Implementación sobre FPGA de un algoritmo de etiquetado en tiempo real

E. Calvo-Gallego[1,2], P. Brox[2], S. Sánchez-Solano[2]

Resumen—En esta comunicación se presenta una implementación de un algoritmo de etiquetado de componentes conexos en tiempo real que aprovecha los intervalos de supresión horizontal y vertical en la fuente de la imagen para mejorar la calidad del etiquetado y acelerar la frecuencia de trabajo. El diseño se ha llevado a cabo usando la herramienta System Generator de Xilinx lo que ha permitido reducir los tiempos de implementación y verificación lógica y funcional del modelo.

Palabras clave-- Implementación FPGA, Algoritmos CCL, Procesamiento de imágenes

I. INTRODUCCIÓN

La integración de sistemas de visión en dispositivos específicos, como teléfonos móviles, PDAs o redes de sensores, es cada día más habitual. Las limitaciones que presentan estos dispositivos en cuanto a consumo de potencia y capacidad de cálculo y almacenamiento, así como la necesidad de operación en tiempo real de las aplicaciones que los utilizan, obligan a llevar a cabo una revisión de los algoritmos existentes con el fin de adaptar su implementación a las características de las plataformas emergentes. Este proceso de adaptación es especialmente importante en los algoritmos de etiquetado de componentes conexos o algoritmos CCL (Connected Component Labeling), ya que se sitúan en una etapa intermedia del flujo de procesado de la imagen y juegan un papel fundamental en cualquier sistema de visión real para la realización de aplicaciones de alto nivel (Fig.1).

El objetivo de los algoritmos CCL es realizar la asignación de un identificador único a cada conjunto conexo de píxeles con unas mismas propiedades dentro de la imagen. Este tipo de algoritmos ha sufrido una evolución constante desde los años 60, proponiéndose distintas alternativas de implementación que pueden catalogarse en función de la regularidad en accesos a memoria, la forma de representación de la imagen o el número de exploraciones que se realizan sobre ésta. Además, como consecuencia de los avances en las tecnologías de fabricación de circuitos integrados, muchas técnicas de paralelismo, descartadas durante los años 80 por necesitar recursos elevados (que limitaban las dimensiones de las imagen con las cuales se podía trabajar y/u obligaba al uso de memorias externas), han vuelto a ser consideradas [1-3].

En esta comunicación se describe la implementación sobre FPGA de un algoritmo CCL que opera en imágenes matriciales, exploradas de izquierda a derecha y de arriba abajo, que se transmiten como un flujo constante de píxeles. El algoritmo, que realiza dos pasadas sobre la imagen, se inspira en el propuesto por Bailey en [4]. Este algoritmo es de mucha relevancia en el campo ya que, además de necesitar pocos recursos, aprovecha, para mejorar la calidad del etiquetado y acelerar la frecuencia de trabajo, los tiempos de supresión o borrado horizontal que permiten en los distintos estándares de video identificar el inicio y el fin de cada línea de la imagen (La Fig.2 ilustra un formato de presentación genérico, así como las señales que suelen estar presentes en una interfaz real de video).

A diferencia de las implementaciones presentadas en [5], la realización descrita en este trabajo proporciona la imagen etiquetada para facilitar la realización de etapas posteriores de procesado.

El proceso de diseño ha seguido una metodología basada en System Generator, una herramienta de Xilinx integrada en el entorno Matlab/Simulink que, al facilitar la realización de todas las etapas bajo un mismo marco de referencia, permite acortar el tiempo de desarrollo de la implementaciones, y facilita el análisis de diversas alternativas en cuanto a área, velocidad de operación y tiempo total de procesamiento del algoritmo.

La estructura de la comunicación es la siguiente. En la Sección II se introducen los distintos tipos de algoritmos CCL, clasificados en función del número de barridos que realizan de la imagen. La Sección III se centra en la descripción del algoritmo CCL de dos pasadas propuesto por Bailey et al. La implementación propuesta se detalla en la Sección IV, mientras que la Sección V recoge los resultados obtenidos. Por último, las conclusiones del trabajo se resumen en la Sección VI.

