Análisis de estrategias paralelas basadas en GOP sobre el nuevo estandar de vídeo HEVC

Otoniel López-Granado, Manuel P. Malumbres, Hector Migallón [1] y Pablo Piñol

Resumen— **HEVC es el nuevo estandar de codificación de video, siendo capaz por termino medio de reducir a la mitad el tamaño del bitstream con respecto al anterior estandar H.264/AVC obteniendo la misma calidad.**

Nuestro interés se centra en aplicar técnicas de procesamiento paralelo al codificador HEVC para reducir significativamente las demandas computacionales sin alterar el rendimiento del mismo. Para ello, proponemos varias alternativas de paralelización del codificador HEVC específicas para arquitecturas multicore. Nuestras propuestas están basadas en el uso de OpenMP a un nivel grueso de paralelización llamado 'GOP-based'. Las aproximaciones basadas en GOP codifican simultáneamente varios grupos de frames consecutivos. Dependiendo de cómo estos GOPs (Grupo de imágenes) se agrupan y distribuyen puede ser crítico obtener buenos rendimientos.

Palabras clave— **algoritmos paralelos, Codificación de vídeo, HEVC, multicore, rendimiento**

I. Introducción

EL nuevo standar de codificación de video 'High Efficiency Video Coding (HEVC)' ha sido desarrollado por el 'Joint Collaborative Team on Video Coding (JCT-VC)' que fue establecido por 'ISO/IEC Moving Picture Experts Group (MPEG)' y 'ITU-T Video Coding Experts Group (VCEG)'. Este nuevo estandar reemplazará al actual estandar H.264/AVC [1] para dar respuesta a las actuales y futuras demandas del mercado Multimedia. La resolución 4K de video ya etá en el mercado en la actualdad y no tardará mucho en ser una realidad cotidiana una resolución de 8K. Por otra parte, el nuevo estandar da soporte a una mayor profundidad de color (10 bits). HEVC pretende doblar la eficiencia en codificación con respecto a H.264/AVC mostrando una calidad visual similar con la mitad de bitrate.

En cuanto a complejidad se refiere, el decodificador HEVC no incrementará significativamente su complejidad con respecto a H.264/AVC [2]. Sin embargo, el codificador es varias veces más complejo que H.264/AVC lo que hará que este tema sea de interés científico en los proximos años. La versión actual del software de referencia de este nuevo estandar de vídeo, llamado 'HEVC test model (HM)', es HM 10.0 que se corresponde con el borrador de la especificación del estandar HEVC 10 [3]. En [4] se puede encontrar una buena descripción de las características de este nuevo estandar.

En la literatura podemos encontrar algunos trabajos relacionados con el análisis de complejidad y estrategias de paralelización de este nuevo estandar, como en [5] [6] y [7]. Muchas de las propuestas de paralelización existentes sobre el HEVC están centradas en el decodificador, y tratan de buscar las mejores optimizaciones para decodificar en tiempo real vídeo de alta definición (HD) y Ultra alta definición (UHD).

Actualmente, hay muy pocos trabajos relacionados con la paralelización del codificador HEVC. En [8] los autores proponen una paralelización de grano fino en el módulo de estimación de movimiento, permitiendo realizar la predición de movimiento en todos las unidades de predicción (PUs) disponibles en la unidad de codificación (CU) al mismo tiempo. En [9] los autores proponen una paralelización dentro del modulo de predicción Intra que consiste en eliminar la dependencia de los datos entre subbloques de una CU, obteniendo buenos resultados en términos de speed-up.

En este artículo analizaremos las estrategias de paralelización disponibles en el estandar y su viabilidad de desarrollo en el software de referencia HM. Además presentamos varias alternativas de paralelización a nivel de GOP para el codificador HEVC especialmente diseñadas para los perfiles Low-delay.

El resto del artículo se organiza de la siguiente manera: En la sección II, se muestra un resumen de los diferentes perfiles disponibles en el HEVC y los test de las condiciones comunes de evaluación. La sección III presenta las diferentes estrategias de paralelización de alto nivel propuestas en el estandar HEVC. La sección IV presenta nuestras propuestas de paralelización a nivel de GOP para los perfiles de aplicación Low-delay, mientras que en la sección V se presenta una comparativa de las diferentes estrategias propuestas. Finalmente, conclusiones y trabajo futuro se muestran en la sección VI.

