Desarrollo de Algoritmos de Procesamiento de Imágenes Basados en "Operadores de Ventana" sobre una FPGA

Osio Jorge R., Aróztegui Walter, Rapallini José A., Quijano Antonio A.
Centro de técnicas Analógico y Digitales, CeTAD, Depto. de Electrotecnia, Fac. de Ingeniería, UNLP. Calle 48 y 116 2º Piso, Jorge.osio@ing.unlp.edu.ar
Palabras Clave: Procesamiento Digital de Imágenes; Lógica Programable; Sistemas Embebidos.

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 de los algoritmos de procesamiento Digital de imágenes presentados en este trabajo [1].

Debido a la gran cantidad de algoritmos para procesamiento de imágenes basados en los Operadores de Ventana, se llega a la necesidad de bloques reusables que permitan implementar los diferentes algoritmos basados en dichos operadores. En base a dichos bloques y mediante el agregado de un bloque adicional específico de cada algoritmo de procesamiento se puede implementar de manera eficiente y con un mínimo desarrollo una diversidad de filtros para procesamiento de imágenes que permiten el realce, acondicionamiento y la obtención de información de las diferentes imágenes en el campo de la Medicina.

Para la implementación del sistema basado en Operadores Morfológicos, previa simulación en matlab [3] [4], 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 procesada y acondicionada.

PARTE EXPERIMENTAL

En procesamiento de imágenes [2] hay varios algoritmos que se pueden encasillar en la categoría operadores de ventana. Los operadores de ventaneo usan una ventana, conjunto de píxeles, para calcular su salida. Por ejemplo el operador de ventaneo puede desarrollar una operación calculando el promedio de todos los pixeles en el entorno de un píxel. El píxel que se encuentra en el centro de la ventana se denomina original.

En este trabajo los algoritmos a implementar se basan en la utilización de ventanas de píxeles dentro de la imagen, para realiza el procesamientos de los mismos y obtener la salida correspondiente. Los algoritmos basados en operadores de ventana se pueden dividir en 3 etapas principales. Una etapa de generación de ventana, un contador de filas y columnas y la etapa final específica del algoritmo a implementar. El objetivo del Generador de Ventanas es almacenar en un buffer la cantidad de píxeles necesarios para formar una ventana de nxn, (en donde n define el tamaño de la ventana). Luego de esto, el generador de ventana deberá ir realizando un desplazamiento de la ventana a lo largo de toda la imagen hasta lograr el procesamiento de todos los píxeles. Luego, el contador de filas y columnas se encargará de determinar cuáles son los píxeles de los bordes de la imagen, esto se hace porque en los bordes de la imagen no se puede aplicar el procesamiento de ventana por falta de información.

Dentro de los algoritmos que utilizan estos operadores se desarrollarán los filtros de mediana, (muy útiles para eliminar el ruido sal y pimienta), los basados en operadores morfológicos y la convolución espacial.

- **Operadores Morfológicos**

El término de Operadores Morfológicos en procesamiento de imágenes se refiere a la clase de algoritmos que están interesados en la estructura geométrica de una imagen. La morfología puede ser usada sobre imágenes binarias o en escala de grises, y usada en muchas áreas de procesamiento de imágenes, tales como esqueletización, detección de bordes, restauración y análisis de textura.

Hay 2 operaciones fundamentales en morfología: erosión y dilatación.
Es común pensar erosión como la contracción de un objeto en una imagen. Dilatación es lo opuesto, agranda el objeto en la imagen. Ambos conceptos dependen del elemento estructurado y de cómo se ajusta en el objeto. La salida de una operación de dilatación es un píxel en primer plano para todos los puntos en el elemento estructurado a tal punto que el píxel original se ajusta en un objeto imagen.

