Implementación de la FFT en hologramas de Fourier generados con FPGAs

Castillo-Atoche A.[1], Pérez-Cortés M.[2], Ortiz-Gutiérrez M.[3], Vázquez-Castillo J.[4]

Recibido: 11 de noviembre de 2005 – Aceptado: 29 de junio de 2006

RESUMEN

En la actualidad, algunos hologramas son generados por computadora utilizando algoritmos que simulan la interferencia de la luz. En este artículo una técnica en tiempo real para implementar la Transformada Rápida de Fourier (FFT por sus siglas en inglés) en hologramas de Fourier digitales generados con circuitos digitales VLSI (Very Large Scale System Integration), específicamente FPGAs es utilizada. El diseño digital con FPGAs presenta una gran ventaja debido al procesamiento en paralelo que realiza y por su arquitectura flexible, alta velocidad e integración. El diseño fue verificado e implementado usando la herramienta Xilinx System Generator con la plataforma Matlab/Simulink.

Palabras Clave: Hologramas de Fourier, FPGA, System Generator, FFT.

FFT implementation on Fourier holograms generated with FPGAs

ABSTRACT

Holograms are made using interference of two light beams; nowadays it is also possible to make computer-generated holograms. In this work we propose a technique in real time to implement the 2D-FFT architecture in digital holograms with a VLSI digital component; specifically an FPGA. The digital design with FPGA presents great advantages due to its parallel processing method and its flexible structure, high integration and velocity. The design was implemented and verified with a Xilinx System Generator using the Matlab/Simulink platform.

Keywords: Fourier Holograms, FPGA, System Generator, FFT.

[1] Profesor de Carrera del Cuerpo Académico de Mecatrónica de la Facultad de Ingeniería de la UADY. E-mail: acastill@uady.mx
[2] Profesor de Carrera del Cuerpo Académico de Física de la Facultad de Ingeniería de la UADY.
[3] Profesor de la Escuela de Ciencias Físico-Matemáticas, Universidad Michoacana de San Nicolás de Hidalgo.
[4] Profesor de Carrera del Cuerpo Académico de Redes y Electrónica de la Universidad de Quintana Roo (UQROO).

INTRODUCCIÓN

Las imágenes de la televisión y las fotográficas son imágenes planas o bidimensionales. En ambos casos representan la distribución de intensidades de luz que se pueden interpretar matemáticamente como el cuadrado de la amplitud del campo óptico, o también conocido como irradiancia; esto es, lo que llega a nuestros ojos se interpreta como una imagen. En este tipo de imágenes, la información del objeto que está en relación con la fase óptica de la onda se pierde.

La holografía, descubierta por Dennis Gabor en 1947, incluye la grabación de ondas ópticas y su reconstrucción (D. Gabor, 1948). Es decir, por medio de la holografía se puede obtener una imagen de un objeto sin pérdida de información debido a que se registra en una película o sistema de memoria adyacente el patrón de interferencia formado por las fases de las ondas, a diferencia de las fotografías que solo registran intensidad.

Un holograma es un patrón de interferencia grabado que nos permite tener la mayor parte de la información de la onda y de su fase respectiva, ambas registradas en una película fotosensible. Para ver la imagen grabada en un holograma, es necesario que pase por el holograma el mismo haz de referencia que usó para generar el patrón de interferencia (Hetch, 2004).

En la actualidad, también se cuenta con hologramas generados por computadoras (Lohmann, 1967) y de hologramas generados químicamente (A. Olivares, 2002); de la misma manera, por medio de la configuración de franjas, tienen la información de la forma y la estructura en que se esparce luz. Para observar este tipo de hologramas, también es necesario que se reconstruyan un haz de luz coherente proveniente de un láser.

Algunos tipos de hologramas son de: Fresnel, Fraunhofer, Imagen, y de Fourier. En la actualidad, con el desarrollo de las computadoras y del software científico, y con la ayuda de algoritmos de transformadas discretas de Fourier o de Fresnel es posible diseñar fácilmente hologramas. A menudo, se usa el algoritmo de la transformada rápida de Fourier (J. W. Goodman, 2003). Los hologramas generados por computadora ayudan a producir frentes de onda con una amplitud diseñada y una cierta distribución de fase; las aplicaciones de dichos sistemas se aplican en exploración por láser, en filtros espaciales ópticos, en pruebas de superficies ópticas y sobre todo como memorias holográficas.

