MPCM: Codificador hardware para cámaras de alta velocidad

Estefanía Alcocer, Otoniel López-Granado, Manuel P. Malumbres[1] y Roberto Gutiérrez[2]

Resumen— **En la última década, las mejoras a nivel de integración VLSI y en las tecnologías de captura de imágenes han encabezado una carrera frenética para proporcionar sistemas de procesamiento de vídeo capaces de capturar y comprimir secuencias de vídeo con resoluciones y tasas de frames muy elevadas. Estas cámaras de vídeo de alta velocidad se utilizan en aplicaciones científicas e industriales, como en tests de accidentes automovilísticos, en investigación en combustión, ensayos de materiales, dinámica de fluidos, visualización de flujos, etc, que demandan la captura y almacenamiento de vídeo en tiempo real a muy altas velocidades (más de 1000 fps) y con formatos de alta definición. Por lo tanto, la capacidad de almacenamiento de datos, el ancho de banda de la comunicación, el tiempo de procesamiento y el consumo de energía son parámetros críticos que deben ser considerados cautelosamente en su diseño. En este trabajo, se propone una implementación en FPGA de un codificador rápido y sencillo llamado MPCM, el cual obtiene calidades similares a la codificación PCM tradicional, pero que es capaz de reducir los requisitos de ancho de banda hasta 1.4 veces, lo que permite a cámaras de muy alta velocidad capturar de manera continua en un dispositivo de almacenamiento masivo como una SSD.**

Palabras clave— **PCM, codificación de imagen, diseño FPGA, alta velocidad, circuitos integrados.**

I. Introducción

La compresión de vídeo ha sido una tecnología de gran éxito que ha encontrado su aplicación comercial en muchas áreas, en aplicaciones científicas e industriales como el almacenamiento de vídeo, imágenes médicas de alta calidad y aplicaciones de vigilancia y seguridad, en la industria audiovisual (cine y televisión) y en la amplia gama de aparatos de vídeo disponibles en el mercado como cámaras digitales, DVD, Blue-Ray, DVB, etc.

En la última década, las mejoras a nivel de integración VLSI y en las tecnologías de captura de imágenes han encabezado una carrera frenética para proporcionar sistemas de procesamiento de vídeo capaces de capturar y comprimir secuencias de vídeo con resoluciones y tasas de frames muy elevadas. Hoy en día, se pueden encontrar en el mercado cámaras de vídeo de ultra alta velocidad como Phantom v641 [1], que es capaz de capturar vídeo de alta resolución (2560 x 1600 píxeles) a 1.450 frames por segundo (fps). Estas cámaras de vídeo son especialmente adecuadas para aplicaciones científicas o industriales, como tests de accidentes automovilísticos, explosivos y pirotecnia, balística, seguimiento de proyectiles, investigación en combustión, ensayos de materiales, dinámica de fluidos, visualización de flujos, etc, que demandan la captura de vídeo en tiempo real a muy altas velocidades y con formatos de gran definición. Por lo tanto, la capacidad de almacenamiento de datos, el ancho de banda de la comunicación, el tiempo de procesamiento y el consumo de energía son parámetros críticos que deben ser considerados cautelosamente en el diseño de cámaras de vídeo de alta velocidad.

Actualmente, la mayoría de cámaras de alta velocidad almacenan las imágenes capturadas en un módulo SDRAM de hasta 64 GB [1], [2], sin realizar compresión, usando Pulse Code Modulation (PCM) [3], generando una enorme cantidad de datos sin comprimir que necesitan ser procesados para su transmisión o almacenamiento posterior. Por tanto, el bus de comunicación interna no es lo suficientemente rápido para transferir el vídeo desde la cámara, o la velocidad de escritura del dispositivo de almacenamiento no es lo suficientemente alta como para guardar los datos en tiempo real [4]. Consecuentemente, el enfoque de la utilización de la memoria SDRAM como almacenamiento de vídeo es factible ya que el ancho de banda de memoria es lo suficientemente alto, pero cuando la memoria se agota, la cámara deja de grabar y tiene que guardar el vídeo en un dispositivo de almacenamiento secundario en crudo o en formato comprimido. Esta es una limitación, ya que dependiendo de la resolución de captura de la cámara, sólo se registrarán unos pocos segundos en la memoria, y por lo tanto, no se realizará una captura continua.

Con el fin de superar estas restriciones, sería de interés reducir las necesidades de almacenamiento de vídeo a través de codificadores hardware que cumplan con los requisitos de la aplicación, es decir, alto frame rate y alta definición de imagen. Por lo tanto, si somos capaces de llevar a cabo un tipo de codificación muy rápida, reduciremos los recursos de almacenamiento, y será posible la grabación en tiempo real, al igual que en las cámaras convencionales.