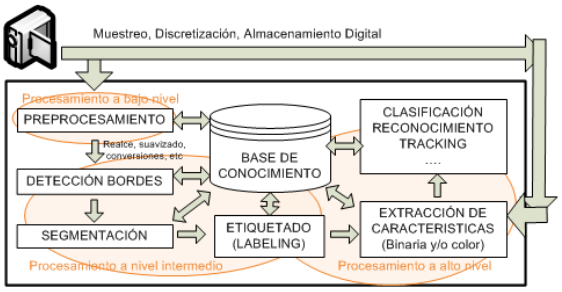

Fig. 1 Sistema de visión genérico

[1] Departamento de Electrónica y Electromagnetismo. Universidad de Sevilla (España), e-mail: calvo@imse-cnm.csic.es
[2] Instituto de Microelectrónica de Sevilla (IMSE). Centro Nacional de Microelectrónica (CNM). Consejo Superior de Investigaciones Científicas (CSIC), e-mail: brox@imse-cnm.csic.es, santiago@imse-cnm.csic.es

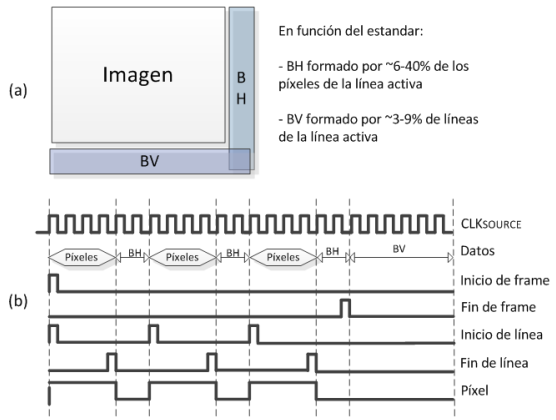

Fig. 2 (a) Imagen y tiempos de supresión (b) Señales de sincronismo proporcionadas por la fuente de la imagen

II. ALGORITMOS CCL

Los algoritmos CCL considerados en este trabajo se aplican a imágenes binarias (B) compuestas por un fondo o Background (FB), formado por píxeles negros ('0'), y un primer plano o Foreground (FO), formado por píxeles blancos ('1'). Estos algoritmos están basados en análisis de conectividad. Dos píxeles pertenecerán a un mismo componente cuando ambos formen parte del primer plano o del fondo y exista un camino de píxeles del mismo tipo entre ellos, es decir, cuando se verifique (1), donde S es un subconjunto de píxeles de la imagen binaria y $N(s_i)$ el entorno de vecindad considerado para el pixel s_i, habitualmente formado por 4 u 8 vecinos. La Fig. 3 muestra un ejemplo de imagen etiquetada en función de la conectividad.

$$p \text{ conectado a } q \Leftrightarrow \exists \{s_i \in S \mid s_1 = p, s_{n+1} = q, \\ s_{i+1} \in N(s_i),\ i = 1, \dots, n\} \quad (1)$$

En base al número de barridos realizados sobre la imagen, los algoritmos CCL se pueden encuadrar en las siguientes categorías:

▪ One-Scan: Realizan una pasada sobre la imagen. Se caracterizan por efectuar accesos irregulares a memoria y por la dificultad que existe para predecir el tiempo de ejecución de los algoritmos. Entre ellos destacan los algoritmos de trazado de contorno, de crecimiento de regiones y de extracción de características [7].

▪ Multi-scan: Recorren la imagen en varias ocasiones realizando accesos regulares a memoria. Aunque son sencillos de implementar, tanto en software como en hardware, su principal inconveniente es la dependencia del tiempo de ejecución del algoritmo con la disposición de los píxeles en la imagen [8],[9].

▪ Two-Scan: Realizan dos barridos de la imagen, efectuando también accesos regulares a memoria. Suelen tener una duración determinada y conocida. Se diferencian entre sí por la forma de asignar etiquetas a los distintos componentes de la imagen y el modo de almacenar y resolver las equivalencias encontradas [10]-[12].

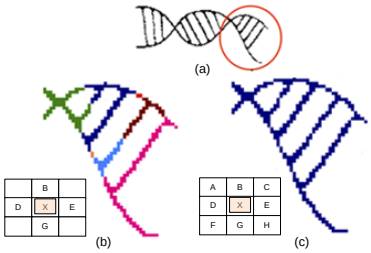

Fig. 3 (a) Imagen binaria. (b) y (c) Imágenes etiquetadas considerando N4 (7 etiquetas) y N8 (1 etiqueta)

III. ALGORITMO CCL PROPUESTO POR BAILEY

El algoritmo CCL propuesto por Bailey [4] realiza dos barridos de la imagen binaria a etiquetar. En el primero se asignan una serie de etiquetas temporales a los píxeles (almacenadas en la matriz denominada g) y se identifican y anotan (en la tabla denominada T) las equivalencias que puedan establecerse entre etiquetas temporales que están siendo asignadas. Una vez concluido este primer barrido se lleva a cabo la resolución de la tabla de equivalencias. En el segundo barrido se sustituyen las etiquetas temporales del primer barrido por los valores equivalentes contenidos en la tabla resuelta.

