IMPLEMENTACIÓN DE FILTROS FIR EN FPGA'S.

Ignacio Bravo, Raúl Rivera, Álvaro Hernández, Raúl Mateos, Alfredo Gardel, Francisco Javier Meca

Departamento de Electrónica. Escuela Politécnica. Universidad de Alcalá.
Ctra. Madrid-Barcelona km 33.600. 28871 – Alcalá de Henares (Madrid)
ibravo@depeca.uah.es http://www.depeca.uah.es

RESUMEN

El presente trabajo tiene como objeto, mostrar diversas alternativas implementadas en laboratorios de diseño electrónico con dispositivos FPGA's (*Field Programmable Gate Array*), de las titulaciones de Ingeniería Técnica de Telecomunicación e Ingeniería Electrónica. Concretamente el artículo presente diversos enfoques en el diseño e implementación de filtros FIR de cuatro coeficientes. Se busca en todos los diseños realizados, estudiar los resultados obtenidos desde el punto de vista de recursos internos consumidos y de la máxima velocidad que se puede lograr.

1. INTRODUCCIÓN.

Desde el desarrollo de los primeros prototipos de FPGA (Field Programmable Gate Array) hace ya tres décadas, estos dispositivos han sufrido importantes avances en cuanto a prestaciones y densidad, al mismo tiempo que los precios han ido disminuyendo. Esta evolución de la tecnología junto con los avances de las herramientas de desarrollo y las tecnologías de soporte han revolucionado el campo del diseño lógico.

A finales de los 70 y principios de los 80 aparecieron un nuevo tipo de circuito llamado ASIC (Application-Specific Integrated Circuit – Circuitos Integrados de Aplicación Específica). Los ASIC's más populares eran los Mask-Programmed Gate Array. Como su nombre indica, éstos están compuestos de un array o matriz de puertas lógicas. Estos circuitos se programan para una aplicación en particular mediante la creación de máscaras metálicas que determinan la interconexión entre las puertas. No obstante, tienen sus inconvenientes como, por ejemplo, el hecho de que el tiempo desde que se realiza el prototipo hasta que puede ser comercializado es muy elevado. Además, dicho tiempo podría incrementarse ya que, a menudo, los prototipos no funcionan como deben y eso añade tiempo y dinero al desarrollo del producto. Por tanto, el desarrollo de una aplicación basada en un Gate Array tiene un considerable riesgo para las empresas.

Así, las FPGA's, inventadas a mediados de los 80, solucionaron muchos de estos problemas. Estos circuitos integrados de alta densidad combinan la flexibilidad y el alto nivel de integración de los anteriores con una sencillez de programación. Como los Gate Arrays, las FPGA's contienen una matriz de elementos lógicos que pueden ser interconectados para implementar una aplicación dada. Estas interconexiones son controladas por switches programables por el usuario. Así, los prototipos pueden ser implementados, probados y modificados rápidamente.

En resumen, las FPGA's en comparación con los demás dispositivos lógicos, tienen como objetivo destacar en aspectos tales como:

- Velocidad: Aumenta la frecuencia de operación de los sistemas.
- Densidad y capacidad: Al ser sistemas complejos, necesitan de gran cantidad de recursos como puertas lógicas y biestables.
- Facilidad de uso: que permita al ingeniero de desarrollo implementar un diseño con la rapidez que exige el mercado. Esto implica software de desarrollo fácil de usar.
- Programabilidad en el sistema y reprogramabilidad en el circuito: Esto es, la posibilidad de programar o reprogramar el dispositivo que se encuentra ya en el circuito impreso al que va destinado.

Desde el punto de vista docente, estos dispositivos proporcionan a los profesores una alternativa para poder implementar diversos sistemas digitales con una plataforma universal. Sin embargo, el camino para poder impartir desde un punto de vista teórico-práctico estos dispositivos en numerosas ingenierías, lleva asociado un continuo reciclaje del profesorado, ante la continua y vertiginosa evolución de estos dispositivos así como sus herramientas de desarrollo.