Muchos codificadores hardware basados en diferentes algoritmos de codificación se utilizan en sistemas reales [5], [6], [7], [8], [9]. La mayoría de ellos son Circuitos Integrados de Aplicación Específica (ASICs) dedicados a algoritmos específicos de codificación que no están diseñados para trabajar en tiempo real con altos frame rates y con formatos de vídeo de alta definición.

Sin embargo, se han hecho varios estudios sobre la codificación en cámaras de alta velocidad. En [10] los autores presentan un codificador JPEG basado en FPGA, que es capaz de comprimir hasta 500

[1]Dpto. de Física y Arquitectura de Computadores, Univ. Miguel Hernández. Elche, e-mail: {ealcocer, otoniel, mels}@umh.es

[2]Dpto. de Ingeniería de Comunicaciones, Univ. Miguel Hernández. Elche, e-mail: roberto.gutierrez@umh.es

frames/s con una resolución de $1280x1024$. Además, en [11] una versión mejorada del algoritmo Fast Boundary Adaptation Rule (FBAR) [12] en conjunto con la modulación de código de pulso diferencial (DPCM) se aplica para aumentar la eficiencia R/D, aunque no se proporcionan los tiempos de codificación.

En general, las limitaciones impuestas por las aplicaciones de captura de vídeo con altos frame rates descartan la mayor parte de las técnicas de codificación existentes (por ejemplo, la codificación por predicción o por transformada), ya que son mucho más complejas que PCM y/o utilizan la codificación predictiva y entrópica (lo que elimina las propiedades de acceso aleatorio y escalabilidad). Por ello, un algoritmo de codificación que tenga propiedades similares a las de PCM (baja complejidad, acceso aleatorio, y escalabilidad), pero con mejor eficiencia de codificación, sería interesante . El codificador de imagen Modulo-PCM (MPCM) [13] cumple estos requisitos. Para codificar una imagen, MPCM elimina ciertos bits de cada píxel siendo éste un procesamiento muy simple. La complejidad se puede encontrar en el decodificador, donde los bits eliminados de cada píxel, se predicen usando los restantes bits del píxel y la información que el decodificador calcula por interpolación entre píxeles vecinos previamente decodificados.

En este trabajo se implementa un códec rápido basado en Modulo Pulse Code Modulation (MPCM) [13] en la FPGA de Xilinx XC7Z020-1CLG484CES. Los resultados muestran que el codificador MPCM basado en FPGA obtiene un rendimiento de hasta 409,84 MBytes/seg a altas tasas de compresión, lo que permite almacenar en una memoria no volátil 2501 frames por segundo a una resolución de $1280x1024$. Además, en este trabajo se presenta una implementación hardware del sistema de decodificación MPCM, que es capaz de reproducir un vídeo de alta definición Full HD a 204 frames por segundo.

El resto del trabajo se organiza de la siguiente manera: En la sección 2 se presenta un breve resumen del codificador Módulo-PCM. En la sección 3 se describe la arquitectura propuesta. Una evaluación detallada de la implementación de esta arquitectura se muestra en la sección 4 en términos de R/D, retardo de codificación, consumo de energía y área ocupada en el dispositivo hardware. Por último, en la sección 5 se presentan algunas conclusiones.

II. Sistema de codificación

En esta sección se describe el algoritmo MPCP para la codificación de una señal unidimensional. Consideramos $\widetilde{x}_n$ $(n \in N)$ una señal continua en amplitud y discreta en tiempo cuyos valores en amplitud se extienden entre $[A_{min}, A_{max}]$. Tomamos x_n como la señal digital resultante de la cuantización escalar uniforme de $\widetilde{x}_n$ con una tasa fija de B bits/muestra y un paso de:
$\Delta = (A_{max} - A_{min})/2^B$.