Para realizar los hologramas de Fourier, se graba el patrón de interferencia entre dos haces provenientes de un láser. Uno llamado haz de referencia y otro que ilumina al objeto y pasa por una lente positiva (convexa) llamado haz objeto. El haz objeto que hace interferencia es la transformada de Fourier del objeto que se forma en el plano de frecuencias de la lente (o también conocido como plano de Fourier). El objeto se coloca en el plano focal anterior de la lente y la transformada de Fourier se localiza en el plano focal posterior. La transformada de Fourier $U(\nu_x, \mu_y)$ de una función $u(x, y)$, es realizada ópticamente por medio de una lente positiva. En este caso se representará la lente por medio de la función matemática de la transformada de Fourier discreta (DFT).

Un algoritmo mucho más eficiente para llevar a cabo la DFT es la transformada rápida de Fourier (FFT). Si se considera el cálculo de la FFT para N puntos, el número de operaciones se reduciría de N^2 a $KN \log_2 N$ sobre el algoritmo de la DFT, donde K es una constante y no depende de N. Por ejemplo, el cálculo de una transformada de 1024 puntos usando la FFT es 200 veces más rápido que usar el algoritmo de la DFT ya mencionada. La ecuación 1 representa la manera de obtener la FFT de N puntos.

$$U_{FFT}(\nu_i, \mu_j) = \sum_{n=1}^{N}\sum_{m=1}^{N} \left| u(n\Delta x_0 + m\Delta y_0) \right| e^{i\phi(n\Delta x_0 + m\Delta y_0)} e^{(i\,2\pi\phi/\lambda z)(nx_i\Delta x_0 + my_i\Delta y_0)} \qquad (1)$$

Si $u(x, y)$ es la amplitud compleja en el plano focal anterior de la lente, entonces $U(x/\lambda\, f, y/\lambda\, f)$ es la amplitud compleja en el plano focal posterior, donde *f* es la longitud focal de la lente y λ es la longitud de onda. Hay que considerar que la transformada de Fourier tiene amplitud compleja, por tal motivo, no puede grabarse directamente debido a que los materiales fotosensibles solo responden a la intensidad de la luz. La Fig. 1, muestra el arreglo experimental para grabar un holograma de Fourier.

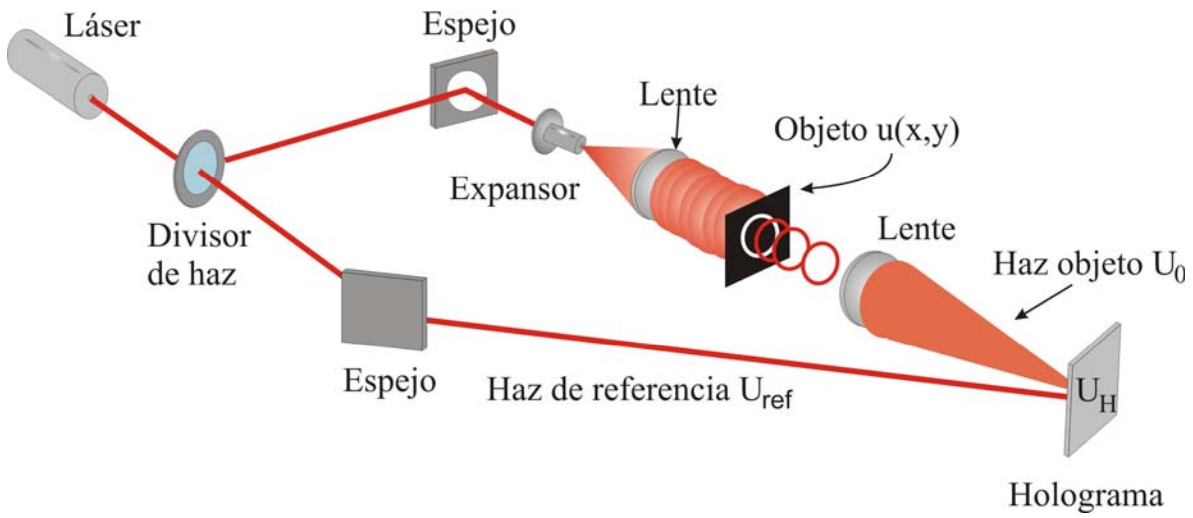


Figura 1. Impresión del Holograma de Fourier

Para mejorar la imagen del objeto transformado, se maximiza el valor de la ecuación (1) por medio de:

$$U_{\max} = \max\left[U_{FFT}(\nu,\mu)\right] \qquad (2)$$

donde el nuevo campo focal estará expresado por:

$$U_0(x,y) = \frac{U_{FFT}(\nu_i,\mu_j)}{U_{\max}} \qquad (3)$$

El grabado holográfico del objeto expresado por la ecuación (3) debe mezclarse con otra onda de referencia expresada por:

$$U_{ref}(x,y) = \frac{Ae^{ik\cdot r}}{r} \qquad (4)$$

la cual representa un frente de onda esférico con amplitud A y vector de posición ***r***. El grabado del holograma se realiza por la superposición de las ondas expresadas por la ecuación (3) y (4).

$$U_H = \left|U_{ref} + U_{\max}\right| \qquad (5)$$

La reconstrucción de la imagen se lleva a cabo mediante la exposición del holograma en un haz de referencia y colocándole una lente en uno de los haces difractados. La función de esta lente será realizar la transformada inversa de Fourier.

El FPGA es un circuito integrado digital de alta velocidad que contiene bloques lógicos programables. A diferencia de los procesadores comunes que tienen una estructura rígida, en el FPGA se diseña Hardware en lugar de programarlos y almacenar el código en una memoria como ocurre con los dispositivos procesadores comunes. Esta diferencia permite que el FPGA se configure para realizar procesamiento concurrente y obtener mayor velocidad en los procesos.

Hoy en día, muchos lenguajes de descripción de hardware (HDL), han emergido en los últimos años (Ulrich Heinkel, 2000). Estos lenguajes describen la implementación y generan una cadena de bits que configura a los FPGAs. Xilinx System Generator es una alternativa para generar código HDL. Este programa adicionado en el ambiente de Matlab/Simulink contiene bloques en alto nivel que implementan arquitecturas diseñadas eficientemente en el FPGA. Consecuentemente Matlab/Simulink y Xilinx System Generator ofrecen un ambiente ideal de prototipado virtual en el cual los modelos de hardware y software de un sistema pueden ser verificados al mismo tiempo (Le Thuyen, 1997; A. Toledo, 2005).

El propósito de este trabajo es describir el proceso en la cual el algoritmo de la FFT en dos dimensiones ($U_{FFT}(\nu_i,\mu_j)$) es obtenido, para su uso en la construcción de un holograma de Fourier. El diseño consistirá de una combinación de bloques suministrados por Xilinx System Generator, para el diseño y verificación del algoritmo de la FFT 2D en el FPGA.

METODOLOGÍA

El algoritmo que ha sido empleado para la obtención de la FFT 2D es el siguiente: primeramente se debe hacer el cálculo de la FFT en una dimensión sobre cada una de las filas de los elementos que conforman la matriz de la imagen. Posterior a ello, para obtener la FFT 2D es necesario aplicar nuevamente la función FFT sobre el resultado obtenido anteriormente, pero ahora sobre las columnas que conforman la imagen, esto representa el procedimiento observado de implementación partiendo de la ecuación (1).

Cuando incide un haz (una onda esférica o plana, U_{ref}), en el plano de Fourier y se superpone con la transformada de Fourier tal como se mostró en la Figura 1, se obtiene el holograma de Fourier. La reconstrucción del objeto contenido en el holograma se realiza por medio de la incidencia del haz y de la transformada inversa de Fourier.

Implementación de la FFT 2D en el FPGA

El diseño de la arquitectura para lograr la implementación del algoritmo del cálculo de la FFT 2D en el FPGA es presentada en la Figura 2.