La arquitectura para la realización del primer barrido, considerando un entorno de 4 vecinos, se muestra en la Fig. 4. En cada ciclo de reloj, llegará un nuevo pixel de la imagen que será etiquetado de acuerdo con (2), es decir:

▪ Si el pixel pertenece al fondo, se le asigna la etiqueta nula.
▪ Si pertenece a un objeto y está rodeado por píxeles del fondo, se le asigna una nueva etiqueta (m) cuyo valor se incrementará cada vez que se dé este mismo caso.
▪ Si pertenece a un objeto y algún vecino ya ésta etiquetado, se le asigna el mínimo valor no nulo entre el equivalente de la etiqueta del pixel superior y la etiqueta del píxel izquierdo.

$$g(x,y) = \begin{cases} F_B & si \quad b(x,y) = F_B \\ m & si \ \{b(i,j-1) = b(i-1,j) = F_B \\ & \quad y\ b(x,y) = F_O\} \\ L & en\ otro\ caso \end{cases} \quad (2)$$

$$L = min[g(i,j-1), T(g(i-1,j))]$$

Cuando la etiqueta asignada al píxel izquierdo y el equivalente de la etiqueta asignada al píxel superior son diferentes se dice que se ha encontrado una equivalencia entre $g(i,j-1)$ y $T(g(i-1,j))$. En ese caso, si el elemento del par equivalente de mayor valor es el del pixel superior, es decir, $T(g(i-1,j))$, la tabla de equivalencias se actualiza salvando el valor menor en la posición de memoria del mayor (3). Si, por el contrario, el elemento del par equivalente de mayor valor es el del pixel izquierdo, $g(i,j-1)$, el par equivalente se almacena en una estructura FIFO de dos filas (denotada con la letra F) que será utilizada para actualizar la tabla de equivalencia durante el intervalo de supresión horizontal (4).

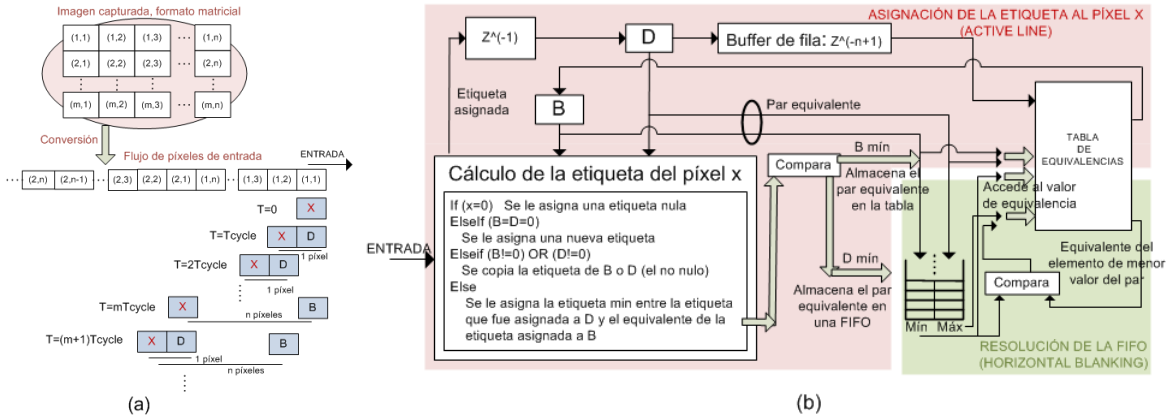

Fig. 4 (a) Formatos de imagen de captura y entrada al algoritmo y superposición de las máscara mostrada en la Fig.3 (Conectividad 4) en cada ciclo de reloj a los píxeles de entrada (b) Diagrama del algoritmo propuesto e implementado por Bailey et al. [4-6]

$$T([T(g(i-1,j))]) = [g(i,j-1)] \quad (3)$$

$$F(1, FIFOLast) = g(i,j-1) \quad (4)$$
$$F(2, FIFOLast) = T(g(i-1,j))$$

El hecho de que se traten las equivalencias de forma diferente en función de la posición del elemento de mayor valor se debe a un intento de propagar las etiquetas de menor valor con el fin de reducir el número de etiquetas propagadas y el número de equivalencias que tienen lugar, evitar pérdidas de equivalencias y facilitar una implementación eficiente para la resolución de la tabla.

