RIELAC, Vol.XXXIV 2/2013 p.9-18 Mayo - Agosto ISSN: 1815-5928

RIELAC
Revista de Ingeniería Electrónica, Automática y Comunicaciones

Diseño de bloques para el procesado de imágenes en lenguaje de descripción de hardware

Elias Augusto Perdomo Hourné[1], Luis Manuel Garcés-Socarrás[2], Alejandro José Cabrera Sarmiento[3]

RESUMEN / ***ABSTRACT***

El presente trabajo, "Diseño de bloques para el procesado de imágenes en lenguaje de descripción de hardware", desarrolla bloques de procesado de imágenes para la biblioteca de XSGImgLib, utilizando el flujo de diseño basado en lenguaje de descripción de hardware (HDL), incrementando las opciones de configuración de los algoritmos implementados en el flujo de diseño basado en modelos de XSG y la velocidad de ejecución.

Para el desarrollo del trabajo se realiza un estudio de las arquitecturas de procesado de imágenes para hardware reconfigurable utilizando lenguaje de descripción de hardware. A partir de este estudio se diseñan, comprueban e implementan bloques de procesado de imágenes para la biblioteca XSGImgLib utilizando lenguaje de descripción de hardware.

Como logros fundamentales del trabajo se obtienen bloques configurables que pueden cambiar el tamaño de la ventana en tiempo de diseño. Estos bloques no disminuyen el desempeño con respecto a los bloques anteriores si no que en muchos casos lo mejoran.

Palabras claves: Procesado de imágenes, XSGImgLib, HDL, FPGA.

The present paper, "Design of Processing Image Blocks in Hardware Description Language", develop images processing blocks for the XSGImgLib library, using the model design flow based on hardware description language (HDL), increasing the configuration options of the implemented algorithm and the execution speed.

In order to develop this project an study of images processing architectures for FPGA using hardware description language was carried out. As a result of this study were designed, tested and implemented processing images blocks for the XSGImgLib library using hardware description language.

As main result of the project was obtained settable blocks that can change the Windows size in design time. Compared with the XSGImgLib previous blocks, in many cases the behavior is improved.

Key words: Image Processing, XSGImgLib, HDL, FPGA.

Design of Processing Image Blocks in Hardware Description Language

INTRODUCCION

Los algoritmos de procesamiento digital de imágenes (DIP) presentan, en general, una gran carga computacional e imponen serias restricciones temporales para asegurar la operación en tiempo real. Por este motivo la inclusión de algoritmos de procesado de visión en sistemas empotrados con recursos de cálculo limitados, requiere la aplicación de métodos que permitan acelerar su ejecución [1], [2]. La implementación sobre hardware reconfigurable de estos algoritmos surge de la necesidad de cumplir con una serie de requisitos, entre ellos: velocidad de procesamiento, flexibilidad, costo, fiabilidad y tiempo de desarrollo [3], [4].

La utilización de estos dispositivos para la implementación de algoritmos de procesado de imágenes, permite aumentar la velocidad de ejecución con respecto a las soluciones software. No obstante, su uso suele estar condicionado por la disponibilidad de recursos en el dispositivo. La estructura de arreglos de compuertas y registros en paralelo de los arreglos de compuertas programables (FPGA) hace de las mismas una opción viable para explotar el paralelismo de datos de las imágenes o tramas de vídeos. Estas pueden ser utilizadas para realizar operaciones completas o para pre-procesar los datos antes de enviarlos a un Procesador digital de señales (DSP) estándar o a un microprocesador [5].

Un diseño para FPGA presenta varias opciones para la implementación del algoritmo, producto de la gran variedad de lenguajes existentes en el mercado. Estos lenguajes en su mayoría no permiten la portabilidad de una arquitectura a otra, por lo tanto se hace necesario escoger bien el lenguaje de di-seño de hardware (HDL) en la que los algoritmos FPGA van a ser diseñados.

En los últimos años, el lenguaje de diseño de hardware VHSIC (Very High Speed Integrated Circuit) (VHDL), se ha convertido en una especie de están-dar de la industria para el diseño de hardware de alto nivel. Dado que es un estándar abierto de la norma IEEE, es apoyado por una gran variedad de herramientas de diseño El desarrollo de código VHDL tiene la ventaja de ser en gran medida independiente del dispositivo, lo que significa que la mayor parte del código puede ser compilado para cualquier dispositivo.

El VHDL permite crear nuevos bloques que no existen en las librerías de XSG para que se adapte a las necesidades del usuario y dado que permite la reutilización de código puede configurarse estas arquitecturas en tiempo de diseño, lo cual es imposible en XSG, para la inclusión del código VHDL se usa el bloque Black Box de Matlab, este modelo reproduce las entradas y las salidas descritas en el código para que interactúen con el resto del flujo basado en modelos que posibilita el Matlab.

