píxeles contenidos en una zona de la imagen.

El filtro de orden estadístico más conocido en el procesamiento digital de imágenes es el filtro de mediana. Dicho filtro es muy eficiente para la reducción de ruido "sal y pimienta" en una imagen.

La Motivación de este trabajo es la corrección de defectos en imágenes médicas, más específicamente las imágenes radiográficas y algunas tomografías que tienen frecuentemente defectos del tipo antes mencionado.

Para la simulación y análisis del sistema se ha utilizado Matlab y el toolbox IPT (toolbox de procesamiento de imágenes), dichas herramientas permiten realizar la simulación del filtro de media obteniendo resultados muy satisfactorios.

Para la implementación final se ha utilizado una FPGA de xilinx, en la cual se ha realizado mediante la creación de 3 módulos principales, (generador de ventanas, ordenador de píxeles y contador de filas y columnas), un filtro de media de 9 píxeles.

Palabras Clave – Procesamiento de Imágenes Digitales; Lógica Programable; FPGA.

I. INTRODUCCIÓN

Con el avance de la tecnología las FPGAs se han convertido en una herramienta indispensable en el diseño rápido y eficiente de proyectos que requieren procesamiento digital y lógica programable. Es por eso que se ha convertido en la principal herramienta para la implementación del algoritmo de procesamiento de imágenes digitales realizado en este trabajo [1].

El Filtrado espacial no lineal permite mediante el filtro de mediana solucionar defectos en imágenes que dificultan o confunden al observador en el momento del análisis visual. Es por esto que este filtro es muy útil para la solución de defectos en imágenes médicas del tipo radiográfico.

Para la implementación del filtro, previa simulación en matlab, se realizó una interfaz serial que permite enviar los pixeles a procesar en la FPGA y luego del procesamiento mediante tres bloques que implementan el algoritmo, recuperar la imagen filtrada y acondicionada.

II. DESCRIPCIÓN DEL FILTRADO ESPACIAL NO LINEAL

El filtrado es un tipo de operación que altera el valor de un píxel en función de los valores de los píxeles que lo rodean, es por ello que a este tipo de procesamiento de imágenes también se le denomina procesamiento basado en la vecindad u operación de vecindad. El término "espacial" se utiliza para distinguir que la alteración del píxel se realiza dependiendo de los valores de los píxeles del entorno sin realizar ninguna modificación previa de sus valores [2].

Cuando se aplica un filtro no lineal se sustituye el píxel central por el resultado de aplicar una función no lineal que depende de los píxeles de la vecindad. Su aplicación fundamental es para suprimir el ruido de la imagen, haciendo que los puntos con niveles de gris distintos sean más parecidos a los de su vecindario.

Cuando la degradación presente es solo ruido, entonces se describe el modelo

$$g(x,y)=f(x,y) + ?(x,y) \quad (1)$$

El método usado para reducción de ruido en este caso es filtrado espacial. El filtro de mediana se implementa mediante una ventana de pixeles que se va desplazando a los largo de la imagen como se muestra en la figura 1.

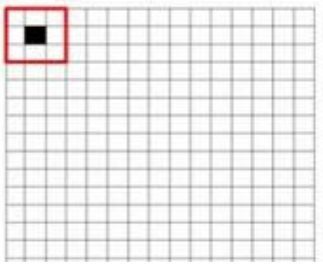


Figura 1. Ventana de 9 pixeles

En cada ventana se realiza el ordenamiento (ranking) de los píxeles y se reemplaza el valor del píxel central llamado "origen" por el valor medio determinado por el resultado del ordenamiento [2]. En la figura 2 se muestra una ventana de nueve píxeles, en donde luego del ordenamiento de los mismos, se toma como salida el pixel que se encuentra en la posición 5, ya que este es un filtro de ordenamiento por ranking de orden 5 y de esta manera se implementa el filtro de mediana.

