IMPLEMENTACIÓN DE UN CIRCUITO PARA COMPRESIÓN DE IMÁGENES APLICANDO LÓGICA DIFUSA

Angel Barriga and Nashaat Mohamed

Instituto de Microelectrónica de Sevilla,
Centro Nacional de Microelectrónica,
Consejo Superior de Investigaciones Científicas (CSIC)
y Universidad de Sevilla
Avda. Reina Mercedes s/n, Edif. CICA
41012-Sevilla, España

barriga@imse.cnm.es
nashaat@imse.cnm.es

RESUMEN

Presentamos en esta comunicación un circuito que realiza la compresión de imágenes aplicando una estrategia basada en el uso de lógica difusa. El circuito ha sido implementado sobre FPGA. El grado de compresión es programable ya que se puede seleccionar diferentes niveles de compresión en función de la calidad de imagen requerida. El sistema realiza tres niveles de compresión de imágenes. Así se dispone de la realización de un mecanismo de compresión sin pérdidas y dos técnicas de compresión con pérdidas.

1. INTRODUCCIÓN

El uso de técnicas neuro-difusas en la compresión de imágenes constituye una aplicación muy adecuada por la propia naturaleza del tipo de procesado que se realiza. De hecho la propuesta de este tipo de estrategias no es un hecho reciente [1] [2]. La propia naturaleza de los algoritmos de razonamiento aproximado permite ajustar la precisión de las aproximaciones estableciendo compromisos entre la calidad de la imagen y la razón de compresión. Así en [3] se muestra como un incremento en la fuzzificación da lugar a mejores resultados de compresión si bien como contrapartida se obtiene menor calidad en la imagen comprimida.

Los buenos resultados que se han obtenido en la aplicación de la lógica difusa en la compresión de imágenes se ven empañados a la hora de su utilización práctica debido a la complejidad computacional de los algoritmos. Por ello nuestro interés se ha centrado en la realización de estrategias de compresión que permitan ser implementadas en hardware con un bajo coste de recursos, una alta velocidad de procesado y una adecuada relación entre calidad y razón de compresión.

En el siguiente apartado se discute la fuzzificación de la imagen en función del compromiso calidad-razón de compresión. A continuación, en el apartado 3 se muestra el esquema de codificación empleado. En el apartado 4 se realiza la implementación hardware del circuito que realiza la compresión sobre un dispositivo FPGA. Finalmente se analizan los resultados obtenidos en la aplicación de la estrategia de compresión propuesta.

2. FUZZIFICACIÓN DE LA IMAGEN

Con objeto de simplificar la presentación vamos a considerar en esta comunicación una imagen monocolor de NxM pixels. La imagen se codifica con 8 bits por pixel lo que significa que se distinguen 256 tonos de grises. Por lo tanto el universo de discurso corresponde al rango entre 0 y 255. Dicho universo de discurso se divide en un conjunto de etiquetas lingüísticas que representan a conjuntos difusos. En nuestro caso hemos considerado conjuntos representados por funciones de pertenencia triangulares iguales, equiespaciadas y solapadas entre sí de acuerdo con el esquema de la figura 1. En el ejemplo mostrado en la figura 1 se han empleado 8 etiquetas.

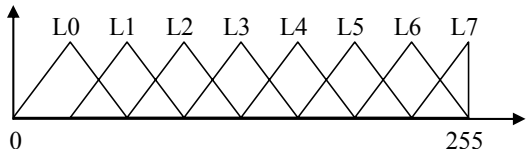

Figura 1. Funciones de pertenencia distribuidas en el universo de discurso de la variable que representa el color de un pixel.

Este trabajo ha sido financiado parcialmente por los proyectos TEC2005-04359/MIC del Ministerio de Educación y Ciencia y TIC2006-635 de la Junta de Andalucía.