En este artículo se describe el empleo de técnicas de diseño basadas en HDL, en combinación con la herramienta XSG, para evaluar distintas opciones de implementación de bloques configurables de procesamiento de imágenes y se analizan los efectos (en cuanto a recursos consumidos y velocidad de operación) relacionados con la configuración de los distintos parámetros de diseño.

ARQUITECTURAS DE PROCESADO DE IMÁGENES

El filtrado espacial basado en ventana es una operación espacial en el procesado digital de imagen, en el cual se modifica el valor de cada píxel de acuerdo a los píxeles que lo rodean; se trata de transformar los niveles originales de tal forma que se parezcan o diferencien más de los correspondientes a los píxeles cercanos. Estas operaciones se clasifican de acuerdo a la linealidad de los algo-ritmos a aplicar en filtros de procesado lineal y filtros de procesado no lineal.

El procesado lineal de imágenes es todo aquel algoritmo de procesamiento que utiliza una transformación lineal para lograr la imagen resultante. La convolución de dos dimensiones es la operación característica de este procesamiento.

El filtrado espacial no lineal permite reemplazar el píxel a procesar por el resultado de una operación no lineal. La operación característica de este tipo de procesado es el ordenamiento, aunque otros tipos de procesados no espaciales (exponencial y logarítmico) se incluyen en esta categoría.

1.1. FILTRO DE PROCESADO LINEAL

La técnica de convolución permite diversos resultados en el procesamiento de imágenes, ya que con esta se realizan operaciones tales como la detección de bordes, el realce y el suavizado de la imagen. La implementación de este tipo de algoritmo sobre hardware reconfigurable ha encontrado aplicaciones prácticas en diferentes áreas, siendo una necesidad común la rapidez en la ejecución de los mismos [2], [3], [6], [7], [8].

1.1.1. CONVOLUCIÓN DE IMÁGENES

La convolución es un tipo de morfología que surge antes que la misma, y genera más efectos de gradientes en escala de grises, en lugar de los efectos de forma binaria que la morfología típicamente genera. Esto se debe a que, a menudo, está presentada como una operación muy distinta a la morfología y una que es fundamental para el procesamiento de imágenes. Muchas de las etapas de un sistema de procesado lineal de imágenes se basan en la operación matemática de convolución de la imagen, representada por una matriz de dos dimensiones, con otra matriz (de 3×3 ó 5×5 píxeles en la mayoría de los casos) que es lo que se denomina núcleo o kernel de convolución [7].

La operación de convolución en dos dimensiones realiza un promedio del valor de todos los píxeles en el vecindario especificado. Es decir multiplica el valor de cada píxel cercano por la cantidad dada en el kernel y luego suma el resultado de las operaciones anteriores para producir el píxel modificado. Así, cada píxel en la imagen final contiene una pequeña parte de los píxeles alrededor de este.

La función A de una imagen de $(m \times n)$ píxeles, y la matriz de convolución C de $(p + 1) \times (q + 1)$ elementos (siendo normalmente $p = q = 3, 5, 7$ ó 9), dado en la Ecuación 1, define la convolución, A_c entre la imagen A y el kernel C mediante la operación matemática mostrada en la Ecuación 2, que se traduce en la sumatoria de la multiplicación de todos los píxeles de la imagen con el correspondiente valor del coeficiente del kernel rotado 180° [6].

$$C = \begin{bmatrix} C_{-\frac{p}{2},-\frac{q}{2}} & C_{-\frac{p}{2},-\frac{q}{2}+1} & \cdots & C_{-\frac{p}{2},\frac{q}{2}} \\ C_{-\frac{p}{2}+1,-\frac{q}{2}} & C_{-\frac{p}{2}+1,-\frac{q}{2}+1} & \cdots & C_{-\frac{p}{2}+1,\frac{q}{2}} \\ \vdots & \vdots & \ddots & \vdots \\ C_{\frac{p}{2},-\frac{q}{2}} & C_{\frac{p}{2},-\frac{q}{2}+1} & \cdots & C_{\frac{p}{2},\frac{q}{2}} \end{bmatrix} \quad (1)$$

$$A_c(x,y) = \sum_{i=-\frac{p}{2}}^{\frac{p}{2}} \sum_{j=-\frac{q}{2}}^{\frac{q}{2}} C(i,j) A(x-i, y-j) \quad (2)$$

La descripción del algoritmo de convolución, puede resumirse en ubicar el centro del kernel sobre un elemento de la imagen de entrada a procesar. Luego multiplicar cada elemento del kernel con el valor correspondiente en la ventana de píxeles de la imagen de entrada y posteriormente sumar el resultado de las multiplicaciones, y por último ubicar el resultado de la suma en el elemento correspondiente a la función bidimensional que forma la imagen de salida.

