Análisis de la paralelización basada en Slices del codificador HEVC

H. Migallón[1], Pablo Piñol[1], O. López-Granado[1] y M.P. Malumbres[1]

Resumen— **El estándar en codificación de video más reciente, desarrollado y publicado por el *ITU-T Video Coding Experts Group* junto a el *ISO/IEC Moving Picture Experts Group* es el estándar HEVC. Este nuevo estándar mejora sustancialmente la eficiencia en la compresión de video. Esta mejora implica una contraprestación que se traduce en un incremento en la carga computacional de dicha codificación. En este trabajo, nuestros objetivos son, por un lado reducir el tiempo de cómputo, mediante el uso de computación paralela, del codificador HEVC, y por otro que el rendimiento de la codificación no se vea afectado. Presentamos en este trabajo una propuesta de paralelización del codificador HEVC para arquitecturas de memoria compartida, haciendo uso de OpenMP para manejar el entorno paralelo. La algoritmo paralelo desarrollado se basa en el concepto de *slice* del codificador HEVC, siendo codificado cada slice de un frame por un core del multiprocesador. En los resultados numéricos presentados se comprueba que se obtienen buenos rendimientos paralelos para los diferentes modos de codificación utilizados (*All Intra, Low-Delay B, Low-Delay P y Random Access*), con apenas pérdida en el rendimiento de la codificación.**

Palabras clave— **Algoritmos paralelos, multicore, OpenMP, codificación de video, HEVC.**

I. Introducción

El reciente estándar HEVC (*High Efficiency Video Coding) ha sido desarrollado por el Joint Collaborative Team on Video Coding (JCT-VC), grupo compuesto por el ISO/IEC Moving Picture Experts Group (MPEG) y por el ITU-T Video Coding Experts Group (VCEG). Este nuevo estándar reemplaza al, hasta ahora, estándar H.264/AVC [1] para adaptarse a las nuevas necesidades de las aplicaciones multimedia, por ejemplo, ya existen contenidos de video con definición 4K y en el futuro se necesitará una resolución de 8K. Es por ello que uno de los objetivos del estándar HEVC es la eficiencia de la codificación, lo que se traduce en que, respecto a su predecesor (H.264/AVC High profile), disminuye el bitrate a la mitad manteniendo la misma calidad de imagen [2].*

Esta mejora en la eficiencia de la codificación no es gratuita, el codificador HEVC es sensiblemente más complejo que el codificador H.264/AVC. Este aumento de la complejidad hace muy interesantes las propuestas que permitan disminuir los tiempos de codificación, aumentando para ello los recursos computacionales utilizados. En este trabajo hemos hecho uso de la versión 10.0 del software de referencia (HM) [3], que corresponde a la especificación preliminar 10 [4]. Información más detallada respecto al estándar HEVC puede verse en [5].

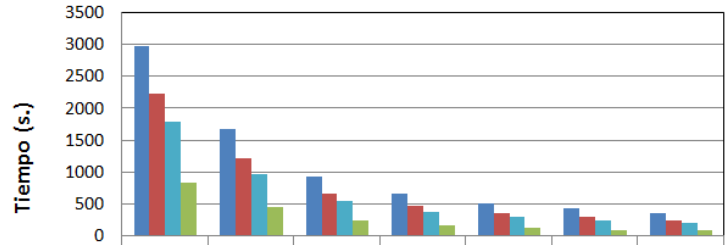
[1]Departamento de Física y Arquitectura de Computadores, Universidad Miguel Hernández, e-mail: {hmigallon,pablop,otoniel,mels}@umh.es

Existen diversos trabajos publicados que versan sobre análisis de la complejidad y estrategias paralelas del estándar HEVC, ver por ejemplo [6] [7] y [8]. No obstante, la mayoría de estos trabajos se centran en la decodificación y sólo un pequeño número se centran en la codificación. Por ejemplo, en [7] se presenta una propuesta paralela, sólo para el decodificador, que permite decodificar en tiempo real contenidos de video en High-Definition (HD) y Ultra-High-Definition (UHD). En [9] y [10] también se presenta una propuesta paralela para el decodificador. Esta propuesta, denominada Overlapped Wavefront (OWF), se basa en el Wavefront Parallel Processing (WPP) incluido en el estándar HEVC, en la cual se solapa la codificación de frames consecutivos. Más específicamente, cuando un hilo finaliza el procesamiento de un Coding Tree Block (CTB) de una fila puede continuar con el siguiente frame en lugar de esperar a que se finalice la decodificación del frame actual. En [11] se presenta otro decodificador en tiempo real, en este caso basado en *Tiles*, WPP e instrucciones SIMD.

Más recientemente se han publicado algunos artículos con propuestas para acelerar el codificador HEVC. En [12], los autores proponen una optimización de grano fino en la estimación de movimiento del codificador HEVC, esta optimización consiste en obtener la predicción del vector de movimiento para todas las *prediction units (PUs)* disponibles de un *Coding Unit (CU)* al mismo tiempo. En [13] los autores proponen una paralelización del módulo de predicción Intra, para lo cual eliminan las dependencias de datos existentes entre subbloques de un CU, y obtienen buenos resultados de speed-up. Otro grupo de trabajos recientes se centran en cambiar el orden de búsqueda. Por ejemplo, en [14], se hace uso de una búsqueda de CUs en diamante que permite un procesamiento paralelo. Además, en [15], se propone un cambio en el orden de procesamiento del *deblocking filter* del HEVC.