Con objeto de sacar el mayor partido a estos dispositivos, su impartición se suele ubicar en cursos superiores (último curso de ingenierías técnicas y cuarto y/o quinto curso de ingenierías superiores), donde el alumno ya posee unos conocimientos previos, tanto en asignaturas de electrónica como otras ramas de ingeniería, necesarios para poder explotar todas las características de estos.

El presente artículo va centrado dentro de esta línea. Se pretende explicar diversas alternativas para desarrollar en un laboratorio de filtros FIR diseñados e implementados en una FPGA. Obviamente el alumno debe tener conocimientos previos de electrónica digital, filtros y análisis de la señal. Todo el trabajo ha sido desarrollado sobre dispositivos Xilinx.

3. TEORÍA DE FILTRADO

El filtrado es una etapa dentro del procesado de señal muy utilizado habitualmente [1]. Se usa para dejar pasar ciertas frecuencias de una señal a través del sistema sin ninguna distorsión y no dejar pasar otras. El sistema que realiza esta función se denomina filtro.

Se pueden definir varios tipos de filtros, dependiendo de la naturaleza de la operación de filtrado. Filtros paso bajo, paso alto, paso banda, paso todo.

Dentro del filtrado también se pueden distinguir distintos tipos de filtros dependiendo de la señal a filtrar. Si la señal a filtrar es analógica y no se desea un filtrado digital, los tipos de filtros que se pueden realizar son:

- Filtros analógicos pasivos: Se realizan con componentes discretos pasivos.
- Filtros analógicos activos: Se realizan con componentes activos (amplificadores con ayuda de estructuras recursivas).

Si por el contrario se desea realizar un filtrado digital, de una señal previa conversión a digital, existen de dos tipos de filtrado:

- Filtros digitales IIR (Infinite Impulse Response, Respuesta Infinita al Impulso).
- Filtros digitales FIR (Finite Impulse Response, Respuesta Finita al Impulso).

Como en el presente trabajo, se ha realizado un procesado de imágenes basado en un filtro FIR los fundamentos teóricos a exponer, irán encaminados a este tipo de filtros.

Un filtro FIR de longitud L (número de coeficientes) y orden del filtro L-1, entrada x[n] y salida y[n] se describe por la ecuación en diferencias siguiente:

$$y[n] = b_0 x[n] + b_1 x[n-1] + \cdots + b_{L-1} x[L-n+1] = \sum_{k=0}^{L-1} b_k x[n-k]$$

En los filtros FIR los $a_k = 0$ y los $b_k = h[k]$, donde b_k es el conjunto de coeficientes del filtro. Alternativamente podemos expresar la secuencia de salida como la convolución de la respuesta impulsiva del sistema h[n] con la señal de entrada.

Los filtros FIR son sistemas que por definición presentan una respuesta al impulso de duración finita. Si se considera al sistema causal, entonces la expresión siguiente caracteriza a un filtro FIR de coeficientes b_k.

$$h[n] = \sum_{k=0}^{L-1} b_k \delta[n-k]$$

Donde δ[n-k] representa el impulso unitario discreto.

También se puede expresar en el dominio transformado Z como:

$$H(z) = \sum_{n=-\infty}^{\infty} h[n] z^{-n} = \sum_{k=0}^{L-1} b_k z^{-k}$$

Como, la transformada de Fourier discreta:

$$H(\Omega) = \sum_{n=0}^{L-1} h[n] e^{-j\Omega n} = H(z)\Big|_{z=e^{j\Omega}}$$

Como, la transformada de Fourier discreta muestreada:

$$H[k] = \sum_{n=0}^{L-1} h[n] e^{-j\frac{2\pi}{L}\cdot k\cdot n} \qquad k = 0, 1, \ldots, L-1$$

Entonces tenemos:

$$H(\Omega) = \Re e\{H(\Omega)\} + j\Im m\{H(\Omega)\} = |H(\Omega)| e^{j\Phi_H(\Omega)} = A(\Omega) e^{j\theta(\Omega)}$$

Donde A(Ω) es la función amplitud, que es la que se utiliza para analizar los filtros.

$$|H(\Omega)| = \sqrt{\Re e^2\{H(\Omega)\} + \Im m^2\{H(\Omega)\}} \quad ; \quad \Phi_H(\Omega) = arctg \frac{\Im m\{H(\Omega)\}}{\Re e\{H(\Omega)\}}$$

Si H(Ω) es real |H(Ω)| par y Φ(Ω) impar.