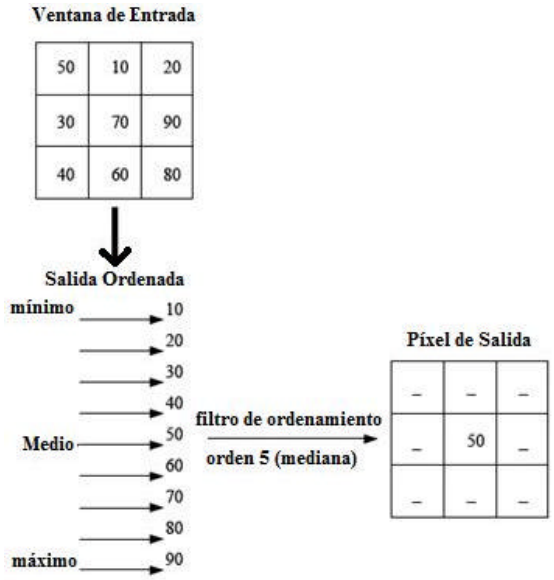


Figura 2. Implementación del Filtrado de mediana

III. SIMULACIÓN EN MATLAB DEL FILTRO DE MEDIANA

La simulación del algoritmo de mediana se realizó en Matlab [3], mediante la lectura de una imagen de 512x512 píxeles con ruido del tipo "sal y pimienta" como se muestra en la figura 3. En dicha figura, los píxeles de color blanco se denominan ruido sal y tiene una intensidad de 00 y los pixeles de color negro se identifican como ruido pimienta con una intensidad de 255.

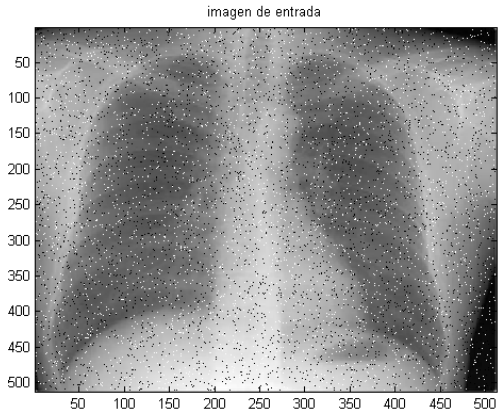


Figura 3. Imagen radiográfica con ruido

El filtrado en matlab consiste en la implementación de una ventana móvil de 9 píxeles y una función de ordenamiento "sort" que permita encontrar el píxel de valor medio, el cual formará parte de la imagen de salida [4],[5] y [6].

La Figura 4 muestra la imagen de salida del filtro sin indicios de ningún tipo de ruido.

Figura 4. Resultado de la implementación del filtro en Matlab.

IV. DESCRIPCIÓN DEL SISTEMA

El sistema está en la FPGA está formado 3 módulos principales que realizan el filtrado de la imagen y un módulo de comunicación serial para la adquisición y el envío de datos procesados [7].

El filtro de mediana en VHDL está formado por un módulo "generador de ventanas", el cual genera ventanas de 3x3 píxeles, un módulo de "ordenamiento de píxeles" que ordena los píxeles según su intensidad (su valor puede estar entre 0 y 255). Por último, el módulo contador de filas y columnas que tiene por objeto determinar cuáles son los píxeles que se encuentran en los bordes de la imagen y asignarles el valor 0, ya que estos píxeles no pueden ser evaluados por no tener información suficiente para formar la ventana de 3x3.

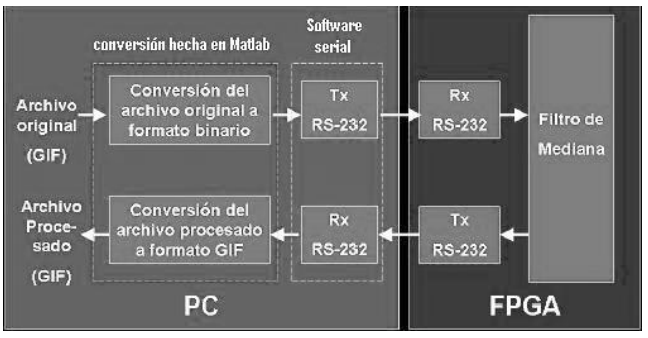


Figura 5. Diagrama en Bloques del Sistema completo

El la Figura 5 se muestra el diagrama en bloques del sistema completo, en donde la imagen se convierte del formato Gif a binario mediante el programa gif_a_bin.m, luego es enviado a la FPGA mediante un software de comunicación serial, en donde se procesa y se envía dicha imagen nuevamente a la PC vía RS-232. Por último, mediante el

programa bin_a_gif.m se transforma la imagen procesada al formato original.