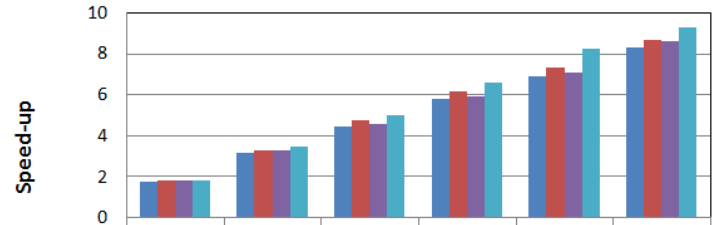
En este trabajo presentamos un esquema de paralelización del codificador del HEVC basado en slices. El número de slices en que se divide cada frame depende del número de procesos utilizados para codificar una secuencia de video. Por tanto el número de CUs consecutivos de cada slice, es decir el tamaño del slice depende del número de procesos utilizado en la codificación. Analizaremos tanto el rendimiento computacional como la eficiencia de la codificación de la propuesta paralela presentada.

El resto de este artículo está organizado de la siguiente forma: en la sección II presentamos una visión general del estándar HEVC y del concepto de



	1P	2P	4P	6P	8P	10P	12P
LowDelay-B	2964.7	1672.9	937.1	661.9	512.3	430.3	358.1
Random	2219.8	1215.4	668.7	467.8	361.0	302.5	255.8
LowDelay-P	1781.0	972.6	545.9	388.0	301.0	252.8	207.1
Intra	842.4	458.7	242.9	169.5	128.2	102.7	91.3

(a) Tiempo.



	2P	4P	6P	8P	10P	12P
LowDelay-B	1.8	3.2	4.5	5.8	6.9	8.3
Random	1.8	3.3	4.7	6.1	7.3	8.7
LowDelay-P	1.8	3.3	4.6	5.9	7.0	8.6
Intra	1.8	3.5	5.0	6.6	8.2	9.3

(b) Speed-up.

Fig. 1. Tiempo computacional y speed-ups. Secuencia FP. 120 frames. Modos AI, LB, LP y RA. QP=32.

slice. En la sección III se describe el algoritmo paralelo propuesto. En la sección IV analizamos el rendimiento, tanto en términos computacionales como en eficiencia de codificación, del algoritmo propuesto. Por último en la sección V resumimos las conclusiones.

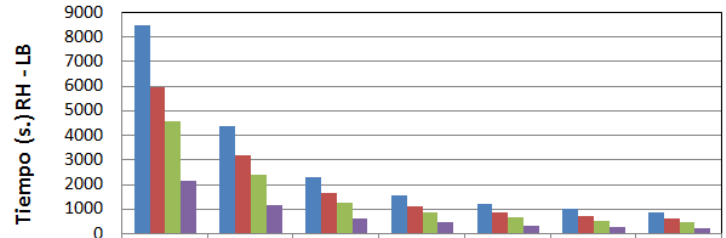
II. HEVC y slices

El estándar HEVC es un codificador de video que utiliza un esquema de codificación híbrido basado en bloques. Este esquema de codificación se basa en la estimación/compensación de movimiento y en la transformación de la codificación. De modo que en el proceso de codificación cada frame se divide en bloques denominados *coding units (CU)*. El tamaño máximo de un CU es de 64×64 píxeles. Un CU, en el proceso de codificación, puede dividirse en bloques más pequeños. Y estos bloques pueden ser codificados de tres modos diferentes: a) sin ningún tipo de predicción, b) con predicción espacial o c) con predicción temporal.

La predicción espacial intenta beneficiarse de la redundancia espacial dentro de un mismo frame. Para codificar un bloque utiliza las regiones ya codificadas del mismo frame para obtener un bloque candidato a ser el más (o muy) parecido, y de esa manera codificar sólo el error respecto al bloque seleccionado. La predicción temporal hace una búsqueda del candidato a ser parecido en los frames que ya han sido codificados, estos frames son denominados frames de referencia. La predicción temporal intenta beneficiarse del hecho de que frames consecutivos suelen tener bloques muy similares, siendo, en muchos casos, casi nulo el error de la compensación. En los tres casos se hace uso de la Transformada Discreta del Coseno (DCT) para pasar los datos obtenidos al dominio de la frecuencia. Los coeficientes obtenidos son cuantizados y comprimidos por el codificador entrópico.

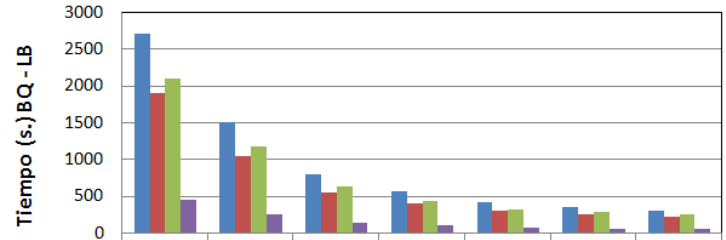
Se dice que un frame es *intra* cuando no se usa información de otros frames para codificarlo, tanto el modo a) como el modo b) cumplen esta condición. En cambio se dice que un frame es *inter* cuando sí se usa predicción temporal, es decir se usa el modo c).