La manera más fácil para reducir el bit-rate de

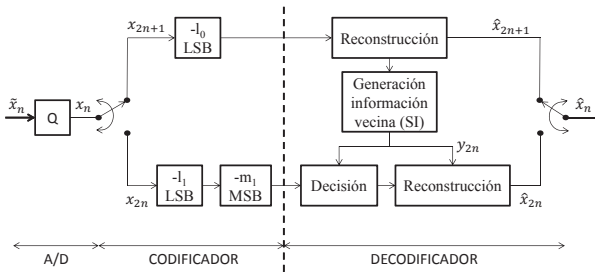

Fig. 1. Diagrama de bloques del algoritmo MPCM

$\widetilde{x}_n$ es eliminar los l-LSBs (bits menos significativos) de cada palabra-código de $\widetilde{x}_n$. Para lograr una reducción de tasa más eficiente, nos basamos en el algoritmo de codificación MPCM (Figura 1), donde las muestras de $\widetilde{x}_n$ se dividen en dos grupos: $S_0 = \{x_{2n+1} | n \in i = 0, 1, 2, \ldots\}$ y $S_1 = \{x_{2n} | n = 1, 2, \ldots\}$, los cuales se codifican con precisiones distintas. Como se muestra en la Figura 1, cada muestra de S_0 se codifica eliminando los l_0-LSBs de su palabra-código mientras que en cada muestra de S_1 se eliminan los m_1-MSBs (bits más significativos) y l_1-LSBs. Entonces, el codificador trabaja a una tasa media de $R = B - (l_0 + l_1 + m_1)/2$ bits/muestra.

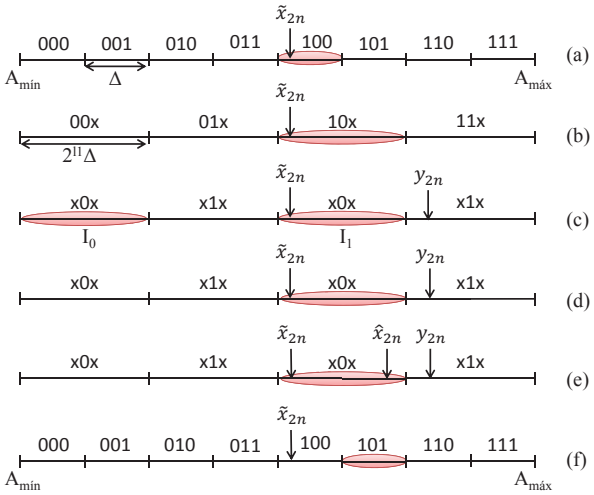

Fig. 2. Intervalos de cuantización y palabras-código para B=3, l_1=1, m_1=1: (a)Tras la conversión A/D, (b)Tras la eliminación de l_1 bits, (c)Tras la eliminación de m_1 bits, (d)Tras la decisión del intervalo I_0 o I_1, (e) tras la reconstrución y (f) tras la cuantización final. El símbolo X representa los bits eliminados. Los intervalos marcados son los representados por la palabra de código seleccionadas en cada paso para los valores mostrados de x_{2n} y y_{2n}.

Ya que la codificación de x_{2n+1} es equivalente a una cuantización uniforme con un incremento en el intervalo de cuantización igual a $2^{l_0}\Delta$, el decodificador puede directamente reconstruir las muestras de S_0 (Figura 1). En cuanto a la codificación de las muestras en S_1 (Figura 2(a)), la eliminación de los l_1-LSBs de x_{2n} es equivalente a cuantizar uniformemente su valor original con un incremento en el intervalo de cuantización igual a $2^{l_1}\Delta$ (Figura 2(b)). Tras eliminar los m_1-MSBs, la palabra código resultante identifica a un conjunto de 2^{m_1} intervalos no consecu-

tivos $\{I_i | i = 0, \ldots, 2^{m_1} - 1\}$, teniendo cada intervalo una longitud de $2^{l_1}\Delta$ (Figura 2(c)).

En el decodificador MPCM, para decidir que intervalo I_i pertenece a $\hat{x}_{2n}$, se explota la correlación entre las muestras de la señal mediante la ayuda de predicciones para cada muestra en S_1 basadas en muestras de S_0 previamente decodificadas. La predicción y_{2n} actúa como SI (Side Information) para decidir el intervalo (Figura 2(d)). La precisión del SI depende del grado de correlación entre las muestras x_n y la distorsión introducida por la codificación de S_0. Con el fin de limitar el impacto de la distorsión de codificación en la calidad del SI, a la hora de codificar l_0 debe ser menor que l_1 ($l_0 < l_1$). Si el proceso de decisión se realiza sin error para x_{2n}, sus m_1-MSBs se recuperan adecuadamente. Una vez que el decodificador ha estimado los m_1-MSBs de x_{2n}, intenta recuperar sus l_1-LSBs que finalmente proporciona la reconstrucción de $\hat{x}_{2n}$. Esta reconstrucción se realiza usando el intervalo de cuantización I_i donde supuestamente se encuentra x_{2n} y su SI (y_{2n}).

Para una descripción más detallada del algoritmo MPCM, se remite al lector a [13], [14].