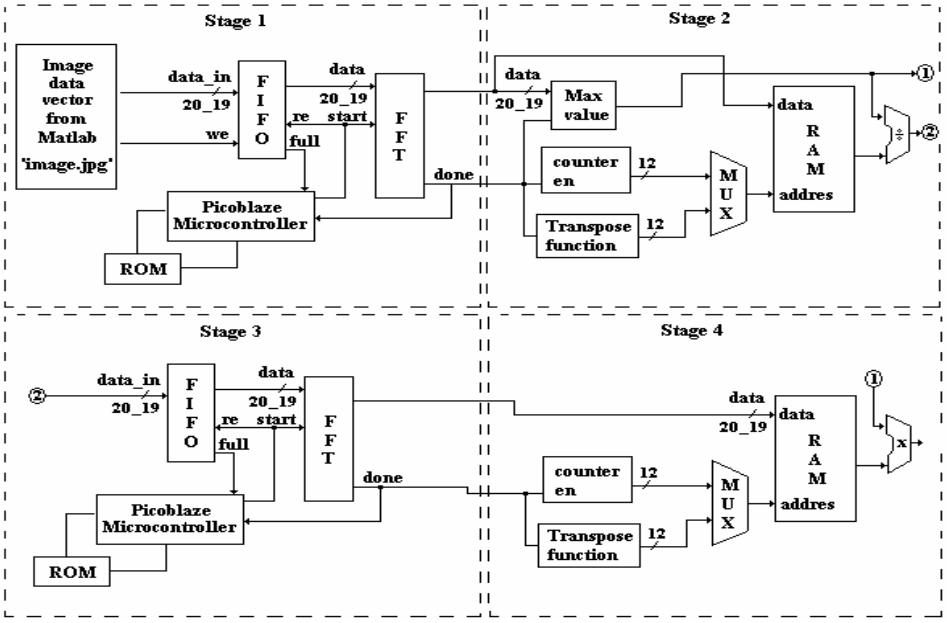


Fig. 2 Diseño de la Arquitectura FFT 2D implementada en el FPGA

Básicamente la arquitectura se divide en 4 etapas, la primera etapa almacena los datos que conforma la imagen. Esto se realiza con el envío serial de datos a través del puerto serial RS-232 de una computadora personal hacia el FPGA. La herramienta de envío de datos fue diseñada en los laboratorios de Mecatrónica de la Universidad Autónoma de Yucatán y básicamente permite elegir cualquier imagen, adecuarla en formatos de mapas de bits de 16x16, 32x32, 64x64 y 128x128 pixeles respectivamente y enviarlos a cualquier dispositivo que soporte el protocolo RS-232. Posteriormente, la misma herramienta recibe la imagen tratada que proviene del FPGA y la almacena de nueva cuenta en la computadora personal. Lo anterior se realiza para facilitar la captura y obtención de una imagen que proviene del exterior, pero se espera que la imagen sea suministrada a través de una cámara serial.

Una vez enviados los datos al FPGA, estos son almacenados en un elemento de memoria o buffer para que posteriormente se obtenga la FFT. La primera etapa es manipulada con un microcontrolador de 8 bits empotrado dentro del FPGA (PicoBlaze), el cual junto con la etapa 2 realiza la obtención de la FFT y el almacenamiento de cada una de las filas de la matriz de la imagen contenidos en el elemento de memoria. Posterior al cálculo de la FFT, nuevamente los datos ya manipulados, son almacenados en otra memoria. Si se trata de una imagen de 64x64 pixeles, donde cada píxel representa un nivel en la escala de grises y éste a un elemento de la matriz de la imagen, se tendría una memoria de almacenamiento de 4096 datos, considerando a cada dato como un elemento de 8 bits, dando un total de 256 niveles de grises. El bloque de cálculo de la FFT es obtenido por bloques

suministrados con la herramienta Xilinx System Generator (XSG), los cuales son instanciados desde la herramienta Simulink de Matlab.

La FFTx es un bloque Xilinx que implementa un algoritmo eficiente para calcular la transformada rápida de Fourier (FFT). Los N-puntos (donde, N = 2m, m = 4 – 14) de la FFT/IFFT se obtiene con un vector de N valores complejos representados con un bus de datos de 8, 12, 16, 20, ó 24 bits. La transformada de Fourier utiliza el algoritmo Cooley-Tukey. El bloque Xilinx FFTx es compatible solamente para los FPGAs Virtex-II, Virtex-II Pro y Spartan-3.

