DESARROLLO DE SISTEMAS DE PROCESADO DIGITAL DE SEÑAL SOBRE DISPOSITIVOS PROGRAMABLES

P. Brox, S. Sánchez-Solano, I. Baturone, A. Barriga

Instituto de Microelectrónica de Sevilla, Centro Nacional de Microelectrónica
Avda. Reina Mercedes s/n, 41012 Sevilla, España. brox@imse.cnm.es

RESUMEN

Las FPGAs se han convertido en una alternativa competitiva para el desarrollo de sistemas de procesdo digital de señal (DSP) para aplicaciones específicas, frente al empleo de los tradicionales DSPs de propósito general o el diseño de estos dispositivos mediante ASICs. En esta comunicación se describe un flujo de diseño de DSPs sobre FFGAs basado en la herramienta *System Generator* de Xilinx, se presenta la implementación en hardware reconfigurable de un algoritmo de desentrelazado de imágenes de vídeo y se analizan, en términos de área y velocidad, diferentes arquitecturas diseñadas.

1. INTRODUCCIÓN

Los constantes avances en el desarrollo de dispositivos lógicos programables como la FPGAs (*Field-Programmable Gate Arrays*) han propiciado la utilización de este tipo de elementos para implementar sistemas de procesado digital de señal (DSP) de alto rendimiento, especialmente en áreas como las comunicaciones [1] y el tratamiento de imágenes de vídeo [2]. La estructura lógica de las FPGAs actuales incluye no sólo un elevado número de bloques lógicos configurables (tablas look-up, registros, multiplexores, etc.), sino también circuitos dedicados que implementan sumadores, multiplicadores y bloques de memoria. Estas características hacen que el empleo de FPGAs presente numerosas ventajas para el desarrollo de aplicaciones específicas de procesado digital de señal frente al uso de los DSPs tradicionales, cuya funcionalidad puede programarse mediante software pero que disponen de una estructura hardware poco flexible: tamaño de datos predeterminado, capacidad de memoria fija y número limitado de bloques acumuladores [3].

En los últimos años se han desarrollado una serie de herramientas, como el entorno *System Generator* (*SysGen*) de Xilinx, que facilitan el diseño de algoritmos DSP sobre FPGAs [4]. *SysGen* es una herramienta software para la descripción y síntesis de sistemas de procesado de señal basado en *Simulink* (la herramienta interactiva para el modelado, la simulación y el análisis de sistemas dinámicos integrada en Matlab). El entorno permite manejar un nivel de abstracción elevado para el diseño DSP y proporciona un entorno gráfico que facilita la descripción hardware del algoritmo que se va a implementar. Una vez verificada la funcionalidad del diseño, *SysGen* lo traslada de forma automática en una implementación hardware optimizada en términos de área y velocidad.

Por otra parte, la disponibilidad de placas de prototipado que facilitan la implementación física de los diseños, permite recorrer de forma rápida todo el ciclo de desarrollo y verificación del sistema. Estas plataformas suelen incluir dispositivos lógicos programables, junto a una serie de periféricos que permiten ampliar la funcionalidad de las mismas. Los

Este trabajo ha sido parcialmente financiado por el proyecto TIC2001-1726 (CICYT).

principales fabricantes de placas y dispositivos programables ofrecen, a través de sus programas universitarios [5] [6], paquetes de desarrollo que incluyen la placa, las herramientas software más indicadas para la programación de nuevos diseños y una serie de demostradores y notas de aplicación de gran valor didáctico. La adquisición de este material a través de dichos programas universitarios permite una reducción del costo y el acceso a documentación relacionada con aplicaciones, consultas y forums de discusión.

En esta comunicación se describe una experiencia de diseño de algoritmos de DSP sobre FPGAs donde se analizan, en términos de área (ocupación) y velocidad de operación, diferentes arquitecturas para la implementación de un algoritmo de desentralazado de imágenes de video. En la Sección 2 se presentan la plataforma de desarrollo, las herramientas utilizadas y el flujo de diseño. A continuación se detalla la aplicación utilizada como ejemplo de diseño. Posteriormente, se analizan las distintas arquitecturas estudiadas y, finalmente, se exponen los resultados y conclusiones del trabajo.

2. PLATAFORMA DE DESARROLLO, HERRAMIENTAS Y FLUJO DE DISEÑO

La herramienta *System Generator* (*SysGen*) de Xilinx facilita el desarrollo de los algoritmos de DSP sobre FPGAs [7]. Dicha herramienta incluye la librería de módulos "*Xilinx Blockset*" para *Simulink*, así como el software necesario para trasladar el modelo matemático descrito en *Simulink* a un modelo hardware descrito en lenguaje VHDL. La descripción VHDL permite verificar la funcionalidad del diseño con la herramienta *ModelSim* de Mentor Graphics y constituye la entrada a las herramientas de síntesis e implementación de FPGAs. *SysGen* chequea los parámetros indicados en cada uno de los bloques de *Simulink* y determina las entidades, arquitecturas, puertas, señales y atributos de la descripción hardware en VHDL. Adicionalmente, *SysGen* genera un proyecto para completar el diseño en el entorno de FPGAs ISE de Xilinx. De este modo el código VHDL obtenido puede ser sintetizado con la herramienta XST integrada en ISE, aunque también puede ser exportado a otra herramienta de síntesis. En el ejemplo de diseño desarrollado la síntesis se ha realizado con la herramienta *FPGA Compiler II* de Synopsys. La implementación del diseño se realiza con las herramientas de Xilinx incorporadas en ISE en cualquier caso. La Figura 1 muestra las diferentes etapas de diseño de un sistema con esta herramienta.