II. Perfiles HEVC

En [10] el JCT-VC define las condiciones comunes de test y las configuraciones del software de referencia que se deben usar para realizar los experimentos con el HEVC, con el fin de poder comparar las diferentes aportaciones realizadas a éste.

Se definen un total de 24 secuencias de vídeo catalogadas en 6 clases. También se definen los parámetros de cuantización (QP) y los ficheros de configuración para el codificador. El hecho de usar estas condiciones comunes permite realizar fácilmente comparativas entre las diferentes propuestas que hacen los investigadores. También el JCT-VC indica cómo calcular las curvas 'Rate-Distortion (RD)' y el % de ganancia en bitrate, usando la medida Bjontegaard-Delta (BD) [11].

Las categorías desde A hasta E incluyen secuencias de vídeo natural a diferentes resoluciones y frame-

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

rate, mientras que la categoría F contiene vídeo sintético.

Cuatro valores de QP se indican a la hora de realizar las comparativas: 22, 27, 32, 37. Para cada punto se obtiene el bitrate y el Peak Signal-to-Noise Ratio (PSNR), pudiendo realizarse las curvas de Rate-Distortion y calcular el BD.

Con el software de referencia [12] se distribuyen también los ficheros de configuración. Hay 8 test diferentes que son una combinación de 2 profundidades de color: Main (8 bits) y Main10 (10 bits) con 4 modos de codificación: All Intra (AI), Random Access (RA), Low-Delay B (LB), y Low-Delay P (LP).

En el modo All Intra, cada imágen se codifica como un I-frame, es decir, se codifica sin estimación/compensación de movimiento. De esta forma, cada frame es independiente de los demás en la secuencia de vídeo. Este modo proporciona menores niveles de compresión que los otros 3, porque el uso de P-frames y B-frames obtienes mayores niveles de compresión que los I-frames para el mismo nivel de calidad. Por otra parte, el proceso de codificación para el modo All Intra es más rápido que en los otros 3 porque no se pierde tiempo en realizar estimación de movimiento. Este modo es perfecto para aquellas aplicaciones que requieren de una codiifcación rápida y no están limitadas en el ancho de banda o el almacenaje.

El modo Random Access combina I-frames y B-frames. Un B-frame es un frame que usa estimación/compensación de movimeinto para obtener mejores niveles de compresión. Cada bloque de un B-frame puede usar hasta 2 frames de referencia, así en el proceso de codificación se mantienen 2 listas de frames de referencia. El tamaño del GOP GOP (Group Of Pictures / Grupo de imágenes) es de 8. Los frames de referencia se pueden localizar antes o después del frame que se está codificando actualmente. De esta manera, en este modo, el orden de codificación no es el mismo que el orden de visualización. Con el fin de permitir el acceso rápido a la secuencia (fast forward), se inserta un I-frame periódicamente. Dependiendo del frame-rate de la secuencia, este periodo varía, siendo siempre múltiplo de 8 (el tamaño del GOP).

Los modos Low-Delay (LP and LB) codifican cada frame en orden de visualización. Primero se inserta un I-frame y después sólo se usan P-frames (o B-frames) para el resto de la secuencia. El tamaño de GOP es 4, y todos los frames de referencia son previos al que se está codificando actualmente. Estos dos modos obtienen mayores tasas de compresión que el modo AI y evitan el retraso que introduce el modo RA. Este modo se usa para aplicaciones como vídeo conferencia que tienen restricciones de ancho de banda y tiempo.

III. Estrategias de paralelización de alto nivel en el estandar HEVC

Las estrategias de paralelismo de alto nivel se pueden clasificar siguiendo un esquema jerárquico. Esta clasificación se debe aplicar con cautela teniendo en cuenta los recursos hardware disponibles con el fin de realizar una implementación adecuada y eficiente. De esta forma, definimos los siguientes niveles desde el mas grueso hasta el más fino: GOP, tile, slice, y wavefront. Cuando se diseña una versión paralela del HEVC, primeramente analizamos el hardware disponible para determinar qué nivel es el más adecuado.