Los slices son particiones de un frame que pueden ser decodificados independientemente unos de otros. Dado que un slice puede ser decodificado sin necesitar información de otros slices (del mismo frame), no puede usarse ni predicción inter ni predicción intra fuera de los límites de un slice. Por tanto cada slice puede decodificarse sin que existan dependencias con el resto de slices, ni a nivel intra ni a nivel inter. Teniendo en cuenta que un slice es un grupo de CUs, un I-slice está compuesto por CUs de tipo intra, mientras que los P-slices y los B-slices pueden contener tanto CUs de tipo intra como de tipo inter. Hay que remarcar que "P" implica una predicción unidireccional y "B" implica una predicción bidireccional. Por tanto los CUs de un P-slice utilizan un



	1P	2P	4P	6P	8P	10P	12P
LowDelay-B	8467.8	4341.9	2259.2	1553.2	1184.3	971.6	836.5
Random	5958.1	3142.4	1627.3	1108.2	843.1	690.4	603.0
LowDelay-P	4555.1	2351.7	1229.3	832.2	634.7	515.6	456.2
Intra	2114.7	1135.8	612.9	423.0	304.2	250.4	218.1

(a) Secuencia RH.



	1P	2P	4P	6P	8P	10P	12P
LowDelay-B	2710.5	1504.6	785.4	556.2	405.8	354.4	300.6
Random	1895.8	1045.4	551.3	390.1	290.5	248.1	212.0
LowDelay-P	2098.6	1167.8	619.3	435.3	319.1	279.8	239.9
Intra	438.9	239.0	123.9	91.2	64.4	56.2	46.3

(b) Secuencia BQ.

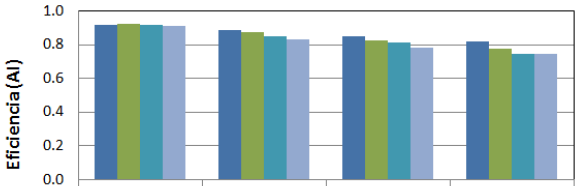
Fig. 2. Tiempo computacional. Secuencias RH y BQ. 120 frames. QP=32.

único frame de referencia, mientras que los CUs de un B-slice pueden usar hasta dos frames de referencia; es decir, el mecanismo de compensación y estimación puede usar una combinación de dos bloques de dos frames diferentes, de la lista de frames de referencia. Un frame puede estar compuesto por un único o por varios slices.

El software de referencia [3] utilizado propone cuatro modos de codificación: All Intra, Random Access, Low-Delay B, y Low-Delay P. En el modo All Intra (AI) cada frame es codificado como un I-frame, siendo todos los slices de tipo, es decir no se usa la estimación/compensación de movimiento. Siendo, por tanto, cada frame independiente del resto de frames de la secuencia. Este modo alcanza menores tasas de compresión que el resto de modos, dado que los frames de tipo B o P suelen obtener mejores tasas de compresión respecto a los frames de tipo I, manteniendo el mismo nivel de calidad. En cambio el proceso de codificación de un I-frame es mucho más rápido comparado tanto con B-frames como con P-frames, debido a que no se realiza la estimación de movimiento. En aquellos casos en los que se requiera velocidad de procesamiento y que el ancho de banda o la capacidad de almacenamiento no sea un problema debe utilizarse el modo AI.

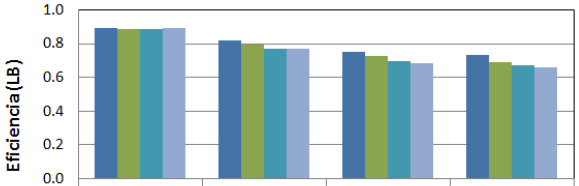
El modo Random Access (RA) combina I-frames y B-frames, agrupándolos en GOPs (Group Of Pictures) de 8 frames. Habitualmente los B-frames consiguen mejores tasas de comprensión. Teniendo en cuenta que cada B-frame usa hasta dos frames de referencia, en el proceso de codificación hay que disponer de dos listas de frames de referencia. Los frames de referencia pueden ser tanto frames pasados como frames futuros, no coincidiendo, por tanto, el orden de codificación y decodificación con el orden de la secuencia de video. Por este motivo y para poder moverse por la secuencia o disponer de funciones como el avance rápido, es necesario insertar un I-frame periódicamente. El periodo entre dos I-frames depende del número de frames por segundo de la secuencia, ya que se inserta un I-frame cada segundo aproximadamente, pero cumpliendo que este intervalo sea múltiplo del tamaño de GOP, 8 en este caso. En aquellas aplicaciones en las que además de poder moverse por la secuencia o disponer de funciones como el avance rápido, el tiempo de codificación no sea un gran problema debe utilizarse este modo de codificación.

Los modos LP y LB codifican los frames en el mismo orden que el de la secuencia de video en grupos de 4 frames. En estos modos hay un I-frame inicial y el resto de frames se codifican como frames tipo P o tipo B. Como se ha comentado los frames de tipo B pueden usar hasta dos frames de referencia y los de tipo P sólo usan uno, pero en ambos casos los frames utilizados son frames pasados. En el caso del modo RA se usan tanto frames pasados como futuros, lo



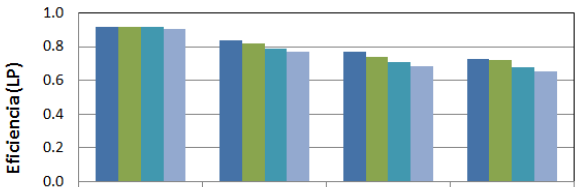
	2P	4P	8P	12P
QP=37	0.92	0.88	0.85	0.82
QP=32	0.92	0.87	0.82	0.77
QP=27	0.92	0.85	0.81	0.74
QP=22	0.91	0.83	0.78	0.74

(a) Modo All Intra.



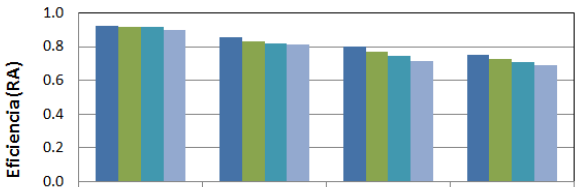
	2P	4P	8P	12P
QP=37	0.89	0.81	0.75	0.73
QP=32	0.89	0.79	0.72	0.69
QP=27	0.88	0.77	0.69	0.67
QP=22	0.89	0.77	0.69	0.66

(b) Modo Low-Delay B.