La salida se puede calcular mediante convolución directa de la entrada con la respuesta al impulso, según se muestra en la siguiente expresión.

$$y[n] = \sum_{k=0}^{L-1} h[k]x[n-k]$$

Las raíces de este último polinomio, como se puede observar, constituyen los ceros del filtro (donde la respuesta de esta función se hace cero).

Los filtros FIR son muy fáciles de realizar. La mayoría de los procesadores de señales digitales tienen unas arquitecturas internas que hacen factible su construcción.

Los filtros FIR no recursivos son menos sensibles a los efectos de longitud de palabra finita que los IIR, ofreciendo diversas ventajas en los cálculos que conlleva el filtrado.

4. IMPLEMENTACIÓN DE FILTROS FIR EN FPGA'S.

En el presente trabajo se muestran dos filtros, el primero de ellos utiliza un algoritmo de multiplicación acumulación [2] en el que los datos de salida se calculan siguiendo la retardando, sumando y multiplicando los valores de las muestras, y un segundo filtro utilizando aritmética distribuida [3] precalcula el valor de los coeficientes y permite ahorrar gran número de recursos internos de la FPGA. Los filtros serán aplicados sobre imágenes capturadas en formato bmp [4].

La filosofía de trabajo a emplear se describe en la Figura 1. En primer lugar del archivo bmp se extraen los datos correspondientes a la imagen utilizando Matlab, quedando estos almacenados en un archivo de texto. Este archivo de texto es utilizado por el banco de pruebas para introducir datos en el filtro. Los datos que proporcionados a la salida del filtro son también almacenados en un archivo de texto, cuyos datos son introducidos en un archivo bmp que representa la imagen filtrada.

Figura 1.- Filosofía de trabajo.

4.1. Algoritmo MAC.

Existen sistemas de procesado digital que utilizan algoritmos MAC (multiplicación acumulación) para realizar dicho procesado. El almacén de un dato significa retrasar su uso en una cantidad igual a un periodo de muestreo. Este retraso se representa mediante z^{-1}.Por lo que los filtros digitales pueden implementarse usando los elementos correspondientes a la multiplicación adición y almacenaje de datos en la FPGA. Matemáticamente el algoritmo

MAC se representa por la multiplicación de dos matrices o un vector por un escalar donde el escalar multiplica a todos los elementos del vector.). El producto de dos matrices C = A x B es una matriz de orden (m x p), que contiene el elemento resultado de la suma de la multiplicación de los elementos de las matrices.

$$c_{ij} = \sum_{k=1}^{N} a_{ik} b_{kj}$$

Como en este diseño no se utilizan los multiplicadores específicos del FPGA, los términos a_j x b_k se deben subdividir en varios términos elementales. El proceso es un sumatorio de productos parciales.

$$a_j \times b_k = \sum_{i=0}^{n-1} \left[\left(a_{ji} b_k \right) 2^i \right] = a_0 \cdot (b_k) 2^0 + a_1 \cdot (b_k) 2^1 + a_2 \cdot (b_k) 2^2 + \ldots + a_{n-2} \cdot (b_k) 2^{n-2} + a_{n-1} \cdot (b_k) 2^{n-1}$$

Esta ecuación no es más que una serie de operaciones AND (multiplicaciones) que determinan una suma seguida por un desplazamiento, una función ADD (acumulación).

La multiplicación de dos números binarios se puede realizar a partir de varias suboperaciones de circuitos combinacionales que forman el producto bit a bit. Esta forma de multiplicar dos números es el modo más rápido. Por otro lado el array de multiplicadores requiere un número elevado de puertas introduciendo un gran número de retardos y provoca un consumo considerable del área de la FPGA. Como consecuencia de esto se utilizan otros métodos para implementar la multiplicación en las FPGA´s que se explicarán mas adelante.