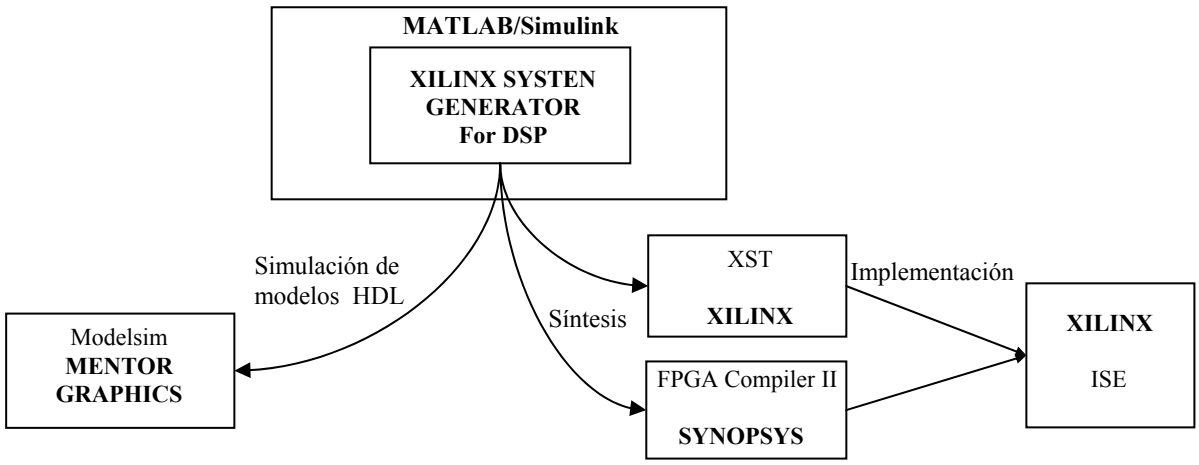


Figura 1. Flujo de diseño de DSPs con la herramienta *SysGen* de Xilinx.

Como plataforma de desarrollo se ha empleado una placa RC100 de Celoxica. Dicha placa dispone de una FPGA Spartan II y un CPLD XCR3128XL, ambos de Xilinx. La placa

incorpora asimismo dos bancos de memoria SSRAM independientes con capacidad para almacenar 256k x 36 bits. El sistema incluye dos entradas de vídeo, una correspondiente a vídeo compuesto (CVBS) y la otra a supervídeo, y un chip decodificador de vídeo (el SAA7111 de Philips) capaz de capturar vídeo en formato NTSC y PAL. Todas las señales de control están directamente conectadas a los terminales de la FPGA. Por último, la placa permite generar una salida VGA que facilita la visualización de los datos en un monitor o proyector de vídeo. Para ello, incorpora un convertidor DA que convierte la salida digital de la FPGA en las señales analógicas apropiadas que se transmiten a través del conector VGA.

Junto con la placa, Celoxica proporciona un entorno para el desarrollo de hardware a partir del lenguaje Handel-C (DK1). Asimismo, proporciona una librería de macros y funciones para la plataforma RC100 que facilitan el diseño de los sistemas que se implementan en dicha placa [8]. Esta librería incluye macros para el acceso de escritura y lectura de las SRAMs, así como drivers para el convertidor DA que proporciona la salida VGA y el decodificador de vídeo SAA7111.

El diseño global de la aplicación, descrito en Handel-C, establece la comunicación de la FPGA con los restantes dispositivos de la placa e incorpora la descripción hardware del algoritmo, obtenida con *SysGen* y sintetizada con la herramienta *FPGA Compiler II* de Synopsys. El fichero EDIF que contiene el resultado de la síntesis se introduce en el flujo de diseño de DK1. Para llevar a cabo esta tarea es necesario definir una interfaz similar a una "caja negra" que tenga las mismas entradas y salidas que el diseño de *SysGen*. Los diseños generados con los entornos de Celoxica y Xilinx son combinados en la etapa final de implementación del sistema. La Figura 2 ilustra el flujo de diseño indicando las herramientas utilizadas.

