SoC basado en un FPGA para Cuantización Vectorial

José Yair Martínez Cantellán
Instituto Tecnológico de Celaya
yair_cantellan@hotmail.com

Agustín Ramírez Agundis
Instituto Tecnológico de Celaya
agustin.ramirez@itcelaya.edu.mx

Javier Díaz Carmona
Instituto Tecnológico de Celaya
javier.diaz@itcelaya.edu.mx

Resumen

En el presente trabajo se presenta la implementación de un cuantizador vectorial en un dispositivo FPGA con el propósito de reducir el tiempo de procesamiento en comparación con su implementación puramente software. El sistema SoC lleva a cabo la clusterización vectorial utilizando dos sistemas independientes, uno hardware y otro software, comunicados entre sí para realizar la cuantización vectorial de una imagen en escala de grises. La introducción y recuperación de datos en y desde el SoC se realiza utilizando Matlab en una computadora personal. También en Matlab se desarrolló un cuantizador vectorial software para comparar los tiempos de procesamiento. Los resultados muestran que existe una reducción en tiempo de más del 87% cuando se utiliza el cuantizador vectorial SoC.

Palabra(s) Clave(s): Cuantizador vectorial, FPGA, SoC.

1. Introducción

La cuantización vectorial (VQ, por sus siglas de Vector Quantizier) es una técnica clásica de procesamiento de señales que permite el modelado de funciones de densidad de probabilidad mediante la distribución de vectores prototipo, utilizada en el procesamiento de imágenes como un algoritmo de compresión con pérdidas.

Normalmente la VQ se desarrolla en software, pues es una secuencia simple de implementar en lenguajes de alto nivel, pero es posible acelerar el procesamiento mediante su implementación SoC (por las siglas de System on Chip).

La frecuencia de trabajo de un procesador computacional, al tiempo de este escrito, es marcadamente superior a la de un dispositivo FPGA, sin embargo una descripción VHDL adecuada y su implementación en un dispositivo FPGA permite la ejecución independiente de varias secciones de código, es decir procesamiento paralelo.

El objetivo de este trabajo es reducir el tiempo de procesamiento mediante la implementación hardware (SoC) del cuantizador vectorial en comparación con la implementación puramente software.

A continuación se exponen brevemente algunos conceptos fundamentales de la cuantización vectorial. En el siguiente apartado se describe con cierto detalle la estructura y características del sistema. En la sección 3 se presentan los resultados obtenidos mediante pruebas en software y hardware. La sección 4 está dedicada a elaborar una breve discusión del trabajo. Finalmente, la sección 5 incluye las conclusiones y recomendaciones para trabajos futuros.

Cuantizador Vectorial

La cuantización vectorial es un método clásico de aproximación de señales, que usualmente construye una aproximación cuantizada para cada uno de los vectores de entrada, usando un número finito de vectores llamados *vectores código,* los cuales

constituyen un *diccionario*. Partiendo de un diccionario predefinido, un vector X se codifica por medio del índice i del vector código más cercano, es decir, el vector X se aproxima mediante el vector código i [1]. En la Fig. 1 se muestra la manera como se cuantiza una imagen de $4x4$ pixeles usando bloques de imagen de $1x4$ pixeles y 10 vectores código, cada uno de 4 elementos.

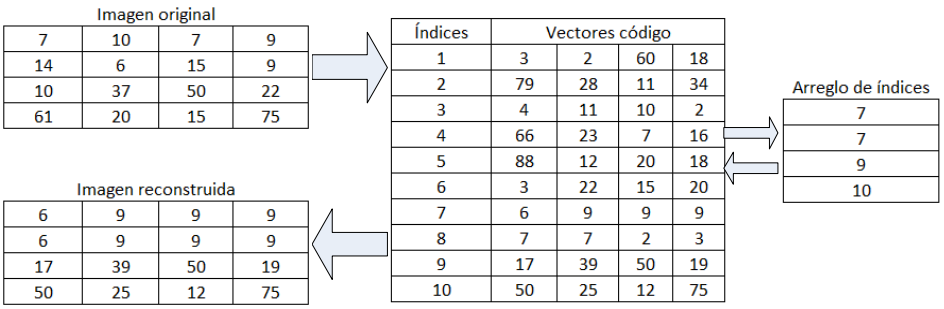

Fig. 1. Cuantización vectorial.