	2P	4P	8P	12P
QP=37	0.91	0.83	0.77	0.72
QP=32	0.92	0.82	0.74	0.72
QP=27	0.91	0.79	0.71	0.68
QP=22	0.90	0.77	0.68	0.65

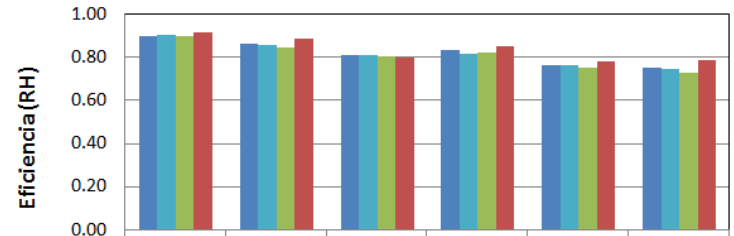
(c) Modo Low-Delay P.



	2P	4P	8P	12P
QP=37	0.92	0.85	0.80	0.75
QP=32	0.91	0.83	0.77	0.72
QP=27	0.91	0.81	0.74	0.71
QP=22	0.90	0.81	0.71	0.69

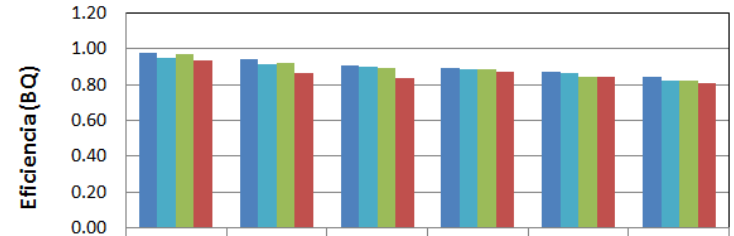
(d) Modo Random Access.

Fig. 3. Eficiencia. Secuencia FP. 120 frames. Modos AI, LB, LP y RA.



	2P	4P	6P	8P	10P	12P
LowDelay-B	0.90	0.86	0.81	0.83	0.76	0.75
Random	0.91	0.86	0.81	0.82	0.76	0.75
LowDelay-P	0.90	0.85	0.80	0.82	0.75	0.73
Intra	0.92	0.89	0.80	0.85	0.78	0.79

(a) Secuencia RH.



	2P	4P	6P	8P	10P	12P
LowDelay-B	0.98	0.94	0.91	0.89	0.87	0.84
Random	0.95	0.92	0.90	0.88	0.86	0.82
LowDelay-P	0.96	0.92	0.89	0.88	0.84	0.82
Intra	0.93	0.86	0.83	0.87	0.84	0.81

(b) Secuencia BQ.

Fig. 4. Eficiencia. Secuencias RH y BQ. 120 frames. QP=32.

que introduce un retraso dado que es necesario esperar a decodificar futuros frames para poder codificar o decodificar el frame actual. Ambos modos obtienen mejores tasas de compresión que el modo AI y

no sufren el retraso comentado introducido en modo RA. Estos modos deben usarse en aplicaciones tipo videoconferencia, en las cuales se alcanza un compromiso entre el ancho de banda necesario y los tiempos de cómputo.

III. Algoritmos paralelos

Nuestra propuesta de paralelización del codificador HEVC se basa en dividir cada frame en tantos slices como procesos paralelos se vaya a utilizar. Por tanto si se utilizan 10 procesos se dividiría cada frame en 10 slices, siendo codificado cada uno de estos slices por un proceso diferente. El tamaño de los slices se calcula automáticamente asignando a cada slice un número de CUs igual o parecido para intentar mantener equilibrada la carga computacional. En los modos LP, LB y RA un frame que sea utilizado como referencia debe estar disponible por completo para que los siguientes frames a codificar lo utilicen en su proceso de estimación/compesación de movimiento, lo que implica un proceso de sincronización. Este proceso de sincronización se produce cuando todos los procesos han finalizado la codificación del slice que tenían asignado, antes, por tanto, de comenzar la codificación de un nuevo frame. En el modo AI este proceso de sincronización no es estrictamente necesario, dado que no se hace uso de frames de referencia. No obstante se ha hecho uso del mismo esquema con el fin de permitir la escritura ordenada en el bitstream. En los modos LP, LB y RA los frames de referencia son almacenados en una estructura denominada *Decoded Picture Buffer*, esta estructura es compartida por todos los procesos haciendo uso de la memoria compartida del multiprocesador.

En nuestros experimentos hemos utilizado tres secuencias de video: "Four People (FP)"(con una resolución de 1280x720 píxeles), "Race Horses (RH)"(con una resolución de 832x480 píxeles) y "BQ Terrace (BQ)"(con una resolución de 1920x1280 píxeles). Hemos analizado el algoritmo paralelo utilizando 1, 2, 4, 6, 8, 10 y 12 procesos, dividiendo por tanto cada frame en 1, 2, 4, 6, 8, 10 y 12 slices respectivamente. Hemos utilizado secuencias de video en las cuales el número de CUs por slice sea similar en todos los casos. No obstante esto no es un requisito, en ningún caso, del algoritmo desarrollado. El objetivo de seleccionar tamaños similares de slices para cada uno de los procesos es evitar que los procesos queden ociosos tras finalizar la codificación de su slice hasta que se produzca el proceso de sincronización para comenzar a codificar un nuevo frame. No obstante sólo escogiendo tamaños de slice similares, y en algunos casos iguales, no se puede asegurar que la carga computacional asignada a cada slice sea la misma. Este comportamiento es debido a que procesos como la estimación de movimiento o la codificación entrópica, pueden necesitar mayor o menor tiempo computacional en función de las características de la región del frame que se está codificando.