La erosión y dilatación pueden ser representadas matemáticamente mediante las siguientes relaciones:

Erosión: $A \ominus B = \{x: B + x < A\}$ y
Dilatación: $A \oplus B = U \{A + b: b \in B\}$, Donde A es la imagen de entrada y B es el elemento estructurado.

La erosión y dilatación en escala de grises puede ser obtenida también mediante un filtro de ordenamiento por rango. La erosión corresponde a un filtro de ordenamiento por rango de orden mínimo, y la dilatación corresponde a un filtro de ordenamiento por rango de orden máximo. La razón de esto es que el resultado de un filtro de ordenamiento por rango de orden mínimo es el valor mínimo en la zona del píxel, que es exactamente lo que una operación de erosión hace. Esto también sigue siendo válido para un filtro de ordenamiento por rango de orden máximo y la operación de dilatación. De cualquier manera, el filtro de ordenamiento por rango solo trabaja como una operación morfológica con un elemento estructurado plano. Esto es así porque el filtro de ordenamiento por rango trabaja como una clase de elemento estructurado formada por todos unos. Aun así, esta es una característica poderosa, ya que la morfología en escala de grises usando elementos estructurados planos describe los usos más comunes de la morfología.

- **Filtros de Ordenamiento por Rango**

Dentro de los filtros de Ordenamiento por rango se encuentra el filtro de mediana que se implementa mediante una ventana de pixeles que se va desplazando a los largo de la imagen.

En cada desplazamiento 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 1 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.

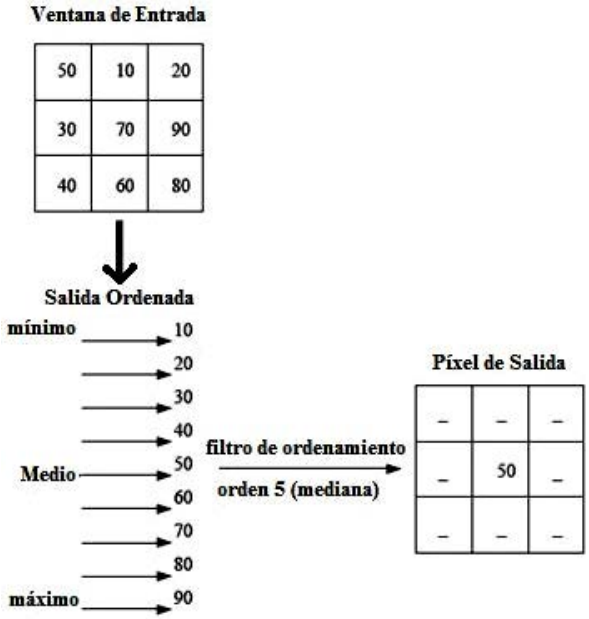


Figura 1. Filtro de Mediana.

De manera similar pero tomando el píxel de mayor intensidad, se obtiene el filtro de máxima el cual permite realizar la erosión que describen los operadores morfológicos. Tomando el píxel de menor intensidad se obtendrá el filtro de mínima que permite implementar la dilatación que también forma parte de dichos operadores.

- **Convolución Espacial**

Los filtros espaciales usan una amplia variedad de máscaras, también conocidas como kernels, para calcular diferentes resultados, dependiendo de la función deseada. Por ejemplo, ciertas mascaras implementan el alisado, mientras otras realizan el filtrado pasa bajos o detección de bordes. La convolución se puede calcular multiplicando cada píxel de una ventana por la mascara (kernel), para luego realizar la suma de cada producto y por último dividir por el número de pixeles de la ventana. Este valor es el píxel de salida en la ubicación del píxel original en la imagen de salida. Matemáticamente la convolución se representa mediante la siguiente ecuación:

$y(n_1,n_2)=\sum_{k_1=-\infty}^{\infty}\sum_{k_2=-\infty}^{\infty}A(k_1,k_2)k(n_1-k_1,n_2-k_2)$, Donde A es la imagen de entrada y K es la máscara de la convolución.