2. Desarrollo

Estructura de la plataforma experimental

La plataforma está compuesta por un dispositivo FPGA y una PC, comunicados mediante el puerto serial (protocolo RS232). La PC hace uso del software Matlab y Office Excel, mientras que el dispositivo FPGA aloja al SoC desarrollado, estando éste constituido por un sistema software (procesador *Microblaze*) y un sistema hardware. La estructura de la plataforma se aprecia en la Fig. 2.

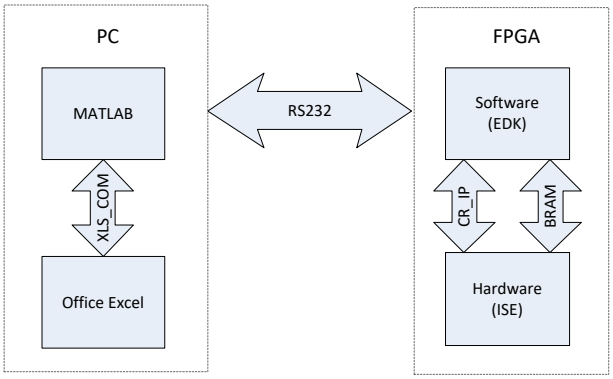

Fig. 2. Esquema de la plataforma experimental.

El sistema software tiene como elemento central un microprocesador *soft Microblaze* y fue desarrollado en EDK de Xilinx, incluyendo varios controladores de dispositivos IP que fungen como periféricos conectados mediante el bus local del procesador PLB v4.6 [2].

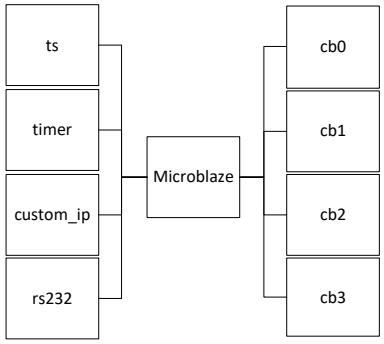

Fig. 3. Esquema de arquitectura software.

En la Fig. 3 se aprecia un esquema de la arquitectura del sistema software, la cual incluye 8 controladores, siendo éstos:

- RS232, permite la comunicación serial mediante el protocolo RS232.
- Timer, permite el conteo de periodos de los relojes disponibles en la arquitectura.

- Custom IP, controlador de diseño original que permite la transferencia de señales tipo DWORD entre el microprocesador y cualquier bloque externo a éste.
- TS, CB0, CB1, CB2, CB3 controladores para memorias BRAM de profundidad 16384 con registros de 32 bits.

Los comandos para cada controlador son escritos en lenguaje C, lo que simplifica la ejecución de varias tareas que comúnmente requerirían del desarrollo de controladores basados en diagramas de tiempo.

Las memorias BRAM, controladas por los bloques mencionados anteriormente (TS, CB0, CB1…), son memorias de doble puerto verdadero [3]. En este caso el puerto A está conectado a los controladores del Microblaze, mientras que el puerto B se utiliza para que el sistema hardware acceda a las memorias. Ver Fig. 4.

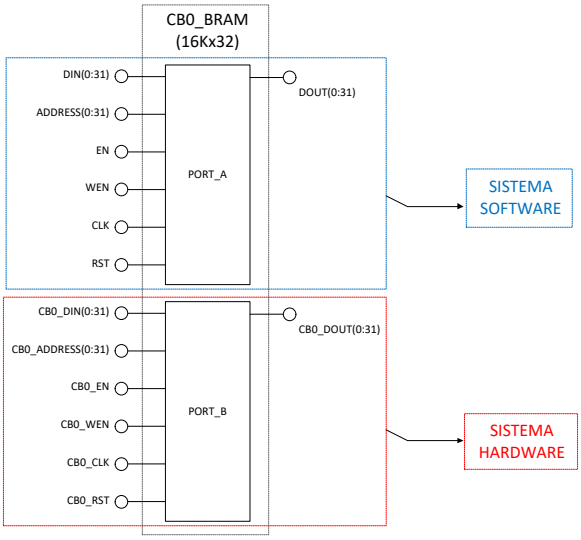

Fig. 4. Disposición de puertos de memorias RAM.

Estas memorias compartidas son un espacio de trabajo común para el sistema software y el sistema hardware, formando junto con el bloque *custom_ip* un puente de enlace entre los dos sistemas que coexisten en el dispositivo FPGA (ver Fig. 5).

