ARQUITECTURA PIPELINE DE LA FFT BIDIMENSIONAL EN FPGA

Eduardo Magdaleno Castelló, Manuel Rodríguez Valido, José Manuel Rodríguez Ramos, Alejandro Ayala Alfonso.
emagcas@ull.es, mrvalido@ull.es, jmramos@ull.es, aayala@ull.es.
Dpto. de Física Fundamental, Experimental, Electrónica y Sistemas. Universidad de La Laguna
Facultad de Física, Avd. Francisco Sánchez s/n, 38203 La Laguna.

***Abstract*- The purpose of the present work is the design and implementation of an electronic device that allows us to obtain pipeline 2-D fast Fourier transform (FFT) using a Field Programmable Gate Array (FPGA). The basic design methodology followed in the development of this digital system has been to create a hardware description language (VHDL) code. VHDL blocks ensure portability, scalability, configurability and technology independence. The basic advantages of FPGA technology are flexible architecture and extremely high-performance signal processing capability through parallelism.**

The implemented 2D-FFT architecture results faster than the architecture that Uzun et al. provide. The implemented algorithm meets current and future adaptive optics image processing frame rate requirements.

I. INTRODUCCIÓN

Los sistemas electrónicos y en general el diseño electrónico han encontrado un importante nicho o soporte en las FPGAs. En aquellos problemas que se requiere alta velocidad de procesamiento a bajo coste, estos dispositivos superan claramente en muchos problemas a procesadores de propósito general de última generación, debido a que estos dispositivos tienen una arquitectura flexible y muchas fases del cálculo global de un algoritmo pueden realizarse de forma completamente paralela [1-2].

La transformada discreta de Fourier (DFT) y su implementación rápida (FFT) juegan hoy en día un papel muy importante en el campo del procesado digital de señales. En concreto, la implementación de una transformada bidimensional (2D-FFT) se emplea en cualquier disciplina científica en la que se desea analizar imágenes digitales.

Por otra parte, ha surgido la necesidad de realizar lo más rápidamente posible la 2D-FFT en el campo de la astronomía y óptica adaptativa. Efectivamente, en el mundo astrofísico, en donde la calidad de las imágenes recibidas se ve afectada debido a la degradación del frente de onda producido por la turbulencia atmosférica, la fase original se puede recuperar mediante el sensor Shack-Hartmann, que obtiene los gradientes de fase de la imagen, y un procesado de imagen que contiene varias FFT en dos dimensiones. El cálculo de este algoritmo debe realizarse en menos de 10 ms, que es el tiempo de variación atmosférica debido a turbulencias [3-4].

Con el propósito de agilizar la implementación de sistemas electrónicos de procesado digital, el fabricante de FPGAs Xilinx proporciona, en la versión 8.2 de su CoreGenerator, que aparece en la misma versión del entorno de desarrollo ISE Foundation, un *core* para implementar de manera eficiente sobre Virtex-4 varias arquitecturas de la transformada rápida de Fourier [5].

Desafortunadamente, esta librería únicamente proporciona transformadas unidimensionales, por lo que en este trabajo se presenta el diseño de una arquitectura que calcula de manera eficiente la transformada bidimensional. Esta arquitectura es más rápida que otras implementaciones en FPGA, debido a que es completamente *pipeline*. Así, tras un determinado tiempo de latencia, el módulo diseñado va obteniendo cada transformada de manera continua en un número de ciclos de reloj que coincide con el tamaño de la imagen.

II. ARQUITECTURA IMPLEMENTADA

El esquema del módulo transformada implementado tiene el aspecto de la figura 1. El funcionamiento del sistema desarrollado es el siguiente: los datos de la imagen se reciben de manera serie por filas y se van introduciendo un bloque que realiza la FFT unidimensional (directa o inversa). A medida que este módulo va sacando los datos de las transformadas, estos se van almacenando en dos memorias de doble puerto (para la parte real e imaginaria). Para completar la 2D-FFT, los datos almacenados en la memoria anterior se van introduciendo en un segundo bloque FFT pero en columnas, de tal manera que, finalmente, a la salida de este bloque se obtiene la transformada bidimensional deseada.