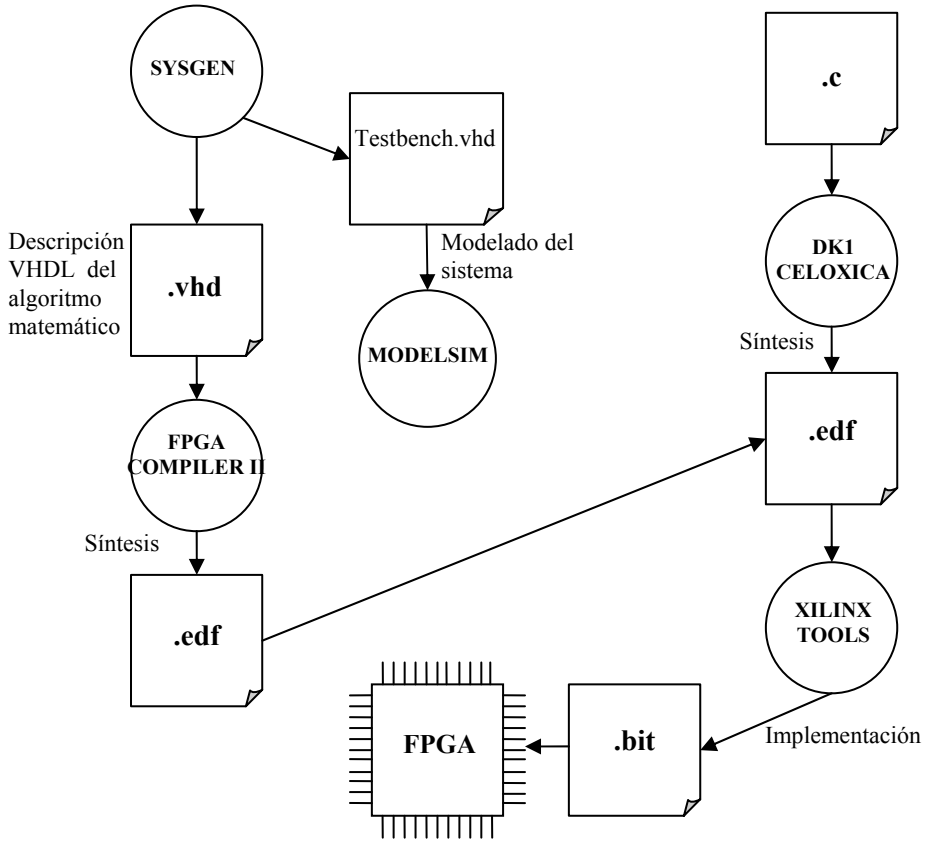


Figura 2. Flujo de diseño realizado indicando las herramientas utilizadas.

3. APLICACIÓN A UN SISTEMA DE DESENTRELAZADO DE VÍDEO

Para incrementar la cantidad de información enviada a través de un canal de transmisión, la industria televisiva utiliza un formato de video entrelazado donde los fotogramas de orden impar contienen solamente las líneas impares y los de orden par sólo incluyen las líneas pares de la imagen. Aplicando diferentes métodos de interpolación, las técnicas de desentrelazado permiten reconstruir un fotograma completo en el receptor, con todas las líneas pares e impares. Existen dos técnicas básicas de desentrelazado de imágenes, las denominadas *"intra-fields"* donde se interpola con datos pertenecientes a un mismo fotograma (interpolación espacial) y las técnicas *"inter-fields"* donde intervienen datos de los fotogramas anteriores y posteriores (interpolación temporal) [9].

La luminancia I(x,y,t) correspondiente a cada píxel de la imagen en un sistema de transmisión de vídeo viene dada por una función que depende de tres variables, x e y son las coordenadas espaciales que determinan el píxel del fotograma, mientras que la tercera variable determina el orden de secuencia del fotograma. Las dos técnicas básicas de desentrelazado (inserción de campo y promediado de líneas) corresponden a la aplicación de las siguientes expresiones:

$$I_T(x,y,t) = I(x,y,t-1) \qquad (1)$$

$$I_S(x,y,t) = \frac{I(x,y-1,t) + I(x,y+1,t)}{2} \qquad (2)$$

La interpolación espacial proporciona buenos resultados cuando existe movimiento de objetos pero provoca distorsiones visuales en fondos de imagen estáticos. Por otro lado, con la inserción de líneas se obtienen mejores resultados en las zonas estáticas aunque en las zonas con movimiento la imagen puede aparecer borrosa. Por este motivo se han desarrollado diversos métodos que combinan ambas técnicas en función del grado de movimiento, de modo que, si existe movimiento predomina la interpolación espacial, mientras que si no existe lo hace la temporal [10] [11]. Si denominamos $\gamma(x,y,t)$ a la salida del detector de movimiento (un valor comprendido en el intervalo [0,1]), el valor de la luminancia de un píxel de coordenadas (x,y,t) de la imagen desentrelazada vendrá dado por la expresión (3):

$$I(x,y,t) = (1-\gamma(x,y,t))\, I_T(x,y,t) + \gamma(x,y,t)\, I_S(x,y,t) \qquad (3)$$

Algunos métodos de desentrelazado, como el propuesto por Van de Ville et al., utilizan técnicas de inferencia basadas en lógica difusa para evaluar la condición de movimiento a partir de un conjunto de reglas descritas de forma lingüística [12]. El algoritmo utilizado como ejemplo de diseño en esta comunicación corresponde a un algoritmo de desentrelazado adaptativo basado también en un detector difuso de movimiento, pero que emplea técnicas de convolución para reducir el coste computacional e incrementar la velocidad de ejecución [13].

Como entrada al detector de movimiento se utiliza una matriz de diferencias, H(x,y,t), cuyos elementos se calculan como la diferencia entre los valores anterior y posterior de la luminancia para un píxel con las mismas coordenadas (x,y):

$$H(x,y,t) = \frac{|I(x,y,t+1) - I(x,y,t-1)|}{2} \qquad (4)$$

Las reglas difusas que rigen el comportamiento del detector combinan en sus antecedentes los valores de diferencias correspondientes a varios elementos vecinos para obtener una medida de certidumbre de la presencia de movimiento. Las reglas difusas

establecen que el movimiento está presente (M(x,y,t)=TRUE) cuando el valor de H(x,y,t) en el píxel actual o en los píxeles situados a ambos lados del píxel actual es "*Large*", siendo "*Large*" la etiqueta lingüística que representa al conjunto difuso ilustrado en la Figura 3a.