El nivel más grueso de paralelización, GOP-based, se basa en romper la secuencia de vídeo en GOPs de manera que cada GOP se procesa de forma independiente a los demás. En general, esta aproximación es la que mejor eficiencia de paralelización reporta. Sin embargo, la eficiencia en codificación puede verse afectada por cómo definimos estos GOPs y cómo eliminamos la interdependencia entre ellos.

Los Tiles se usan para dividir una imágen horizontalmente o verticalmente en varias sub-imagenes. Al usar Tiles las dependencias en la predicción se rompen en los bordes entre Tiles. Tiles consecutivos se representan en orden 'raster scan'. El orden de los Coding Tree Blocks (CTBs) se mantiene también en 'raster scan'.

Los Slices siguen el mismo concepto que en anterior estandar H.264/AVC permitiendo que una imágen se parta en grupos consecutivos de Coding Tree Units (CTUs). Hay una rotura en las dependencias en la predicción en los bordes de los slices que causa una pérdida de eficiencia en la codificación. El uso de slices está más justificado en temas de 'error resilience', más que en técnicas de paralelismo.

Por último, los Wavefronts dividen una imágen en filas de CTU, donde cada fila se procesa por un hilo diferente. Las dependencias entre filas se mantiene expcepto para el estado del contexto del aritmético CABAC [13], que se reinicializa al comienzo de cada fila de CTU. Para mejorar la eficiencia, en vez de reiniciar normalmente los contextos en CABAC, estos se heredan de la fila anterior.

Todas estas técnicas de paralelización son más útiles con resoluciones de imágenes superiores al HD. Para imágenes pequeñas, la decodificacion en tiempo real se puede realizar con un sólo proceso. Para resoluciones grandes de imágen puede ser util forzar un número mínimo de particiones de imágen para garantizar un nivel mínimo de paralelismo en el decodificador.

El software de referencia HM no soporta directamente muchas de estas técnicas de paralelización, sobre todo por el diseño de la implementación. En la siguiente sección presentaremos algunas aproximaciones de paralelización a nivel de GOP (GOP-based) que pueden implementarse en arquitecturas hardware multicore.

IV. algoritmos paralelos

En las secciones anteriores hemos revisdo las pricipales características del estandar de vídeo HEVC. Hemos paralelizado el software de referencia usando los modos LB y AI, ambos combinados con el per-

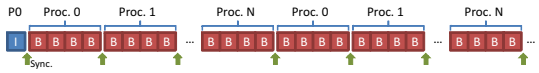


Fig. 1. Opción I: Distribución paralela.

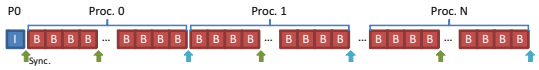


Fig. 2. Opción II: Distribución paralela.

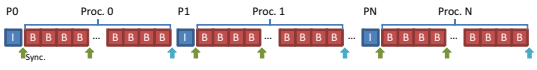


Fig. 3. Opción III: Distribución paralela.

Fig. 4. Opción IV: Distribución paralela.

fil Main (8 bits). Estos dos modos son utiles para aplicaciones con restricciones de tiempo, así que pensamos que pueden beneficiarse de las estrategias de paralelización. Obviamente, este trabajo puede exterderse fácilmente al modo Low-Delay P. Sin embargo, esto nos es así para el modo Random-Access debido a la forma en que usa los frames de referencia, usando referencias pasadas y futuras.