El kernel se va desplazando por la imagen, obteniendo como resultado una imagen de salida que es válida en todos los lugares excepto en los bordes de la imagen donde no se tiene información suficiente para realizar el cálculo.

La operación de convolución realiza operaciones de acuerdo al kernel utilizado. Estas operaciones se clasifican en detección de bordes, suavizado y realce de detalles, como muestra la

Figura 1. **Procesado con diferentes núcleos de convolución.**

Figura 1. Procesado con diferentes núcleos de convolución.

1.1.2. ARQUITECTURA HARDWARE DEL BLOQUE DE CONVOLUCIÓN

El algoritmo general de procesado lineal de imágenes es la convolución no simétrica. La estructura interna de este bloque de convolución se compone de dos bloques principales: un bloque de paralelización de la información y uno para el procesado de la misma, donde además se realizan otras funciones como el cálculo del valor absoluto.

La arquitectura general del bloque de convolución se muestra en la

Figura **2. Arquitectura de convolución no simétrica** Este está basado en la definición matemática de la función de convolución de dos dimensiones, calculando cada píxel de la imagen. Como se muestra en la

Figura **2. Arquitectura de convolución no simétrica**, luego del proceso de paralelización del flujo de datos de la imagen, se obtiene la información paralela necesaria para el cálculo del valor del píxel resultante. Después de este tiempo los $(p + 1) \times (q + 1)$ píxeles se multiplican por los valores correspondientes del kernel rotado $180°$, obteniendo el valor para la posición central de la ventana, este valor puede ser tanto positivo como negativo por lo que resulta necesario hallar el valor absoluto. Al final de la operación del cálculo del valor absoluto del píxel se realiza la conversión a valores enteros de n bits para la representación de una imagen en escala de grises con 256 niveles truncándose la parte decimal del valor. En la próxima entrada, las estructuras de paralelización aseguran el movimiento de la ventana sobre la imagen, permitiendo el cálculo del valor para el píxel siguiente.

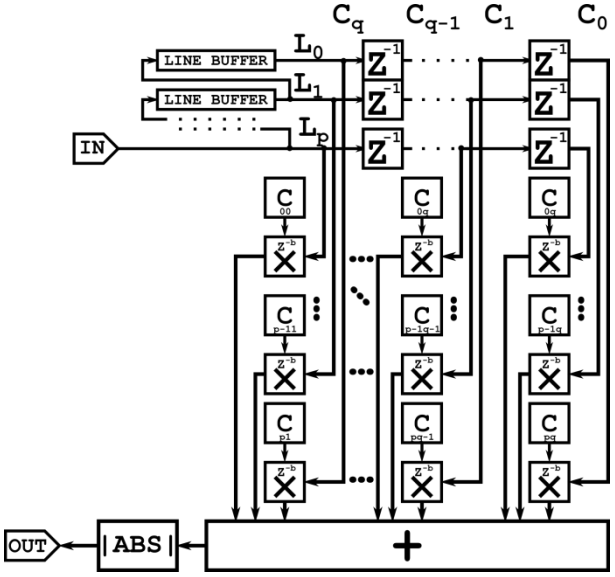

Figura 2. Arquitectura de convolución no simétrica. [9]

El funcionamiento del sistema presenta una latencia (τ_s), dada por el tiempo requerido para completar los almacenadores de línea (τ_{lb}) y ocupar los registros (τ_{rg}), así como la demora interna de los multiplicadores (τ_{op}) como muestra la Ecuación 6.

$$\tau_s = \tau_{lb} + \tau_{rg} + \tau_{op} \qquad (6)$$

La implementación en leguaje de descripción de hardware se realizó en VHDL 200X, el cual limita a que los arreglos sean de tipo constrained, es decir la cantidad de elementos del arreglo puede ser variable pero el tamaño de estos tiene que estar predefinido. Esto provoca que para emplear arreglos sea necesario definir el tamaño del mismo como el máximo permitido y usar de este los bits válidos, lo que aumenta el consumo de recursos y disminuye la frecuencia máxima de trabajo.

ARQUITECTURA HARDWARE DEL BLOQUE DE PARALELIZACIÓN DE LA INFORMACIÓN

El bloque de paralelización de la información es el encargado de que los píxeles en la ventana de procesado se envíen al próximo bloque al mismo tiempo para la ejecución en paralelo del algoritmo. La arquitectura interna de este bloque está compuesta por un arreglo de almacenadores de línea y una matriz de registros. La arquitectura de los almacenadores de línea, expuesta en la

Figura3. Arquitectura **del Bloque de almacenadores de línea.**, utiliza $(p - 1)$ registros de desplazamiento que permiten la configuración de la profundidad de acuerdo con la cantidad de columnas de la imagen. Las señales de L_0 a L_p son las filas de la imagen en la ventana de procesado, mientras que las señales de C_0 a C_q representan las columnas en la misma.