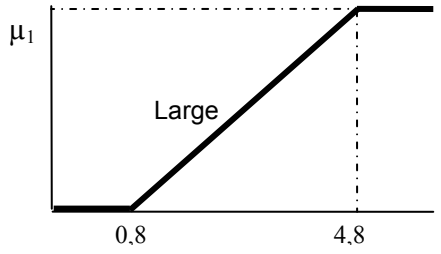
La salida del detector tiene en cuenta asimismo las condiciones de movimiento correspondientes a las líneas superior M(x,y+1,t-1) e inferior M(x,y-1,t-1) del frame anterior. El valor numérico final, γ(x,y,t), se obtiene como una combinación de la medida de certidumbre de movimiento de acuerdo con la expresión:

$$\pi_{M(x,y,t)}(true) = (M(x,y,t) = true) \vee (M(x,y-1,t-1) = true \ ) \wedge (M(x,y+1,t-1) = true) \quad (6)$$

Si se utilizan 5 vecinos para detectar el movimiento de la imagen, es decir, dos píxeles por la izquierda y dos por la derecha del píxel actual, las reglas difusas para detección de movimiento tomarán la forma que se muestran en la Figura 3b. Por tanto, para evaluar cada píxel de la imagen desentrelazada es necesario considerar 15 valores de la matriz de diferencias que, dispuestos matricialmente, pueden expresarse de la forma:

$$\begin{pmatrix} H(x-2,y-1,t-1) & H(x-2,y,t) & H(x-2,y+1,t-1) \\ H(x-1,y-1,t-1) & H(x-1,y,t) & H(x-1,y+1,t-1) \\ H(x,y-1,t-1) & H(x,y,t) & H(x,y+1,t-1) \\ H(x+1,y-1,t-1) & H(x+1,y,t) & H(x+1,y+1,t-1) \\ H(x+2,y-1,t-1) & H(x+2,y,t) & H(x+2,y+1,t-1) \end{pmatrix}$$

Siguiendo la estrategia desarrollada en [13], que explota las propiedades matemáticas de los operadores difusos, se evalúa en primer lugar el movimiento en torno al píxel actual. Para calcular los valores de la matriz de diferencias que afectan al píxel de coordenadas (x,y,t) es necesario considerar los valores de luminancia de los píxeles reflejados en la Figura 4 pertenecientes a cuatro frames consecutivos.



(a)

(b)

$$if \begin{pmatrix} H(x+2,y,t) = LARGE \vee \\ H(x+1,y,t) = LARGE \vee \\ H(x,y,t) = LARGE \vee \\ H(x-1,y,t) = LARGE \vee \\ H(x-2,y,t) = LARGE \end{pmatrix} THEN \ \ M(x,y,t) = true$$

$$if \begin{pmatrix} H(x+2,y-1,t-1) = LARGE \vee \\ H(x+1,y-1,t-1) = LARGE \vee \\ H(x,y-1,t-1) = LARGE \vee \\ H(x-1,y-1,t-1) = LARGE \vee \\ H(x-2,y-1,t-1) = LARGE \end{pmatrix} THEN \ \ M(x,y-1,t-1) = true$$

$$if \begin{pmatrix} H(x+2,y+1,t-1) = LARGE \vee \\ H(x+1,y+1,t-1) = LARGE \vee \\ H(x,y+1,t-1) = LARGE \vee \\ H(x-1,y+1,t-1) = LARGE \vee \\ H(x-2,y+1,t-1) = LARGE \end{pmatrix} THEN \ \ M(x,y+1,t-1) = true$$

Figura 3. a) Función de pertenencia que representa a la etiqueta "Large". b) Base de reglas del sistema de detección de movimiento cuando se consideran 5 vecinos.

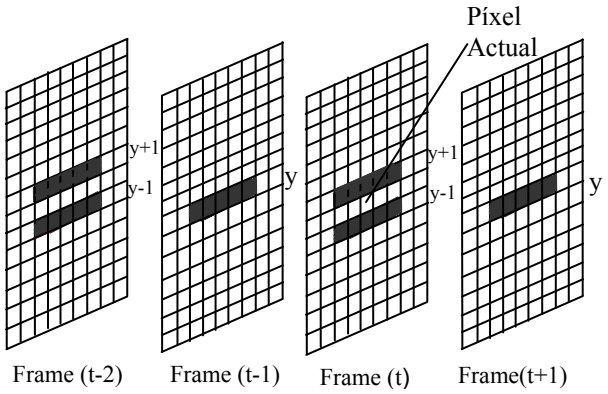


Figura 4. Cálculo de la matriz de diferencias.

A continuación, se realiza un preprocesado de los frames con objeto de evaluar el movimiento de forma simultánea dentro de una ventana rectangular alrededor del píxel. Este preprocesado se realiza utilizando técnicas de convolución bidimensionales de acuerdo con la expresión (7):

$$A = \sum_{i=1}^{5} \left(\sum_{j=1}^{3} H_{ij} \cdot C_{ij} \right) \quad (7)$$

donde:

$$C = \frac{1}{32} \begin{pmatrix} 1 & 1 & 1 \\ 2 & 3 & 2 \\ 3 & 5 & 3 \\ 2 & 3 & 2 \\ 1 & 1 & 1 \end{pmatrix} \quad (8)$$

El uso de técnicas de convolución proporciona una mayor flexibilidad porque permite utilizar distintos pesos en los coeficientes de la matriz C, con el mismo coste computacional. En el algoritmo implementado se ha trabajado con los valores de C reflejados en (8).

El resultado de la convolución, denotado por A, es la entrada al proceso de fuzzificación, en el que el grado de pertencia del sistema difuso $\gamma(x,y,t)$ es evaluado tal y como ilustra la Figura 3a. Su valor se encuentra comprendido en el intervalo [0,1] e indica el grado de movimiento. Para obtener la luminancia del píxel actual con coordenadas (x,y,t) este valor se sustituye en la expresión (3).

4. IMPLEMENTACIÓN HARDWARE DEL ALGORITMO DE DESENTRELAZADO

Se han realizado tres diseños distintos con ayuda de la herramienta *SysGen*. Los diseños implementan el algoritmo explicado en el apartado anterior y se diferencian entre sí en el grado de paralelismo empleado para realizar la convolución. El primero de ellos responde a una arquitectura masivamente paralela que opera simultáneamente con los treinta valores de luminancia de los frames almacenados en memoria que intervienen en el cálculo de la luminancia de la nueva línea insertada (Figura 5a). Como resultado se obtiene un valor para el píxel de la nueva línea en cada ciclo de reloj.