Los algoritmos paralelos desarrollados estan diseñados para un nivel de paralelización a nivel de GOP. Primeramente, decir que el tamaño de GOP para el modo AI es 1, porque todos los frames se computan como I-frames (sin frames de referencia). En el modo LB el tamaño de GOP es 4, sin embargo, este valor puede cambiarse. Por otra parte, hemos considerado algoritmos síncronos donde el proceso de sincronización se realiza después de computar los GOPs. Hemos desarrollado 4 aproximaciones:

- Opción I: (LB) en esta opción asignamos de forma secuencial cada GOP a cada proceso en la ejecución paralela, de manera que cada proceso codificará GOPs aislados.
- Opción II: (LB) en esta aproximación dividimos la secuencia en tantas partes como el número de procesos paralelos disponibles, de manera que cada proceso codificará grupos de GOPs adyacentes.
- Opción III: (LB) similar a Opción II, excepto que cada proceso comienza la codificación insertando un I-frame.
- Opción IV: (AI) similar a Opción I donde cada GOP se asigna secuencialmente a cada proceso, pero el GOP consiste en sólo un I-frame.

La Figura 1 muestra la distribución paralela para la Opción I. Los procesos de sincronización están localizados después de cada GOP. El proceso maestro (Proc. 0 o P0) computa el primer frame como I-frame. Tras esto, todos los procesos codifican un GOP de 4 B-frames. Todos los procesos, excepto el proceso maestro, codifica su primer B-frame sin frames de referencia, de esta manera, el número de bits necesarios para codificar este primer B-frame es similar al que usa el I-frame.

Para aumentar el rendimiento de los algoritmos paralelos propuestos, cada proceso contiene sus propios buffers de trabajo. Este hecho cambia el patrón real de frames de referencia utilizados. Por ejemplo, en la versión secuencial, el segundo B-frame de un GOP usa los frames *-1 -2 -6 -10* como frames de referencia, donde (*-1* significa el frame previo, y así sucesivamente). Como el tamaño de GOP es 4, el frame *-2* apunta al último frame del GOP previo. En el procesamiento paralelo, como asignamos GOPs aislados a cada proceso, el GOP previo no es el adyacente en la secuencia original, así, el frame *-2* no apunta al frame dos posiciones antes del frame actual. Si, por ejemplo, el número de procesos es 6, entonces el GOP previo para este proceso se localiza en la secuencia de vídeo 6 GOPs atrás con respecto al GOP actual. Así pues, el segundo B-frame de un GOP apuntará al frame *-22 (-2-(6-1)x4=-22)* en la secuencia de vídeo original. Se puede concluir que el algoritmo paralelo y el secuencial producirán bitstreams diferentes. Esto se analizará en la sección V, así como su impacto en términos de PSNR y bitrate.

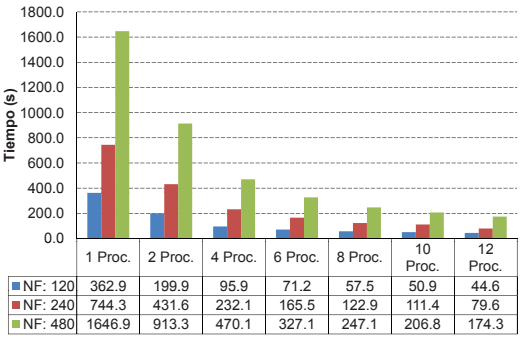
En la Figura 2 se puede ver una representación de distribución de la versión paralela Opción II. A igual que en la Opción I, los procesos de sincronización se localizan después de cada GOP. Así, todos los procesos, excepto el proceso principal, codifican su primer B-frame sin frames de referencia. En este caso, la lista de referencias no se modifica porque cada proceso trabaja con GOPs adyacentes. En el ejemplo anterior, para el segundo frame del GOP, el patrón de referencias solo se altera para los primeros tres GOPs. A partir de ahí, todos los frames de referencia están disponibles en el buffer privado de cada proceso.

La Figura 3 muestra la distribución paralela para la Opción III, donde se usa una estructura similar a la de la Opción II. Aquí cada proceso comienza codificando el primer frame como un I-frame. En este caso, las ejecuciones paralela y secuencial pueden ser identicas si en la versión secuencial realizamos un pequeño cambio en el fichero de configuración, cambiando el parámetro *IntraPeriod* de acuerdo con el número de procesos de la ejecución paralela. La Tabla I presenta los valores del parámetro *IntraPeriod* cuando computamos 240 y 480 frames. Más aún, el I-frame que se incluye debe ser un IDR (Instantaneous Decoding Refresh), así que debemos poner el parámetro *DecodingRefreshType* a 2.