Las Figuras 3 y 4, muestran el bloque correspondiente a la etapa 1 de la arquitectura implementada en Simulink de Matlab y hace uso de la herramienta de Xilinx System Generator. El bloque azul de la Figura 3 se amplia y se presenta en la Figura 4, correspondiente al bloque del cálculo de la FFT.

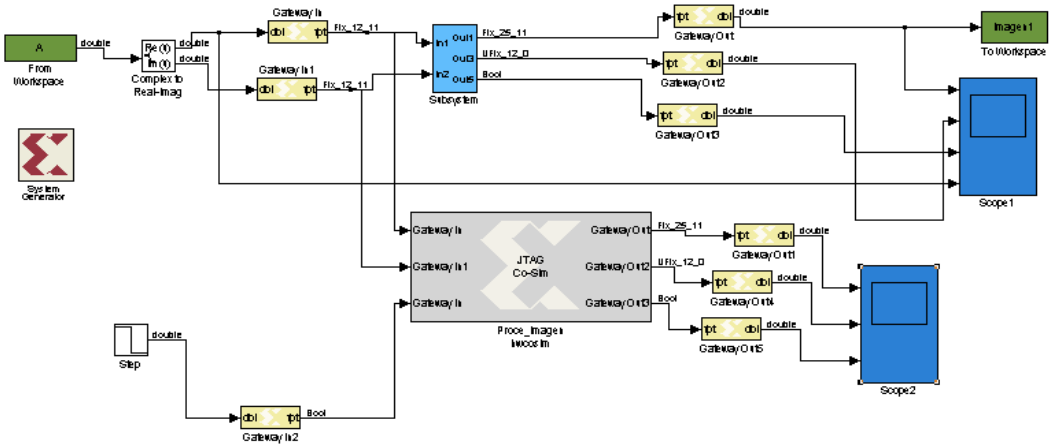


Figura 3. Implementación de la FFT con Xilinx System Generator en la plataforma Simulink

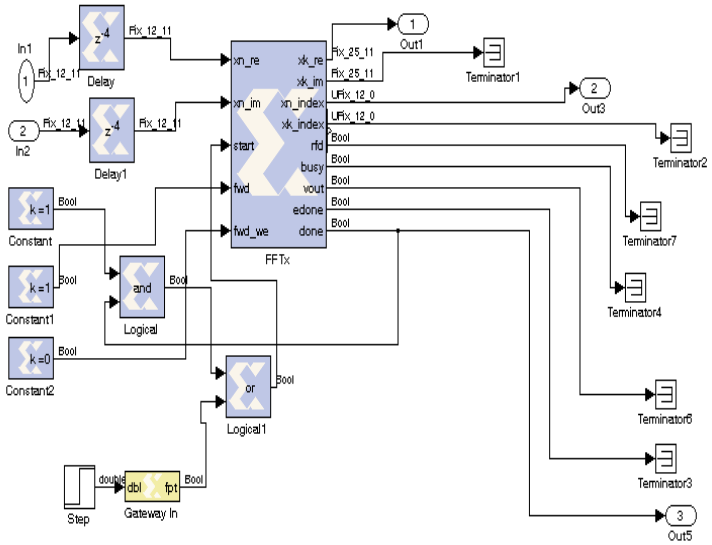


Figura 4. Bloque FFTx de Xilinx System Generator usado en las etapas 1 y 3 de la arquitectura

La etapa 2 se compone de un sub-bloque contador, el cual es el encargado de llevar la cuenta para el control de almacenamiento de los datos en la memoria; el sub-bloque valor máximo que identifica al elemento máximo de la imagen, la cual normaliza los datos ya que el bloque FFTx solamente permite la manipulación de datos que se encuentren en el rango de 0 a 1. Al final de la etapa 2 se observa que los

datos serán normalizados con ayuda del elemento máximo identificado en el sub-bloque valor máximo. Hasta ahora solamente se ha obtenido el cálculo de la FFT en 1D sobre las filas de la matriz, por lo que es necesario obtener la FFT sobre las columnas y así finalizar con la obtención de la FTT 2D.

Para poder realizar en cada columna el cálculo de la FFT sobre la matriz, es necesario obtener su transpuesta y posteriormente aplicar de nuevo la FFT como en la etapa 1. La transpuesta de la matriz es realizada de la siguiente manera.