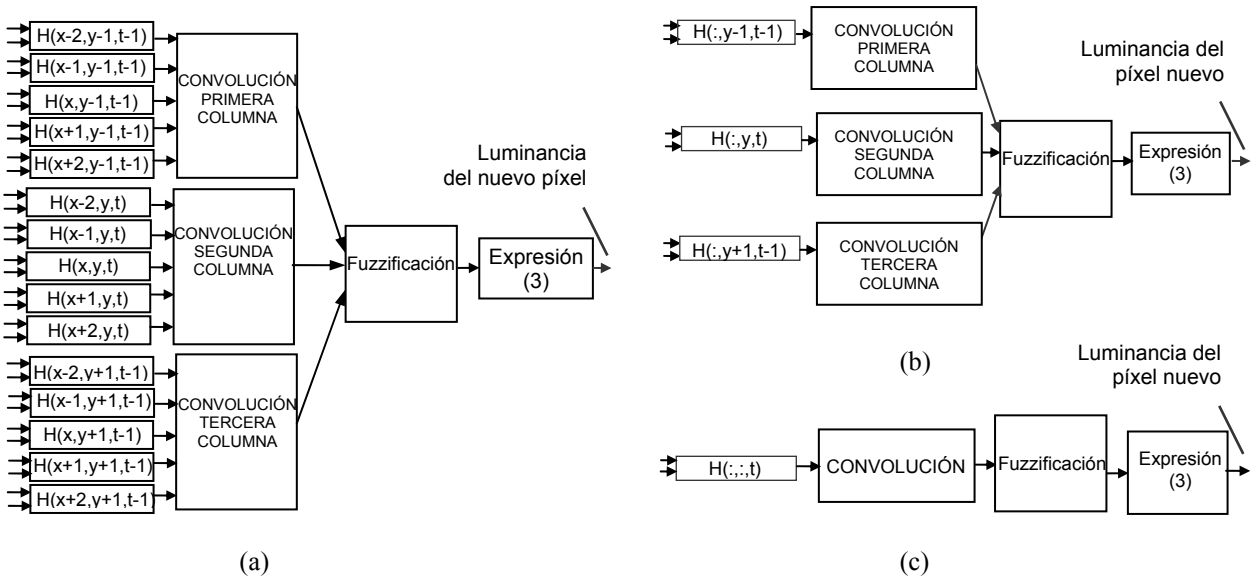


Figura 5. Arquitecturas para la implementación paralela (a), mixta (b) y secuencial (c) del algoritmo de desentrelazado de vídeo.

El segundo diseño presenta una arquitectura mixta que realiza las mismas operaciones pero lo hace de forma secuencial para los cinco valores de una misma columna de la matriz de diferencias, mientras que existe paralelismo entre las operaciones en las distintas columnas de la matriz de diferencias (Figura 5b). Por tanto, se obtiene un resultado transcurrido cinco ciclos de reloj.

El tercer diseño utiliza una arquitectura totalmente secuencial (Figura 5c), de forma que deben transcurrir quince ciclos de reloj para el cálculo del nuevo valor de la luminancia.

Los sistemas digitales generados utilizan señales binarias codificadas en complemento a dos. La Figura 6 muestra el diseño del subsistema que implementa la matriz de diferencias Hxyt. El bloque diferencia realiza dicha operación entre las señales de luminancia de las entradas. De acuerdo con el bit más significativo del resultado, un multiplexor selecciona la entrada d0 (bit igual a 0 – resultado positivo) o la entrada d1 (bit igual a 1 – resultado negativo). En el último caso, el resultado es complementado para implementar el valor absoluto de la expresión (4). La división por dos se realiza desplazando un bit a la derecha.

La Figura 7a muestra el esquemático del circuito que implementa la convolución de la primera y la tercera columna de la matriz C. Para evitar utilizar multiplicadores que requerirían un uso elevado de dispositivos hardware, se ha recurrido al uso de sumadores y desplazamientos a la izquierda. La Figura 7b muestra el diseño análogo para la segunda columna.

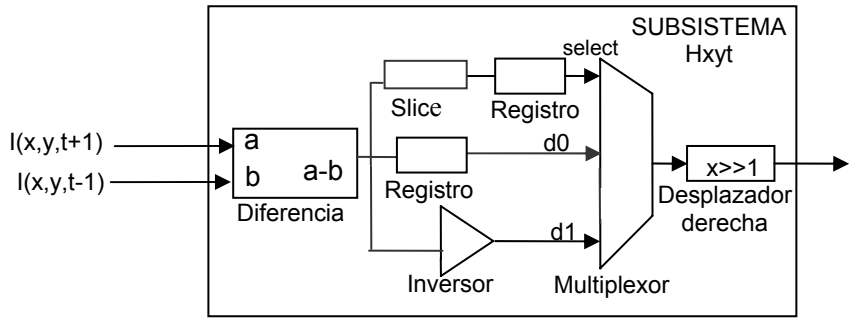


Figura 6. Diagrama de bloques del circuito que realiza el cálculo de la matriz de diferencias.