Finalmente, en la Figura 4 se muestra la distribución paralela para la Opción IV. Remarcar, que la estructura paralela es simalar a la de la Opción I, pero el GOP siempre consiste en un I-frame. Los I-frames son de tipo IDR, con lo cual no hay diferencias entre las ejecuciones paralelas y secuencial.

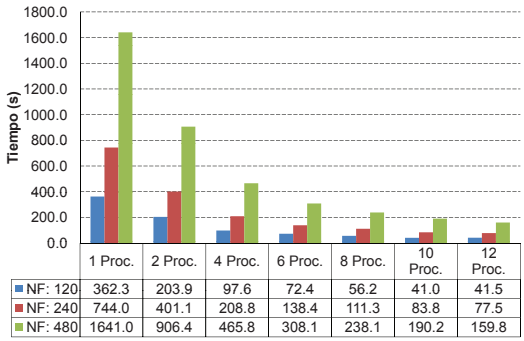
V. Experimentos

Analizaremos los algoritmos paralelos descritos en la sección IV, en términos de rendimiento paralelo, PSNR y bitrate. Se ha utiliado el paradigma de pro-



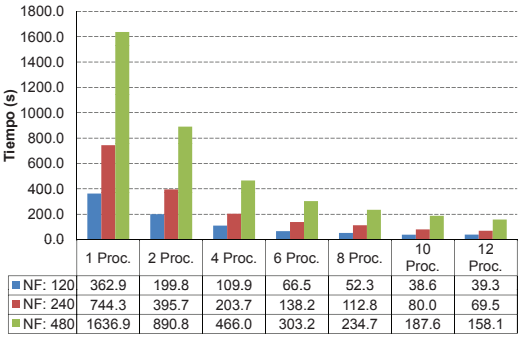
	1 Proc.	2 Proc.	4 Proc.	6 Proc.	8 Proc.	10 Proc.	12 Proc.
NF: 120	362.9	199.9	95.9	71.2	57.5	50.9	44.6
NF: 240	744.3	431.6	232.1	165.5	122.9	111.4	79.6
NF: 480	1646.9	913.3	470.1	327.1	247.1	206.8	174.3

(a) Opción I.



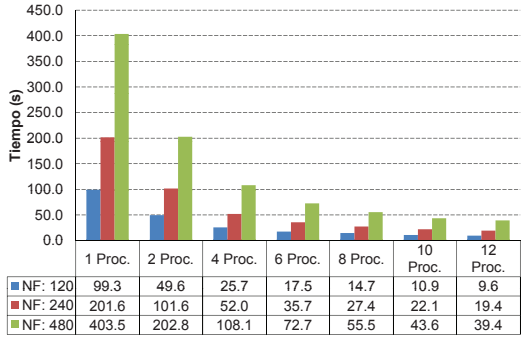
	1 Proc.	2 Proc.	4 Proc.	6 Proc.	8 Proc.	10 Proc.	12 Proc.
NF: 120	362.3	203.9	97.6	72.4	56.2	41.0	41.5
NF: 240	744.0	401.1	208.8	138.4	111.3	83.8	77.5
NF: 480	1641.0	906.4	465.8	308.1	238.1	190.2	159.8

(b) Opción II.



	1 Proc.	2 Proc.	4 Proc.	6 Proc.	8 Proc.	10 Proc.	12 Proc.
NF: 120	362.9	199.8	109.9	66.5	52.3	38.6	39.3
NF: 240	744.3	395.7	203.7	138.2	112.8	80.0	69.5
NF: 480	1636.9	890.8	466.0	303.2	234.7	187.6	158.1

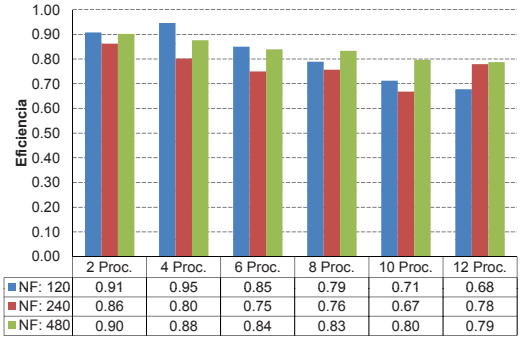
(c) Opción III.