El esquema del filtro se muestra en la Figura 2:

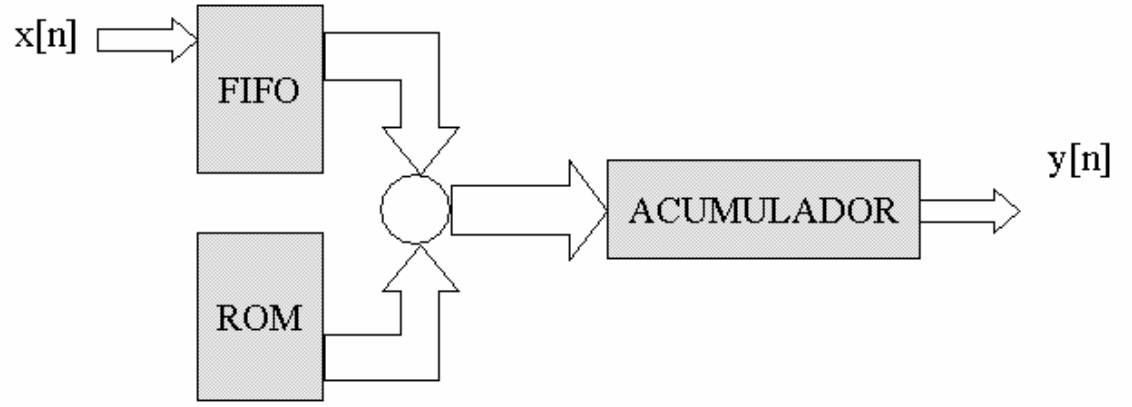

Figura 2. Estructura del filtro MAC.

El sistema esta formado por los siguientes componentes:

- Memoria FIFO, que se encarga de realizar los retardos sobre los datos de entrada.
- Memoria ROM, Los coeficientes del filtro se almacenan en una memoria ROM que se define como un array de palabras de la propia ROM.
- Un multiplicador, se encarga de multiplicar los datos de entrada por los coeficientes.
- Un acumulador, que se encarga de almacenar y sumar los productos parciales.

Con respecto a su funcionamiento básico, los datos son introducidos en el circuito a través de la memoria FIFO. Cada dato se valida mediante la señal load, que se activa a nivel alto cada vez que la cámara suministra un nuevo dato. Cada uno de los datos se multiplica por los

coeficientes del filtro almacenados en la memoria ROM y se suma a los valores anteriores, realizándose la función de un filtro FIR.

Los coeficientes del filtro suelen ser números reales. Esto hecho supone un gran problema a la hora de la realización de un filtro. Para representar estos valores se han desarrollado diferentes sistemas de representación como la aritmética en coma fija, que se utilizará posteriormente. En este primer diseño se ha optado por multiplicar los coeficientes del filtro por una cantidad constante y dividir el resultado obtenido por esa misma cantidad. Esta solución no es la más óptima ya que para no perder precisión en los cálculos los coeficientes se deben multiplicar por una cantidad elevada. Esto provoca el aumento del ancho de las señales internas del filtro con lo que aumenta el número de bits del multiplicador originando un incremento considerable del área que ocupa el circuito en de la FPGA.

El tiempo que tarda el filtro en obtener un dato valido depende del número de coeficientes del filtro y del número de bits de los datos de entrada. Para un filtro de orden cuatro desde que se introduce un dato de entrada deben transcurrir ocho ciclos de la señal de reloj para obtener un dato de salida.

4.1. Algoritmo con aritmética distribuida.

El empleo de la aritmética distribuida (DA) para el procesado digital está justificado por su eficiencia de cálculo. Realizando un buen diseño se puede reducir el número de puertas, utilizando Aritmética distribuida, de un 50% a un 80% con respecto a las técnicas tradicionales de procesamiento (MAC) [5].

La ventaja principal de la DA es su eficiencia de mecanización y la disminución de recursos internos que origina su utilización. Debido a que se trata de un proceso de naturaleza serie la desventaja que presenta es su relativa lentitud, aunque en ocasiones esto no se plantea como una desventaja: el tiempo requerido para una entrada (8 bits) al sistema de 8 palabras de 8 bits cada una, en la que se emplea un modo de transmisión paralelo (una palabra por periodo) es el mismo tiempo que se emplea para la transmisión serie de (8 bits de entrada) 8 palabras de 8 bits conectadas en serie cada una de las palabras a cada una de las entradas y se van desplazando en serie bit a bit (un bit por periodo).