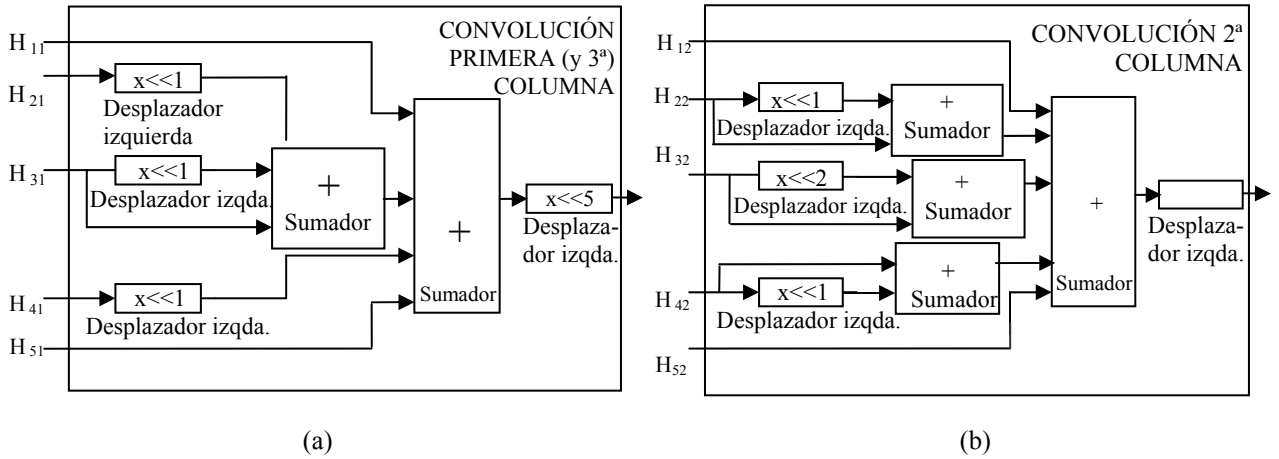


Figura 7. Diagrama de bloques de la convolución: a) primera y tercera columnas de la matriz C. b) segunda columna.

El subsistema que implementa el proceso de fuzzificación queda representado en el diagrama de bloques en la figura 8a. El bloque difencia1 determina si el resultado de la convolución A es menor que el primer parámetro de la función grado de pertenencia. Si el resultado es negativo, el multiplexor1 selecciona la entrada d1 y la salida del detector es '0'. Si, por el contrario, el resultado es positivo, una segunda diferencia evalúa si el resultado es mayor o igual que el segundo parámetro. Si el resultado es mayor o igual, la salida del detector es '1'. En cualquier otro caso, es un valor intermedio entre ambos valores. La Figura 8b muestra el diagrama de bloques que realiza la implementación de la expresión (3). En este circuito son imprescindibles dos multiplicadores, puesto que los factores no son constantes y no pueden sustituirse por desplazamientos y sumas.

El medio físico empleado para la implementación hardware del algoritmo ha sido la placa RC100 de Celoxica. El código que controla todos los procesos necesarios para realizar la implementación ha sido codificado en Handel-C. Este código controla, en primar lugar, la captura de la señal de vídeo a través de la entrada de vídeo compuesto de la placa. Posteriormente se describe el proceso que realiza la escritura y lectura de los frames en las memorias SRAMs. En cada memoria se almacenan 4 frames consecutivos que son requeridos en la implementación del algoritmo. La lectura de la SRAM proporciona los datos de entrada

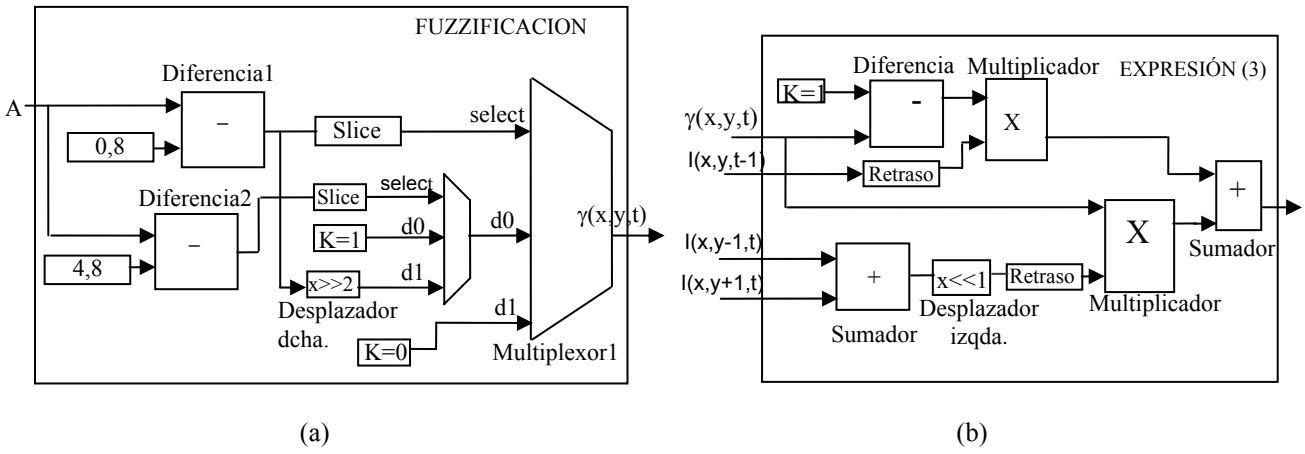


Figura 8. Diagramas de bloques: a) del proceso de fuzzificación; b) de la implementación de la expresión (3).