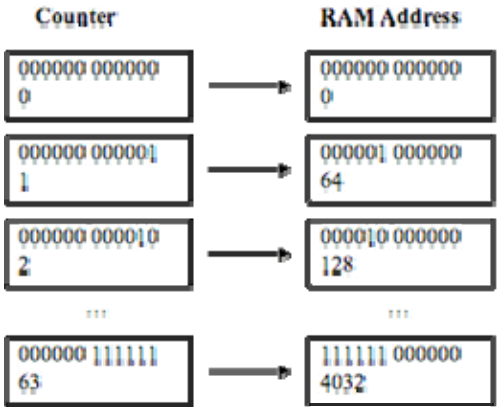


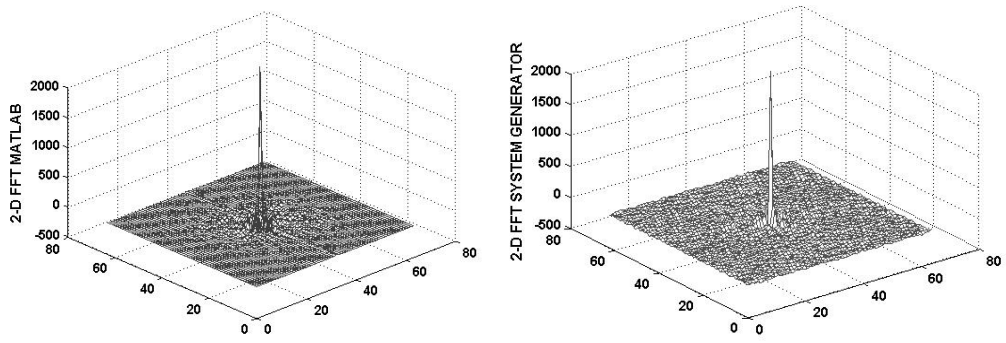
Figura. 5 Función transpuesta de la imagen

El almacenamiento en la memoria RAM se lleva a cabo mediante el sub-bloque contador, el cual manipula las direcciones de la memoria para almacenar un dato. Para obtener la transpuesta de los datos almacenados, simplemente se debe intercambiar los 6 bits menos significativos con los 6 bits más significativos de los 12 bits en total que conforma el bus de direccionamiento del sub-bloque contador (ver Figura 5). El reacomodo se realiza por medio del sub-bloque función transpuesta de la etapa 2 de la Figura 2. La salida de los datos de la etapa 2 son dirigidos a la entrada de la etapa 3, la cual es idéntica a la etapa 1. Una vez obtenida la FFT 2D de los datos de entrada, éstos deben obtener nuevamente su transpuesta, por lo que la etapa 3 es simular a la etapa 4, siendo la única diferencia que en ésta etapa los datos deben ser escalados con el valor máximo identificado en la etapa 2. Así finaliza el proceso de la obtención de la FFT 2D.

Ahora los datos pueden ser enviados a cualquier elemento externo que soporte comunicación serial RS-232, en nuestro caso los resultados obtenidos se regresan a la computadora personal encargada de proporcionar la imagen. La arquitectura presentada en la Figura 2, representa en bloques la obtención de la FFT en 2D de una imagen. Una vez que se realice el diseño de la arquitectura de cada uno de los bloques en Simulink de Matlab y de programar el microcontrolador encargado de llevar el control de todas las etapas de la arquitectura, se procede a su conversión a código en el lenguaje VHDL (Very High Description Language), donde posteriormente se realiza la síntesis del diseño que es la etapa en la cual cada una de la instrucciones escritas en VHDL son convertidas a compuertas lógicas, tablas de verdad y otros elementos.

RESULTADOS

Los resultados obtenidos del FPGA fueron comparados con los resultados obtenidos con la herramienta Simulink de Matlab. La comparación consistió en observar cada uno de los elementos que conforman las filas y columnas de la matriz de la simulación en Matlab. Los resultados son muy similares. La Figura 6 muestra un comparativo en magnitud de la FFT 2D con Matlab y la salida del procesamiento a través del FPGA identificado como 2D FFT System Generator.