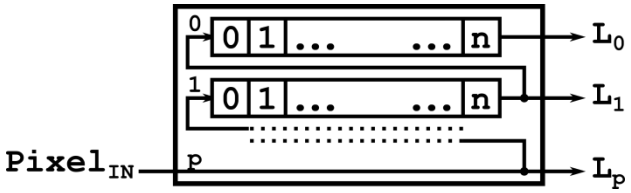

Figura3. Arquitectura del Bloque de almacenadores de línea. [9]

En cada ciclo de reloj un nuevo píxel es entregado por el subsistema de adquisición al bloque de almacenadores de línea devolviendo a su salida $(p + 1)$ píxeles $(L_0 - L_p)$, que se corresponden con una nueva columna a procesar. Este procedimiento implica una latencia inicial en el sistema de procesado debida a la demora en completar este bloque. La Ecuación 3 expone el valor de esta latencia donde n es la cantidad de píxeles de la imagen en el eje horizontal.

$$\tau_{lb} = 2 \times q \times n \qquad (3)$$

La matriz de registros utiliza $p(q - 1)$ bloques de registros genéricos, como se muestra en la Figura 4. **Arquitectura de la matriz de registros**. La función principal de esta arquitectura es mantener las diferentes columnas de píxeles para las entradas de los bloques de procesamiento de la imagen (L_aC_q, donde $1 < a < p + 1$ y $1 < b < q + 1$).

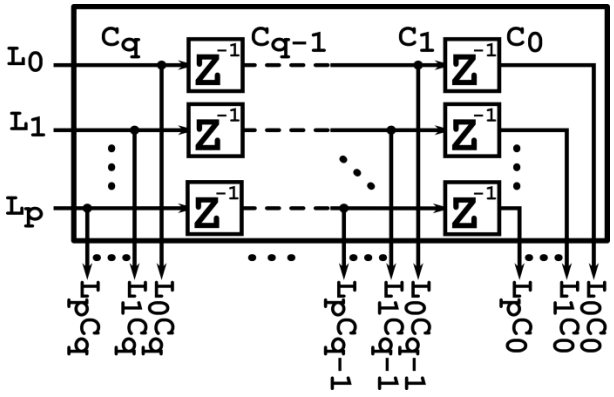

Figura 4. Arquitectura de la matriz de registros. [9]

Las arquitecturas que requieren estos bloques presentan una latencia inicial determinada por la demora en ocupar totalmente el mismo. La Ecuación 4 expone este valor.

$$\tau_{lb} = 2 \times q \qquad (4)$$

ARQUITECTURA HARDWARE DEL BLOQUE DE PROCESADO DE LA INFORMACIÓN.

El bloque de procesado de la información está compuesto por tantos multiplicadores como elementos tenga el kernel de convolución $(p + 1) \times (q + 1)$ y un bloque de suma. El bloque de paralelización envía los $(p + 1) \times (q + 1)$ pixeles presente en la ventana simultáneamente a los multiplicadores, estos están compuestos por dos entradas y una salida, permiten la multiplicación de números con punto fijo y son invocados tantas veces como sea necesario, mostrado en la

Figura 2. **Arquitectura de convolución no simétrica**

Estos multiplicadores están confeccionados con latencia porque así se introduce paralelismo y por lo tanto un aumento de la velocidad de ejecución. Dicha latencia o pipeline en una lógica combinacional se logra agregando niveles de registros en la lógica combinacional. Los Flips-Flops introducidos por un pipeline generalmente llevan consigo un aumento mínimo en el consumo de recursos en el FPGA, ocupando los Flips-Flops en desuso dentro de las celdas lógicas que ya se estén utilizando para implementar la lógica combinacional del diseño.

La latencia (τ_{op}) de cada uno de esos multiplicadores está dada por el logaritmo en base dos de la cantidad de bits a multiplicar como expresa la Ecuación 5.

$$\tau_{lb} = \log_2(\text{DataWidth} + \text{Presition}) \qquad (5)$$

El bloque de suma puede efectuar la adición de hasta 49 elementos de forma concurrente, por lo que el período mínimo de trabajo está dado por el tiempo que toma para hacerse la suma, esto incluye una demora mayor a la que presentan los sumadores en cascada de las arquitecturas presentes en XSGImgLib.

1.2. FILTRO DE PROCESADO NO LINEAL

El filtrado espacial no lineal permite sustituir el píxel a procesar por el resultado de una operación no lineal. La operación característica de este tipo de procesado es el ordenamiento, aunque otros tipos de procesamiento (exponencial y logarítmico) se incluyen en esta categoría. Los filtros de ordenamiento, y en particular el de mediana, permiten eliminar el ruido impulsivo y de alta frecuencia, ruido salt & pepper, en las imágenes sin afectar los bordes de las mismas [7], [8], [10], [11], [12]. Este procesado tiene una amplia aplicación en diferentes sectores, entre ellos en la medicina para eliminar el ruido en las imágenes radiográficas [13].