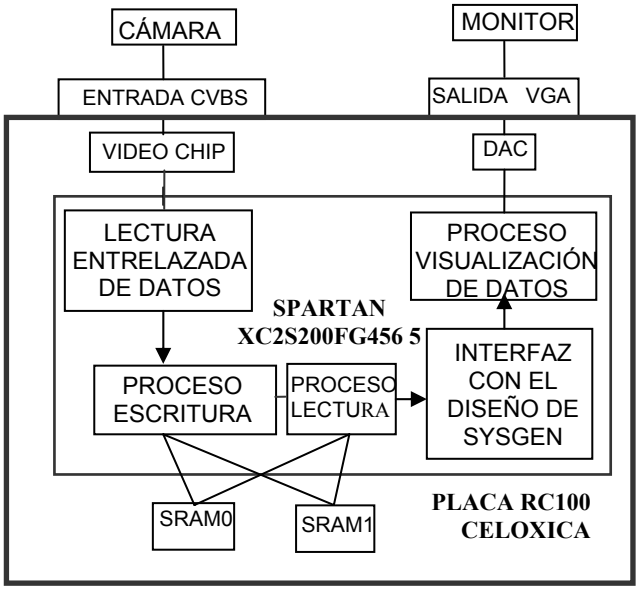


Figura 9. Diagrama de bloques de los procesos implementados en la placa RC100 de Celoxica.

a una caja negra definida en Handel-C que integra el diseño del algoritmo de desentrelazado desarrollado mediante *SysGen*. La Figura 9 muestra un diagrama de bloques indicando los procesos implementados en la FPGA.

5. ANÁLISIS DE LOS RESULTADOS OBTENIDOS

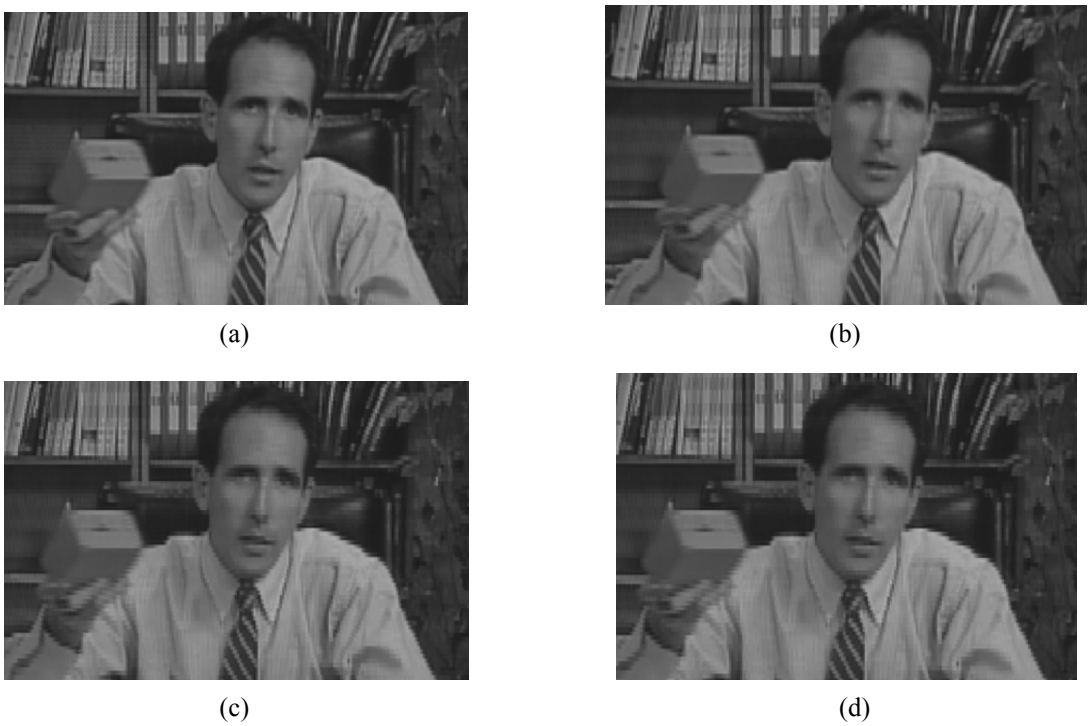
La evaluación del comportamiento del algoritmo de desentrelazado puede realizarse a partir de los resultados obtenidos en el entorno *Simulink* o mediante el modelado, con la herramienta *ModelSim* de Mentor Graphics, de la descripción VHDL generada por *SysGen*. En la Tabla 1 se muestran los errores cuadráticos medios obtenidos para el algoritmo desarrollado en [13] (implementación software) y los generado con *SysGen* (implementación hardware) utilizando 8 y 16 bits de resolución.

Los tests han sido realizados con una película donde aparece un vendedor con un objeto en la mano constantemente en movimiento en contrate con un fondo estático. Por lo tanto, es un vídeo muy adecuado para realizar el análisis de las dos técnicas, espacial y temporal. En concreto, la película está compuesta por 15 fotogramas en blanco y negro con una resolución de 288x360 píxeles de los cuales hemos eliminado un marco exterior de 3x3 píxeles para evaluar correctamente el error.

Tabla 1.- Valores RMSE analizados para el octavo frame de la película.

FPGA: SPARTAN 2 xc2s200 5 fg456		
282x354 PÍXELES FRAME 8		RMSE
ALGORITMO SOFTWARE		1,69%
DISEÑO SYSGEN	8 BITS	2,9%
	16 BITS	2,85%

El error cuadrático medio es superior para el diseño realizado con *SysGen*, ya que el algoritmo descrito en Matlab trabaja con números reales mientras que el modelo en VHDL



(a) (b)

(c) (d)

Figura 10. a) Imagen original. b) Imagen obtenida con la implementación software. c) Imagen obtenida con *SysGen* 8 bits. d) Imagen obtenida con *SysGen* 16 bits.

trabaja con números enteros de longitud finita. Los resultados mejoran ligeramente cuando aumentamos la resolución de ocho a dieciséis bits.

La Figura 10 muestra la imagen original y las imágenes obtenidas mediante la implementación software del algoritmo y las implementaciones realizadas con SysGen. Estas imágenes corroboran los resultados de los errores cuadráticos medios obtenidos, siendo la imagen de peor calidad la de mayor error.