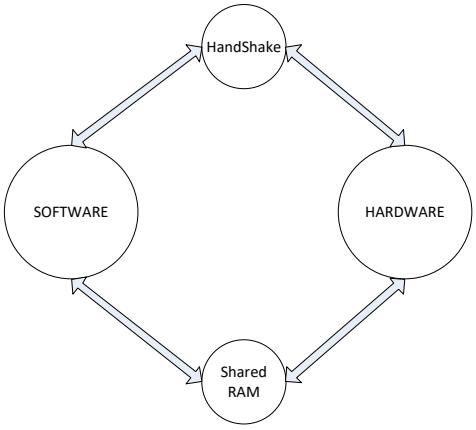

Fig. 5. Comunicación entre sistema software y sistema hardware.

El sistema hardware es un conjunto de bloques combinacionales y secuenciales síncronos controlados por una máquina de estados finita, desarrollados en VHDL en ISE de Xilinx. Un esquema simple de este sistema se aprecia en la Fig. 6.

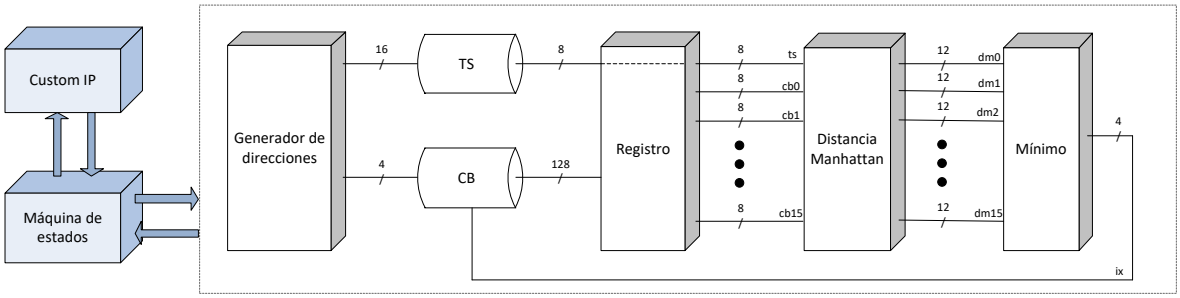

Fig. 6. Esquema de sistema hardware.

Cada uno de los bloques ilustrados representa circuitería digital con la capacidad de trabajar independientemente de los demás, por lo que se obtiene un paralelismo en el procesamiento de datos:

- Máquina de estados, controla el flujo del proceso.
- Custom IP, mismo bloque diseñado en EDK bajo el dominio de Microblaze.

- Generador de direcciones, con base en contadores y multiplexores genera las direcciones de trabajo de las memorias BRAM.
- TS y CB, controladores basados en el diagrama de tiempo de las memorias BRAM diseñados para leer/escribir en éstas.
- Registro síncrono diseñado para almacenar por un tiempo determinado el dato obtenido en el puerto B de las memorias.
- Distancia Manhattan, consiste de 16 procesadores que operan en paralelo, cada uno calcula la distancia Manhattan entre los 16 vectores código y un mismo vector de entrada.
- Mínimo, bloque basado en el concepto *winner takes all* diseñado para obtener el índice correspondiente al menor valor de las 16 distancias Manhattan.

El diagrama expandido de la arquitectura hardware, excluyendo la máquina de estados y el bloque Custom IP, se muestra en la Fig. 7; en ésta se puede apreciar el paralelismo de las operaciones realizadas por los bloques Distancia Manhattan y Mínimo.