Al final de cada línea de la imagen, durante el intervalo de supresión horizontal, se procesa la información almacenada en la FIFO. Para cada par equivalente extraído de la FIFO, se accede al valor de equivalencia del mínimo valor del par, se compara el valor del mínimo con el equivalente obtenido ($\mathrm{F}(2, \mathrm{FIFOFirst})$ con $\mathrm{T}(\mathrm{F}(2, \mathrm{FIFOFirst}))$) y, si son diferentes, se guarda el valor del equivalente del mínimo en la posición del máximo valor del par (5). Para acelerar la operación se utiliza una estructura pipeline de tres etapas, que permite procesar cada entrada de la FIFO en un ciclo de reloj, aunque introduce una latencia de dos ciclos (Fig. 5).

$$T(F(1, FIFOFirst)) = T[F(2, FIFOFirst)] \quad (5)$$

La resolución final de la tabla de equivalencias se lleva a cabo recorriendo las posiciones de la misma y almacenando, si la equivalencia es distinta de la etiqueta, el equivalente del equivalente de cada etiqueta como el nuevo valor de la tabla (6).

$$si\ T(Label_i) \neq Label_i \rightarrow T(Label_i) = T\big(T(Label_i)\big) \quad (6)$$

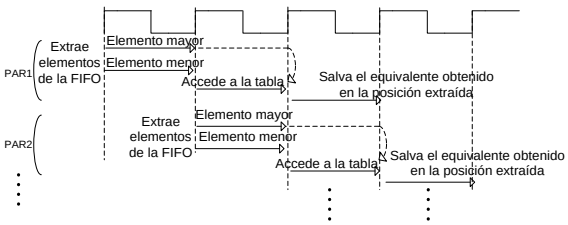

Fig. 5 Arquitectura pipeline para actualizar la tabla de equivalencias al final de cada fila con los datos de la FIFO

El segundo barrido a través de la imagen simplemente usa la tabla de equivalencias como LUT para remplazar las etiquetas iniciales con las finales.

La implementación realizada por Bailey et al. [5] no contempla el algoritmo completo sino únicamente el primer barrido [6]. El objetivo de dicha implementación, que considera conectividad 8, es la extracción de características (permite obtener el número de componentes, el área y el centro de gravedad de los mismos) de imágenes de resolución VGA (480x640) a una velocidad de 60 frames por segundo. El algoritmo se implementó sobre una placa de desarrollo Celoxica RC100 (Spartan-II XC2S200) usando Handel-C como lenguaje de descripción de hardware.

IV. Implementación propuesta

El objetivo de la implementación del algoritmo CCL descrito en este trabajo es que sea capaz de procesar secuencias de video en tiempo real, proporcionando frames etiquetados a la salida. Para obtener este resultado ha sido necesario aprovechar los tiempos de supresión verticales para la resolución de equivalencias entre filas y explotar el paralelismo temporal usando técnicas de pipeline para simultanear las dos fases de ejecución del algoritmo (Fig.6).

El proceso de diseño seguido, así como las herramientas usadas en él, se muestran en la Fig. 7. Se parte de un modelo Matlab del algoritmo (fichero .m) que facilita el análisis de los posibles problemas que pueden aparecer en la implementación en función de distintos parámetros de diseño (como el porcentaje de etiquetas permitido). Esta descripción es posteriormente trasladada a un modelo Simulink (fichero .mdl), compuesto por bloques de la librería Xilinx Blockset, que es verificado a nivel lógico y funcional desde Matlab, y compilado mediante