1.2.1. FILTROS DE ORDENAMIENTO DE IMÁGENES

El filtrado de ordenamiento es una técnica no-lineal que ordena el contenido de la ventana seleccionada y en esta toma la muestra indexada por la magnitud del rango. Para el caso bidimensional (2D), el contenido de una ventana de dos dimensiones, que se desliza a través de la imagen, se ordena numéricamente y se remplaza el elemento central de la ventana a la salida por el que tenga el rango especificado. Cada vez que se desplaza la ventana por la imagen, un conjunto de píxeles obsoletos se descartan y un conjunto de nuevos píxeles se insertan a esta. Los filtros de ordenamiento más típicos son la media, el mínimo y el máximo, aplicándose en el tratamiento previo antes de la detección de bordes o en la eliminación de ciertos tipos de ruidos de transmisión [9], [11], [12], [14].

Una variable importante en el uso de filtros de ordenamiento es el tamaño de la vecindad. Generalmente las formas que se utilizan son cuadradas (por comodidad de cálculo) o circular (para minimizar los efectos direccionales). Sin embargo a medida que el tamaño de la vecindad se incrementa, el esfuerzo computacional en la realización de la clasificación se aumenta rápidamente.

1.2.2. ARQUITECTURA HARDWARE DEL BLOQUE DE FILTRO DE ORDENAMIENTO

En un filtro de ordenamiento unidimensional, una ventana de $(q + 1)$ muestras se desplaza por el arreglo de puntos devolviendo el valor correspondiente a la posición requerida en el arreglo ya ordenado. Sea W_i una ventana ordenada de una dimensión, como muestra la Ecuación 7, donde y_i es el valor de la posición media de W_i definida como $y_i = \text{median}(W_i)$. Para un filtro en dos dimensiones una ventana de $(p + 1) \times (q + 1)$ elementos se mueve sobre la imagen.

Definiendo $W_{(i,j)}$ como una ventana en dos dimensiones, centrada en la posición (i, j), entonces $y_{(i,j)} = \text{median}(W_{(i,j)})$ [4].

$$W_i = \{x_i - N, \dots, x_i, \dots, x_i + N\} \qquad (7)$$

En la Figura5. Arquitectura **del filtro de ordenamiento genérico** se muestra la arquitectura de Chakrabarti para el filtro de ordenamiento de dos dimensiones que es la empleada para desarrollar el bloque de procesado, en esta figura $k = p = q$. Esta se compone de dos bloques principales, uno que calcula la posición de cada uno de los procesadores y otro que en función de estos resultados devuelve a la salida el valor requerido. Chakrabarti propone el uso de un arreglo compuesto por $(p + 1) \times (q + 1)$ procesadores, donde el período de muestreo es función del tiempo de actualización de la posición de los elementos antiguos en la ventana, utilizando los resultados de las comparaciones con las nuevas muestras y con las muestras desechadas.

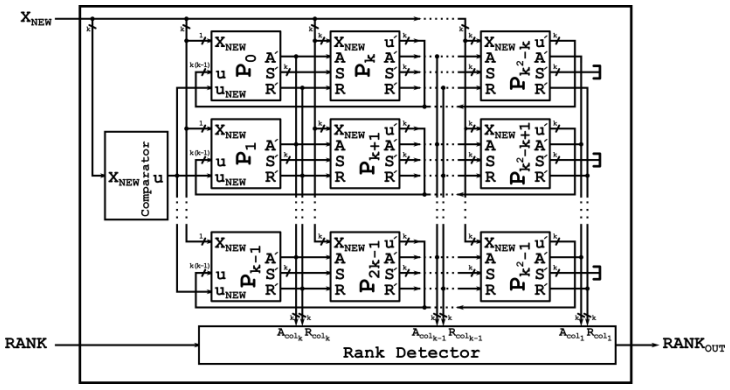

Figura5. Arquitectura del filtro de ordenamiento genérico. [9]

El diseño del filtro de ordenamiento genérico configurable desarrollado se basa en la arquitectura no recursiva definida por Chakrabarti en [11]. El funcionamiento del mismo es una extensión del algoritmo de ordenamiento de una dimensión para un filtro de mediana no recursivo y comienza cuando las nuevas muestras entran al comparador donde son comprobadas entre sí, este resultado pasa a los procesadores del $P_{(0)}$ al $P_{(p)}$ donde se calcula su posición en función de la salida del comparador y de las comparaciones con el resto de los elementos presentes en la ventana, paralelamente el resto de los procesadores, de $P_{(p+1)}$ al $P_{(p+1,q+1)}$, están actualizando su posición dentro de la ventana en función de la comparación con los elementos nuevos y los ya desechados. Posteriormente el Detector de posición devuelve a la salida el valor del pixel que se encuentre en la posición deseada.