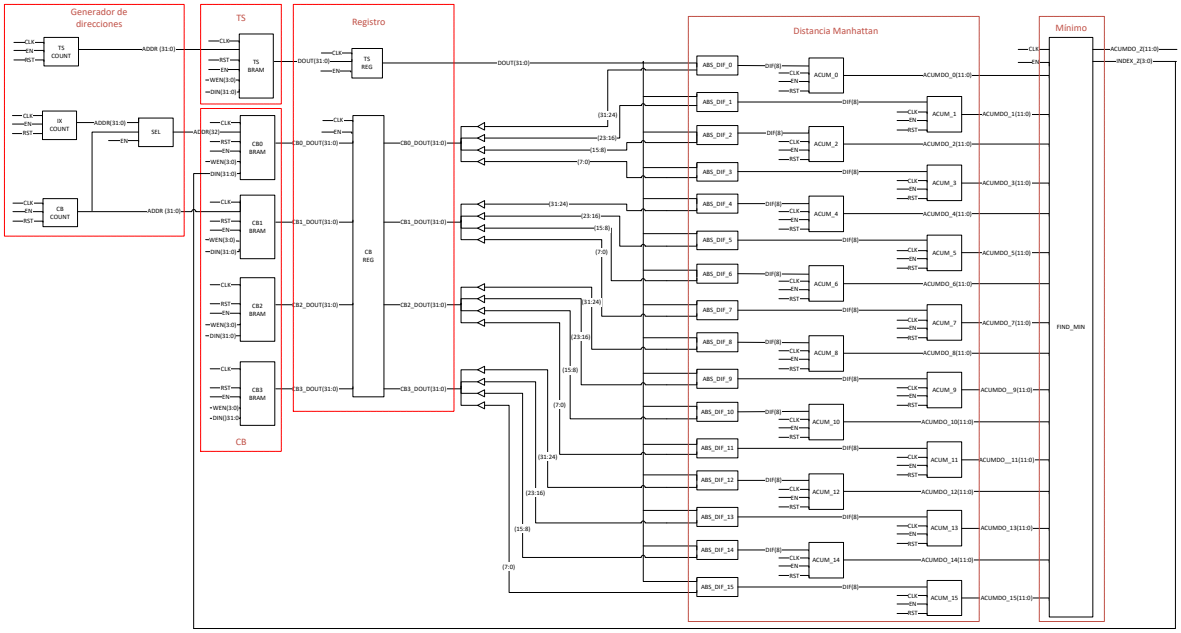

Fig. 7. Arquitectura hardware.

Pre-procesamiento

El VQ desarrollado codifica imágenes de $128x128$ pixeles con tonos de gris de 8 bits. Se utiliza un diccionario constituido por 16 vectores código con 16 elementos cada uno. La imagen se descompone en bloques de $4x4$ pixeles, de manera que se obtienen 1024 vectores de 16 elementos, almacenándose todos ellos en un arreglo de 16384 elementos con valores de 8 bits (0 a 255) llamado *TS*. En la Fig. 8 se aprecia la aplicación de este proceso para la imagen *Lena,* almacenando cada bloque de imagen de tamaño $4x4$, recorriendo la imagen de izquierda a derecha y de arriba a abajo.

Para realizar la VQ se utiliza un diccionario constituido por 16 vectores código generados aleatoriamente. Los vectores código se acomodan de forma progresiva (0 a 15) en un solo arreglo llamado CB de 256 elementos.

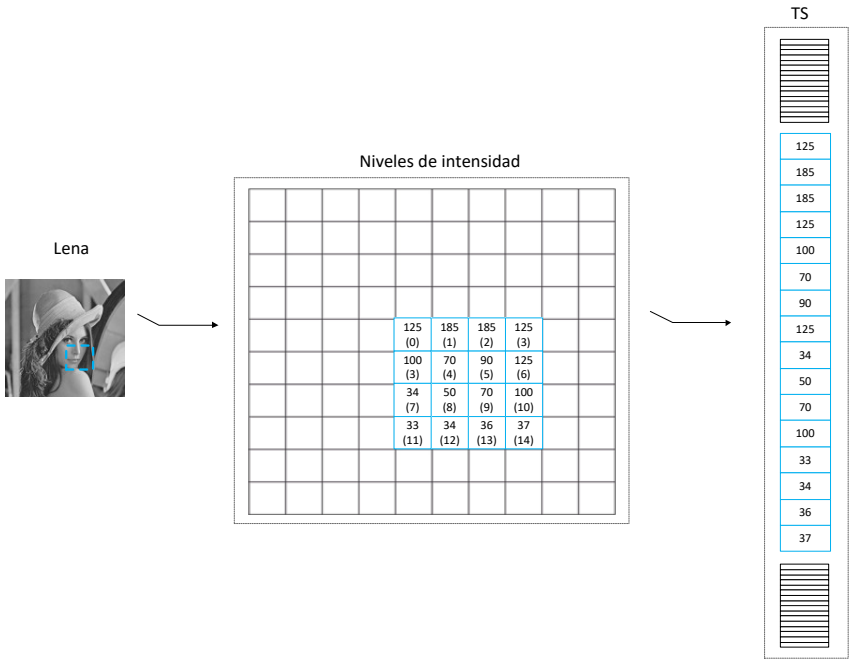

Fig. 8. Almacenamiento de las intensidades de una imagen.