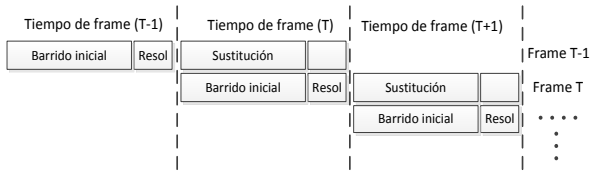

Fig. 6 Pipeline en las distintas fases del procedimiento

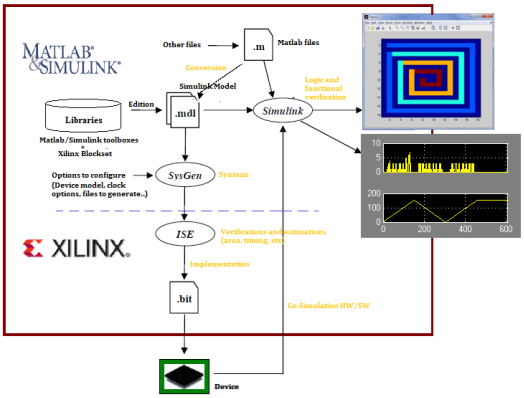

Fig. 7 Metodología de diseño seguida y herramientas usadas

System Generator para generar un proyecto ISE. Desde el entorno ISE es posible realizar estimaciones de área y tiempo, así como otras operaciones de verificación. Por último, para comprobar que el funcionamiento del sistema una vez implementado en la FPGA es idéntico al deseado, se lleva a cabo una co-simulación HW/SW, cerrando el lazo en el flujo de diseño.
En los siguientes apartados se describen los principales componentes del sistema.

A. *Bloques de control*

A partir de las señales de una interfaz de video real, mostradas en la Fig.2, se generan otras que permitirán el control del algoritmo. Las de mayor relevancia son:

- Fila, columna, posición: Indican la fila (x), la columna (y) y la posición (x*Num_Colum+y-1) del píxel en la imagen.
- IsFrame y HB: Permiten separar, respectivamente, el intervalo de supresión vertical del intervalo de imagen y los intervalos de supresión horizontales, y el intervalo de supresión horizontal del intervalo de imagen.

B. *Memorias*

La implementación del algoritmo requiere dos bloques de memoria de acceso aleatorio (para la matriz de etiquetas temporales y la tabla de equivalencia) y una estructura FIFO:

- Memoria de etiquetas temporales: Es implementada mediante una RAM de doble puerto, con Num_Colum * Num_filas posiciones y un ancho de palabra de $\lceil \log_2((\text{Num_Colum} * \text{Num_filas})/2\ )\rceil$[3] bits. El puerto A se configura con la opción 'Read before write' de forma que en cada tiempo de frame se pueda obtener, al salvar la etiqueta temporal asignada a un pixel (barrido inicial del frame t), la etiqueta temporal asignada al mismo pixel del frame anterior (fase de sustitución del frame t-1). El puerto B, por el que se realizará la lectura de la etiqueta del pixel superior, se configura como 'Write before read'.

- Tabla de equivalencias: La tabla de equivalencias se implementa por duplicado (para facilitar la ejecución simultánea de las dos fases del algoritmo ilustradas en la Fig.6), utilizando dos memorias RAM de doble puerto, ambas con (Num_Colum*Num_filas)/2 posiciones y un ancho de palabra de $\lceil \log_2((\text{Num_Colum} * \text{Num_filas})/2\ )\rceil$ bits[4]. Cada dirección de memoria en las tablas corresponde a una etiqueta y el dato contenido en ella a su equivalencia. Durante el barrido inicial, la primera tabla permite salvar las equivalencias detectadas entre etiquetas temporales. En la fase de resolución se usan ambas tablas, almacenándose los valores definitivos en la segunda de ellas. Por último, esta tabla es utilizada en la fase de sustitución de los valores de etiquetas temporales para obtener la etiqueta final asignada al pixel.

- FIFO: En el diseño se utiliza una estructura FIFO con dos columnas: una para almacenar los elementos de mayor valor de los pares equivalentes, y la otra para los elementos de menor valor. Se implementan con dos memorias RAM cuyo tamaño es el máximo número de equivalencias que se pueden dar en una fila, es decir, Num_Colum/2. El ancho de cada una de las palabras es de $\lceil \log_2((\text{Num_Colum} * \text{Num_filas})/2\ )\rceil$ bits. Ambas memorias se escriben y se leen simultáneamente. Para evitar que los pares no resueltos de una imagen interfieran en el etiquetado de la imagen siguiente en la secuencia, se envía una señal de reset a las memorias con la señal de fin de frame.

C. *Acceso a la tabla de equivalencias*

Puesto que las operaciones que se realizan sobre las memorias que forman la tabla de equivalencias dependen de las distintas fases del algoritmo, es necesario emplear bancos de multiplexores para acceder a estos recursos. La Fig. 8 muestra un diagrama de bloques de esta parte del algoritmo en el que se aprecia el banco de multiplexores encargado de seleccionar las entradas de direcciones y datos de las memorias durante el intervalo de imagen activa y los intervalos de supresión horizontal, así como los bancos que separan las operaciones del primer barrido y las realizadas en el intervalo de supresión vertical. Las tablas I y II resumen