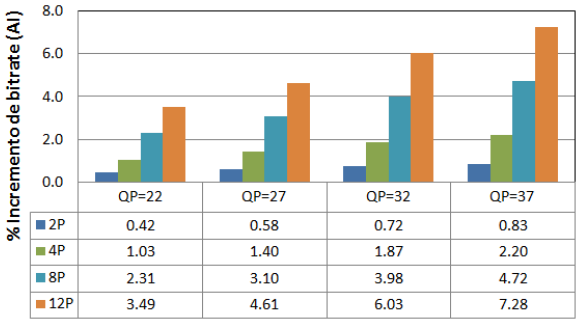
IV. Resultados numéricos

Se ha analizado el rendimiento de los algoritmos propuestos en un multiprocesador de memoria compartida, analizando tanto el redimiento paralelo como la eficiencia de la codificación en términos de PSNR (Peak Signal Noise Rate) y *bitrate*. El multiprocesador utilizado está compuesto por dos procesadores hexacore Intel XEON X5660 a 2.8 GHz, con 12MB de memoria caché por procesador, y 48 GB de memoria RAM. El sistema operativo de dicho multiprocesador es CentOS Linux 5.6 para sistemas x86 de 64 bits. Se ha hecho uso de OpenMP [16] para manejar el entorno paralelo. El compilador utilizado ha sido el compilador $g++$ v,4,1,2.

Como se ha comentado, se han utilizado tres secuencias de video (RH, FP y BQ) para obtener los resultados presentados. Se han codificado 120 frames de cada una de estas secuencias usando los modos de codificación LP, LB, RA y AI, y utilizando diferentes valores del parámetro de cuatización (QP), en particular se han usado los valores 22, 27, 32 y 37.

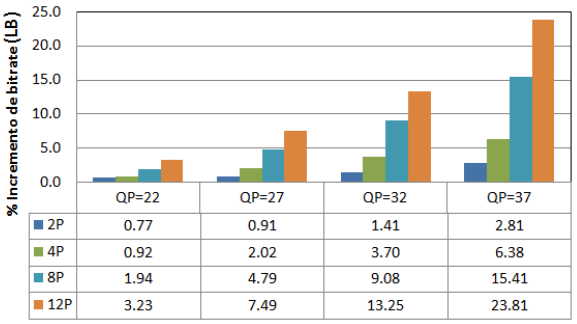
En la figura 1 presentamos los tiempos de cómputo y sus correspondientes speed-ups, para la codificación de 120 frames de la secuencia FP usando los cuatro modos de codificación. En dicha figura podemos observar que para el modo AI se obtiene un speed-up igual a 9.3 utilizando 12 cores, siendo de media las eficiencias obtenidas del 80 %. Se puede observar que el rendimiento paralelo obtenido con el modo AI es mejor que el obtenido con el resto de los modos (LP, LB y RA). Este comportamiento se debe al tiempo de procesamiento de la estimación/compensación de movimiento, el cual puede diferir de unos slices a otros, provocando que algunos cores permanezcan inactivos disminuyendo la eficiencia. En la figura 2 se presentan los tiempos de cómputo para las secuencias de video RH y BQ con el valor de QP igual a 32. Puede comprobarse en las figuras 1 y 2 que el comportamiento computacional es similar para las tres secuencias de video, pudiendo afirmar que también se mantiene el mismo comportamiento para el resto de valores de QP.

La figura 3 muestra la evolución de la eficiencia en función tanto del número de procesos como de la tasa de compresión. En todos los casos se obtienen buenas eficiencias, evidenciándose una ligera pérdida en la eficiencia a medida que el número de procesos utilizado aumenta, lo que implica que el número de slices aumente y el tamaño de dichos slices disminuya. Además esta pérdida de eficiencia al aumentar el número de procesos disminuye para tasas de compresión altas, para las cuales el tiempo de cómputo asociado a cada slice aumenta. Los valores medios de eficiencias obtenidos son 0.75, 0.78, 0.79 y 0.83 para los modos LB, LP, RA y AI respectivamente. En la figura 4 se puede comprobar que los resultados obtenidos para las secuencias RH y BQ, con valor de QP igual a 32, son similares a los obtenidos para la secuencia FP. Es importante remarcar que los mejores resultados se obtienen para la secuencia de mayor resolución (BQ).



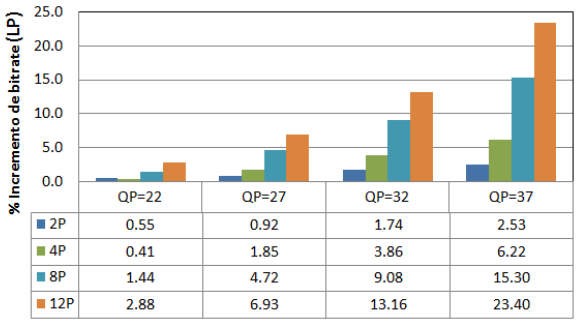
	QP=22	QP=27	QP=32	QP=37
2P	0.42	0.58	0.72	0.83
4P	1.03	1.40	1.87	2.20
8P	2.31	3.10	3.98	4.72
12P	3.49	4.61	6.03	7.28

(a) Modo All Intra.