La ventana del pixel de entrada es siempre del mismo tamaño que la máscara de convolución. El pixel de salida es redondeado al entero más cercano. La figura siguiente muestra la ventana de pixeles de entrada, la máscara de convolución, y el resultado de la salida. La máscara de convolución que se muestra en la Figura siguiente es frecuentemente usada como filtro para eliminar ruido.

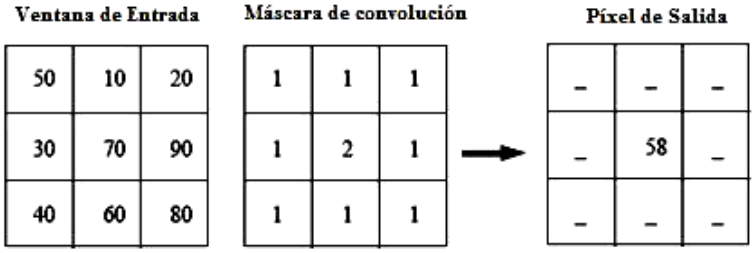


Figura 2. Filtro de la Convolución.

- **Descripción del Sistema**

El sistema en la FPGA está formado por 3 módulos principales; el Generador de Ventanas, el Contador de Filas y Columnas y el Módulo que implementa el algoritmo deseado. Por otro lado, el sistema consta de un módulo de comunicación serial para la adquisición y el envío de datos procesados [5].

El filtro de mediana, de Máxima y de Mínima se implementó en VHDL mediante 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 y un módulo contador de filas y columnas. Por otro lado, El algoritmo de la Convolución Espacial contiene el módulo de generación de ventanas, el contador de filas y columnas y un bloque específico que realiza la convolución entre cada una de las ventanas y la máscara a aplicar.

El la Figura 3 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.

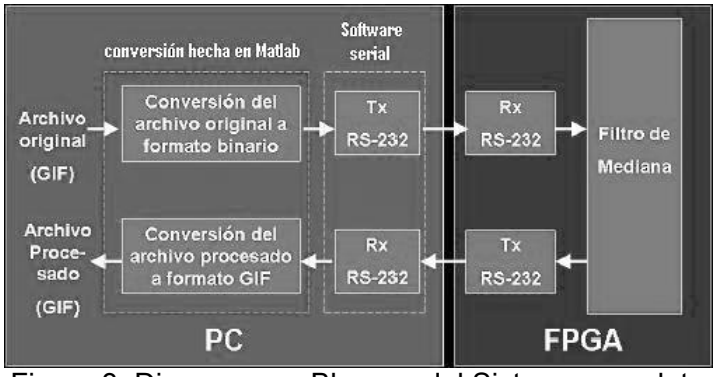


Figura 3. Diagrama en Bloques del Sistema completo

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 [6].

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. 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.

B. Generador de Ventanas

Los filtros de de ordenamiento por rango 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 [7].

El generador de ventanas es un módulo diseñado en VHDL para la selección de los píxeles de la región a procesar. La Figura 4 muestra como se implementa en hardware dicho

generador mediante flip flops y 2 buffers FIFO [8]. También se muestran las salidas de los 9 píxeles de la ventana a ser procesados por el bloque de ordenamiento de píxeles.

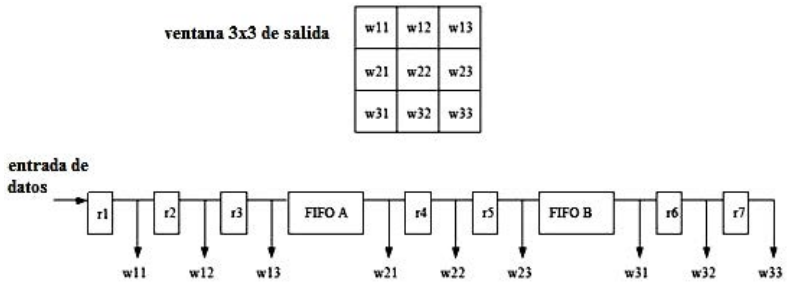


Figura 4. Arquitectura del Generado de Ventanas.