El pre-procesamiento descrito es realizado en la plataforma Matlab mediante código y funciones escritas en archivos .m, para posteriormente transmitir los arreglos resultantes (TS y CB) a la plataforma FPGA mediante el puerto serial.

Recepción y almacenamiento

Ambos arreglos (TS y CB) son recibidos y almacenados en la plataforma FPGA mediante el sistema software. Conforme el bloque RS232 adquiere los valores del puerto serial, éstos son colocados en su correspondiente memoria; en el caso del arreglo TS, los datos se almacenan en la memoria TS a la par de su lectura, mientras que para almacenar el arreglo CB se realiza un acomodo a fin de distribuir el diccionario en las 4 memorias CB, de modo que cada una almacena 4 vectores código, esto es posible ya que los vectores código tienen valores de 8 bits y la memoria es de 32 bits, como se observa en la Fig. 9.

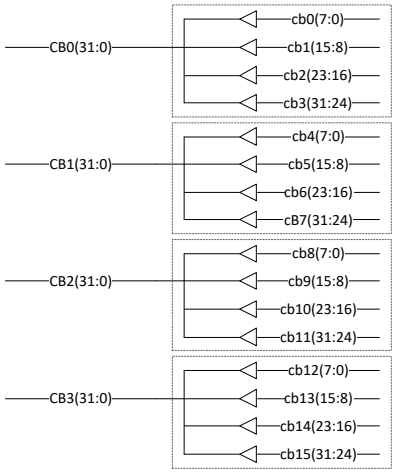

Fig. 9. Disposición de diccionario en memorias BRAM.

Al término del almacenamiento de datos, el sistema software queda a la espera de la instrucción de inicio de procesamiento por parte de Matlab.

Procesamiento

Una vez que Matlab genera la indicación de iniciar el procesamiento hardware, el bloque Custom IP es el encargado de mandar la señal de arranque a la máquina de estados. Un diagrama simplificado del flujo del proceso controlado por la máquina de estados se aprecia en la Fig. 10.

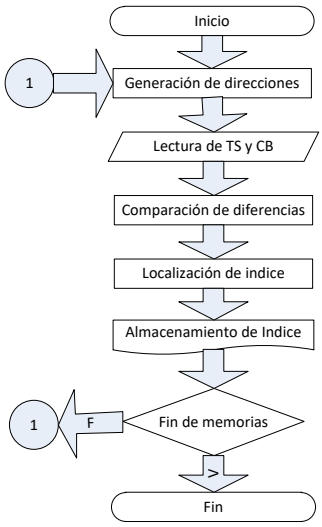

Fig. 10. Diagrama de tareas del sistema hardware.

El índice se obtiene de las operaciones aritméticas aplicadas sobre los elementos del vector correspondiente al bloque de imagen y los vectores código, de acuerdo a las ecuaciones (1) y (2):

$$a_j = \sum_{i=1}^{16} |ts_i - cb_i^j|, \quad j = 0, 2, \dots, 15 \tag{1}$$

$$ix = \min(a_0, a_1, a_2, \dots, a_{14}, a_{15}) \tag{2}$$

Donde ts_i es el $i-esimo$ elemento del bloque de imagen a codificar y cb_i^j es el $i-esimo$ elemento del $j-esimo$ vector código.

Este proceso se repite para cada uno de los bloques de imagen, por lo que se obtendrán 1024 índices en total.

La mayor aceleración del proceso de cuantización vectorial se obtiene en los bloques Distancia Manhattan y Mínimo, al permitir realizar la clusterización de forma paralela [4]. Un diagrama de estos bloques se observa en la Fig. 11.