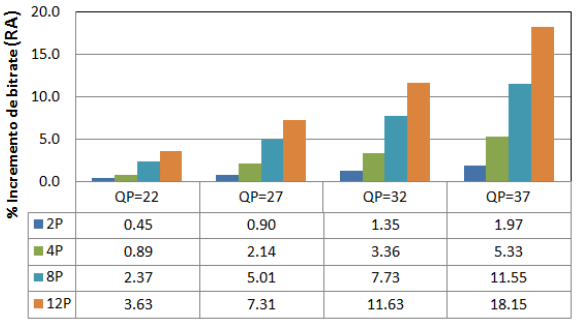
	QP=22	QP=27	QP=32	QP=37
2P	0.77	0.91	1.41	2.81
4P	0.92	2.02	3.70	6.38
8P	1.94	4.79	9.08	15.41
12P	3.23	7.49	13.25	23.81

(b) Modo Low-Delay B.



	QP=22	QP=27	QP=32	QP=37
2P	0.55	0.92	1.74	2.53
4P	0.41	1.85	3.86	6.22
8P	1.44	4.72	9.08	15.30
12P	2.88	6.93	13.16	23.40

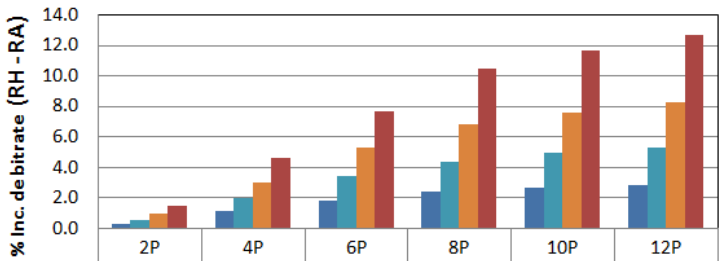
(c) Modo Low-Delay P.



	QP=22	QP=27	QP=32	QP=37
2P	0.45	0.90	1.35	1.97
4P	0.89	2.14	3.36	5.33
8P	2.37	5.01	7.73	11.55
12P	3.63	7.31	11.63	18.15

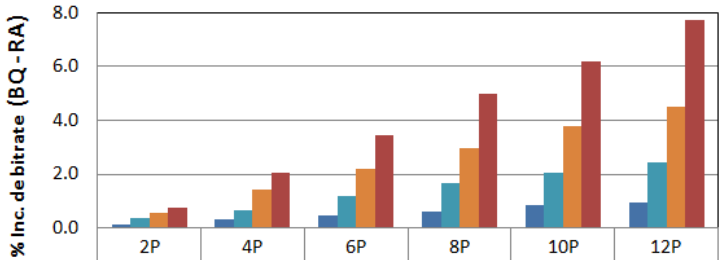
(d) Modo Random Access.

Fig. 5. Evolución del incremento de bitrate (%). Secuencia FP. 120 frames.



	2P	4P	6P	8P	10P	12P
QP=22	0.37	1.20	1.88	2.42	2.73	2.83
QP=27	0.62	2.07	3.44	4.41	4.95	5.30
QP=32	0.99	3.07	5.28	6.80	7.62	8.24
QP=37	1.51	4.68	7.69	10.47	11.62	12.66

(a) Secuencia RH.

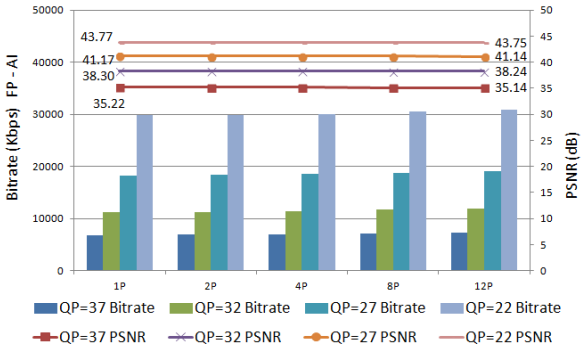


	2P	4P	6P	8P	10P	12P
QP=22	0.12	0.32	0.49	0.61	0.85	0.95
QP=27	0.35	0.66	1.21	1.65	2.06	2.43
QP=32	0.58	1.45	2.20	2.96	3.77	4.50
QP=37	0.74	2.07	3.45	4.97	6.18	7.71

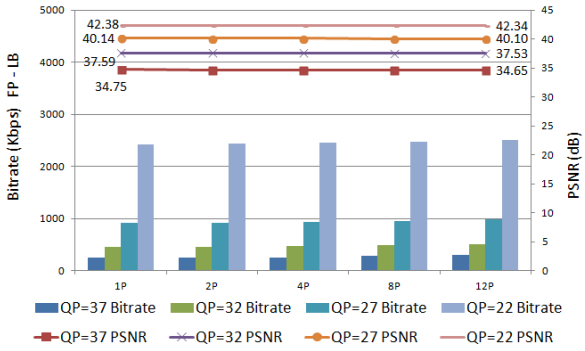
(b) Secuencia BQ.

Fig. 6. Evolución del incremento de bitrate (%). Secuencias RH y BQ. Modo RA. 120 frames.