A. Módulo de Comunicación Serial

El módulo serial implementado en VHDL consiste de un Bloque "generador de baud rate", el cual se implementa mediante un contador que genera una señal de muestreo cuya frecuencia es 16 veces el baud rate seleccionado para la UART. Está señal es utilizada por el bloque de recepción y transmisión serial para detectar los datos de entrada y generar los datos de salida, respectivamente [8].

El modulo Receptor se implementa mediante una máquina de estados que recibe los bits de datos y los envía a un buffer FIFO en donde son almacenados de a 1 byte. La salida del buffer FIFO entrega un Byte de datos en forma paralela, para ser procesado por el sistema.

Por último, el bloque transmisor funciona del mismo modo que el bloque receptor, solo que recibe los datos de a 1 byte en forma paralela y los transmite de forma serial. La Figura 6 muestra el diagrama en bloques completo de una UART implementada en Hardware [8].

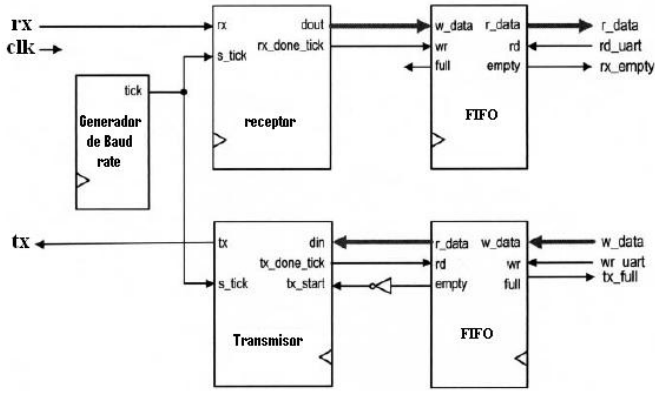


Figura 6. Diagrama en Bloques del módulo UART

B. Generador de Ventanas

Los filtros de mediana forman parte del procesamiento de imágenes basado en regiones. Para realizar este tipo de procesamiento es necesario seleccionar los píxeles correspondientes a la región de interés. Esto se hace mediante un "generador de ventanas"[9].

Figura 7. Arquitectura del Generado de Ventanas

El "generador de ventanas" es un módulo diseñado en VHDL para la selección de los píxeles de la región a procesar. La Figura 7 muestra como se implementa en hardware dicho generador mediante flip flops y 2 buffers FIFO [10]. También se muestran las salidas de los 9 píxeles de la ventana a ser procesados por el bloque de ordenamiento de píxeles.

Cabe aclara que los Bloques FIFO fueron diseñados utilizando una herramienta de Xilinx muy potente llamada CORE GENERATOR que permite personalizar el diseño de una FIFO de manera rápida. En este caso se seleccionó una entrada de 8 bit y 512 bytes de almacenamiento implementados en RAM distribuida.

C. Ordenador de Píxeles

El filtro de mediana es un subconjunto que pertenece a los filtros de orden estadístico [9]. La característica principal de estos filtros es que requieren el ordenamiento de menor a mayor (o viceversa) del valor de los píxeles involucrados en la operación. Este bloque se realiza en VHDL para obtener el ordenamiento de los píxeles de la ventana de interés.