Cabe aclarar 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. 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 [9].

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.

D. Ordenador de Píxeles

El filtro de mediana es un subconjunto que pertenece a los filtros de orden estadístico (ordenamiento por rango) [7]. 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.

Como muestra la Figura 5, 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 [8].

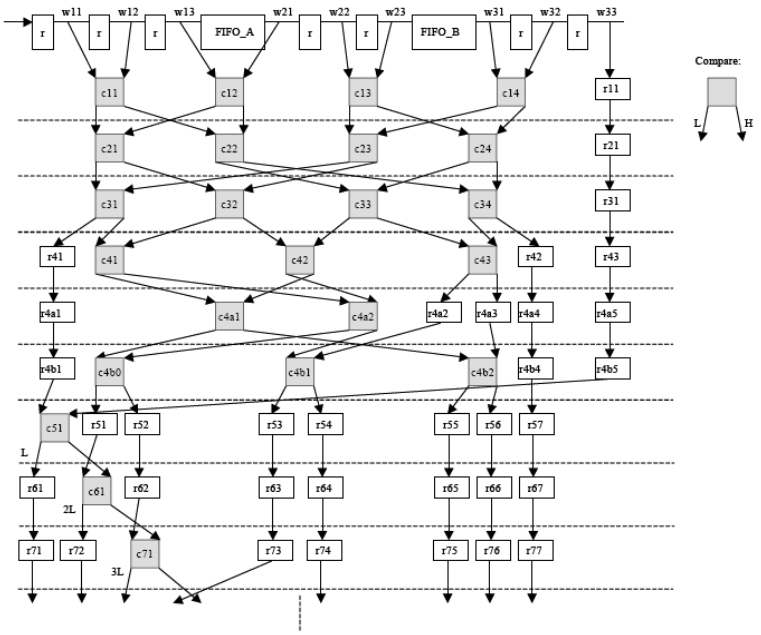


Figura 5. Diseño de Hardware del algoritmo de ordenamiento.

E. Convolución Espacial

El módulo de convolución espacial realiza el producto entre cada ventana y la máscara píxel a píxel, para luego sumar cada resultado y dividirlo por la cantidad de píxeles de la ventana. En la Figura 6 se muestra el diagrama de HW del bloque que realiza la convolución. Cabe aclarar que se realizo una división por 8 mediante desplazamientos, para evitar la utilización de excesivos recursos y la operación de redondeo.

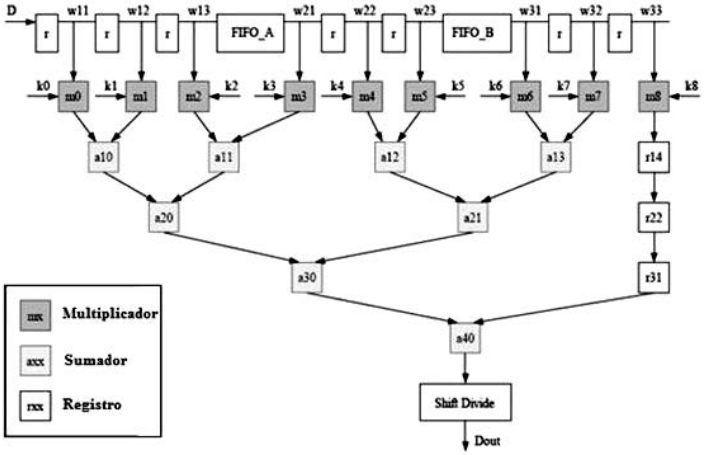


Figura 6. Diseño de Hardware del algoritmo de Convolución.