III. Implementación hardware

Con el fin de hacer frente al gran ancho de banda que necesitan las cámaras de alta velocidad de hoy en día, tanto el codificador como el decodificador MPCM se han implementado en una arquitectura hardware. El lenguaje de descripción utilizado para construir el diseño es VHDL. La implementación propuesta en hardware se ha desarrollado en una FPGA Zynq-7000 de la familia Xilinx, específicamente sobre el modelo ZC702 que incluye el XC7Z020-1CLG484CES SoC (System-on-Chip) [15].

A. Arquitectura del codificador

La arquitectura implementada del codificador se ilustra en la Figura 3. La imagen original capturada por los sensores de la cámara se almacena en un bloque de memoria cuya lectura está determinada por un bloque de control. En esta estructura, 16 píxeles se leen en cada ciclo de reloj con el fin de acelerar el proceso de codificación tanto como sea posible dentro del alcance de la memoria interna del dispositivo. Esta memoria actúa como un buffer de frames interno para leerlos en aplicaciones de alta velocidad y es implementado con block RAMs. De esta manera, utilizamos 52 blocks RAMs Dual-port de 36 Kb, configurando sus puertos a $512x64$ bits, donde 64 bits de salida, es decir 8 píxeles, se leen en cada puerto de salida de la memoria. Estos píxeles son procesados en el siguiente bloque, sin retardo, en el mismo ciclo de reloj, donde son codificados eliminando los correspondientes bits l_0 o l_k y m_k. Finalmente, obtenemos las muestras codificadas de la imagen las cuales serán enviadas al dispositivo final de almacenamiento.

En el diseño propuesto, la imagen se divide en bloques de 4 muestras diferentes $x_{0,0}[n_1, n_2]$, $x_{0,1}[n_1, n_2]$, $x_{1,0}[n_1, n_2]$ y $x_{1,1}[n_1, n_2]$ de tal manera

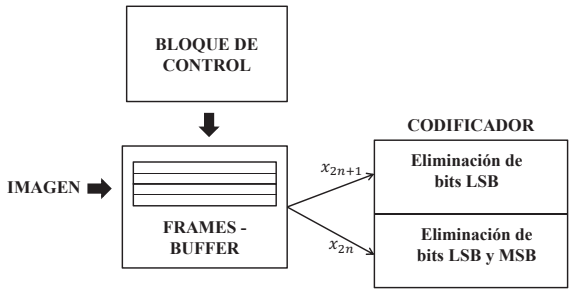

Fig. 3. Diseño de la arquitectura del codificador

que:

$$x_{p,q}[n_1, n_2] = x[2n_1 + p, 2n_2 + q]$$

con p y $q \in \{0, 1\}$. Así, la primera muestra se codifica usando PCM, eliminando unicamente los LSBs l_0, y el resto de partes utilizando MPCM, al extraer los LSBs l_k y los MSBs m_k. En la Figura 4 se muestra un diagrama con los pasos realizados en nuestro algoritmo hardware. Hay que tener en cuenta que tanto la lectura como la codificación de los 16 píxeles se lleva a cabo en el mismo ciclo de reloj.

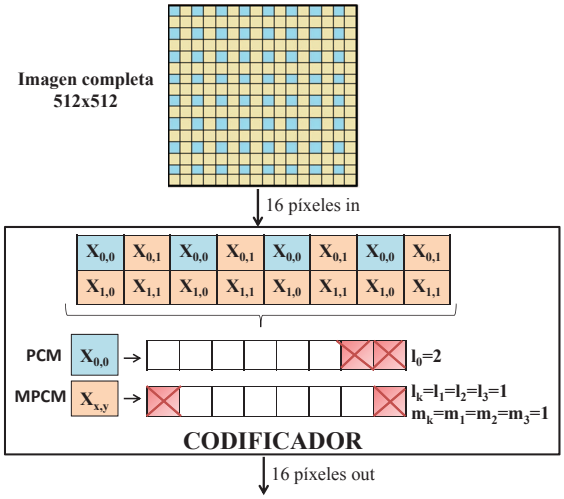

Fig. 4. Ejemplo de algoritmo de codificación con $l_0 = 2$, $l_k = 1$ y $m_k = 1$

B. Arquitectura del decodificador

En la Figura 5 se muestra la arquitectura propuesta para el decodificador. En este enfoque, el buffer intermedio se llena con muestras codificadas hasta que se completan las tres primeras líneas, entonces se inicia el proceso de decodificación. El decodificador está dividido en dos pasos: decisión y reconstrucción. La recuperación de la imagen original se realiza como sigue:

- Tres columnas de datos se procesan en el bloque de decisión, donde la señal PCM se reconstruye. Entonces se generan SIs precisos a partir de las muestras PCM y así se selecciona uno de los posibles intervalos 2^{m_k} donde cada señal decodificada MPCM se situará.