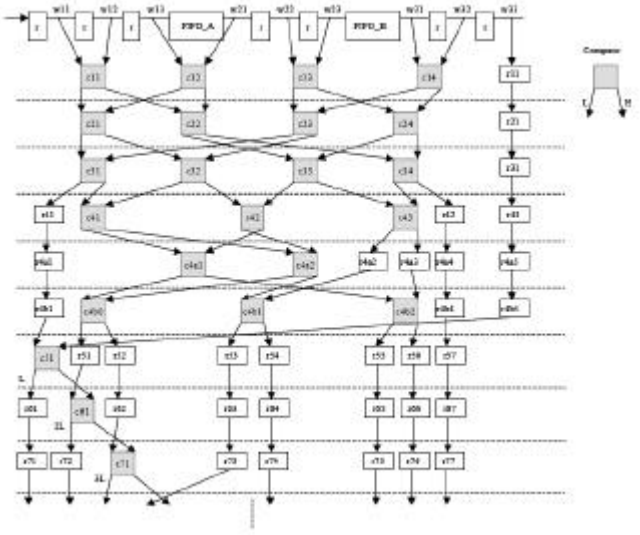


Figura 8. Diseño de Hardware del algoritmo de ordenamiento

Como muestra la figura 8, el bloque de ordenamiento se implementa mediante comparadores que reciben como entradas grupos de 2 píxeles para determinar cuál es mayor y cual es menor. De esta manera y mediante varios ciclos de clock se obtienen los 9 píxeles ordenados. De los cuales se selecciona como salida el de la posición 5, esto es así porque el filtro debe ser de orden intermedio para ser de mediana [10] y [11].

D. Contador de Filas y Columnas

Cada pixel de salida del filtro es asignado a la posición correspondiente del pixel que se encuentra en el centro de la ventana procesada. Por lo tanto, si se pretendiese conocer este valor en los bordes de la imagen, se requerirían valores que no están disponibles por tratarse del borde. En otras palabras para conocer los valores de la mediana en los bordes sería necesario conocer los pixeles adyacentes a los mismos. El contador de filas y columnas indica si el pixel de salida se encuentra en los bordes de la imagen, para poder asignarle un valor nulo al pixel [11].

La implementación en Hardware de dicho contador tiene en cuenta los píxeles correspondientes a la primer y última fila y los correspondientes a la primer y última columna de una

imagen de 512x512. A dichos píxeles les asigna el valor 0 y los envía a la salida.

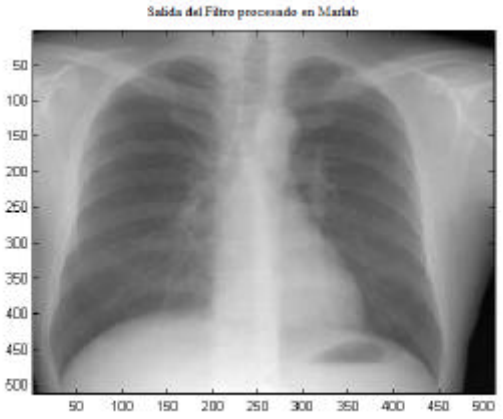
V. RESULTADO OBTENIDOS

En la Figura 9 se muestran los recursos utilizados en la FPGA para la realización del sistema. Se debe tener en cuenta que se ha implementado un sistema completo que procesa imágenes de 512x512 píxeles por lo que es de esperarse que se utilicen una gran cantidad de recursos. De cualquier manera la implementación solo utilizó el 51 por ciento de los slices. El 31 por ciento de flop flops y el 59 por ciento de las 4 input LUTs [12]. Se puede optimizar el uso de recursos mediante el diseño en VHDL personalizado de los bloques FIFO, ya que dichos bloques fueron diseñados por una herramienta que provee Xilinx llamada Core Generator [8].

Device Utilization Summary (estimated values)			
Logic Utilization	Used	Available	Utilization
Number of Slices	980	1920	51%
Number of Slice Flip Flops	1223	3840	31%
Number of 4 input LUTs	2268	3840	59%
Number of bonded IOBs	19	173	10%
Number of GCLKs	1	8	12%

Figura 9. Reporte de recursos empleados por la FPGA

En la Figura 10(a) se muestra la imagen filtrada mediante la simulación en Matlab y en la Figura 10 (b) se muestra la imagen procesada en la FPGA, a simple vista se puede decir que son similares. Para verificar dicha apreciación se realizó la diferencia entre las dos imágenes filtradas y se comprobó que el error es nulo, dicho resultado se puede observar en la Figura 11.