El bloque de ordenamiento genérico no presenta ciclos de demora en la ejecución, obteniendo una salida válida en cada ciclo de reloj después del primer pulso, pero presenta una latencia debido a la demora en completar los píxeles en la ventana, la cual se calcula mediante la Ecuación 3

El bloque de ordenamiento implementado da la posibilidad de seleccionar la posición del píxel de salida posibilitando la obtención de cualquier valor dentro de la ventana, siendo las más usadas las posiciones mínima, media y máxima, permitiendo de esta manera implementar operadores morfológicos para las imágenes a procesar. Una imagen con ruido sal y pimienta al ser procesada con un filtro de ordenamiento para la mínima posición, resalta las tonalidades bajas de la imagen, aumentando los puntos negros en la misma, la selección de la posición media suprime el ruido de la imagen evitando los cambios bruscos en los niveles de grises de la imagen, y en cambio, la posición máxima destaca las tonalidades altas, amplificando los puntos blancos de la imagen, Figura6.Resultado del **procesado de imágenes con filtros de ordenamiento.**.

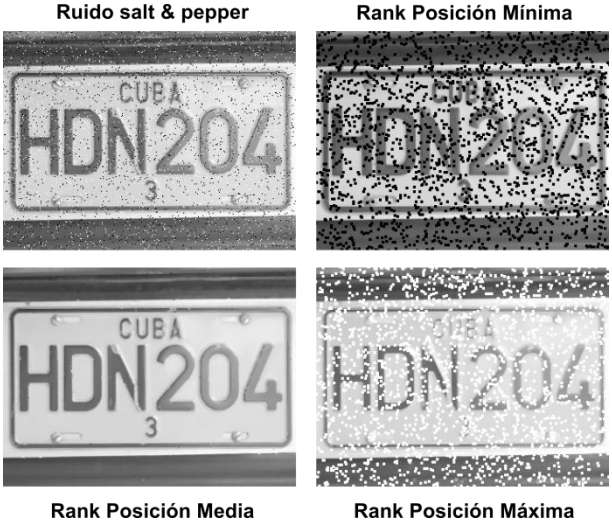

Figura6.Resultado del procesado de imágenes con filtros de ordenamiento.

ANÁLISIS DE LOS RESULTADOS

La comprobación de los resultados del diseño se realiza entre versiones software de estos algoritmos en MATLAB, los cuales requieren la representación de la imagen de forma matricial, y los bloques desarrollados. Esto posibilita obtener una medida de la exactitud del cálculo, y un análisis de la parametrización de los bloques.

Se utilizan para el diseño la herramienta Project Navigator de ISE Design Suite 12.4 y Xilinx System Generator (XSG) con MATLAB R2010b. El Project Navigator se trabaja con las características que trae por defecto para la síntesis, mapeo y ruteo. El diseño se implementa sobre una placa Spartan 3A DSP 1800 de Xilinx, para esto se usan dos opciones, con 6 bits de precisión (la mayor cantidad de bits que puede necesitar un kernel de convolución) y con 3 bits. Los resultados del diseño se compran con otras arquitecturas en cuanto a la cantidad de celdas lógicas utilizadas y a la frecuencia máxima de operación.

El diseño se caracteriza sobre una placa Spartan 3A DSP 1800 de Xilinx, para esto se usan dos opciones, con 6 bits de precisión (la mayor cantidad de bits que puede necesitar un kernel de convolución) y con 3 bits. Los resultados del diseño son comparados con otras arquitecturas en cuanto a la cantidad de celdas lógicas utilizadas y en cuanto a la frecuencia máxima de operación.

BLOQUE DE CONVOLUCIÓN

Este bloque de convolución permite configurar diversos parámetros como son la cantidad de bits para representar la parte entera (Data_Width) y la parte decimal (Presition) del kernel donde hay que tener en cuenta además de la cantidad de bits necesarios para la parte entera, un bit más para el signo, el número de columnas que va a tener la imagen, la latencia de los multiplicadores y el kernel a utilizar en la convolución de la imagen, el cual puede tener un tamaño de 3×3, 5×5, 7×7 o cualquier combinación de filas y columnas cuya multiplicación tenga como resultado máximo 49, el uso de las opciones de replicación de código de VHDL, nos permite poder ajustar en este bloque el tamaño del kernel en tiempo de diseño, provocando un gran avance en el nivel de configurabilidad de los bloques y también un salto en la versatilidad de los mismos.