A. Módulo FFT 1-D

Para implementar los bloques FFT se ha usado un *core* que proporciona Xilinx para realizar transformadas unidimensionales. Este *core* dispone de tres arquitecturas para implementar la transformada, las de *radix-2* (recursos mínimos) y *radix-4*, que constan de fase de carga/descarga de datos y fase cómputo, y la arquitectura *pipeline*. Todas realizan el cálculo de la 1D-FFT usando el algoritmo de Cooley and Tukey [6]. Se ha optado por la arquitectura *pipeline* ya que, aunque ocupa más espacio en la FPGA, es la única opción que permite un procesamiento de datos

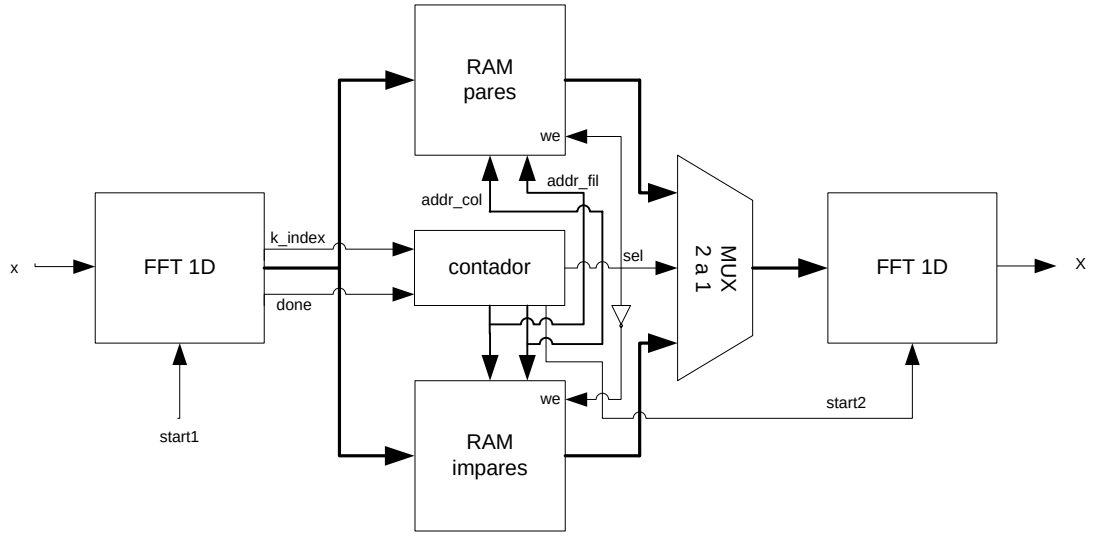

Fig. 1. Esquema de la transformada bidimensional implementada.

continuo. De esta manera, el número de ciclos de reloj en los que se obtiene cada una de las transformadas es, directamente, el número de puntos de los datos cuya transformada se desea calcular.

El módulo permite escalar los datos en cada etapa del proceso de cálculo de la transformada, ya que los datos pueden experimentar un cierto crecimiento intrínseco a los cálculos que conllevan las mariposas de la transformada. No obstante, en esta implementación se ha preferido no truncar los datos, para no propagar el error de truncamiento a la segunda etapa del módulo. Con esta opción también se eliminan posibles desbordamientos u *overflow*. Para tener en cuenta este crecimiento, los datos a la salida contienen un mayor número de bits que a la entrada del módulo, lo que hace que los dos bloques FFT de los que consta el diseño no sean idénticos. La precisión de salida de los datos viene dada por el siguiente cálculo:

$$ancho\,salida = ancho\,entrada + \log_2 n^o\ puntos + 1 \qquad (1)$$

Así, para una transformada 8x8, si primer bloque tiene 16 bits de datos de entrada y 20 de salida, mientras que la FFT de las columnas tiene 20 bits de entrada 24 de salida.

La implementación de las FFT a través de este *core* se basa en el uso de módulos DSP48 que se encuentran distribuidos en la FPGA. Estos módulos son circuitos aritméticos dedicados que realizan eficientemente operaciones tales como multiplicaciones, sumas y restas [7-8].