RESULTADOS Y DISCUSIÓN

En las Figuras 7, 8 y 9 se muestra los tesbench respectivos del Generador de Ventanas, el algoritmo de convolución espacial y el algoritmo de ordenamiento (utilizado para los filtros de ordenamiento por rango).

Las Figuras 10, 11 muestran los recursos utilizados en la FPGA para la implementación del bloque generador de ventana, de convolución y de ordenamiento, respectivamente.

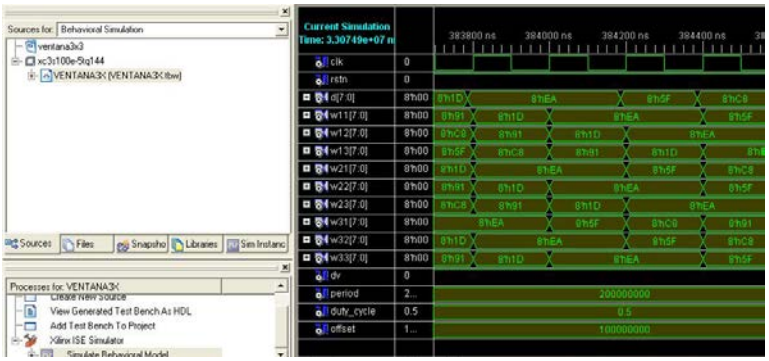


Figura 7. Testbench del Generador de Ventanas.

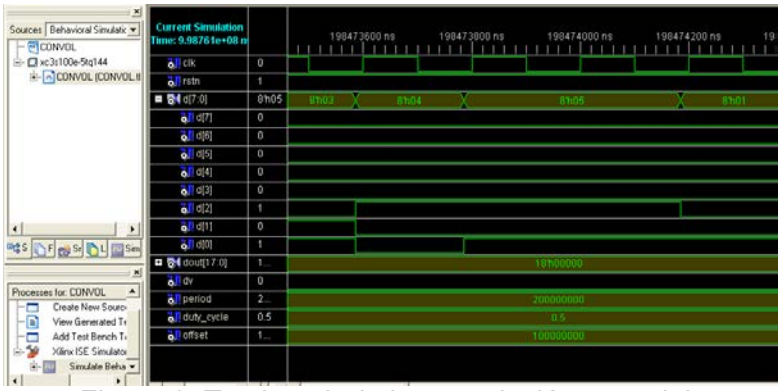


Figura 8. Testbench de la convolución espacial.

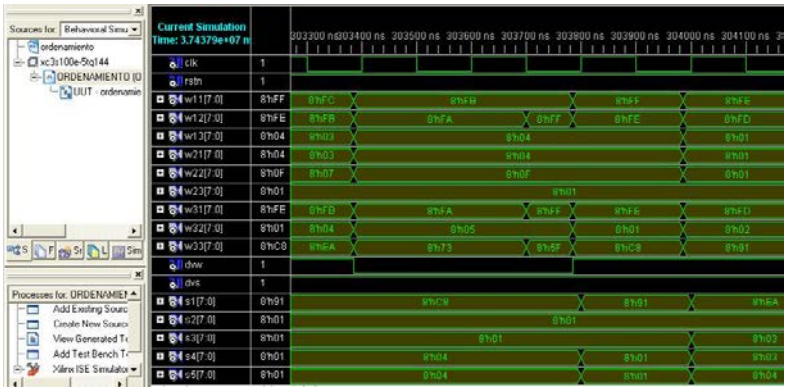


Figura 9. Testbench del algoritmo de ordenamiento

La implementación del Generador de Ventanas solo utilizó el 5 por ciento de los slices, como se muestra en la Figura 10. El 4 por ciento de flip flops y el 0 por ciento de las 4 input LUTs. La implementación del Ordenador de Píxeles utilizó el 50 por ciento de los slices, el 4 por ciento de flip flops y el 0 por ciento de las 4 input LUTs, como se muestra en la Figura 11.