6. (a) **6. (b)**

Figura 6. (a) Magnitud de la FFT obtenida mediante Simulink de Matlab (b) Magnitud de la FFT obtenida a través del FPGA

Los resultados observados en la Figura 6a, han tenido que ser manipulados debido a que cuando se obtiene la FFT 2D de una imagen, los componentes de mayor frecuencia se encuentran distribuidos en la esquinas de la matriz de salida, por lo que es necesario hacer un reacomodo e intercambiar los datos correspondientes a los cuadrantes pares e impares de la matriz de salida con el objetivo de colocar esos coeficientes en la posición central del arreglo matricial. En la Figura 7 se muestra el proceso que es necesario realizar para finalizar con la obtención de la FFT 2D.

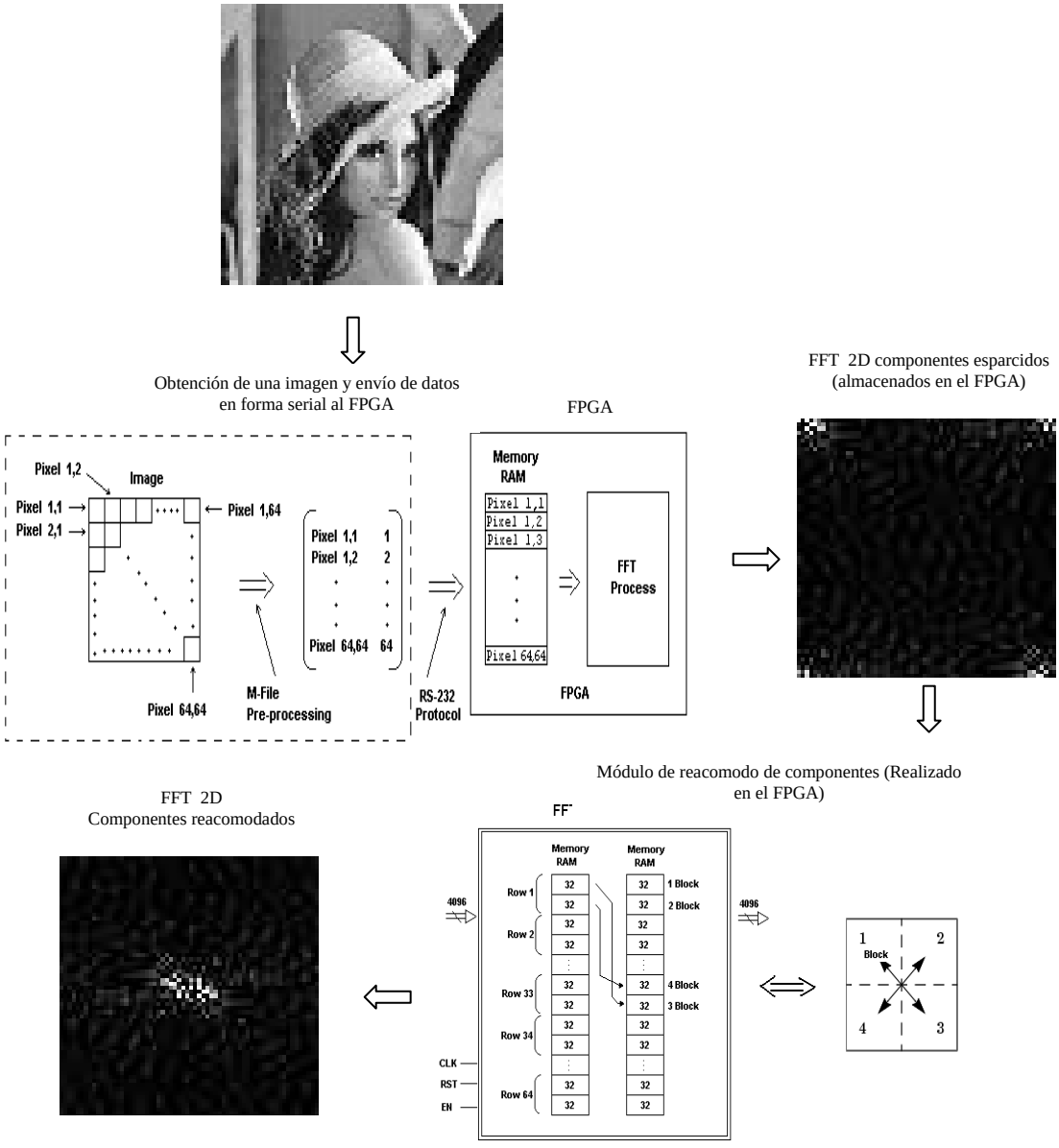


Figura 7. Procedimiento de reacomodo de componentes de la FFT 2D obtenida del FPGA