B. Pipeline con memoria duplicada

Los valores obtenidos en la FFT de las filas se almacenan según van saliendo en una memoria RAM (por ejemplo la RAM par), que consta en realidad de dos memorias de doble puerto, una para la parte real y otra para la parte imaginaria, como ya se ha comentado. Para ir direccionando la memoria en modo escritura, se usa un contador. El *core* de Xilinx proporciona un índice de los datos de salida, denominado *k_index*, por lo que el contador extiende añadiendo bits adicionales para direccionar todos los elementos de la matriz. Este contador se activa cuando se recibe la señal *done* de la FFT de las filas, que indica que se van a recibir los datos transformados. El contador proporciona también el direccionamiento de las columnas que se consigue fácilmente conmutando los bits más significativos de la cuenta por los menos significativos.

Puede apreciarse que, con una sola memoria, no es posible ir introduciendo los datos en la segunda FFT (de las columnas) a medida que se reciben datos de la transformada de las filas, pues estos datos sobrescribirían la memoria antes de que se proporcionaran los datos a la segunda etapa. Por otra parte, si esperásemos a dicha introducción se perdería la interesante propiedad *pipeline* del módulo FFT de Xilinx. Para solventar este problema se introdujo dos memorias en lugar de una, mientras se van almacenando las transformadas de las imágenes pares, se va introduciendo los datos de las imágenes impares en el segundo bloque. De esta manera, el flujo de datos permanece continuo en todo el cálculo de la transformada bidimensional. Los modos de escritura y lectura de las dos memorias son siempre alternos y los gobierna el propio contador. Esta misma señal conmuta el multiplexor que selecciona los datos que entran en la transformada de las columnas.

Este módulo y el resto de bloques básicos que aparecen en la figura 1 se realizaron con el lenguaje de descripción de hardware VHDL [9]. El módulo implementado se generalizó para poder realizar la transformada inversa, cambiar la precisión y el tamaño de la imagen, a través de los genéricos *fwd_inv*, *data_width* y *lognpoints* (el logaritmo en base 2 del número de elementos de cada fila/columna, que coincide con el número de etapas en el cálculo de una transformada unidimensional), respectivamente.

III. PRINCIPIOS DE OPERACIÓN

En la figura 2 puede verse el cálculo continuo de la transformada bidimensional directa para un tamaño de 8x8 y16 bits de precisión (la señal *start* permanece en alta en todo momento). El bloque contiene las clásicas señales de