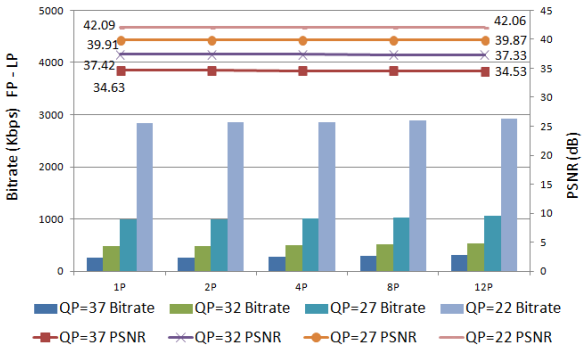
En la figura 5 se muestra el incremento de bitrate en función tanto del número de procesos como del valor de QP. Para todos los modos el bitrate aumenta tanto al aumentar el número de procesos (lo que



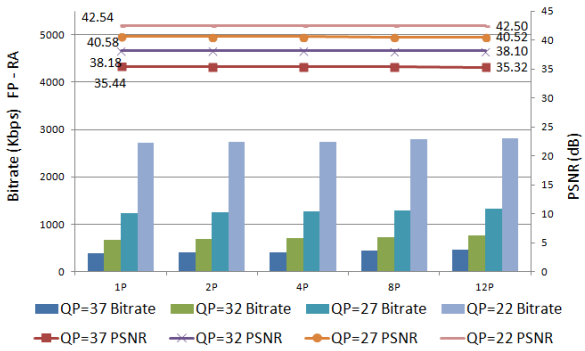
(a) Modo All Intra.



(b) Modo Low-Delay B.

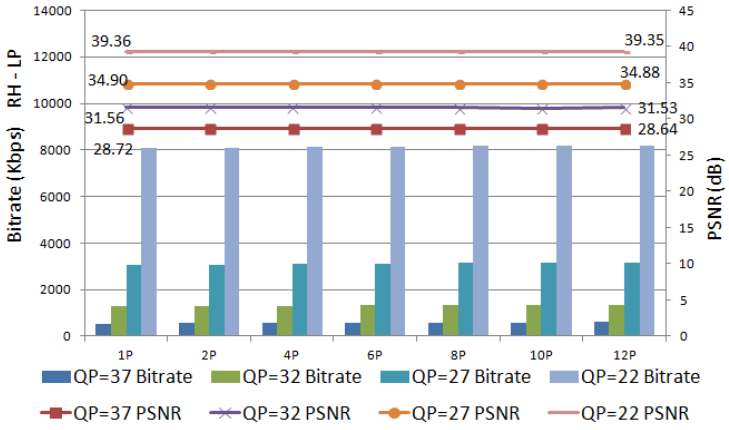


(c) Modo Low-Delay P.

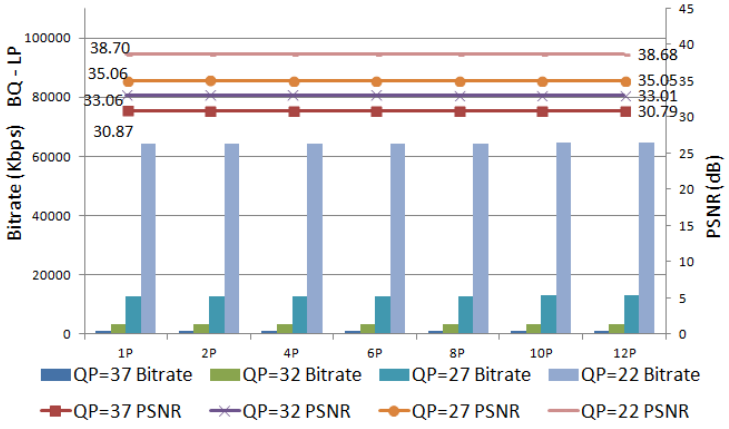


(d) Modo Random Access.

Fig. 7. Comportamiento R/D. Secuencia FP. 120 frames.



(a) Secuencia RH.



(b) Secuencia BQ.

Fig. 8. Comportamiento R/D. Secuencias RH y BQ. Modo LP. 120 frames.

implica un aumento del número de slices) como al aumentar la tasa de compresión. Este comportamiento, para el modo AI, se debe por un lado al aumento de las cabeceras incluidas en el bitstream y por otro a la disminución de CUs disponibles para realizar la predicción intra, ya que esta predicción se realiza sobre los CUs del propio slice. El máximo aumento del bitrate mostrado en la figura 5(a) para el modo AI usando 12 procesos es igual al 7.28 %, siendo el valor de QP igual a 37. De media el incremento es inferior al 3 %. En el resto de modos el comportamiento es similar al comportamiento reseñado.

El incremento de bitrate en los modos LP, LB y RA se debe tanto al incremento de cabeceras que deben ser incluidas en el bitstream, como a que en la predicción de los vectores de movimiento sólo se utiliza la región del slice, y no el frame completo. En porcentaje, el incremento de bitrate es mayor en los modos LB, LP, y RA que en el modo AI. En la figura 5 el máximo incremento de bitrate mostrado es 23.81 %, 23.40 % y 18.15 % para los modos LB, LP y RA respectivamente. Los incrementos de bitrate mostrados en la figura 6 para las secuencias RH y BQ son menores que los obtenidos para la secuencia FP.

Atendiendo a la calidad, en la figura 7 se muestran resultados de R/D (Rate/Distorsion) para todos los modos. En estas figuras se representa el bitrate (barras verticales) y el PSNR (líneas horizontales) utilizando hasta 12 procesos para diferentes valores de QP. Puede observarse que la diferencia que existe entre los algoritmos paralelos y el algoritmo secuencial es pequeña, tanto a nivel de PSNR como a nivel de bitrate. Hay que remarcar que el algoritmo secuencial se caracteriza por tener un único slice. En los resultados presentados el mayor incremento de bitrate es igual a un incremento del 23.8 %, para tasas altas de compresión (QP=37) y 12 procesos. Tal y como se ha comentado, al aumentar el número de slices se incrementa el número de cabeceras, y este incremento tiene mayor impacto cuando se utilizan tasas de compresión altas. Los resultados de R/D mostrados en la figura 8 correspondientes a las secuencias RH y BQ mantienen el mismo comportamiento que el correspondiente a la secuencia FP.