La principal característica de la DA es que los multiplicadores se implementan mediante bancos de memoria ROM que almacenan el valor de los coeficientes precomputados. La expresión que permite obtener la salida de un filtro FIR de k etapas se muestra a continuación:

$$y[n] = \sum_{K=0}^{K-1} h[k]x[n-k]$$

Rescribiendo la ecuación anterior con un cambio de variable, se obtiene:

$$y = \sum_{n=0}^{N-2} 2^n \cdot \left[\sum_{k=0}^{K-1} h[k] \cdot b_n[k] \right] - 2^{N-1} \cdot \sum_{k=0}^{K-1} h[k] b_{N-1}[k]$$

Considérese el término que aparece entre corchetes en la expresión anterior:

$$\left[\sum_{k=0}^{K-1} h[k] \cdot b_n[k] \right]$$

Puesto que el término $b_n[k]$ solo puede tomar los valores 0 o 1 la expresión anterior puede tomar 2^k valores distintos. Estos valores pueden precalcularse y almacenarse en una memoria

ROM. Los bits de los datos de entrada se emplean para generar la dirección con que se accede a la memoria. La Figura 3 ilustra esto.

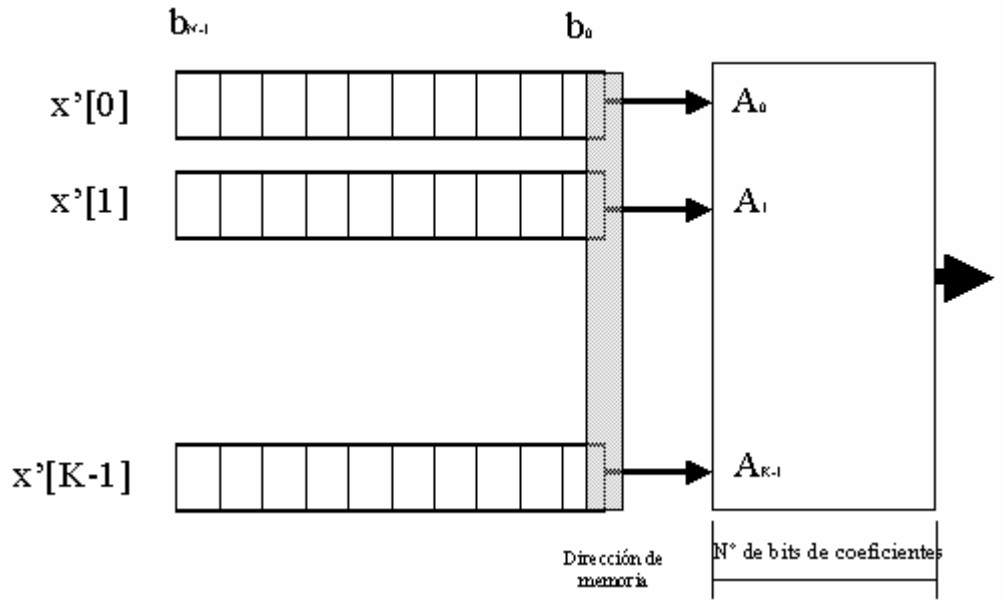

Figura 3.-Empleo de la ROM de coeficientes precalculados.

La dirección de memoria se obtiene concatenando los bits de orden n de las muestras de entrada. Dada una dirección de memoria el contenido de esa posición de memoria viene dado por la expresión:

$$mem(adrs) = \sum_{k=0}^{K-1} h[k] \cdot A_k$$

La Tabla 1 muestra el contenido de la memoria para un filtro FIR de cuatro coeficientes

A0	A1	A2	A3	Contenido
0	0	0	0	0
0	0	0	1	h[0]
0	0	1	0	h[1]
0	0	1	1	h[1]+h[0]
0	1	0	0	h[2]
0	1	0	1	h[2]+h[0]
0	1	1	0	h[2]+h[1]
0	1	1	1	h[2]+ h[1]+h[0]
1	0	0	0	h[3]
1	0	0	1	h[3]+h[0]
1	0	1	0	h[3]+h[1]
1	0	1	1	h[3]+ h[1]+h[0]
1	1	0	0	h[3]+ h[2]
1	1	0	1	h[3]+ h[2]+h[0]
1	1	1	0	h[3]+ h[2]+h[1]
1	1	1	1	h[3]+ h[2]+h[1]+h[0]

Tabla 1.-Contenido de la memoria ROM para cuatro coeficientes.