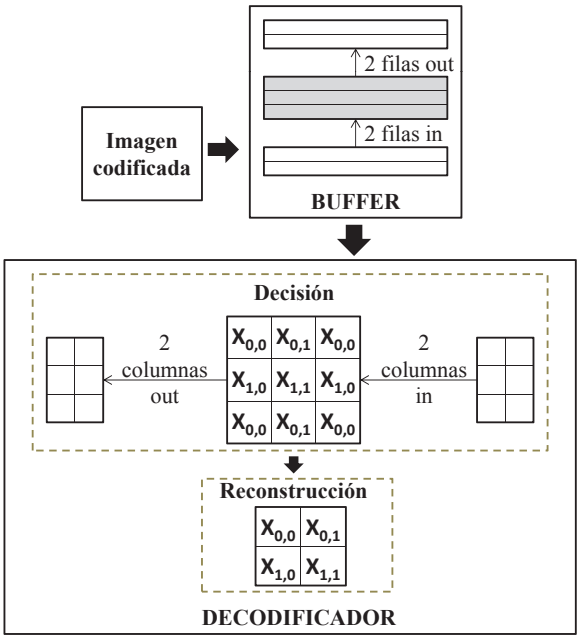

Fig. 5. Diseño de la arquitectura del decodificador

- En el bloque de reconstrucción del decodificador, se recuperan los bits menos significativos (LSB) que fueron eliminados en el proceso de codificación mediante la utilización de su SI y el intervalo correspondiente a cada muestra MPCM. Tras completar estos pasos, se obtienen cuatro píxeles decodificados.
- En cada ciclo de reloj, dos columnas de muestras codificadas salen del bloque del decodificador y dos nuevas entran en él. Este proceso continua iterativamente hasta que todas las muestras de las tres filas del buffer son decodificadas.
- Entonces, el buffer se desplaza. Dos filas salen del buffer y dos nuevas entran. Todas las operaciones anteriores continuan hasta que todas las muestras codificadas de entrada son procesadas.

El diseño propuesto del decodificador ha sido completamente segmentado, esto hace que cada una de las etapas mencionadas anteriormente para la decodificación se realicen concurrentemente. Además, como se explicó anteriormente, se obtienen 4 píxeles decodificados en cada ciclo de reloj. En esta propuesta, la frecuencia de operación se ha establecido para conseguir estos 4 píxeles, pero con el fin de organizar toda la imagen decodificada, se ha utilizado el módulo Phase-Locked-Loop (PLL), mediante el cual se generan múltiples relojes a partir de la señal de reloj de entrada determinada. Por lo tanto, en cada ciclo, se almacenan 4 píxeles en una memoria intermedia con una frecuencia fija, pero estos píxeles se leen con una frecuencia 4 veces mayor, logrando así una salida en serie sin retardo. Los buffers utilizados en esta arquitectura se han implementado usando únicamente block RAMs de 18 Kb de doble puerto.

IV. Resultados

En esta sección se presenta la evaluación del sistema completo en términos de PSNR, tiempos de codificación/decodificación, uso de área del dispositivo FPGA, máximo frame rate y mejoras en velocidad comparándolos con algoritmos secuenciales en CPUs. Las arquitecturas han sido sintetizadas, emplazadas y enrutadas utilizando la herramienta Xilinx ISE 14.3, y han sido simuladas y verificadas mediante Matlab/Simulink través de la herramienta System Generator. Estas propuestas han sido diseñadas para el dispositivo Zynq AP SoC previamente mencionado. El área ocupada del dispositivo, la frecuencia máxima y la estimación de consumo de energía se han medido desde la herramienta Xilinx ISE 14.3. En nuestros experimentos, hemos evaluado los resultados de cinco imágenes en escala de grises (Zelda, Lena, Peppers, Barbara y Baboon) con una resolución de $512x512$ píxeles y 8 bits por muestra. Por otra parte, hemos asignado los valores óptimos de los parámetros de codificación/decodificación con el fin de obtener el máximo PSNR para una tasa de bits dada. La asignación de valores de los parámetros para el codificador/decodificador MPCM se proponen en [14].

A. Evaluación del codificador

En la Tabla I se muestra el PSNR obtenido para todas las imágenes evaluadas en función del bit-rate (R). Como se esperaba, para tasas altas de (R), lo que significa eliminar pocos bits en el proceso de codificación, el algoritmo MPCM porporciona generalmente mejor PSNR debido a que no se produce una pérdida significativa en el proceso, y en consecuencia, no se introducen grandes errores en el proceso de decodificación. Por lo tanto, a menos bit-rate (R), menor valor de PSNR. Hay que tener en cuenta que, para cada tasa, los parámetros l_0, l_1, m_1 no son necesariamente los mismos para todas las imágenes. Cada imagen tiene unos parámetros apropiados para obtener la calidad óptima en la imagen recuperada.