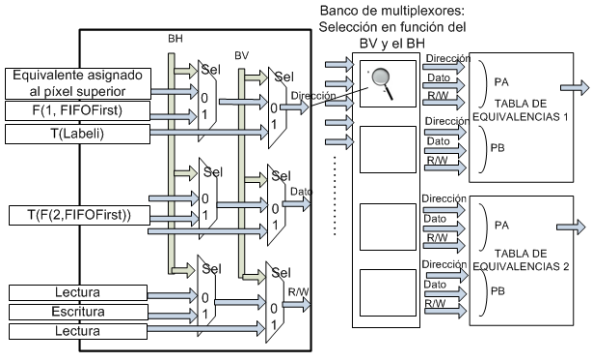

Fig. 8 Bancos de multiplexores para el acceso a las tablas de equivalencias. Ampliado, como ejemplo, el banco que proporciona las señales de dirección, datos y R/W a la tabla 1 por el puerto A.

[3] $(\text{Num_Colum} * \text{Num_filas})/2$ es el máximo número de componentes diferentes que puede contener una imagen considerando conectividad 4 (corresponde a una imagen binaria que representa un tablero de ajedrez).

[4] En la mayoría de las aplicaciones habituales el tamaño de estas memorias puede reducirse sin riesgo a perder etiquetas.

los accesos ilustrados en Fig. 4 (barrido inicial), Fig. 5 (intervalos de supresión horizontal) y Fig. 9 (intervalos de supresión vertical).

D. Cálculo de la etiqueta que se asignará al píxel

En función de la posición del píxel considerado (fila y columna) y del valor de la etiqueta del píxel izquierdo y el equivalente de la etiqueta del píxel superior, se copia uno de los dos valores, se calcula el mínimo entre ambos valores, o se asigna una nueva etiqueta. En este último caso se incrementa un contador de etiquetas, además de almacenar en un registro el número de etiquetas totales asignadas con objeto de facilitar la realización de una fase de resolución de duración variable así como otras operaciones y/o presentación de datos a alto nivel. En este caso también se extrae la señal de habilitación de nueva etiqueta, de forma que sea la entrada de habilitación de escritura de la primera tabla de equivalencias para la inicialización de la misma.

Este bloque también lleva a cabo la detección de la equivalencia y su tipo, separando las señales correspondientes para la actualización de las memorias.

E. Generación de direcciones y asignación de nueva equivalencia en la fase de resolución

Este bloque se muestra en la Fig.10. Utiliza un contador (que se resetea con cada inicio de frame y se incrementa cada ciclo de reloj hasta llegar al número máximo de etiquetas asignadas en el primer barrido) para generar las señales de dirección con las que se accede en la fase de resolución al puerto A de la primera tabla de equivalencias y al B de la segunda. La señal de escritura por el puerto B de la segunda memoria se habilita durante toda la fase de resolución dos ciclos después de que ésta haya comenzado. El dato que se almacena como equivalencia definitiva de cada etiqueta se calcula con los bloques centrales del diagrama. Si el valor de equivalencia ha sido modificado en la segunda tabla será éste el que se seleccione como equivalente del equivalente. Por el contrario, si el valor no ha sido modificado (tendrá el valor inicial o el perteneciente al frame anterior) se toma como equivalente del equivalente el valor leído de la primera tabla de equivalencia. Se comprueba también que el valor de equivalencia de una etiqueta sea de la propia etiqueta y si, como consecuencia de una posible cadena de equivalencias en la tabla, se va a leer un valor de equivalencia ya modificado pero no salvado.

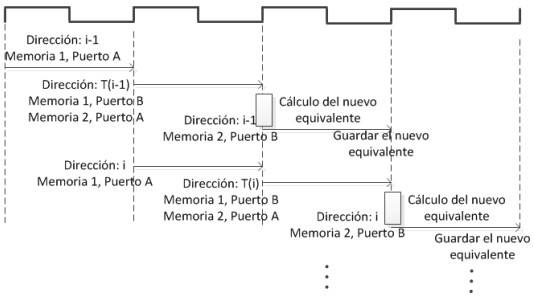

Fig. 9 Acceso a las tablas de equivalencia durante la fase de resolución del algoritmo (intervalo de supresión vertical)

TABLA I

ACCESOS A LAS TABLAS DE EQUIVALENCIAS DURANTE EL INTERVALO DE IMAGEN ACTIVA

	T 1. Imagen Activa	
A	Dir: Equivalente de la etiqueta asignada al píxel superior	
	R/W: Lectura	
B	Dos casos posibles	
	1. Asignación de una nueva etiqueta	2.Equivalencia entre etiquetas
	Dir: Valor de la nueva etiqueta	Dir: $T([T(g(i-1,j))])$
	Dato: Valor de la nueva etiqueta	Dato: $g(i,j-1)$
	R/W: Escritura	R/W: Escritura
	T 2. Imagen Activa	
B	Dir: Etiqueta temporal asignada al pixel	
	R/W: Lectura	