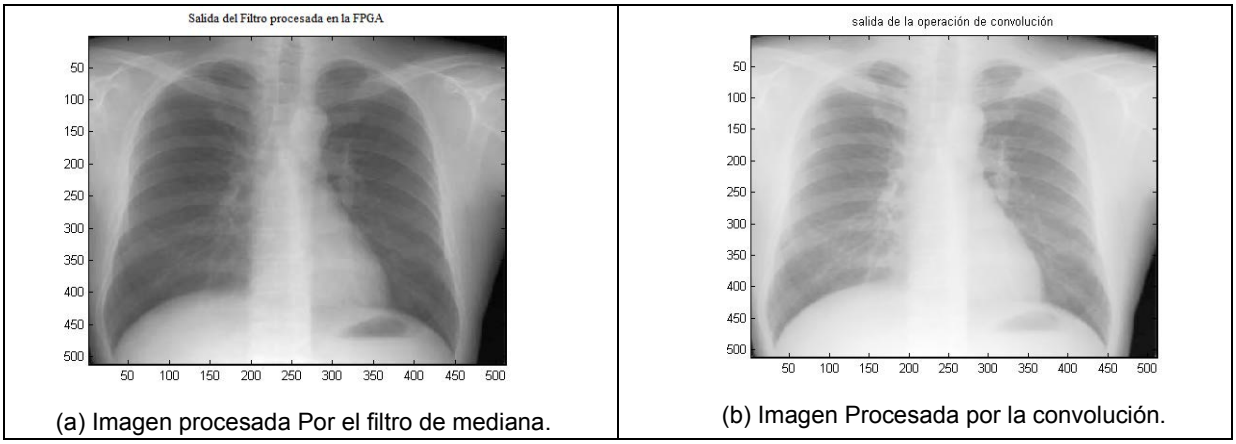
Device Utilization Summary (estimated values)			[-]
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slices	56	960	5%
Number of Slice Flip Flops	91	1920	4%
Number of 4 input LUTs	7	1920	0%
Number of bonded IOBs	83	108	76%
Number of GCLKs	1	24	4%

Figura 10. Reporte de recursos empleados para el Generador de Ventanas.

Device Utilization Summary				[-]
Logic Utilization	**Used**	**Available**	**Utilization**	**Note(s)**
Number of Slice Flip Flops	972	1,920	50%	
Number of 4 input LUTs	750	1,920	39%	
Logic Distribution				
Number of occupied Slices	619	960	64%	
Number of Slices containing only related logic	619	619	100%	
Number of Slices containing unrelated logic	0	619	0%	

Figura 11. Reporte de recursos empleados para el ordenador de píxeles.

En la Figura 12(a) se muestra la imagen procesada en la FPGA por el filtro de mediana y en la Figura 12 (b) la imagen procesada por la convolución.



(a) Imagen procesada Por el filtro de mediana.

(b) Imagen Procesada por la convolución.

Figura 12. Resultados del Procesamiento en la FPGA.

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

Bibliografía

[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, P. : —MATCH: A MATLAB Compiler for Configurable Computing Systems, Electrical and Computer Engineering, Northwestern University, 1999.
[4] Rafael C. González, Richard E. Woods, Steven L. Eddins, —Digital Image Processing using Matlab, Prentice-Hall, New Jersey, 2004.
[5] Nelson, A.: —An Implementation of t he Optical Flow Algorithm on FPGA Hardware, Independent Study Paper, December 1998.
[6] Pong P. Chu, —FPGA Prototyping by VHDL Examples, Xilinx Spartan 3 Version, John Wiley Cleveland, 2008
[7] Nelson, A.: —Further Study of Image Processing Techniques on FPGA Hardware, Independent Study Paper, May 1999.
[8] The Designer's Guide to VHDL 2nd Edition, Peter Ashenden, Editorial Morga Kaufman.