En cuanto al retardo de codificación/decodificación, el codificador propuesto funciona a una frecuencia de reloj máxima de 204,96 MHz. Además, el algoritmo requiere 16.387 ciclos para llevar a cabo el proceso completo de codificación para una resolución de imagen de 512x512 píxeles. Por lo tanto, necesitamos 79,952 ms para codificar cualquier imagen de la mencionada resolución, siendo 12 veces más rápido que el algoritmo secuencial en una CPU Intel Core 2 a 1,8 Ghz con 5 GBytes de RAM. Ya que el proceso de codificación sólo depende de la resolución de la imagen, en la Figura 6 se muestra el máximo framerate obtenido en la arquitectura propuesta. Como se observa, la implementación hardware del codificador MPCM es capaz de comprimir hasta 3558 frames por segundo para la resolución HD-Ready y hasta 1668 frames por segundo para la resolución Full-HD.

El proceso de codificación de alta velocidad hace que las cámaras de alta velocidad sean capaces de capturar y grabar continuamente sin las restricciones

TABLA I
PARÁMETROS ÓPTIMOS Y VALORES DE PSNR PARA TODAS LAS IMÁGENES EVALUADAS

	R=1bpp		R=2bpp		R=4bpp		R=6bpp	
Imagen	(l_0,l_1,m_1)	PSNR	(l_0,l_1,m_1)	PSNR	(l_0,l_1,m_1)	PSNR	(l_0,l_1,m_1)	PSNR
Zelda	(4,8,0)	33.83	(0,8,0)	36.57	(1,3,2)	40.07	(2,1,1)	48.07
Lena	(4,8,0)	31.79	(3,6,1)	33.71	(1,4,1)	37.74	(2,1,1)	47.80
Peppers	(4,8,0)	30.57	(3,7,0)	32.14	(4,4,0)	34.88	(2,2,0)	44.62
Barbara	(4,8,0)	24.87	(3,7,0)	26.59	(4,4,0)	33.67	(2,2,0)	44.56
Baboon	(4,8,0)	22.53	(3,7,0)	24.22	(4,4,0)	32.20	(2,2,0)	43.85

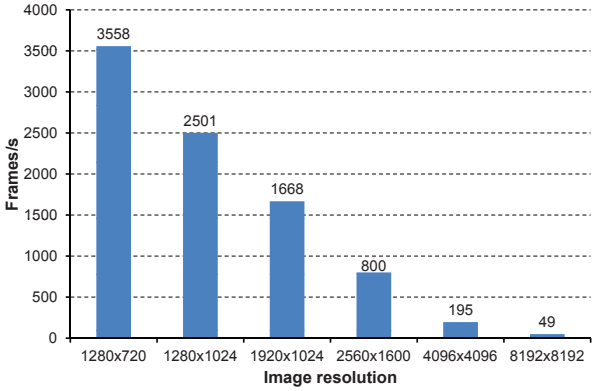

Fig. 6. Máximo frame por segundo en el codificador para diferentes resoluciones de imagen

de tamaño de la memoria RAM interna. Por ejemplo, a una tasa de compresión de 1 bpp, el sistema de codificación tiene un rendimiento de 409,84 MBytes/s, que es menor que el ancho de banda disponible en dispositivos de almacenamiento típicos como un SSD (Solid State Drive) (de hasta 600 MBytes/s). Dependiendo de la aplicación final, si es necesaria una calidad de imagen mayor, la implementación hardware MPCM propuesta podría comprimir a una tasa de 4 bpp con buena calidad y un rendimiento de ancho de banda de 1640 Mbytes/s, que extenderá el tiempo de captura en el módulo de memoria interna de la cámara hasta 1,4 veces o permitirá su transmisión a través de un enlace punto a punto Ethernet 40 Gbps.

Los elementos básicos de una FPGA son los CLBs (Bloques de Lógica Configurable). La arquitectura CLBs incluye: LUTs de 6 entradas, capacidad de memoria y de registro dentro de la LUT y la funcionalidad de registro de desplazamiento. Las LUTs en el dispositivo Zynq-7000 AP SoC pueden ser configuradas o bien como una LUT de 6 entradas (ROMs de 64-bit) con una salida, o como dos LUTs de 5 entradas (ROMs de 32-bit) con salidas separadas pero direcciones comunes o entradas lógicas. Cada salida de una LUT puede opcionalmente ser registrada en un flip-flop. Cuatro de estas LUTs y sus ocho flip-flops además de multiplicadores y lógica aritmética forman un slice, y dos slices forman un bloque lógico configurable (CLB). Cuatro de los ocho flip-flops por slice (un flip-flop por LUT) pueden configurarse como latches opcionalmente. Entre el 25-50% de todos los slices pueden también utilizar sus LUTs como RAM-distribuidas de 64-bit o como registros de desplazamiento de 32-bit. [15].