En las comparaciones del diseño con los bloques genéricos de convolución de la biblioteca XSGImgLib configurada para obtener un mejor rendimiento en área, se puede observar que los nuevos diseños además de ganar mucho en configurabilidad, disminuye la frecuencia máxima de trabajo para los resultados de implementación provistos por Project Navigator de 1.299 a 1.322 y para los provistos por XSG disminuye en 1.013 veces para la convolución de 3 ×3 y aumenta para la convolución de 5 ×5 en 1.086 veces, estos resultados varían en función de la cantidad de bits para la precisión y el tamaño del kernel.

Cuando la biblioteca XSGImgLib se configura para obtener un mejor rendimiento en velocidad, la frecuencia máxima de trabajo disminuye en 1.069 veces para la convolución de 5 ×5 y aumenta para la convolución de 3×3 en 1.008 veces para los resultados de implementación provistos por Project Navigator y para los provistos por XSG aumenta de 1.315 a 1.319 veces.

Al comparar los resultados de las implementaciones en cuanto al consumo de recursos se obtiene un aumento de 1.86 veces para la convolución de 3×3 y una disminución de 1.023 para la convolución de 5 ×5 de los nuevos diseños con respecto la biblioteca XSGImgLib configurada para obtener un mejor rendimiento en velocidad. Al configurar XSGImgLib para obtener un mejor rendimiento en área los resultados de comparación aumentan de 1,632 a 2,557 veces. Estas comparaciones se muestran de la Tabla I: Consumo de recursos: Convolución 3 ×3 no simétrica. y la

Tabla II: Consumo de recursos: Convolución 5 ×5 no simétrica

Tabla I: Consumo de recursos: Convolución 3 ×3 no simétrica.

Recursos	Project Navigator	XSG	XSGImgLib speed	XSGImgLib área
Slices	532	532	286	208
Frecuencia [MHz]	179.827	137.817	181.321	136.054

Tabla II: Consumo de recursos: Convolución 5 ×5 no simétrica

Recursos	Project Navigator	XSG	XSGImgLib speed	XSGImgLib área
Slices	914	914	935	560
Frecuencia [MHz]	176.709	125.266	165.344	136.054

BLOQUE DE ORDENAMIENTO

El bloque de ordenamiento implementado da la posibilidad de seleccionar la posición del píxel de salida posibilitando la obtención de cualquier valor dentro de la ventana, siendo las más usadas las posiciones mínima, media y máxima, permitiendo de esta manera implementar operadores morfológicos para las imágenes a procesar. Además al igual que el bloque anterior permite configurar el tamaño de la ventana a utilizar, en tiempo de diseño.

Las comparaciones del diseño con los bloques de ordenamiento genéricos de la biblioteca XSGImgLib en cuanto a la frecuencia de trabajo muestran una disminución en de 1.62 a 1.715 veces para los resultados provistos por el Project Navigator y de 1.035 a 1.091 veces para los resultados provistos por el XSG.

Al comparar el diseño en cuanto a la cantidad de recursos ocupados se tiene que los resultados de comparación aumentan de 1,896 a 5 veces. Estos resultados se aprecian en la Tabla III: Consumo de recursos: Filtro de ordenamiento 3 ×3 no simétrica y la

Tabla IV: Consumo de recursos: Filtro de ordenamiento 5 ×5 no simétrico

Tabla III: Consumo de recursos: Filtro de ordenamiento 3 ×3 no simétrica

Recursos	Project Navigator	XSG	XSGImgLib
Slices	495	495	99
Frecuencia [MHz]	116.14	71.582	65.608

Tabla IV: Consumo de recursos: Filtro de ordenamiento 5 ×5 no simétrico

Recursos	Project Navigator	XSG	XSGImgLib
Slices	967	974	510
Frecuencia [MHz]	81.591	52.091	50.345

CONCLUSIONES

El paralelismo de los algoritmos de procesado de imágenes posibilita un desarrollo más eficiente de sistemas empotrados que los algoritmos secuenciales sobre procesadores de propósito general. El diseño de aplicaciones para FPGA no sólo realiza un uso más intensivo del paralelismo, también de la portabilidad del diseño y el bajo consumo de potencia del mismo, posibilitando el desarrollo de sistemas modulares y autónomos.

La implementación de módulos de procesado de imágenes parametrizables para System Generator permite el desarrollo de sistemas basados en modelos con un elevado nivel de eficiencia. El bloque Black Box que este propone brinda la opción de incluir código VHDL en sistemas basados en modelos, lo cual ofrece un salto en el grado de configurabilidad, pero para que estos diseños mantengan un desempeño similar a los ofrecidos Project Navigator, se hace necesario que las opciones de configuración de implementación sean modificadas.

