Análisis del efecto del particionado en Tiles sobre una versión paralela del codificador HEVC

Otoniel López, Pablo Piñol, Héctor Migallón, Vicente Galiano, Manuel P. Malumbres [1]

Resumen— **El nuevo estándar de codificación de vídeo HEVC introduce un nuevo concepto llamado 'Tiles'. Los Tiles son regiones rectangulares de un frame de vídeo que pueden ser codificadas de manera independiente. Con el fin de reducir el tiempo necesario para codificar una secuencia de vídeo con HEVC, hemos utilizado un enfoque paralelo basado en Tiles y se han evaluado los beneficios y desventajas de esta aproximación. En las pruebas realizadas se obtienen aceleraciones de hasta 9.3x utilizando 10 procesos con una pérdida en R/D despreciable.**

Palabras clave— **HEVC, openmp, tiles, parallelismo, codificación de video.**

EL Joint Collaborative Team on Video Coding (JCT-VC) ha desarrollado un nuevo estándar de codificación de vídeo llamado High Efficiency Video Coding (HEVC)[1]. El JCT-VC está formado por miembros de la norma ISO/IEC Moving Picture Experts Group (MPEG) y la UIT-T Video Coding Experts Group (VCEG). El nuevo estándar alcanza casi un 50% de ahorro de tasa de bits en comparación con el anterior estándar de codificación de video H.264/AVC (Advanced Video Coding) [2]. El aumento de la eficiencia está ligada a un aumento de la complejidad computacional. Para hacer frente al aumento de la complejidad se utilizan técnicas de paralelización para aprovechar las arquitecturas de computación paralela disponibles.

HEVC incluye nuevas características que permiten la computación paralela de alto nivel, como el Wavefront Parallel Processing (WPP) o los Tiles y también incluye nuevas características que permiten la computación paralela de bajo nivel (dentro del proceso de codificación), como el Parallel Method [3] que permite la estimación de movimiento de forma paralela.

Los tiles son regiones rectangulares de un frame de vídeo que pueden ser codificados de forma independiente. En este trabajo se analiza el comportamiento paralelo cuando se usan tiles y cómo el diseño de particionado afecta en el rendimiento tanto en la aceleración como en el Rate/Distortion (R/D).

Algunos trabajos como [4] se centran en la paralelización del decodificador HEVC, ya que se busca una decodificación rápida de contenidos multimedia pre-codificada, como el cine digital y de vídeo bajo demanda. Sin embargo, pensamos que se deben realizar más esfuerzos en el lado del codificador, donde se encuentra la mayor complejidad computatioonal. La paralelización de la parte de codificación de vídeo puede ser útil para aplicaciones como la grabación de vídeo o el streaming de eventos en vivo.

Otros trabajos analizan y proponen técnicas paralelas de bajo nivel para la codificación de secuencias de vídeo con HEVC, como la paralelización del módulo de estimación de movimiento [5] o la paralelización del módulo de predicción intra [6]. Nuestro trabajo se basa en aplicar técnicas paralelas de alto nivel, mediante el uso de tiles sobre arquitecturas de memoria compartida.

En [7], los autores comparan slices y tiles sobre HEVC. Los resultados se muestran en términos de porcentaje de incremento o decremento de la tasa de bits. En nuestro estudio evaluamos el rendimiento de los tiles, pero nos centraremos tanto en la reducción de la complejidad relacionada con la codificación como en el R/D obtenido. Además, se evalúa el impacto del particionado de los tiles.

El resto del trabajo se organiza de la siguiente manera. En la Sección I presentaremos los principales aspectos relacionados con los tiles en HEVC. En la Sección II se presentan y analizan los resultados de las pruebas realizadas. Finalmente, en la Sección III se presentan las conclusiones obtenidas.