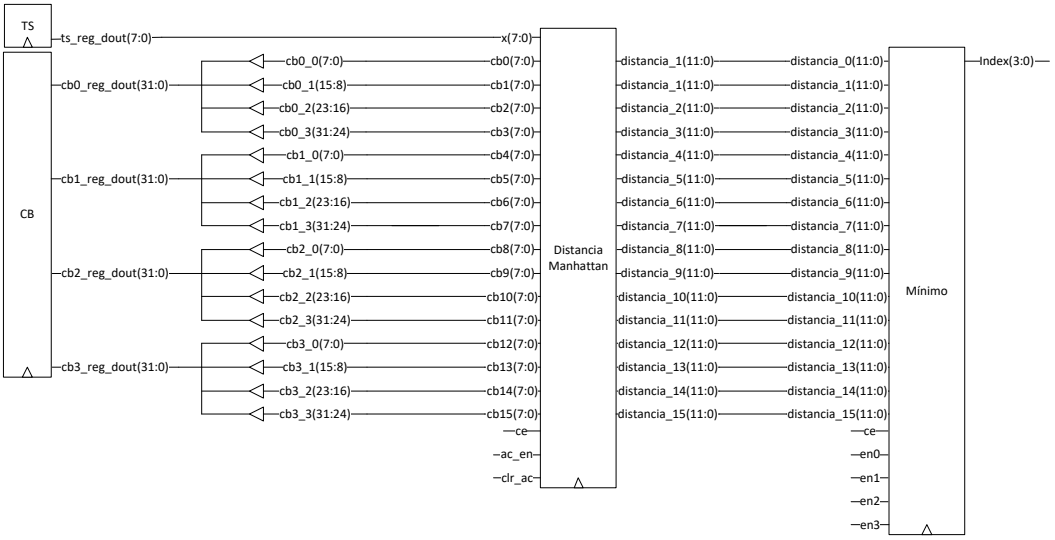

Fig. 11. Diagrama de los bloques Distancia Manhattan y Mínimo.

La implementación hardware del bloque Distancia Manhattan se obtiene de manera directa y está formado por un extractor de valor absoluto de las diferencias y un acumulador de valores absolutos. La componente a procesar del vector de entrada y la parte de la componente respectiva del vector código se extiende un bit (de 8 a 9 bits) y después se restan. El bit más significativo de la diferencia así obtenida es el bit de *borrow*. Si éste es '0', el valor absoluto será igual a la diferencia; si el bit de *borrow* es '1', entonces el valor absoluto es igual al complemento a dos de la diferencia.

Debido a que para el diseño se consideraron vectores de 16 elementos y a que los valores absolutos son de 8 bits, el acumulador que produce la distancia Manhattan es de 12 bits.

El bloque Mínimo es un árbol de comparadores/multiplexores con reutilización. Para cada uno de los módulos se tiene un árbol de 4 niveles. En su primer nivel tiene 8 comparadores/multiplexores, cada uno recibe un par de distancias, una asociada a un índice par y una asociada a un índice non. En su salida se tendrá la distancia menor y un bit que será '0' si está fue la correspondiente al índice par o un '1' en caso contrario. Las distancias se van propagando a través del árbol al mismo tiempo que los bits se van concatenando, hasta que finalmente a la salida del cuarto nivel, se tendrá la distancia mínima y el índice de 4 bits correspondiente [5].

Cada índice obtenido es almacenado en la memoria CB0, pero sobre localidades no utilizadas por el diccionario (0 a 15), mediante el bloque generador de direcciones.

Recuperación de índices

Cuando se han obtenido los 1024 índices, la máquina de estados manda una respuesta al sistema software mediante el bloque Custom IP informando de la finalización del proceso.

Para obtener los valores de los índices generados, el sistema software realiza la lectura de los datos escritos por el sistema hardware en la memoria CB0, específicamente en las localidades 8192 a 9216 y los transmite por el puerto serial a Matlab.

Fases de la plataforma SoC

El proceso llevado a cabo por la plataforma SoC se realiza de acuerdo con las siguientes fases:

- Fase 1: Matlab genera los vectores de imagen y diccionario.

- Fase 2: Matlab transmite vía RS232 los vectores de imagen y diccionario. El sistema software almacena los vectores en las memorias compartidas respectivas.
- Fase 3: Matlab genera la orden requerida por el SoC para iniciar la VQ. El sistema software arranca la máquina de estados mediante un registro (*custom ip*) y queda a la espera de la respuesta de terminación.
- Fase 4: El sistema hardware realiza el proceso descrito por la máquina de estados para realizar la cuantización vectorial utilizando los arreglos TS (imagen) y CB (diccionario) almacenados en las memorias compartidas y almacena los índices resultantes en la memoria CB0. Al alcanzar el último estado de la máquina, se envía la señal de terminación al sistema software (Microblaze).
- Fase 5: El sistema software transmite vía RS232 el arreglo de índices a Matlab.
- Fase 6: Matlab posee la capacidad de almacenar los arreglos en hojas de cálculo.

3. Resultados

A fin de comparar los resultados obtenidos por la plataforma SoC con una implementación puramente software, se elaboró un programa en Matlab que realiza las mismas tareas del sistema hardware descritas en la sección 2.