TABLA II

ACCESOS A LAS TABLA DE EQUIVALENCIAS DURANTE LOS INTERVALOS DE SUPRESIÓN HORIZONTAL Y VERTICAL

	Sup. Horizontal		Sup. Vertical	
T	1	2	1	2
A	Dir: $F(1, \text{FIFOFirst})$	--	Dir: $T(Label_i)$	Dir: $T(T(Label_{i-1}))$
	Dato:T(2, $FIFOFirst$))	--	R/W: Lectura	R/W: Lectura
	R/W: Escritura	--		
B	Dir: $F(2, \text{FIFOFirst})$	--	Dir: $T(T(Label_{i-1})$	Dir: $T(Label_i)$
				Dato:$T(T(T(Label_{i-1})))$
	R/W: Lectura	--	R/W: Lectura	R/W: Escritura

V. RESULTADOS DE IMPLEMENTACIÓN

Desde el punto de vista de su implementación hardware, las principales limitaciones del algoritmo CCL propuesto vienen determinadas por la memoria RAM utilizada para mantener las tablas de etiquetas (que establecerá unas dimensiones máximas de la imagen con la cual es posible trabajar) y por el tiempo de resolución de equivalencias (que viene acotado por la duración del intervalo de supresión vertical). No obstante, el rango de aplicación de la implementación dentro de ellas es bastante amplio.

Mediante simulaciones realizadas con Matlab, el algoritmo se probó con imágenes reales tomadas de las bases de datos de USC-SIPI [14] y del Berckley Computer Vision Group [15], así como con imágenes binarias generadas aleatoriamente de 256x256 y 1000x1000 píxeles que siguen una distribución de Bernoulli con distintos factores de probabilidad. Se comprobó cómo las etiquetas asignadas en el primer barrido no excedieron en ningún caso el 30% de las totales en el caso de las imágenes aleatorias, y un 3-5% en el caso de imágenes reales.

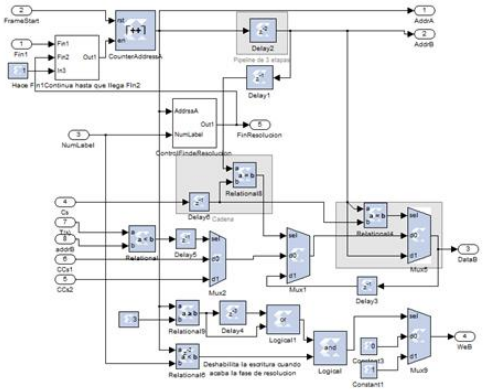

Fig. 10 Bloque de generación de direcciones y asignación de nueva etiqueta

Estos resultados permiten una reducción importante de los tamaños de las tablas de equivalencias y, en menor medida, de la memoria de etiquetas temporales (al reducir el número de bits por palabra). De esta forma, utilizando una placa de desarrollo SPARTAN 3A DSP 3400 es posible implementar un sistema que trabaja con resolución QCIF (176x144) considerando el tamaño máximo de la tabla de equivalencias. El tamaño de imagen puede ampliarse hasta WCIF (512x288) si se acota la tabla al 5% de su valor máximo. Si en lugar de almacenar la tabla de etiquetas temporales en BlockRAMs se almacena en un frame buffer externo, se podría trabajar con resolución CIF (352x288), para tamaños máximos de la tabla de equivalencias, y en resolución WUXGA (1920x1200), tamaños de tablas del 5%.

Los resultados muestran asimismo una disminución del número de casos en los cuales no se completará la resolución de las tablas de equivalencias. Por ejemplo, en las estándares descritos por VESA para una tasa de refresco de 60 Hz, en el tiempo de supresión vertical, el algoritmo podría resolver, si se usa como reloj el mismo que rige el flujo de llegada de pixeles, entre un 6 y un 20% de la tabla de equivalencias, lo cual, como ya se ha dicho, sería suficiente en la mayoría de los casos.

Por otra parte, la frecuencia de operación obtenida tras la síntesis del algoritmo en la FPGA utilizada es de 40 MHz, lo que permite trabajar con estándares de resoluciones inferiores a SVGA (800x600).