I. Particionado en Tiles

La división de un frame de vídeo en tiles es una técnica nueva incluida en HEVC que no existía en los estándares anteriores de codificación de vídeo. Los tiles son regiones rectangulares de un frame de vídeo que puede ser codificadas de forma independiente (y posteriormente decodificadas), con el uso de algunos datos comunes para todo el bitstream. Esta independencia permite la paralelización de los procesos de codificación y decodificación a nivel 'subpicture'. Los tiles contienen un número entero de Coding Tree Units (CTUs). Cada región rectangular resulta de la división de un frame en varias columnas y filas. El ancho de cada columna (en unidades CTU) y también el alto de cada fila se puede ajustar individualmente. En el ejemplo que se muestra en la Figura 1, un frame de resolución Full HD (1920x1080) se divide en 10 tiles utilizando un esquema de particionado de 5 columnas con un ancho de 6 CTUs y 2 filas con una altura de 8 y 9 CTUs, respectivamente.

La independencia de los tiles permite la paralelización del proceso de codificación, pero tiene un inconveniente principalmente: la eficiencia de la codificación respecto al rendimiento en R/D, se ve reducida. Esto sucede porque los tiles no pueden

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

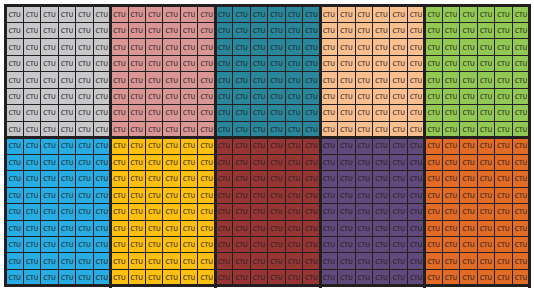


Fig. 1. Division de un frame full-HD (1920x1080 pixels) en 10 tiles (5 columnas de 6 CTUs de ancho; 2 filas con 8 and 9 CTUs de alto cada una).

utilizar la información de los CTUs perteneciente a los otros tiles vecinos para realizar cualquier tipo de predicción, y por lo tanto, la redundancia existente entre los CTUs vecinos que pertenecen a diferentes tiles no puede ser explotada.

En este trabajo se ha implementado una paralelización a nivel de tile en el codificador de vídeo HEVC para plataformas de memoria compartida. La codificación de cada tile se asigna a un núcleo diferente. Hemos evaluado el rendimiento de la versión paralela para 2, 4, 6, 8, 9, y 10 procesos y se han comparado los resultados obtenidos con los proporcionados por la versión secuencial, tanto con respecto al rendimiento en R/D, como en tiempo de cómputo. Como se ha comentado anteriormente, se mapean los tiles de cada frame en el mismo número de procesos. Para un cierto número de tiles por frame, podemos encontrar diferentes particionados del frame. Por ejemplo, si queremos dividir el frame en 9 tiles podemos elegir tres diferentes distribuciones: 9x1 (9 columnas por fila 1), 1x9 (1 columna por 9 filas) y 3x3 (3 columnas por filas 3). Cada una de estas distribuciones se podria, además, haber diseñado de manera diferente si cambiamos el ancho y el alto de cada una de las columnas y las filas. Con el fin de obtener la máxima eficiencia de computación en paralelo vamos a utilizar los diseños que proporcionan la distribución de la carga más equilibrada, es decir, los tiles que producen un número similar de CTUs. Una distribución de carga equilibrada no siempre garantiza una distribución equilibrada de trabajo debido a que los recursos necesarios para codificar un solo CTU pueden variar, pero una distribución desequilibrada de la carga probablemente implicará una baja eficiencia computacional. Como una medida del equilibrio en la distribución de la carga, hemos calculado la eficiencia máxima teórica que se podría obtener en la versión paralela, teniendo en cuenta la misma complejidad computacional para cada CTU. Un valor del 100% significa que todos los procesos codificarán el mismo número de CTUs. En la Tabla I, se enumeran los diferentes particionados de tiles que utilizaremos en nuestras pruebas con diferente número de núcleos y dos formatos de vídeo.