V. Conclusiones

En este artículo se ha propuesto un algoritmo paralelo para el codificador de video HEVC. Dicho algoritmo está basado en el concepto de slice, dividiendo cada frame en tantos slices como procesos a utilizar. Dicha propuesta se ha implementado sobre la versión 10.0 del software de referencia del HEVC, y se han utilizado los cuatro modos propuestos (All Intra, Low-Delay B, Low-Delay P y Random Access) por dicho software de referencia para obtener los resultados computacionales. Se han obtenido speed-ups de hasta 9,8 para el modo AI y de hasta 8,8 para el resto de modos. Para todos los modos se incrementa el bitrate al aumentar el número de procesos (que implica el aumento del número de slices), de media este incremento es del 6 % para los modos LP, LB y RA y de sólo el 3 % para el modo AI. La pérdida de calidad es irrelevante en todos los experimentos realizados. Atendiendo a los resultados obtenidos el algoritmo propuesto es un buen candidato para reducir el tiempo de codificación del estándar HEVC utilizando plataformas de memoria compartida

Agradecimientos

El presente trabajo ha sido parcialmente financiado mediante el proyecto TIN2014-53522-REDT financiado con fondos FEDER (CAPAP-H5 network) y por los proyectos CICYT TIN2011-27543-C03-03 y TIN2011-26254.

Referencias

[1] ITU-T and ISO/IEC JTC 1, "Advanced video coding for generic audiovisual services," *ITU-T Rec. H.264 and ISO/IEC 14496-10 (AVC) version 16, 2012*.

[2] J. Ohm, G.J. Sullivan, H. Schwarz, Thiow Keng Tan, and T. Wiegand, "Comparison of the coding efficiency of video coding standards - including high efficiency video coding (HEVC)," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 22, no. 12, pp. 1669–1684, 2012.

[3] HEVC Reference Software, https://hevc.hhi.fraunhofer.de/svn/svn_HEVCSoftware/tags/HM-10.0/, ," .

[4] B. Bross, W.J. Han, J.R. Ohm, G.J. Sullivan, Y-K Wang, and T. Wiegand, "High efficiency video coding (HEVC) text specification draft 10," *Document JCTVC-L1003 of JCT-VC, Geneva*, January 2013.

[5] G.J. Sullivan, J.R. Ohm, W.J. Han, and T. Wiegand, "Overview of the high efficiency video coding (HEVC) standard," *Circuits and systems for Video Technology, IEEE Transactions on*, vol. 22, no. 12, pp. 1648 –1667, December 2012.

[6] F. Bossen, B. Bross, K. Suhring, and D. Flynn, "HEVC complexity and implementation analysis," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 22, no. 12, pp. 1685–1696, 2012.

[7] M. Alvarez-Mesa, C.C. Chi, B. Juurlink, V. George, and T. Schierl, "Parallel video decoding in the emerging HEVC standard," in *International Conference on Acoustics, Speech, and Signal Processing, Kyoto*, March 2012, pp. 1–17.

[8] E.A. Ayele and S.B.Dhok, "Review of proposed high efficiency video coding (HEVC) standard," *International Journal of Computer Applications*, vol. 59, no. 15, pp. 1–9, 2012.

[9] Chi Ching Chi, M. Alvarez-Mesa, B. Juurlink, G. Clare, F. Henry, S. Pateux, and T. Schierl, "Parallel scalability and efficiency of HEVC parallelization approaches," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 22, no. 12, pp. 1827–1838, Dec 2012.

[10] ChiChing Chi, Mauricio Alvarez-Mesa, Jan Lucas, Ben Juurlink, and Thomas Schierl, "Parallel HEVC decoding on multi- and many-core architectures," *Journal of Signal Processing Systems*, vol. 71, no. 3, pp. 247–260, 2013.

[11] Benjamin Bross, Mauricio Alvarez-Mesa, Valeri George, Chi Ching Chi, Tobias Mayer, Ben Juurlink, and Thomas Schierl, "HEVC real-time decoding," 2013, vol. 8856, pp. 88561R–88561R–11.

[12] Qin Yu, Liang Zhao, and Siwei Ma, "Parallel AMVP candidate list construction for HEVC," in *VCIP'12*, 2012, pp. 1–6.

[13] Jie Jiang, Baolong Guo, Wei Mo, and Kefeng Fan, "Block-based parallel intra prediction scheme for HEVC," *Journal of Multimedia*, vol. 7, no. 4, pp. 289 –294, August 2012.

[14] Adam Luczak, Damian Karwowski, Slawomir Mackowiak, and Tomasz Grajek, "Diamond scanning order of image blocks for massively parallel HEVC compression," in *Computer Vision and Graphics*, Leonard Bolc, Ryszard Tadeusiewicz, LeszekJ. Chmielewski, and Konrad Wojciechowski, Eds. 2012, vol. 7594 of *Lecture Notes in Computer Science*, pp. 172–179, Springer Berlin Heidelberg.

[15] Chenggang Yan, Yongdong Zhang, Feng Dai, and Liang Li, "Efficient parallel framework for HEVC deblocking filter on many-core platform," in *Data Compression Conference (DCC), 2013*, March 2013, pp. 530–530.

[16] "OpenMP application program interface, version 3.1," *OpenMP Architecture Review Board.* `http://www.openmp.org`, 2011.