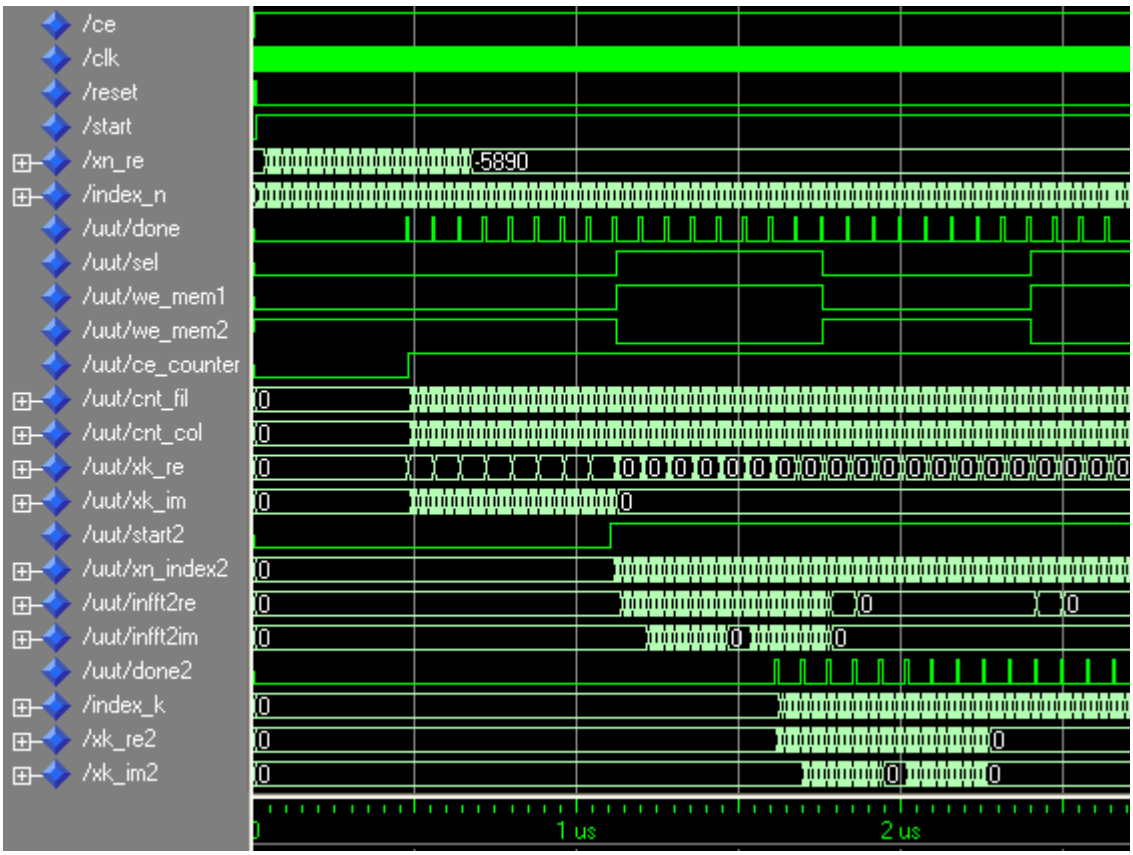

Fig. 2. Simulación funcional de un 2D-FFT de 8x8 puntos.

reset, *ce* y *clk* que no se incluyeron en la figura 1 para no complicar en exceso el esquema. Cuando *start* se activa se van introduciendo los datos (en este caso reales, *xn_re*) y se lleva a cabo una indexación de los datos (*index_n*). En el instante en el que el módulo va a sacar la transformada de la primera fila de la imagen (*done* en alta), se activa el contador (*ce_counter*), para almacenar los datos en una de las memorias, con lo que se direcciona la memoria (*cnt_col* y *cnt_fil*). Puede apreciarse que las señales *we_mem1* y *we_mem2* son siempre complementarias (mientras se lee en una, se escribe en la otra). Una vez introducidos todos los datos (*cnt_fil*=63), se activa la segunda FFT (*start2*=1) y se introducen los datos en columnas (señales *infft2re* e *infft2im*). Al igual que con el primer módulo, cuando se van a sacar los datos de la primera columna el *core* activa una señal (*done2* en este caso), y se inicia la indexación de los datos (*index_k*) y la presentación de los mismos (*xk_re2* y *xk_im2*).

IV. Resultados

El módulo diseñado para realizar la transformada 8x8 anteriormente comentado emplea 162 ciclos de reloj en realizar la transformada bidimensional. Como el sistema es *pipeline*, cada 64 ciclos se obtiene una nueva transformada. Con un reloj de 100 MHz esto implica 1620 ns y 640 ns respectivamente.

Los resultados numéricos fueron comparados satisfactoriamente con simulaciones realizadas con Matlab. Se sintetizaron transformadas de varios tamaños y precisión sobre la Virtex-4 XC4VSX35. Este dispositivo consta de 15360 slices 30720 flip-flops, 192 BRAM y 192 DSP48s. Para realizar el diseño se empleó en entorno de desarrollo ISE Foundation 8.2 y el sintetizador XST de Xilinx [10].

La tabla 1 presenta los recursos empleados para implementar una 2D-FFT *pipeline* cuando los datos de entrada tienen una precisión de 8 bits. La transformada más grande que puede implementarse es de 128x128 puntos. Puede observarse que, a medida que la transformada a implementar es mayor, la frecuencia máxima a la que puede operar la FPGA disminuye. La columna ciclos muestra el tiempo de latencia. Después de este periodo, los ciclos de reloj de obtención de cada transformada es el número de píxeles de la imagen, conservando esta propiedad de la transformada *pipeline* de una dimensión que suministra Xilinx. Puede apreciarse (tabla 2), que para una precisión de 16 bits, se necesitan más recursos y la frecuencia máxima de funcionamiento es menor, debido al aumento en la complejidad intrínseco al aumento del ancho de los datos.