La eficiencia óptima paralela teórica sería proporcionada por la columna 'balance %' en la Tabla I. Si dividimos, por ejemplo, un frame de 2560x1600 píxeles en 10 tiles, y elegimos el particionado 10x1, entonces tendremos 10 tiles con 100 CTUs cada uno, lo que significa un 100% de balanceo de carga (todos los tiles tienen el mismo número de CTUs). Si nosotros, por lo contrario, seleccionamos el particionado de 1x10, entonces vamos a tener 5 tiles con 80 CTUs cada uno, y 5 tiles con 120 CTUs cada uno. El escenario más probable es que los 5 procesos que gestionan los tiles más 'pequeños' permanezcan a la espera de los procesos responsables de los tiles más 'grandes'. En este caso, el índice de balanceo de carga máxima sería el 83%. Por lo tanto, para un número específico de procesos, el diseño seleccionado puede afectar a la eficiencia de la versión paralela. Se ha de tener también en cuenta que un solo diseño puede proporcionar diferentes porcentajes de equilibrio de carga en función de la resolución de la secuencia de vídeo. Por ejemplo, la disposición 4x1 obtiene el 100% y el 94% de equilibrio de carga para resoluciones de video de 2560x1600 y 1920x1080 respectivamente.

En la siguiente sección se presentarán los resultados para las secuencias de vídeo seleccionadas, usando todos los particionados que se indican el la Tabla I.

II. Experimentos

El algoritmo paralelo propuesto se ha lanzado sobre una plataforma de memoria compartida que consiste en dos hexacores Intel Xeon X5660 con frecuencias de hasta 2.8GHz, una caché de 12 MB por procesador y 48 GB de RAM. El sistema operativo utilizado es CentOS Linux 5.6 para x86 de 64 bits. El entorno paralelo usado es OpenMP [8]. El compilador utilizado es *g ++* compilador v.4.1.2. La versión del software de codificación de referencia utilizado es HM 16.3 [9].

Las secuencias de vídeo de test utilizadas en nuestros experimentos son Traffic (2560x1600), People on Street (2560x1600), Tennis (1920x1080) y Park Scene (1920x1080). Los resultados obtenidos son para los modos de codificación Low-delay B (LB) y All Intra (AI), codificando 150 frames las secuencias Traffic y People y 240 frames para ParkScene y Tenis con diferentes parámetros de cuantificación (QPs) (22, 27, 32, 37).

En la Figura 2, se presenta la evolución del tiempo de codificación para las secuencias Traffic y People para los modos AI y LB en función del número de procesos según el particionado de tiles propuesto. Como se puede ver, el tiempo de codificación puede verse reducido hasta 9.3 veces cuando usamos 10 procesos. Como puede verse en la Figura 3, el algoritmo de paralelización a nivel de tiles obtiene un buen rendimiento y también buenos resultados en cuanto a escalabilidad. Si nos fijamos en la Figura 3, para 10 procesos, existen diferencias en el rendimiento paralelo cuando usamos diferentes particionados de tiles. En concreto, en el modo de codificación AI podemos encontrar diferencias de hasta 1.58x de un esquema de partición de tiles de 1x10 con respecto al esquema de partición de tiles de 10x1. Por lo gen-

TABLA I

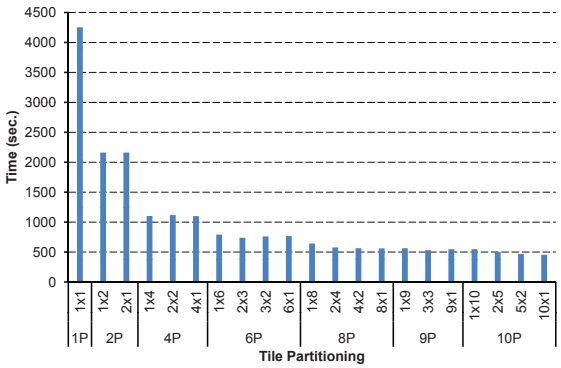
PARTICIONADO Y PORCENTAJE DE BALANCEO DE CARGA

(a) 2560x1600 (40x25 CTUs)