	1 Proc.	2 Proc.	4 Proc.	6 Proc.	8 Proc.	10 Proc.	12 Proc.
NF: 120	99.3	49.6	25.7	17.5	14.7	10.9	9.6
NF: 240	201.6	101.6	52.0	35.7	27.4	22.1	19.4
NF: 480	403.5	202.8	108.1	72.7	55.5	43.6	39.4

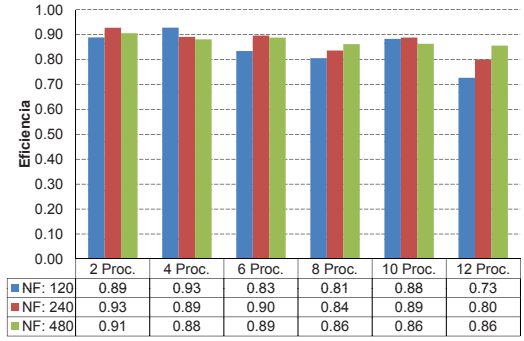
(d) Opción IV.

Fig. 5. Tiempos de cómputo de los algoritmos paralelos. Nº frames 120, 240 y 480.



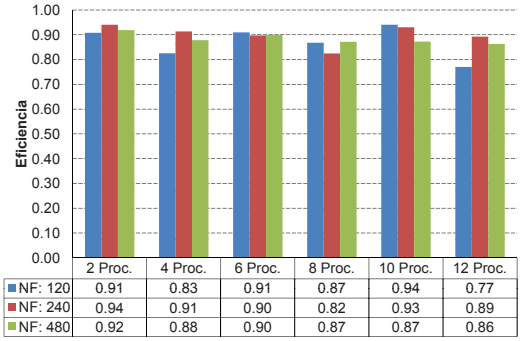
	2 Proc.	4 Proc.	6 Proc.	8 Proc.	10 Proc.	12 Proc.
NF: 120	0.91	0.95	0.85	0.79	0.71	0.68
NF: 240	0.86	0.80	0.75	0.76	0.67	0.78
NF: 480	0.90	0.88	0.84	0.83	0.80	0.79

(a) Opción I.



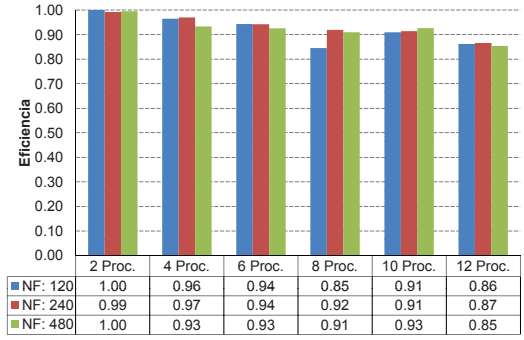
	2 Proc.	4 Proc.	6 Proc.	8 Proc.	10 Proc.	12 Proc.
NF: 120	0.89	0.93	0.83	0.81	0.88	0.73
NF: 240	0.93	0.89	0.90	0.84	0.89	0.80
NF: 480	0.91	0.88	0.89	0.86	0.86	0.86

(b) Opción II.



	2 Proc.	4 Proc.	6 Proc.	8 Proc.	10 Proc.	12 Proc.
NF: 120	0.91	0.83	0.91	0.87	0.94	0.77
NF: 240	0.94	0.91	0.90	0.82	0.93	0.89
NF: 480	0.92	0.88	0.90	0.87	0.87	0.86

(c) Opción III.



	2 Proc.	4 Proc.	6 Proc.	8 Proc.	10 Proc.	12 Proc.
NF: 120	1.00	0.96	0.94	0.85	0.91	0.86
NF: 240	0.99	0.97	0.94	0.92	0.91	0.87
NF: 480	1.00	0.93	0.93	0.91	0.93	0.85

(d) Opción IV.