El empleo de VHDL 200X limita a que los arreglos sean de tipo constrained, es decir la cantidad de elementos del arreglo puede ser variable pero el tamaño de estos tiene que estar predefinido. Esto provoca que para emplear arreglos sea necesario definir el tamaño del mismo como el máximo permitido y usar de este los bits válidos, lo que aumenta el consumo de recursos y disminuye la frecuencia máxima de trabajo.

Los bloques desarrollados para la biblioteca de procesado de imágenes posibilitan la configuración de los diseños genéricos con diferentes parámetros, logrando unidades de procesado versátiles al alcance del usuario. El nivel de configurabilidad que presentan los nuevos bloques diseñados para la biblioteca XSGImgLib viene dado por la utilización de la replicación de código, que permite cambiar los parámetros del bloque en tiempo de diseño.

REFERENCIAS

1. K BENKRID, D CROOKES, J SMITH, AND A BENKRID. High Level Programming for FPGA Based Image and Video Processing using Hardware Skeletons. In 9th Annual IEEE Symposium on Field-Programmable Custom Computing Machines, pages 1–8, 2001.

2. MAREK WNUK. Remarks on Hardware Implementation of Image Processing Algorithms. International Journal of Applied Mathematics and Computer Science, 18(1):105–110, 2008.

3. CARLOS RODRÍGUEZ CRUZ, RAZIEL RIVERO FLORES, ALEJANDRO CASTILLO ATOCHE, AND JAVIER VÁZQUEZ CASTILLO. Procesamiento de Imágenes con Xilinx System Generator. Primer Congreso Internacional de Sistemas Computacionales y electrónicos, pages 1–8, 2006.

4. T SAIDANI, D DIA, W ELHAMZI, M ATRI, AND R TOURKI. Hardware Co-simulation For Video Processing Using Xilinx System Generator. Proceedings of the World Congress on Engineering 2009, I: 3–7, 2009.

5. C T JOHNSTON, D G BAILEY, AND P LYONS. A Visual Environment for Real-Time Image Processing in Hardware. EURASIP Journal on Embedded Systems, 2006:1–8, 2006.

6. K BENKRID AND S BELKACEMI. Design and implementation of a 2D convolution core for video applications on FPGAs. Third International Workshop on Digital and Computational Video, 2002.DCV 2002.Proceedings., (November):85–92, 2002.

7. RAFAEL C GONZÁLEZ AND RICHARD E WOODS. Digital image processing. In Digital image processing, chapter 2, 3, 4, 6, pages 66–70,116–134,205–208,283–302,308–313,519–560. Prentice Hall, 2nd edition, 2002.

8. ALBA M SÁNCHEZ G., RICARDO ALVAREZ G., AND SULLY SÁNCHEZ G. Architecture for filtering images using Xilinx system generator. In International Journal of Mathematics and Computer in Simulation, volume 1, pages 101–107, 2007.

9. LUIS MANUEL GARCÉS-SOCARRÁS. Aceleración de Algoritmos Mediante Hardware Reconfigurable Biblioteca de Procesamiento de Imágenes para System Generator. Master thesis, Instituto Superior Politécnico "José Antonio Echeverría", 2011.

10. SHEETAL U BHANDARI, SHASHANK S PUJARI, SHAILA SUBBARAMAN, AND RASHMI MAHAJAN. Real Time Video Processing on FPGA Using on the Fly Partial Reconfiguration. In 2009 International Conference on Signal Processing Systems, pages 245–247, 2009.

11. C CHAKRABARTI. High sample rate array architectures for median filters. IEEE Transactions on Signal Processing, 42(3):707–712, 1994.

12. TEXAS INSTRUMENTS. Implementation of an Image Processing Library for the TMS320C8x (MVP), 1997.

13. JORGE OSIO, JOSE RAPALLINI, A QUIJANO, AND JESÚS OCAMPO. Implementación de un Algoritmo para procesamiento de imágenes en una FPGA. Congreso de MicroelectrónicaAplicada 2010, pages 38–42, 2010.

14. GABOR SZEDO. Two-dimensional rank-order filter by using max-min sorting network. Xilinx Application Notes, pages 1–17, 2006

AUTORES

Elias Augusto Perdomo Hourné. Ingeniero en Automática por el Instituto Superior Politécnico "José Antonio Echeverría" en 2012. Profesor Adiestrado del departamento de Automática y Computación del ISPJAE, La Habana, Cuba.

e-mail: elias@electrica.cujae.edu.cu

Luis Manuel Garcés Socarrás. Master en Sistemas Digitales, Ingeniero en Automática. Profesor Asistente del departamento de Automática y Computación del ISPJAE, La Habana, Cuba. e-mail: lmgarcess@electrica.cujae.edu.cu

Alejandro José Cabrera Sarmiento Doctor en Ciencias Técnicas. Ingeniero Electricista. Profesor Titular del departamento de Automática y Computación del ISPJAE, La Habana, Cuba. e-mail: alex@electrica.cujae.edu.cu