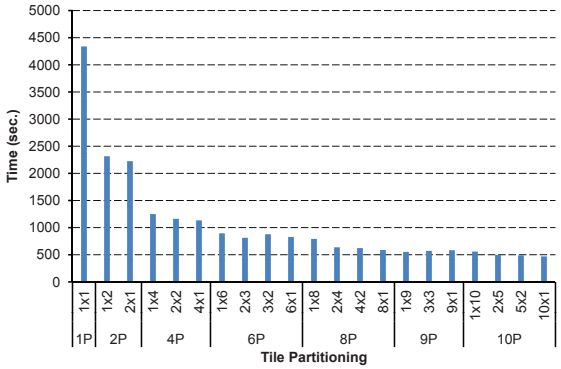
#Proc	Particionado	AvgCTU	MaxCTU	Bal %
1P	1x1	1000	1000	100%
2P	1x2	500	520	96%
	2x1	500	500	100%
4P	1x4	250	280	89%
	2x2	250	260	96%
	4x1	250	250	100%
6P	1x6	166.7	200	83%
	2x3	166.7	180	93%
	3x2	166.7	182	92%
	6x1	166.7	175	95%
8P	1x8	125	160	78%
	2x4	125	140	89%
	4x2	125	130	96%
	8x1	125	125	100%
9P	1x9	111.1	120	93%
	3x3	111.1	126	88%
	9x1	111.1	125	89%
10P	1x10	100	120	83%
	2x5	100	100	100%
	5x2	100	104	96%
	10x1	100	100	100%

(b) 1920x1080 (30x17 CTUs)

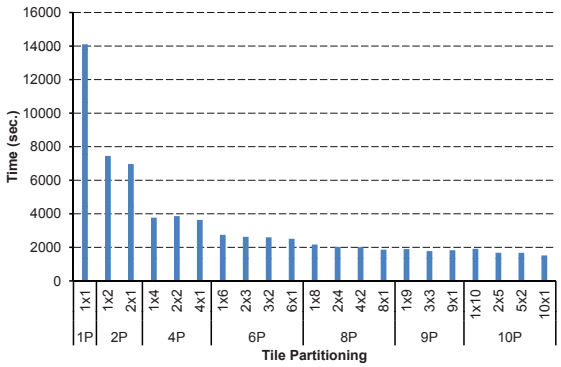
#Proc	Particionado	AvgCTU	MaxCTU	Bal %
1P	1x1	510	510	100%
2P	1x2	255	270	94%
	2x1	255	255	100%
4P	1x4	127.5	150	85%
	2x2	127.5	135	94%
	4x1	127.5	136	94%
6P	1x6	85	90	94%
	2x3	85	90	94%
	3x2	85	90	94%
	6x1	85	85	100%
8P	1x8	63.8	90	71%
	2x4	63.8	75	85%
	4x2	63.8	72	89%
	8x1	63.8	68	94%
9P	1x9	56.7	60	94%
	3x3	56.7	60	94%
	9x1	56.7	68	83%
10P	1x10	51	60	85%
	2x5	51	60	85%
	5x2	51	54	94%
	10x1	51	51	100%



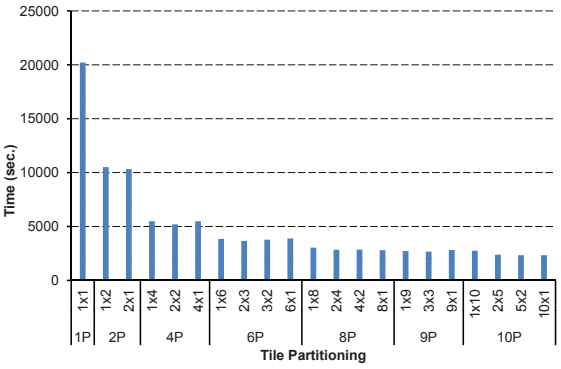
(a) Traffic AI mode.