Fig. 6. Eficiencia de los algoritmos paralelos. Nº frames 120, 240 y 480.

gramación OpenMP [14].

La prataforma multicore utilizada es un HP Proliant SL390 G7 con dos Intel Xeon X5660, cada CPU con seis núcleos a 2.8 GHz, con lo que los experimentos usan hasta 12 procesos. La secuencia de vídeo utiliada es *BasketballPass_416x240.yuv*, que contiene

Nº procesos	240 frames	480 frames
2	120	240
4	60	120
6	40	80
8	*30*	60
10	24	48
12	20	40

TABLA I

OPCIÓN III: PARÁMETRO *IntraPeriod* PARA IGUALAR EJECUCIONES SECUENCIAL Y PARALELA.

500 frames a $50Hz$ con una resolución de $416x240$ píxels. Hemos lanzado los algoritmos paralelos codificando 120, 240 y 480 frames en modo Low-delay. El valor de cuantización usado (QP) es 32;

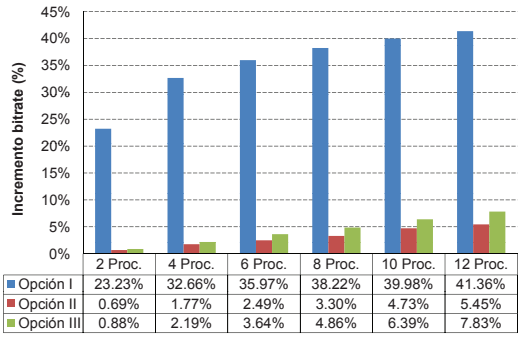
En la Figura 5 mostramos los tiempos de cómputo para la Opción I, Opción II, Opción III, y Opción IV. Los resultados muestran un buen comportamiento paralelo en todos los casos. Cuando usamos sólo 1 proceso, todos los algoritmos propuestos muestran el mismo tiempo que la versión secuencial. Con respecto a la Opción IV, la ejecución secuencial de referencia no es la misma, ya que usamos el modo AI.

La Figura 6 muestra la eficiencia asociada a los resultados de la Figura 5. Esto confirma el buen comportamiento de todos los algoritmos paralelos propuestos, obteniendo cerca de eficiencias ideales en algunos casos y por encima de 0.85 en la mayoría de los experimentos realizados. Remarcar que la Opción IV obtiene una eficiencia media superior a 0.95.

Como se comentó en la sección IV, las versiones paralelas no dan el mismo resultado que la versión secuencial. En la Figura 7 mostramos cómo las versiones paralelas modifican el bitrate de la versión secuencial. Es importante decir que la Figura 7 muesta los resultados para la Opción I, II y III, pero no para la Opción IV, porque en este caso la versiones secuencial y paralela muestran el mismo bitrate. También se puede observar que el incremento del bitrate introducido por la Opción I no es aceptable. Este algoritmo cambia drásticamente la estructura de los frames de referencia y como consecuencia incrementa el tamaño del bitstream generado, como se muestra en la Figura 7. En todos los casos, el incremento del bitrate es mayor conforme aumentamos el número de procesos. Finalmente, el incremento de bitrate en la Opción III, se debe a que el frame inicial (I-frame) se codifica con mayor calidad que el B-frame inicial de la Opción II, como se especifica en la configuración del perfil Low-delay.

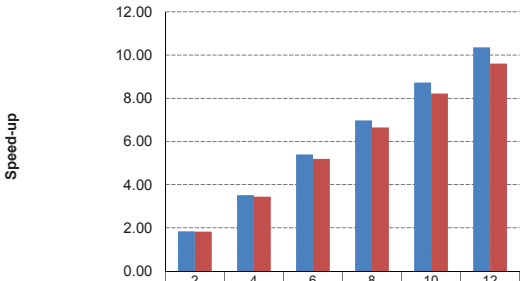
La Tabla II muestra el PSNR, i.e. una medida de calidad para los algoritmos paralelos II y III. Se puede observar que usando la Opción II la calidad del vídeo codificado decrece aunque en la Figura 7, se mostró que el bitrate aumentaba. Por el contrario, el incremento de bitrate de la Opción III mostrado en la Figura 7 se compensa con un aumento de la calidad como se puede observar en la Tabla II.