El grado de fuzzificación de la imagen va a determinar tanto el grado de compresión como la calidad [3]. Así en el caso de aplicar un modelo *crisp* de la imagen se obtiene un mecanismo de compresión sin pérdidas ya que cada pixel está unívocamente codificado. Este caso extremo permite la mejor calidad. Una mayor compresión significa aumentar el grado de fuzzificación de la imagen (modelo fuzzy). En este caso varios pixels pueden pertenecer a un conjunto difuso y tener, por lo tanto, el mismo código. Dependiendo de la granularidad del conjunto difuso tendremos mayor compresión (y mayor error en la imagen final).

Nuestra estrategia se basa en considerar un esquema de fuzzificación en el que controlemos la razón de compresión modificando el número de bits que se requiere para representar cada pixel.

3. CODIFICACIÓN DE LA IMAGEN

La codificación que se realiza de cada pixel se basa en particionar el universo de discurso en conjuntos difusos. En el ejemplo de la figura 1 se requieren 3 bits para codificar las etiquetas. El grado de pertenencia a cada etiqueta se codifica con 5 bits. De acuerdo con esto podemos realizar una representación exacta de la imagen al establecer un código único para cada pixel. En este caso el tipo de funciones de pertenencia que empleamos se ilustra en la figura 2.

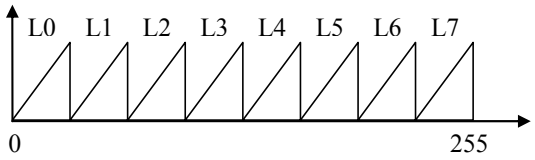

Figura 2. Representación exacta del universo de discurso

El tamaño de la imagen requiere NxMx8. Con objeto de reducir dicho tamaño (compresión) manteniendo la calidad original (compresión sin pérdida) aplicamos un algoritmo de codificación modificando y adaptando la técnica de codificación RLE (*Run Length Encoding)*. Para ello vamos a considerar un imagen como un vector unidimensional de NxM elementos que corresponden a los pixels. Cada uno de ellos se codifica mediante la pareja (etiqueta, grado). De esta manera cada píxel necesita 8 bits. Sin embargo es común que determinadas zonas de la imagen (pixels adyacentes) tengan valores comunes. Nuestra estrategia pretende aprovechar este hecho para reducir el número de bits necesarios ya que cuando los pixels adyacentes corresponden a la misma etiqueta sólo se requiere especificar el grado de pertenencia como elemento distintivo. Este hecho se ilustra en el ejemplo mostrado en la figura 3.

La figura 3a muestra un ejemplo que corresponde a una cadena de 6 pixels codificados con 8 bits cada uno. Los cinco primeros tienen como etiqueta el valor L1 y los grados de pertenencia corresponden a valores g1 (los tres primeros pixels) y g2 (los dos siguientes). El último pixel de la cadena tiene como etiqueta el valor L2 y el grado de pertenencia es g3.

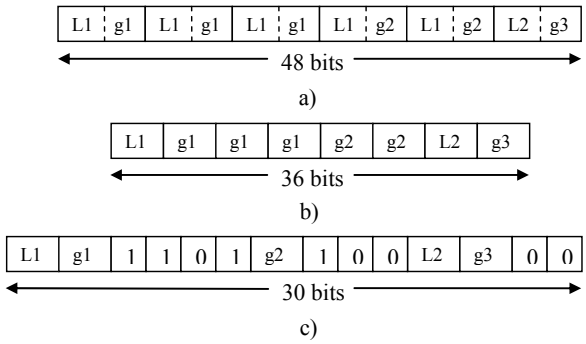

Figura 3. Ejemplo de codificación de la imagen.

Una primera aproximación que permite reducir el tamaño de la cadena es eliminar las etiquetas que se repiten en pixels consecutivos. Así para los 5 primeros pixels sólo tenemos que especificar la etiqueta L1 al comienzo de esa subcadena (figura 3b). En la figura 3b observamos que existe una redundancia en la información ya que hay repeticiones consecutivas en los grados de pertenencia. Así podemos apreciar que el grado g1 se repite tres veces consecutivas y el grado g2 se repite dos veces. Podemos aprovechar esa redundancia para optimizar la longitud de la cadena tal y como se ilustra en la figura 3c. En dicha figura la cadena comienza por especificar la etiqueta L1 y el grado de pertenencia g1. A continuación tenemos 2 bits que actúan de *flags* indicando la repetición del grado g1. Tras dos bits de control se indica el grado g2 y un bit de *flag* indicando su repetición. De esta forma en el ejemplo mostrado se ha reducido en un 37% la longitud de la cadena (de 48 bits a 30).