(b) People AI mode.



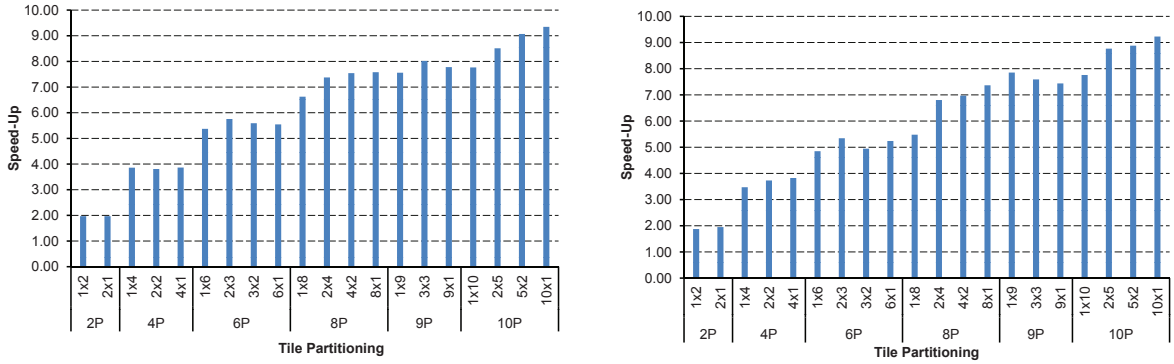
(c) Traffic LB mode.



(d) People LB mode.

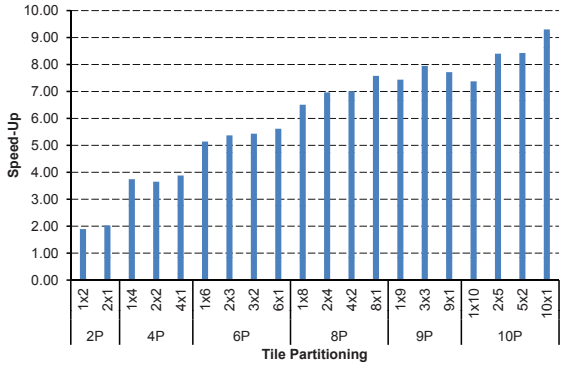
Fig. 2. Evolución del tiempo de codificación para todas las secuencias de video con diferente numero de procesos y particionados de tiles para un QP=37.

eral, los diseños de particionado de tiles basados en columnas de CTUs o los tiles cuadrados obtienen un mejor rendimiento paralelo. Como es de esperar, este efecto depende de la resolución de vídeo. Siguiendo con el ejemplo anterior, para una resolución de vídeo de 2560x1600, y un tamaño de CTU de 64x64, el

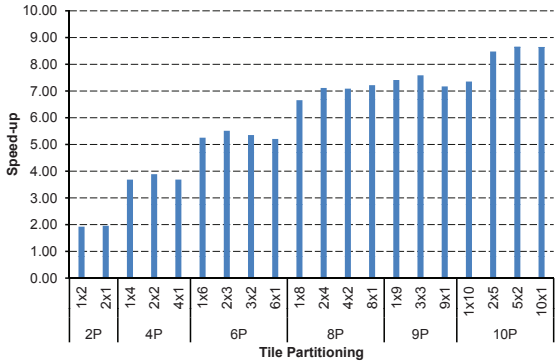


(a) Traffic AI mode.

(b) People AI mode.



(c) Traffic LB mode.



(d) People LB mode.