(a) Imagen procesada en Matlab

(b) Imagen Procesada en la FPGA

Figura 10. Comparación entre el filtrado hecho en Matlab y en la FPGA

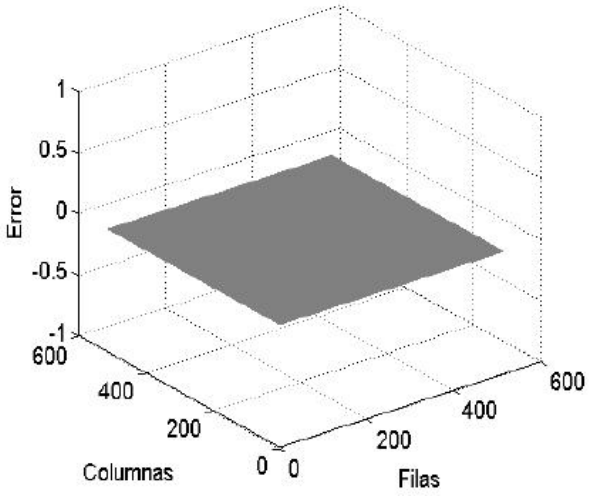


Figura 11. Cálculo de error mediante la diferencia entre la salida en Matlab y en la FPGA

VI. CONCLUSIONES

Se Puede concluir que La FPGA es una herramienta muy potente para el procesamiento de imágenes, ya que mediante una Spartan 3 con las mínimas prestaciones se puede implementar un algoritmo de procesamiento de imágenes para la corrección de errores que requiere mucho procesamiento y en imágenes de un tamaño aceptable de 512x512 pixeles. Se debe tener en cuenta que solo se están utilizando la mitrad de los recursos de HW. De cualquier manera sería conveniente utilizar una FPGA más potente para imágenes de mayor tamaño o para una ventana de procesamiento mayor a la utilizada de 3x3 píxeles.

También se pudo comprobar el buen funcionamiento del filtro en HW mediante la comparación de la imagen obtenida con una procesada en Matlab.

VII. TRABAJO A FUTURO

Como trabajo a futuro se pretende implementar otros algoritmos de Procesamiento de Imágenes, como la convolución y operadores morfológicos, los cuales tienen en común el generador de ventanas que utilizan los filtros de ordenamiento por rango.

También se pretende implementar una salida VGA para mostrar los resultados directamente en pantalla.

Por otro lado, se pretende implementar algoritmos más complejos como la transformada wavelet y el filtro de sobel [13], mediante FPGAs (Spartan 6 o Virtex 5) más potentes con Módulos DSPs y otras características que permitan operaciones a muy altas velocidades.

REFERENCIAS

[1] A. Domingo Ajenjo, "Tratamiento digital de imágenes", Anaya, Madrid, 1993.

[2] Rafael C. González, Richard E. Woods, "Digital Image Processing", Addison-Wesley, Massachusetts, 1992.

[3] Banerjee, N., et. al.: "MATCH: A MATLAB Compiler for Configurable Computing Syst ems,"

[4] Rafael C. González, Richard E. Woods, Steven L. Eddins, "Digital Image Processing using Matlab", Prentice-Hall, New Jersey, 2004.

[5] www.prenhall.com/gonzalezwoods

[6] www.prenhall.com/gonzalezwoodseddins

[7] Nelson, A.: "An Implementation of t he Optical Flow Algorithm on FPGA Hardware," Independent Study Paper, December 1998.

[8] Pong P. Chu, "FPGA Prototyping by VHDL Examples",Xilinx Spartan 3 Version, John Wiley Cleveland

[9] Nelson, A.: "Further Study of Image Processing Techniques on FPGA Hardware," Independent Study Paper, May 1999.

[10] The Designer's Guide to VHDL 2nd Edition, Peter J. Ashenden, Editorial Morgan Kaufman, 2002

[11] Chou, C., Mohanakrishnan, S., Evans, J.: "FPGA Implementation of Digital Filters," Proc. ICSPAT, 1993.

[12] Diseño Digital Principios y Prácticas Tercera Edición, John F. Wakerly, Editorial Prentice Hall, 2001

[13] Russ, J.: "The Image Processing Handbook," CRC Press, Boca Raton, FL, 1992.