La Fig. 11 muestra una imagen etiquetada en tiempo real por la implementación hardware del algoritmo propuesto.

VI. Conclusiones

Se ha descrito la implementación sobre FPGA de un algoritmo de etiquetado de componentes conexos que aprovecha los intervalos de supresión horizontal y vertical en la fuente de imágenes para operar en tiempo real. El diseño se ha llevado a cabo usando la herramienta System Generator de Xilinx y ha sido validado mediante co-simulación hardware en una placa SPARTAN 3A DSP 3400 para una resolución CIF operando a 40 MHz.

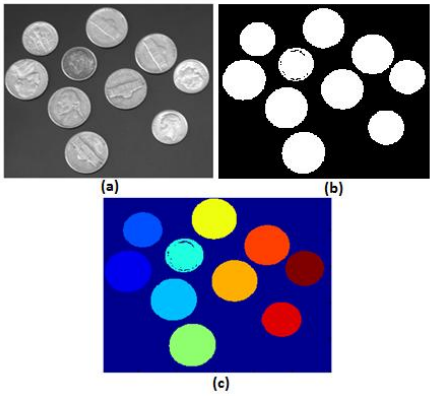

Fig. 11 (a) Imagen original. (b) Imagen umbralizada. (c) Imagen etiquetada con la implementación hardware realizada

AGRADECIMIENTOS

El presente trabajo ha sido financiado por los proyectos: MOBY-DIC FP7-IST-248858 de la CEE, TEC2011-24319 del Ministerio de Ciencia e Innovación y P08-TIC-03674 de la Junta de Andalucía (con soporte FEDER). P.Brox posee un contrato post-doctoral del programa 'Juan de la Cierva' financiado por el Ministerio de Economía y Competitividad del Gobierno de España.

REFERENCIAS

[1] D. Nassini, S. Sahni, "*Finding connected components and connected ones on a mesh-connected parallel computer*", SIAM Journal on Computing, vol. 9, n. 4, pp. 744-757, 1980

[2] R. Cypher, J. L. C Sanz, L. Snyder "*Algorithms for image component labeling on SIMD mesh connected computers*", IEEE Transactions on Computers, vol. 39, n. 2, pp. 276-281, 1990

[3] S.-W Yang et al, "*Parallel 3-Pixel Labeling Method and its Hardware Architecture Design*", Proc. 5th Int. Conf. on Information Assurance and Security, 2009

[4] D. G. Bailey, "*Design for Embedded Image Processing on FPGAs"*, Wiley, 2011

[5] C. T. Johnston, D.G. Bailey, "*FPGA Implementation of a Single Pass Connected Components Algorithm"*, Proc. 4th IEEE Inte. Symp. on Electronic Design, Test & Applications, 2008

[6] D. G, Bailey, C. T. Johnston, "*Single Pass Connected Components Analysis"*, Proc. of Image and Vision Computing, New Zeland, 2007

[7] F. Chang, C. Chen, "*A component-labeling algorithm using contour tracing technique*", Proc. Int. Conf. Document Analysis and Recognition, pp. 741–745, 2003

[8] R. M. Haralick, "*Some neighborhood operations*". M Onoe, K. Jr. Preston y A Rosenfeld. "Real Time Parallel Computer Image Analysis". New York: Plenum Press, pp. 11-35, 1981.

[9] K. Suzuki, I. Horiba, N. Sugie, "*Linear-time connected-component labeling based on sequential local operations*". Computer Vision and Image Understanding, vol. 89, pp. 1-23, 2003.

[10] A. Rosenfeld, J. L. Plazt, "*Sequential operator in digital pictures processing*", Journal of ACM, vol. 13, n. 4, pp. 471-494,1966.

[11] L. Di Stefano, A. Bulgarelli, "*A Simple and Efficient Connected Components Labeling Algorithm*", Proc. 10th Int. Conf. on Image Analysis and Processing, IEEE Computer Society, Washington, DC USA.

[12] L. He, Y. Chao, L. Suzuki, "*A run-based two-scan labeling algorithm*", IEEE Transactions on Image Processing, vol. 17, n. 5, pp. 749–756, 2008.

[13] Matlab example: Modeling a Video Processing System for an FPGA Target. Computer Vision System Toolbox

[14] USC-SIPI (University of Southern California- Signal and Image Processing Institute. [En línea] http://sipi.usc.edu/ database/

[15] Berkeley Computer Vision (University of California). [En línea] http://www.eecs.berkeley.edu/Research/Projects/CS/vision/grouping/resources.html