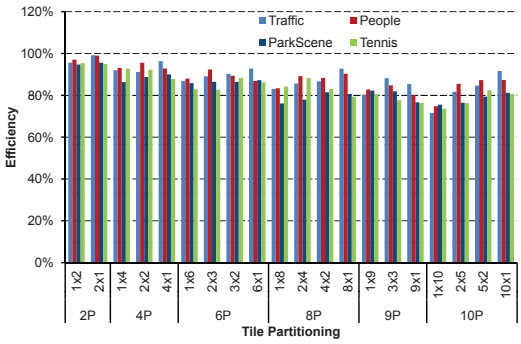
Fig. 3. Evolucion del Speed-Up para todas las secuencias de video con diferente numero de procesos y particionados de tiles para un QP=37.

número de CTUs en un frame son 40x25. Si dividimos el frame con el particionado 1x10, tenemos 5 procesos con CTUs de 40x2 y 5 procesos con 40x3 CTUs. Por otra parte, si dividimos el frame con la disposición 10x1, contamos con 10 procesos de 4x25 CTUs. En el primer caso (1x10), 5 procesos tienen que realizar un 50% más trabajo que los otros 5 procesos. Por lo general, cuanto más equilibrada esté la carga computacional, mejor rendimiento paralelo se consigue, a excepción de algunas secuencias en las que esto no se logra. En estas excepciones, incluso cuando cada proceso tiene el mismo número de CTUs, la complejidad computacional inherente a cada CTU difiere, produciendo que algunos procesos terminen antes que los otros.

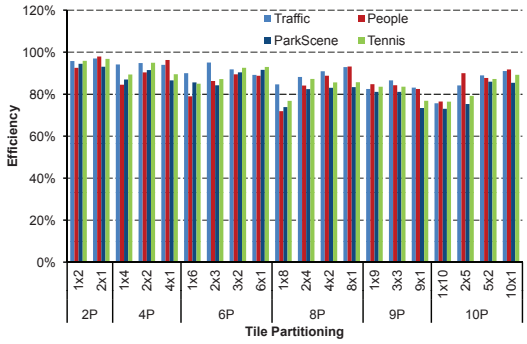
En la Figura 4 se muestra la eficiencia media de la versión paralela para todos los valores de QP de las diferentes secuencias codificadas para ambos modos LB y AI. Como se puede ver, se obtienen buenas eficiencias para los dos modos de codificación LB y AI, siendo la eficiencia media del 87% . Sin embargo, si nos centramos en los diseños de particionado de tiles cuadrados, la eficiencia media se eleva hasta el 91%.

En cuanto al rendimiento en R/D, en la Figura 5 se presenta el % de BD-rate para las secuencias de vídeo de resolución 4K en función del número de procesos y del esquema de particionado en tiles. Como se puede ver, el % de BD-rate aumenta cuando aumenta el número de procesos. Este es un comportamiento esperado porque los tiles son estructuras independientes y por lo tanto el codificador aritmético funciona de una manera independiente en cada tile, no disponiendo de información de los tiles previamente codificados. Por otra parte, la partición en tiles cuadrados obtiene ligeramente mejor resultado tanto en el modo LB como en AI, debido a que hay más informacion de los CTUs vecinos para la predicción inter e intra.

Según los resultados obtenidos, podemos asegurar que la paralelización del codificador HEVC en tiles es interesante, ya que reduce drásticamente el tiempo de codificación (hasta 9.3x para 10 procesos) con un ligero incremento en R/D, especialmente si se utilizan esquemas de particionado cuadrados (BD-rate de 0.75% para el modo AI y 1.2% para el modo LB, en promedio). El incremento máximo debido al esquema de particionado del tile es de un 5% para la secuencia Tennis en el modo LB utilizando 10 procesos. Por lo tanto, el principal aspecto que afectará al rendimiento de la versión paralela es la carga computacional y es por eso que debemos dividir los tiles de tal manera que todos los procesos tengan el mismo número de CTUs. En el caso de las secuencias de vídeo analizadas en este trabajo, un particionado de tiles cuadrados o con más columnas que filas de CTUs, tienen una mejor distribución de la carga computacional.