TABLA II
ÁREA USADA EN FPGA - IMPLEMENTACIÓN CODIFICADOR

	Used	**Available**	**% use**
No. of Slices	14	13300	1%
No. of Slice Registers (as Flip Flops)	17	106400	1%
No. of Slice LUTs	35	53200	1%
No. of RAMB36	52	140	37%
FMax(MHz)	204,96	-	-
Consumption (mW)	305	-	-

En la Tabla II se presentan los resultados de la implementación del codficador en términos de recursos hardware usados, indicando el número de Slices, Flip-Flops, LUTs y bloques RAMs de 36 KB utilizados. Adicionalmente, se muestra una estimación del consumo de potencia que proporciona la herramienta XPower de Xilinx ISE 14.3, obteniendo solamente un consumo de 305 mW debido a la gran segmentación realizada en el diseño del codificador. Como se puede observar, sólo un 1% de toda el área disponible en la FPGA es utilizada, por tanto dada la gran cantidad de área disponible en la FPGA, ésta podríamos usarla para implementar múltiples codificadores idénticos que podrían ejecutarse concurrentemente. De este modo, diferentes frames podrán ser codificados simultáneamente con el fin de incrementar el tiempo disponible de grabación de una cámara de alta velocidad. Para aprovechar las ventajas de ello, simplemente tendríamos que considerar el uso de una memoria externa para almacenar los diferentes frames, teniendo en cuenta el uso de los bloques de RAMs como buffers intermedios.

B. Evaluación del decodificador

En lo que se refiere al decodificador, la frecuencia máxima de reloj se ha establecido en 100MHz, siendo la menor latencia 713 ciclos. Esta frecuencia se toma como compromiso debido al uso de otra frecuencia 4 veces mayor, proporcionada por el módulo PLL, ya que analizando el retardo de la implementación, se podrían conseguir frecuencias mayores (aprox. 160 MHz), como se discutierón en la sección III-B. De esta manera, el decodificador MPCM es capaz de recuperar 400 Mpíxels por segundo a esa frecuencia. Por otro lado, el algoritmo requiere 66.240 ciclos para realizar el proceso de decodificación de imagen con

una resolución de 512x512 píxeles, por tanto se necesitan 662 μs para decodificar cualquier imagen con esta resolución, llegando a ser 70 veces más rápido que el algoritmo de decodificación secuencial en un Intel Core 2 CPU a 1.8 Ghz con 5 GBytes de RAM.

La Figura 7 muestra el máximo frame rate obtenido en la decodificación de la arquitectura propuesta. Como se muestra, la implementación hardware del decodificador MPCM es capaz de recuperar más de 434 frames por segundo para una resolución HD-Ready o más de 204 frames por segundo para una resolución Full-HD, que corresponde a un rendimiento de 50 MBytes/s, pudiendo así reproducir vídeo de alta definición a altas tasas de frames por segundo.

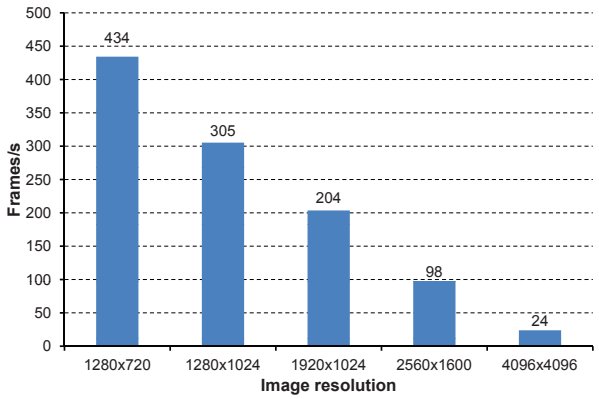

Fig. 7. Máximo frame por segundo en el decodificador para diferentes resoluciones de imagen

De forma análoga al estudio realizado en el codificador, el área utilizada en el dispositivo es menor del 1%. El área ocupada puede variar dependiendo de los parámetros l_0, l_1, m_1, pero en cualquier caso será menor que un 1%. Como se indicaba en la sección III-B, los buffers utilizados han sido modelados en bloques RAMs de 18 Kb de doble puerto con el fin de aprovechar la ventaja de un consumo menor comparado con las memorias distribuidas, además de ser más rápidas.