Especímenes

Los especímenes utilizados para realizar las pruebas son cuatro imágenes con características deseables, resolución $128x128$ pixeles y escala de grises, ver Fig. 12.

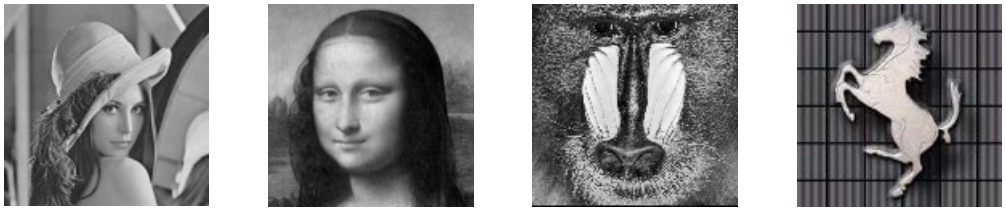

Fig. 12. Imágenes de prueba de izquierda a derecha: Lena, Gioconda, Mandril y Ferrari.

Se emplearon tres diccionarios diferentes generados al azar pero idénticos para ambas implementaciones.

Características de la plataforma

Las características de la PC son: *Intel® Core™ i7 4710HQ, 2.5-3.5GHz; Corsair DDR3 RAM 8GB, 1600MHz; Matlab R2013a 64 bits* [6].

El único proceso en ejecución es Matlab y la prioridad del mismo es *default*.

Las características del SoC son: *XUPV5-LX110T Evaluation Platform; Virtex-5 XC5VLX110T FPGA; ISE Design Suite 14.5*, de Xilinx [7].

El reloj del procesador Microblaze y el del hardware se encuentran a una frecuencia de trabajo de 100MHz.

Comparación de resultados de la VQ

Se realizaron 120 pruebas con ambas implementaciones, generando cada una de ellas los mismos índices, es decir, los 1024 índices obtenidos en cada prueba con la implementación hardware coinciden con los obtenidos con la implementación software.

Tiempo de ejecución del VQ en Matlab.

Haciendo uso del comando TIC TOC, se cuenta el tiempo transcurrido desde el inicio del procesamiento hasta la obtención de último índice [8].

Realizando mil pruebas por diccionario a cada imagen con el algoritmo de cuantización vectorial se obtuvieron los resultados (promedio) mostrados en la Tabla 1.

	Tiempo transcurrido (s)			
	Lena	**Gioconda**	**Mandril**	**Ferrari**
Diccionario 1	0.0075	0.0074	0.0077	0.0073
Diccionario 2	0.0075	0.0075	0.0075	0.0075
Diccionario 3	0.0075	0.0073	0.0075	0.0073
Promedio	0.0075	0.0074	0.0076	0.0074

Tabla 1. Resultados cuantizador vectorial software.

Tiempo de ejecución del VQ en plataforma SoC

Haciendo uso del bloque IP Timer, se cuentan los tics (periodos) del reloj del sistema que transcurren desde el inicio de la máquina de estados finita hasta su último estado [9].

Se realizaron diez pruebas por cada diccionario a cada imagen con la plataforma SoC de cuantización vectorial. El tiempo de procesamiento fue de 0.0009s (en todos los casos).

Comparación de tiempos de ejecución

La comparación porcentual del tiempo de ejecución de la plataforma SoC con respecto a la implementación software del cuantizador vectorial bajo las configuraciones descritas se muestra en la Tabla 2.

Imagen	Software	Hardware	Ahorro
Lena	0.0075s	0.0009s	88.00%
Gioconda	0.0074s	0.0009s	87.84%
Mandril	0.0076s	0.0009s	88.16%
Ferrari	0.0074s	0.0009s	87.84%

Tabla 2. Resultados comparativos.

4. Discusión

Los valores del tiempo de ejecución no son iguales para todas las pruebas, aun en los casos en que se trabaje con la misma imagen. Lo valores varían debido a razones propias del software y la forma de trabajo del PC.

A fin de reducir la incertidumbre en el tiempo de ejecución de la cuantización vectorial en Matlab, se realizó un ciclo de $1{,}000$ repeticiones de esta misma. El tiempo obtenido, medido desde antes de entrar al ciclo y hasta terminar éste fue divido entre $1{,}000$ obteniendo así el tiempo que tarda cada cuantización. Se experimentó con diferentes valores para la cantidad de cuantizaciones. Se encontró que el tiempo por cada cuantización varía en función de la cantidad de cuantizaciones siguiendo una tendencia a la baja. El tiempo por cuantización se estabiliza a partir de 500. Por esta razón se realizó el ciclo de $1{,}000$ repeticiones.