En un diseño real usando un ancho de palabra de 16 bits para la memoria ROM es suficiente para conseguir una buena precisión y no aumentar excesivamente el área ocupada por el circuito.

Se puede obtener el circuito que realiza la función de un filtro FIR que se muestra en la Figura 4. El primer elemento de la cadena de registros es un registro paralelo serie que recibe los datos de entrada, la salida de cada registro, se conecta a la entrada del siguiente de esta forma en cada registro aparece un nuevo dato en un número de ciclos de la señal de reloj igual al número de bits de las muestras de entrada.

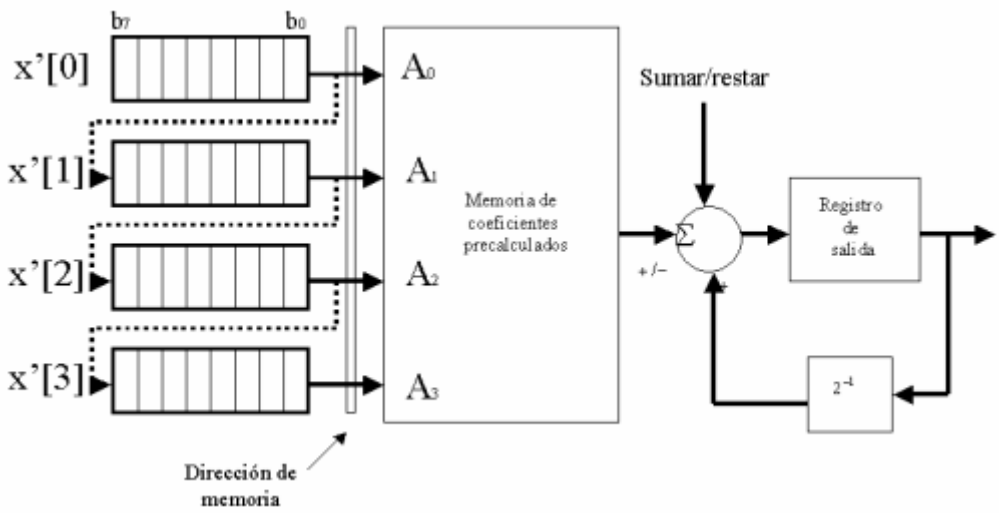

Figura 4. -. -Filtro FIR utilizando DA.

5. RESULTADOS Y CONCLUSIONES

Como resultado de la simulación temporal se obtiene un archivo de texto. El archivo de texto es transformado a un archivo bmp utilizando la función bmp2txt obteniéndose la imagen filtrada de salida. Los resultados obtenidos con Matlab coinciden con los obtenidos mediante VHDL. La Figura 5.a muestra la imagen original. Los resultados se han obtenido utilizando los siguientes coeficientes: h[0] = 0.017578125, h[1] = 0.046875, h[2]= 0.123046875, h[3] = 0.19716505, las imágenes obtenidas mediante Matlab y VHDL se muestran en la Figura 5.b y c.