La Tabla 2 refleja los resultados post-síntesis para las distintas arquitecturas de implementación cuando se utilizan números enteros de 8 y 16 bits. Los resultados muestran que la arquitectura secuencial es la más lenta, pero es la que menos dispositivos requiere.

Tabla 2.- Resultados post-síntesis de los diseños generados con *SysGen*.

FPGA: SPARTAN 2 xc2s200 5 fg456			
		SLICES UTILIZADOS	PERIODO/FRECUENCIA DE PROCESADO PARA UN PÍXEL NUEVO (ns / MHz)
PARALELO	8 bits	716 (30%)	36 / 27,7
	16 bits	1235 (52%)	39 / 25,6
MIXTO	8 bits	323 (13,7%)	115 / 8,7
	16 bits	525 (22,3%)	125 / 6,9
SECUENCIAL	8 bits	117 (7%)	345 / 2,8
	16 bits	305 (12%)	390 / 2,6

La Tabla 3 muestra el número de frames que pueden ser generados por segundo en función de la frecuencia a la que trabajan los sistemas y la resolución de las imágenes transmitidas.

Tabla 3.- Tiempo requerido para generar una imagen.

FPGA : SPARTAN 2 xc2s200 5 fg456			
	TIEMPO PARA COMPLETAR UN FRAME (ms)	NÚMERO DE FRAMES POR SEGUNDO (frames/s)	RESOLUCIÓN MÁXIMA PARA 30 frames/s
8-Paralelo	1,868	535	924.480
16-Paralelo	2,036	491	848.448
8-Mixto	6,096	164	283.392
16-Mixto	6,7	149	257.472
8-Secuencial	18,49	54	93.312
16-Secuencial	20,33	49	84.672

Analizando los resultados que se muestran en la Tabla 3, la arquitectura más adecuada para implementar el algoritmo en tiempo real es el diseño paralelo porque permite trabajar con resoluciones estándares: VGA (640x480), SVGA (800x600) y XGA(1024x768). Finalmente, la Tabla 4 muestra los resultados del nivel ocupacional de la FPGA cuando se implementa el sistema global descrito en Handel-C.

Tabla 4.- Resultados post-implementación de los diseños globales descritos en Handel-C.

FPGA SPARTAN 2 xc2s200 5 fg456		NÚMERO DE SLICES
PARALELO	8 BITS	2042 of 2352 (86 %)
MIXTO	8 BITS	1649 of 2352 (70%)
	16 BITS	1851 of 2352 (78%)
SECUENCIAL	8 BITS	1443 of 2352 (61%)
	16 BITS	1631 of 2352 (69%)

6. CONCLUSIONES

Los resultados obtenidos muestran que las FPGAs son una alternativa válida para la implementación microelectrónica de sistemas para procesado de imágenes en tiempo real. La disponibilidad de herramientas de CAD especializadas y placas de desarrollo disponibles a través de los programas universitarios de los principales fabricantes de dispositivos programables facilita la exploración del espacio de diseño, ya que permite recorrer de forma rápida y con costos asequibles las diferentes etapas de desarrollo de los sistemas.

El uso de código escrito e Handle-C facilita la implementación de otros algoritmos de procesado de imágenes, sin más que modificar la llamada a nuevos modelos desarrollados con *SysGen*. Esta característica convierte al sistema presentado en esta comunicación en una plataforma de desarrollo configurable cuyas únicas limitaciones vienen impuestas por las dimensiones de la FPGA y los recursos de memoria disponibles.

7. BIBLIOGRAFÍA

[1] C. H. Dick, F. J. Harris "Configurable logic for digital communications: some signal processing perspectives". *IEEE Conm. Magazine*, vol. 2, pp. 107-111, August 1999.

[2] A. Bainbridge-Smith, P. Dunne. "FPGAs in computer vision applications". *Image and Vison Computing*. pp.347-352, New Zealand 2002.

[3] Xilinx. “A guide to using Field Programmable Gate Arrays (FPGAs) for application-specific digital signal processing performance”. Application note, 1995.

[4] J. Hwang, B. Milne, N. Shirazi, J. Strooner. “System level tools for DSPs in FPGAs”. Application note from Xilinx.

[5] http://www.xilinx.com/univ/index.htm

[6] http://www.celoxica.com/partner/university/default.asp

[7] Xilinx. “Xilinx System Generator v2.1 for Simulink”. User Guide. Xilinx Blockset Reference Guide.

[8] Celoxica. “RC100 Function Library Manual”. 2002.

[9] C. J.Kuo, C. Liao, C. C.Lin. “Adaptive interpolation technique for scanning rate conversion”. *IEEE Trans on Circuits and Systems for Video Technology*, vol. 6, no.3, June 1996.

[10] K. Sugiyama, H. Nakamura. “A method of de-interlacing with motion compensated interpolation”. *IEEE Trans Consumer Elec*., vol.45, no.3, pp.611-616, August 1999.

[11] J. Deame. “Motion compensated de-interlacing: The key to the digital video transition”. *SMPTE 141st Technical Conference*. NY, November, 1999.

[12] D. Van De Ville, B. Rogge, W. Philips, I. Lemahieu. “De-interlacing using fuzzy-based motion detection”. *Knowledge-Based Intelligent Information Engineering System*, pp. 263-267, 1999.

[13] J. Gutiérrez-Ríos, F. Fernández Hernández, J. C. Crespo, G. Treviño. “Motion Adaptive Fuzzy Video De-interlacing method based on convolution techniques”. *Information Processing and Management of Uncertainly in knowledge-Based Systems (IPMU)*, July 2004.