Para la transformada de 128x128 se tarda en realizar la operación 170.84 us, esto es, 5853 imágenes por segundo. Uzun et. Al. realizaron diferentes implementaciones de 2D-FFT paralelas modificando en número de etapas implementadas sobre una Virtex-2000E [11]. La más rápida de sus implementaciones para una transformada del mismo tamaño fue de 420 imágenes por segundo. Nuestra implementación resulta más de diez veces más rápida.

V. Conclusiones

Los resultados de las tablas 1 y 2 muestran que la arquitectura de transformada bidimensional implementada maneja unos

2D-FFT 8 bits	Slices	Slices Flip-Flops	RAMB16	DSP48s	Fmax	Ciclos	Duración [a 100 MHz]
8x8	1141 (7%)	1958 (6%)	4 (2%)	6 (3%)	305.599	162	1620 ns
16x16	1667 (10%)	2839 (9%)	4 (2%)	6 (3%)	304.404	412	4120 ns
32x32	2855 (18%)	4517 (14%)	4 (2%)	15 (7%)	292.757	1304	13.04 µs
64x64	4613 (30%)	6318 (20%)	16 (8%)	15 (7%)	189.519	4516	45.16 µs
128x128	8135 (52%)	9755 (31%)	64 (33%)	25 (13%)	161.300	17084	170.84 µs

Tabla 1. Recursos empleados en la Virtex-4 XC4VSX35 para una precisión de entrada de 8 bits.

2D-FFT 16 bits	Slices	Slices Flip-Flops	RAMB16	DSP48s	Fmax	Ciclos	Duración [a 100 MHz]
8x8	1650 (10%)	2809 (9%)	4 (2%)	10 (5%)	285.193	162	1620 ns
16x16	2302 (15%)	4076 (13%)	4 (2%)	10 (5%)	278.384	412	4120 ns
32x32	3992 (25%)	6315 (20%)	8 (4%)	22 (11%)	195.223	1304	13.04 µs
64x64	6472 (42%)	8870 (28%)	24 (12%)	22 (11%)	165.272	4516	45.16 µs
128x128	11326 (73%)	13522 (44%)	24 (12%)	36 (18%)	161.300	17084	170.84 µs

Tabla 2. Recursos empleados en la Virtex-4 XC4VSX35 para una precisión de entrada de 16 bits.

tiempos de cálculo sensiblemente menores a los 10 ms de límite que se dispone para recuperar la fase de imágenes astronómicas distorsionadas debido a la turbulencia atmosférica. Puede apreciarse que la mayor transformada que puede realizarse usando la arquitectura *pipeline* propuesta en esta FPGA es de 128x128. En un futuro será necesario realizar transformadas mayores, por lo que se va a adquirir una Virtex-5 XC5VSX50T. Este dispositivo cuenta con 52224 slices, 32640 slices flip-flops, 288 DSP48s y 264 BRAM blocks, operando a una frecuencia hasta de 550 MHz [12]. Al tener un mayor nivel de integración, consta de un número de recursos más elevado y permitirá realizar transformadas mayores.

Próximamente se realizará el algoritmo completo de recuperación de fase. Éste consta de dos transformadas bidimensionales directas, un filtro y una transformada inversa. El esquema del sistema a implementar puede verse en la figura 3.

AGRADECIMIENTOS

Este trabajo está subvencionado parcialmente por "Programa Nacional de Diseño y Producción Industrial" (Project DPI 2006-07906) del "Ministerio de Ciencia y Tecnología" y por la "European Regional Development Fund" (ERDF).