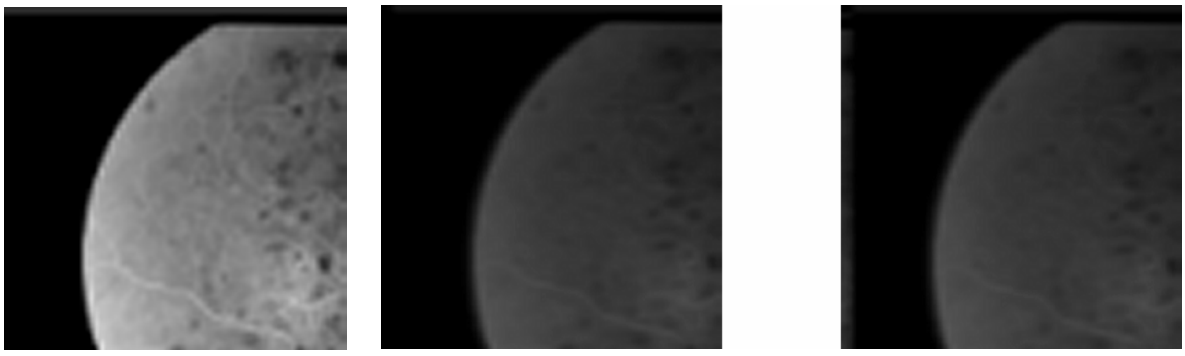

Figura 5. a.Imagen patrón (izda.), b. filtrada mediante Matlab (centro) , c. con VHDL (derecha.)

Para poder comparar las dos imágenes obtenidas se aplicado la función resta. La imagen diferencia umbralizada que se obtiene tras aplicar la función a la imagen obtenida mediante Matlab y VHDL se muestra en la Figura 6. Se puede apreciar que el mayor número de diferencias se encuentra en la parte izquierda de la imagen. Esto se debe a que la forma de operar de VHDL y Matlab es distinta. La diferencia radica en que la función filter toma los

datos de entrada en grupos del tamaño del vector de coeficientes coeff, realizando las operaciones sobre este paquete de datos, por otro lado en VHDL los datos se introducen secuencialmente. Las operaciones se realizan para cada dato de entrada y no agrupados con lo que al calcular los primeros datos de cada fila no se tiene constancia de los anteriores, al ser estos los primeros, produciéndose resultados distintos. Esto sólo afecta a las tres primeras columnas de la imagen ya que al recibirse el cuarto (número de coeficientes del filtro) dato de cada fila la forma de operar es idéntica. El resto de diferencias en la imagen son debidas a la precisión. Matlab utiliza una precisión prácticamente infinita mientras que los datos en VHDL están escalados, con la consiguiente perdida de precisión en los cálculos. El cálculo del porcentaje de coincidencia calculado mediante la función resta es del 94.6777%, debido a lo comentado anteriormente.

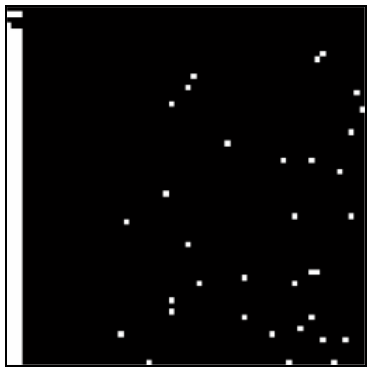

Figura 6.-Imagen diferencia para el filtro MAC.

Con respecto al segundo algoritmo, se ha escogido una imagen distinta donde la simulación temporal coincide con la simulación funcional. Al igual que en el diseño anterior los datos de la salida son almacenados en un fichero de texto que posteriormente mediante Matlab se transforma en un archivo de imagen. Los coeficientes empleados son los siguientes: h[0] = 2.3788104585828560e-001, h[1]=2.4640238752833240e-001, h[2] = 2.4640238752833240e-001, h[3]=02.3788104585828560e-001. La imagen original y la imagen filtrada mediante Matlab y usando VHDL se muestran en la Figura 7.