El esquema de la figura 4 muestra la codificación empleada. La subcadena de etiqueta está compuesta por un conjunto de campos que son los siguientes: 1) el campo etiqueta contiene el código de la etiqueta lingüística; 2) el campo grado contiene el código del grado de pertenencia; 3) el campo FCRG (*Flag* de Control de Repetición del Grado) indica con si el grado aparece un vez y con el valor '0' si ya no aparece; 4) el campo FCRL (*Flag* de Control de Repetición de etiqueta/*Label*) indica si la etiqueta se repite en el siguiente pixel. En caso de repetición de la etiqueta el esquema que continúa la secuencia corresponde a la subcadena de grado (FCRG), mientras que en caso contrario (un '0' en el bit FCRL) significa que el siguiente pixel corresponde a otra etiqueta lingüística por lo que se añadirá la subcadena de etiqueta.

4. IMPLEMENTACIÓN HARDWARE

La implementación hardware del circuito de compresión difuso ha sido realizada sobre dispositivos FPGA Spartan3 XC3S200PQ208 de Xilinx. El diseño se ha realizado a partir de la descripción VHDL de los módulos correspondientes al circuito de compresión y de

descompresión. El flujo de diseño se ha basado en las herramientas de síntesis e implementación disponibles en el entorno de desarrollo ISE de Xilinx

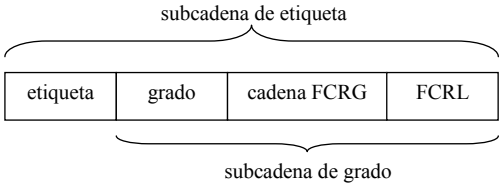

Figura 4. Esquema de codificación.

El esquema del sistema se ilustra en la figura 5. Los bloques `compresión` y `descompresión` leen la imagen secuencialmente por el bus de entrada `Din` y generan las salidas en el bus `Dout`. La selección de uno de los bloques se realiza con la señal `codec`. El sistema permite tres niveles de codificación que se selecciona con las señales de control `fuzz`.

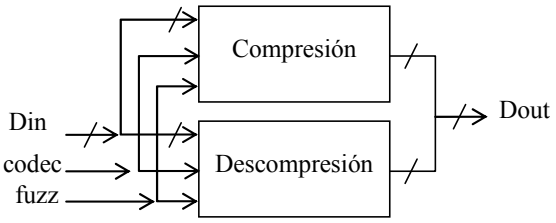

Figura 5. Esquema del sistema de compresión/descompresión.

Los bloques `compresión` y `descompresión` se han diseñado mediante maquinas FSM que realizan el algoritmo descrito en el apartado anterior. La figura 6 muestra el algoritmo de la FSM del circuito de compresión de imágenes.

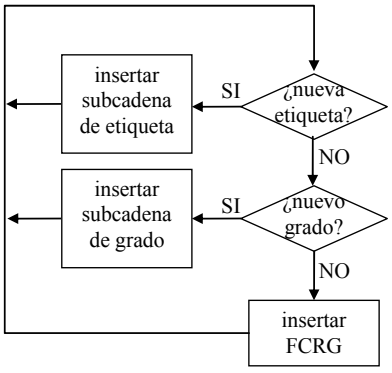

Figura 6. Algoritmo de compresión.

La selección de la granularidad de la compresión de la imagen es programable pudiéndose seleccionar entre las aproximaciones *Crisp*, Fuzz1 y Fuzz2. El circuito recibe la imagen de manera secuencial y procesa un pixel en cada ciclo de reloj. De esta forma el tiempo que se requiere para comprimir una imagen dependerá del tamaño de la misma. La tabla 1 muestra los tiempos de compresión necesarios para imágenes de diferente tamaño. La frecuencia de operación del circuito es de 100 MHz.