Finalmente, modificamos la configuración del perfil Low-delay para obtener el mismo PSNR y bitrate tanto en la versión paralela como en la secuencial



	2 Proc.	4 Proc.	6 Proc.	8 Proc.	10 Proc.	12 Proc.
Opción I	23.23%	32.66%	35.97%	38.22%	39.98%	41.36%
Opción II	0.69%	1.77%	2.49%	3.30%	4.73%	5.45%
Opción III	0.88%	2.19%	3.64%	4.86%	6.39%	7.83%

Fig. 7. Porcentaje de incremento de bitrate en los algoritmos paralelos. 480 frames.



	2 Proc.	4 Proc.	6 Proc.	8 Proc.	10 Proc.	12 Proc.
Speed-up NON_EQUIV	1.84	3.51	5.40	6.97	8.73	10.36
Speed-up EQUIV	1.82	3.44	5.19	6.65	8.22	9.60

Fig. 8. Speed-up del algoritmo Opción III. 480 frames.

del algoritmo Opción III. La Figura 8 muesta como *Speed-up NON_EQUIV* el speed-up cuando los algoritmos secuencial y paralelo obtienen resultados ligeramente diferentes, y como *Speed-up EQUIV* el speed-up cuando dan el mismo resultado. Se puede concluir que la Opción III obtiene buenas eficiencias que van desde 0.80 a 0.91.

VI. CONCLUSIONES

En este artículo hemos propuesto varios algoritmos paralelos para el codificador HEVC. Estos algoritmos se basan en un nivel de paralelización grueso a nivel de GOP. Estos algoritmos están especialmente diseñados para arquitecturas multicore. Tras implementar estos algoritmos en el software de referencia del HEVC en el perfil Low-delay se han realizado varios experimentos que muestran resultados interesantes como (a) La organización del GOP determina el rendimiento final de la codificación, siendo la mejor aproximación la Opción IV (modo AI) en términos de speed-up/eficiencia con respecto a su homólogo secuencial, y (b) Aunque la Opción III introduce un incremento de bitrate conforme aumentamos el número de procesos, el rendimiento paralelo global y las mejoras en PSNR lo convierten en un buen candidato para ser usado en modo LB (Low-delay B).

En general, todas las versiones desarrolladas obtienen buenas eficiencias, mostrando que las aproximaciones de paralelización basadas en GOP deben ser tenidas en cuenta para reducir la complejidad del codificador HEVC. Como trabajo futuro, exploraremos una aproximación jerarquica, combinando técnicas basadas en GOP junto con paralelización a

Algoritmos	1 Proc	2 Proc	4 Proc	6 Proc	8 Proc	10 Proc	12 Proc
Opción II	33.23	33.23	33.20	33.18	33.17	33.16	33.15
Opción III	33.23	**33.26**	**33.31**	**33.35**	**33.39**	**33.44**	**33.47**

TABLA II

PSNRs (dB) de los algoritmos paralelos (Luminancia). 480 frames.

nivel de Slice o de Wavefront.

Agradecimientos

Esta investigación ha sido financiada por el Ministerio de Educación y Ciencia TIN2011-27543-C03-03, por el Ministerio de Ciencia e Innovación TIN2011-26254 y por la Generalitat Valenciana ACOMP/2013/003.

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] 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.

[4] 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.

[5] 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.

[6] 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.

[7] 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.

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

[9] 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.

[10] Frank Bossen, "Common test conditions and software reference configurations," Tech. Rep. JCTVC-L1100, Joint Collaborative Team on Video Coding, Geneva, January 2013.

[11] Gisle Bjontegaard, "Improvements of the BD-PSNR model," Tech. Rep. VCEG-M33, Video Coding Experts Group (VCEG), Berlin (Germany), July 2008.

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

[13] D. Marpe, H. Schwarz, and T. Wiegand, "Context-based adaptive binary arithmetic coding in the H.264/AVC video compression standard," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 13, no. 7, pp. 620–636, 2003.

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