Para recrear la imagen original de los datos que contienen el resultado de la FFT 2D, fue necesario utilizar la función IFFT2 de Matlab, la cual es la función encargada de realizar la transformada inversa de Fourier en 2 dimensiones. El resultado obtenido se presenta en la Figura 8. Se puede notar que en la Figura 8b, la imagen es la misma que la imagen original excepto por la pérdida de alguno de los datos. Esto es debido a que la imagen tratada es muy pequeña, y donde las pequeñas pérdidas de

información representan errores visibles. Lo anterior ya es despreciable en las imágenes apartir de 128 x 128 pixeles.

Figura 8. Reconstrucción de la imagen. (a) Imagen original 64x 64 (b) Imagen reconstruida 64 x 64 (c) Imagen reconstruida128 x 128

Como puede observarse en la Tabla 1, el tiempo máximo de procesamiento de 1.35 ms por imagen, lo que nos da una tasa de procesamiento de imágenes de 740 imágenes por segundo que es un número considerable para otro tipo de aplicaciones como vídeo. En la tabla 1, se presentan los resultados de síntesis del FPGA.

Tabla 1. Reporte de síntesis del FPGA

Resultados de la Síntesis		
Característica	**Imágenes 64 x 64**	**Imágenes 128 x 128**
Número de Flip Flops	55%	60%
Número de IOBs	14%	61%
Área del FPGA utilizado *	65%	83%
Tiempo de procesamiento	1.35 ms	6.1109 ms
Máxima Frecuencia	115.6 MHz	114.936MHz

*Virtex2 cx2v2000-6ff896

CONCLUSIONES

En este artículo se presentó la implementación de la FFT para construir hologramas digitales con FPGAs y se propuso una solución para reconstruir los hologramas de Fourier. La simulación del proceso y la programación en VHDL para la síntesis del FPGA trabajan adecuadamente en monitores de computadora sin la necesidad del CPU. Las adaptaciones de una pantalla LCD con el FPGA serán realizados en un trabajo futuro. El resultado de la Fig. 8 representa que a partir de imágenes de 128 x 128 pixeles las posibles pérdidas de datos en el procesamiento son despreciables. Actualmente se está trabajando en la representación del haz de referencia y en la recreación del holograma utilizando técnicas de diseño digital en VHDL y se esperan obtener resultados muy pronto.

REFERENCIAS BIBLIOGRÁFICAS

A. Olivares, M. Pérez (2002). Dynamic diffraction ring projector by doped polymer dispersed liquid crystal, Optical Letters, 2, 1020-1024.

A. Toledo. (2005). Experiences on developing computer vision harware algorithms using xilinx system generator, Microprocessors and Microsystems 29, p. 411-419, Elsevier.

D. Gabor. (1948). A new microscopic principle, Nature, 161, 780-3.

Hetch. (2004). Optics, USA: Adison Wesley.

J. W. Goodman. (2003). Introduction to Fourier Optics, McGraw Hill, New York.

Le Thuyen, Renner Frank, Glesner M.(1997). "Hardware in-the-loop Simulation –A Rapid Prototyping Approach for Designing Systems", Proceedings, 8th IEEE International Workshop on 24-26 June. Page(s) 116-121.

Lohmann. (1967). Binary Fraunhofer holograms generated by computer, Applied Optics, 6, 1739-48.

Ulrich Heinkel. (2000), "The VHDL Reference: A Practical Guide to Computer-Aided Integrated Circuit Design including VHDL-AMS", Chichester, New York: Wiley.

Este documento se debe citar como:
Castillo-Atoche A., Pérez-Cortés M., Ortíz-Gutiérrez M., Vázquez-Castillo J. (2006). **Implementación de la FFT en hologramas de Fourier generados con FPGAs**. Ingeniería, Revista Académica de la FI-UADY, 10-2, pp.47-55 ISSN: 1665-529X