	Crisp		Fuzz2	
	comp	decomp	comp	decomp
Soccer (64x64)	125.4	100	108	85
Peppers(115x115)	435.7	344.5	355.2	281
Lena (222x208)	1466.2	1166.7	1250.4	990.5
Cameraman (256x256)	2016.8	1602.5	1509.8	1226.7
Nosveo(389x433)	4481	3606.8	2850.9	2451.8
Goldhill (512x512)	8630.7	6800.6	6984.9	5517.6

Tabla 1. Tiempo de compresión y descompresión de imágenes (en μseg) para una frecuencia de 100 MHz.

Básicamente las operaciones que se necesitan consisten en comparaciones, desplazamientos y contadores. Ello da lugar a un circuito que requiere de pocos recursos de procesado. El área ocupada por el sistema completo ha sido de 330 *slices* lo que supone un 17% de ocupación del dispositivo FPGA seleccionado. El tamaño en número de puertas equivalentes es de 4377 puertas.

La implementación del sistema para imágenes en color se ha realizado sobre un dispositivo FPGA XC2V1500. El circuito ha ocupado 630 slices y 51 IOBs que suponen un total de 8709 puertas equivalentes. La frecuencia de operación máxima ha sido de 158 MHz. Por otro lado se ha implementado el compresor JPEG de [4]. Dicho circuito ha ocupado un área de 6105 slices, 13 BRAM, 2 multiplicadores de 18x18 bits y 77 IOBs que suponen un total de 970707 puertas equivalente. La frecuencia máxima de operación ha sido de 26 MHz.

Podemos observar que la propuesta presentada en esta comunicación representa un menor coste en área ya que dicho circuito supone un 0,6% del tamaño del compresor JPEG. Por otro lado se multiplica por 6 la frecuencia de operación lo que da lugar a menores tiempos requeridos para la compresión de imágenes.

5. RESULTADOS

Vamos a analizar a continuación los resultados obtenidos en la compresión de un conjunto de imágenes. Dicho conjunto de imágenes esta constituido por 12 imágenes utilizadas por diversos autores en la literatura. El análisis se ha realizado para estudiar tanto la razón de compresión como la calidad de las imágenes. La razón de compresión se define como el cociente entre el tamaño original de la imagen y el tamaño de la imagen comprimida.

$$CR = \frac{T_{orig}}{T_{comp}} : 1$$

La razón de compresión se referencia frente a la unidad. Así el objetivo es tener una razón de compresión lo mayor posible. Los resultados obtenidos sobre la razón de compresión se ilustran en la tabla 2. Junto al nombre de cada imagen se ha indicado su tamaño. Para cada imagen de la tabla se muestra la razón de compresión para diversos formatos. Junto a los formatos

estándar JPEG (con pérdidas) y PNG (sin pérdidas) se han considerado tres aproximaciones de nuestro algoritmo. La aproximación *Crisp* corresponde al caso de compresión sin pérdidas mientras que las soluciones Fuzz1 y Fuzz2 son versiones de compresión difusa con distinta granularidad.

Se puede observar que a medida que disminuimos la granularidad (pasamos del algoritmo *Crisp* al Fuzz1 y luego al Fuzz2) se aumenta la razón de compresión. De hecho el caso Fuzz1 es competitivo con JPEG mientras que Fuzz2 lo mejora.

	JPEG	PNG	Crisp	Fuzz1	Fuzz2
Lena(222x208)	1.44	1.43	1.12	1.31	1.62
Soccer(64x64)	1.39	1.39	1.19	1.42	1.73
Peppers(512x512)	1.7	1.53	1.12	1.37	1.79
Baboon(512x512)	1.28	1.26	0.99	1.16	1.4
Cameraman (256x256)	1.62	1.6	1.20	1.53	2.06
Goldhill(512x512)	1.58	1.51	1.10	1.34	1.73
Barbara(512x512)	1.63	1.45	1.05	1.25	1.58
Nosveo(389x433)	2.03	2.32	1.51	2.19	3.29
Peppers_small (115x115)	1.39	1.53	1.18	1.42	1.53
Baboon_small (115x115)	1.33	1.41	1.09	1.26	1.53
Pirata(150x200)	3.37	2.36	1.65	2.22	2.98
Tuneldevertigo (368x381)	1.92	2.02	1.48	1.97	2.68

Tabla 2. Razón de compresión de imágenes.