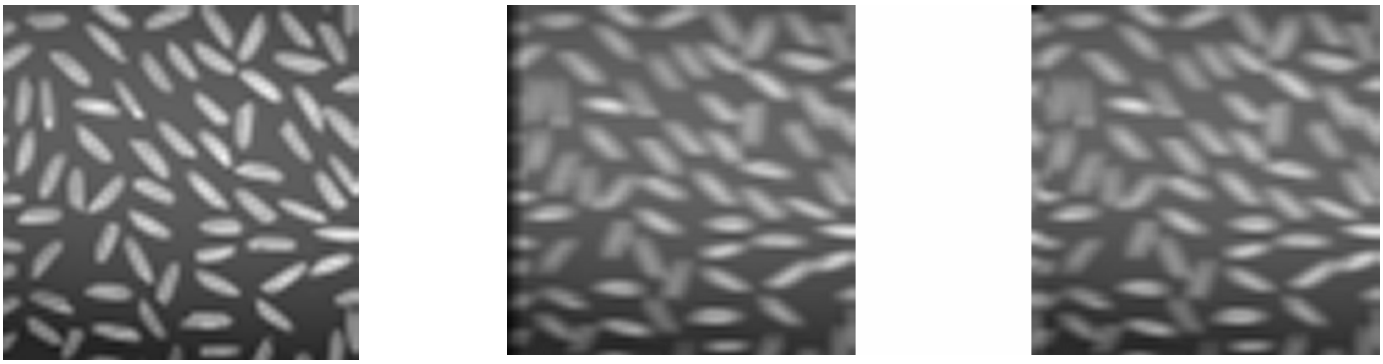

Figura 7. Imagen original (izda.). Imagen filtrada con Matlab (centro) y con VHDL (derecha.)

Al aplicar la función resta de imágenes se obtiene para esta imagen un porcentaje de igualdad de un 90%. Las causas son exactamente las mismas que para el filtro FIR utilizando el algoritmo MAC [6].

La principal ventaja que presenta el diseño utilizando aritmética distribuida es la reducción del número de CLB's que necesita el diseño, por otro lado, la ventaja que presenta el filtro utilizando el algoritmo MAC es que permite frecuencias de funcionamiento muy elevadas. Además aunque el código del filtro DA es más largo y complicado es más uniforme y el

proceso de síntesis es mas controlado por el diseñador en todo momento, mientras que en el filtro MAC permite demasiada libertad al sintetizador y su eficacia depende de la capacidad del sintetizador de implementar la operación de multiplicación, que consume gran número de recursos internos de la FPGA y que es fuertemente dependiente del número de bits de los operandos.

Para diseños con un número de coeficientes mayor, la diferencia entre el número de CLB`s consumidos por un diseño y por otro aumenta considerablemente. En el filtro MAC un aumento del número de coeficientes del filtro supone un aumento del número de multiplicaciones disminuyendo la frecuencia de funcionamiento del filtro, sin embargo en un filtro DA un aumento del número de coeficientes supone un aumento del número de elementos de la memoria ROM que es menos significativo, además la frecuencia de funcionamiento se ve menos afectada.

Aunque se pueda pensar que los diseños no son del todo comparables ya que la forma de codificar los números reales sea distinta en un diseño y otro la reducción empleando DA es de mas del 80 %. La Tabla 2 muestra las diferencias en cuanto al área y la frecuencia para el filtro utilizando el algoritmo MAC y el de DA.

	MAC	DA
Área (slices)	46	181
Frecuencia (Mhz)	287	109

Tabla 2. Comparativa MAC y DA.

6 AGRADECIMEINTOS

Este trabajo ha sido posible gracias al proyecto “Sistema de localización y posicionamiento absoluto de robots. Desarrollo de un espacio inteligente” del Ministerio de Ciencia y Tecnología (ref:DPI2003-05067).

7 BIBLIOGRAFÍA

[1] Discrete-Time signal processing. A. V. Oppenheim, R.W. Schafer. Prentice Hall International. 1989-

[2] Implementación de filtros digitales en FPGAs. Francisco Javier Torres Valle.Universidad Autónoma de Guadalajara.(México)

[3] Distributed Arithmetic FIR FILTER. Product Specification. White Papers Xilinx web (www.xilinx.com). 1999

[4] Implementation of image processing algorithms on FPGA hardware. Anthony Edward Nelson. Vanderbilt University. Master Thesis. May 2000.

[5] Distributed Arithmetic Design. www.geocities.com/SiliconValley/pines/6639/ip/da.htm.

[6] Implementación de algoritmos de tratamiento de señales en FPGA’s. Raúl Rivera Navas. Trabajo Final de Carrera. Departamento de Electrónica. Universidad de Alcalá. Julio 2003.