V. Conclusiones

En este artículo hemos presentado una implementación eficiente del codec MPCM en una FPGA. Se ha mostrado la calidad de las imágenes reconstruidas en términos de PSNR a diferentes tasas de compresión. En lo concerniente a la velocidad de codificación, los resultados muestran que nuestra propuesta de implementación es capaz de comprimir una imagen de resolución Full-HD a 1668 frames por segundo. El máximo ancho de banda alcanzado por nuestra implementación es de 409.84 MBytes/s lo que permite la grabación continua de una cámara de alta velocidad actual con una resolución de imagen HD-Ready ($1280x720p$) y una calidad razonable. Pero, si la aplicación final requiere una calidad de imagen superior, nuestro codificador es capaz de proporcionar más de 1640 MBytes/s a una tasas de compresión 2:1, duplicando el tiempo de captura sobre la memoria RAM interna de la cámara de alta velocidad. El área utilizada de la FPGA es menor al 1% del área total disponible, lo que nos da la posibilidad de replicar varios sistemas de codificación y por tanto, diferetes frames pueden ser comprimidos de manera paralela.

También hemos desarrollado en hardware el módulo del decodificador MPCM. El diseño propuesto es capaz de recuperar imágenes a 204 frames por segundo para una resolución Full-HD, con un área ocupada del dispositivo menor al 1%.

Agradecimientos

Este trabajo ha sido financiado por el Ministerio de Economía y Competitividad con el proyecto de I+D con referencia TIN2011-27543-CO3-03 y por la Generalitat Valenciana con la ayuda de referencia ACOMP/2013/03.

Referencias

[1] Vision Research PHANTOM v641, "http://www.visionresearch.com/Products/High-Speed-Cameras/v641," .

[2] PHOTRON FASTCAM SA-X, "http://www.photron.com/index.php," .

[3] N. S. Jayant and P. Noll, Prentice-Hall, Inc, 1974.

[4] P. Gemeiner, W. Ponweiser, P. Einramhof, , and M. Vincze, "Real-time slam with high-speed cmos camera," *14th International Conference on Image Analysis and Processing ICIAP*, pp. 297–302, September 2007.

[5] L.H. Chen, W.L. Liu, O.T.C. Chen, and R.L. Ma, "A reconfigurable digital signal processor architecture for high-efficiency MPEG-4 video encoding," in *IEEE Conference on Multimedia and Expo*, 2002.

[6] J. Ritter, G. Fey, and P. Molitor, "Spiht implemented in a XC4000 device," in *IEEE 45th Midwest Symposium on Circuits and Systems*, 2002.

[7] I. Urriza, J.I. Artigas, J.I. Garcia, L.A. Barragan, and D. Navarro, "Vlsi architecture for lossless compression of medical images using discrete wavelet transform," in *Conference on Design Automation and Test in Europe*, 1998.

[8] J. Ahmad and M. Ebrahim, "Fpga based implementation of baseline jpeg decoder," *International Journal of Electrical & Computer Sciences*, vol. 9, no. 9, pp. 371–377.

[9] A. Descampe, Devaux F., Rouvroy G., Macq B., and Legat J.D., *An Efficient FPGA Implementation of a Flexible JPEG2000 Decoder for Digital Cinema*, Ph.D. thesis, Université catholique de Louvain, 2002.

[10] Xi CHEN, Lin ZENG, Qinglin ZHANG, and Wenxuan SHI, "A novel parallel JPEG compression system based on FPGA," *Journal of Computational Information Systems*, vol. 7, no. 3, pp. 697–706, 2011.

[11] Yan Wang, Shoushun Chen, and A. Bermak, "Fpga implementation of image compression using dpcm and fbar," in *Integrated Circuits, 2007. ISIC '07. International Symposium on*, sept. 2007, pp. 329 –332.

[12] Dominique Martinez and Marc M. Van Hulle, "Generalized boundary adaptation rule for minimizing rth power law distortion in high resolution quantization," *Neural Networks*, vol. 8, no. 6, pp. 891 – 900, 1995.

[13] J. Prades-Nebot, A. Roca, and E. Delp, "Modulo-pcm based encoding for high speed video cameras," in *Image Processing, 2008. ICIP 2008. 15th IEEE International Conference on*, oct. 2008, pp. 153 –156.

[14] Marleen Morbee Ph.D. Thesis, "Optimized information processing in resource-constrained vision systems," 2011.

[15] Xilinx Zynq-7000, "Zynq-7000 all programmable soc overview, advance product specification - ds190 (v1.2) available on: http://www.xilinx.com/support/documentation-/data_sheets/-ds190-Zynq-7000-Overview.pdf," August 2012.