Un sistema computacional es construido con sus propios relojes internos; el número y variedad dependen específicamente del hardware. La arquitectura del hardware tiene la posibilidad de incluir varios CPU, o núcleos, todos corriendo a diferentes velocidades. El sistema operativo puede ejecutarse en hardware o depender de la virtualización de hardware. El sistema operativo provee varios servicios de tiempo que dependen del acceso a relojes hardware. Algunos valores obtenidos están sincronizados con relojes externos y otros no. Por ejemplo, para obtener resultados en segundos TAI (International Atomic Time), se requiere una sincronización con fuentes TAI. El NTP (Network Time Protocol) puede ser usado para realizar esto vía internet, pero mantener una precisión alta constantemente es difícil por causa de la variabilidad de retardo del servidor. Software cliente/servidor especializado es requerido para suavizar la variabilidad, y los proveedores de sistemas operativos no han acordado una solución. Como resultado, los relojes internos de un sistema computacional parecen tener una frecuencia efectiva de solo 10^2 Hertz (10 mili segundos) [10].

Para la descripción hardware, la reiteración del proceso con la misma cantidad de datos bajo circunstancias y condiciones normales, no va a cambiar el tiempo de ejecución.

Por simple inspección es notorio que cada uno de los 16 procesadores implícitos en la plataforma SoC sólo realiza 1/16 de las operaciones que ejecuta el procesador de la PC, reduciendo el tiempo de ejecución, por lo cual la implementación SoC es una mejor opción cuando el tiempo de procesamiento es crítico.

La complejidad de la descripción hardware con respecto a la programación de alto nivel es un factor importante en la elaboración de trabajos de este tipo si el plazo del proyecto es corto, una instrucción de alto nivel puede requerir cientos de líneas de código en la descripción hardware.

5. Conclusiones

Se desarrolló, implementó y probó una plataforma SoC que realiza la tarea de cuantización vectorial con datos obtenidos de una imagen.

Utilizando Matlab como herramienta para el desarrollo de un VQ software con la misma lógica de procesamiento, se obtienen índices idénticos en ambos sistemas.

Los resultados de las pruebas demuestran que utilizando la plataforma SoC existe un ahorro de tiempo de procesamiento de más del 87% en comparación con la implementación software del mismo cuantizador vectorial.

Es necesario que en trabajos futuros se analice mejorar el tiempo de ejecución de la implementación software del VQ mediante el aumento de prioridades en el proceso, bloques de tiempo real, loop unrolling de secciones de código, uso de funciones propias de Matlab, etc.

Así mismo sería conveniente implementar el diseño de la plataforma SoC en una FPGA reciente.

6. Referencias

[1] A. M. Montenegro Díaz, J.E. Ortiz Pinilla, Modelamiento Estadístico. Primera Edición. 2005. Universidad Nacional de Colombia. Colombia. Página 100.

[2] ML505/ML506/ML507 Evaluation Platform User Guide. Xilinx®. Estados Unidos. 2011.

[3] EDK Concepts, Tools, and Techniques, A Hands-On Guide to Effective Embedded Design. Xilinx®. Estados Unidos. 2012.

[4] A. Ramírez, R. Gadea, “A mixed HW-SW system for fast codebook generation with the LBG algorithm”. Advances in Electronics and Micro-electronics. ENICS ’08. Septiembre 2008. International Conference On.

[5] A. Ramírez, R. Gadea, “A wavelet VQ system for real-time video compression”. Journal of Real-Time Image Processing. Springer Link. Diciembre, 2007. Volume 2. Issue 4.

[6] Intel® Core™ i7 4710HQ Processor. ark.intel.com/es/products/78930/Intel-Core-i7-4710HQ-Processor-6M-Cache-up-to-3_50-GHz. Julio 2015.

[7] Xilinx® University Program XUPV5-LX110T Development System. www.xilinx.com/univ/xupv5-lx110t.html. Julio 2015.

[8] Tic. www.mathworks.com/help/mathlab/ref/tic.html. Julio 2015.

[9] LogiCORE IP XPS Timer/Counter. Xilinx®. Estados Unidos. 2010.

[10] M. Knapp-Cordes, B. Mckeeman. “Improvements to tic and toc Functions for Measuring Absolute Elapsed Time performance in Matlab”, Matlab Digest. mathworks.com. 2011.