REFERENCIAS

[1] J. Deschamps, G. Bioul, G. Sutter, "Synthesis of Arithmetic Circuits. FPGA, ASIC and Embedded Systems", Wiley-Interscience, 2006.
[2] U. Meyer-Baese, "Digital Signal Processing with Field Programmable Gate Arrays", Springer-Berlag, 2001.
[3] L. A. Poyneer, D. T. Gavel, and J. M. Brase, "Fast wave-front reconstruction in large adaptive optics systems with use of the Fourier transforms", J. Opt. Soc. Am. A 19, pp. 2100-2111, 2002.
[4] F. Roddier and C. Roddier, "Wavefront reconstruction using iterative Fourier transforms", Applied Optics 30, pp. 1325-1327, 1991.
[5] Xilinx, "Fast Fourier Transform v3.2", 2006.
[6] J. W. Cooley and J. W. Tukey, "An algorithm for the machine calculation of complex Fourier series", Mathematics of Computation 19, pp. 297-301, April 1965.
[7] Xilinx, "XtremeDSP for Virtex-4 FPGAs user guide", 2006.
[8] G. C. Hawkes, "DSP: Designing for Optimal Results. High-Performance DSP Using Virtex-4 FPGAs", 2005.
[9] IEEE Standard VHDL Language Reference Manual, IEEE-1076-2000, 2000. 11.
[10] Xilinx, "XST User Guide", pp. 118-217, 2006.
[11] I. S. Uzun, A. Amira and A. Bouridane, "FPGA implementations of fast Fourier transforms for real-time signal and image processing", IEE Proc.- Vis. Image Signal Processing, vol. 152, no. 3, pp. 283-296, 2005.
[12] Xilinx, "Virtex-5 Family Overview. LX, LXT and SXT Platforms", pp. 2-6, December 2007.

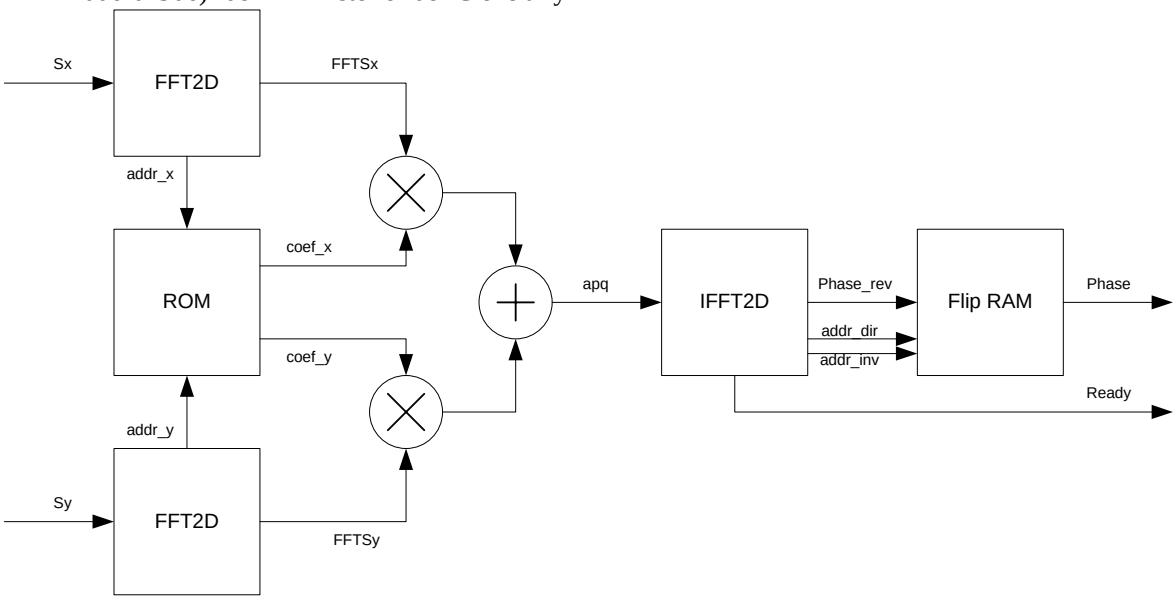

Fig. 3. Diagrama de bloques del recuperador de fase. Se han omitido las señales de clock, reset y chip enable por claridad.