En lo que respecta a la calidad hemos empleado el error cuadrático medio como indicador de dicho parámetro. Dicho error se calcula sumando el cuadrado del error, esto es, de la diferencia entre cada pixel de la imagen original y la final.

$$RMSE = \sqrt{\frac{\sum_{i,j}(img_{orig}(i,j) - img_{final}(i,j))^2}{N \times M}}$$

Los formatos PNG y *Crisp* son estrategias de compresión sin perdidas por lo que el error de compresión es cero en ambas aproximaciones. Los casos de algoritmos con pérdidas se ilustran en la tabla 3.

	JPEG	Fuzz1	Fuzz2
Lena	0.468	0.737	1.975
Soccer	0.474	0.756	2.033
Baboon	0.48	0.708	1.873
Cameraman	0.475	0.705	1.857
Goldhill	0.48	0.708	1.874
Barbara	0.479	0.707	1.870
Nosveo	0.451	0.718	1.939
Peppers_small	0.481	0.709	1.889
Baboon_small	0.483	0.709	1.874

Tabla 3. RMSE de las imágenes.

Se puede apreciar que el algoritmo JPEG suministra mejor calidad. También se observa el efecto mencionado de reducción en la calidad a medida que se disminuye la granularidad de la aproximación difusa (mayor fuzzificación significa menor calidad).

6. CONCLUSIONES

Se ha presentado la realización de un circuito que realiza la compresión de imágenes aplicando diferentes niveles de fuzzificación. El esquema de codificación descrito permite realizar una compresión sin pérdidas. Otros esquemas de compresión con pérdidas implementados en el circuito permiten seleccionar la granularidad de la fuzzificación lo que da lugar a diversos resultados en los que se controla la razón entre calidad y grado de compresión.

7. REFERENCIAS

[1] S.-G. Kong, B. Kosko, "Adaptive Fuzzy System for Transform Image Coding", In *International Joint Conference on Neural Networks*, IJCNN-91-Seattle, July 1991.

[2] A. Steudel, M. Glesner, "Image Coding with Fuzzy Region-Growing Segmentation", In *Proc. IEEE Int. Conf. on Image Proc.*, Lausanne, Suisse, September, 1996.

[3] K. Hirota, W. Pedrycz, "Fuzzy Relational Compression", *IEEE Trans. in System, Man and Cybernetics – Part B: Cybernetics*, vol. 29, no. 3, pp. 407-415, June 1999.

[4] http://www.opencores.org

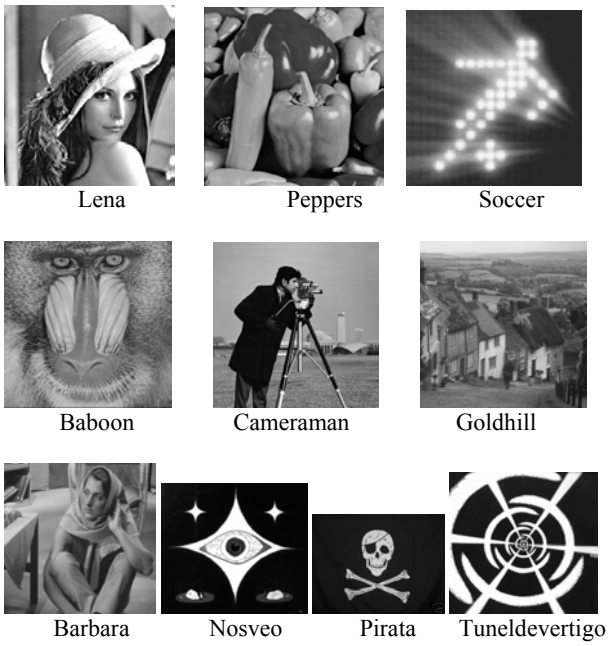

Figura 7. Banco de imágenes de test.