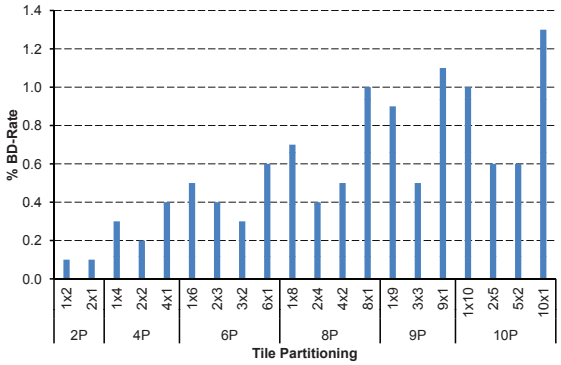


(a) LB mode.

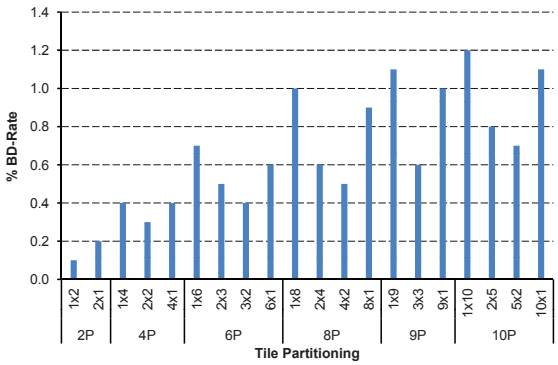


(b) AI mode.

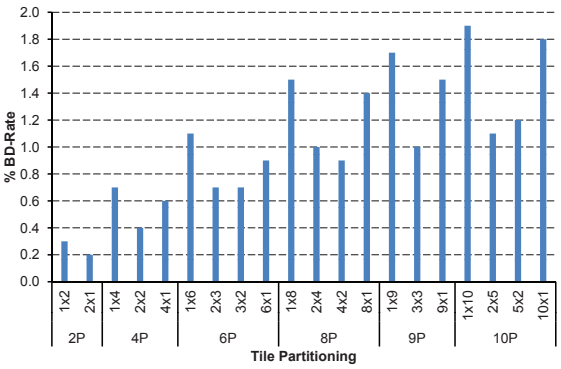
Fig. 4. Efficiencia paralela media para todas las secuencias de video con diferente número de procesos y particionado de tiles para todos los QPs.



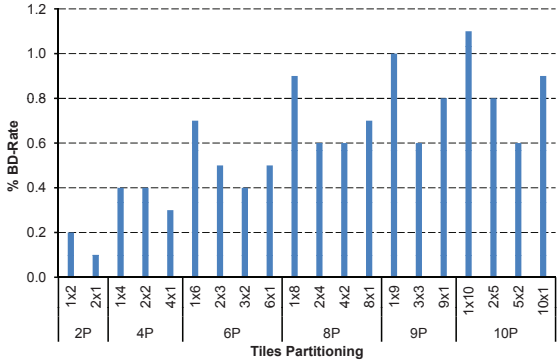
(a) Traffic AI mode.



(b) People AI mode.



(c) Traffic LB mode.



(d) People LB mode.

Fig. 5. % BD-Rate medio para todas las secuencias de video con diferente numero de procesos y particionado de tiles para todos los QPs.

III. Conclusiones

En este artículo se presenta un estudio del la paralelización a nivel de tiles del codificador HEVC, usando diferentes diseños de particionado de frame. Los resultados muestran que tanto los particionados de tiles cuadrados como aquellos con más columnas que filas obtienen los mejores resultados (hasta 9.3x para 10 procesos) en las secuencias de vídeo utilizadas. Aunque en algunos experimentos los particionados de tiles basados en columnas puedan obtener una mejor eficiencia, en promedio, los particionados de tiles cuadrados presentan un mejor comportamiento tanto en speed-up como en R/D. Además, el incremento en el porcentaje de BD-rate es baja en todos los casos, especialmente cuando se aplican particionados de tiles cuadrados, debido a la mayor información disponible de los CTUs vecinos para los procesos de predicción inter e intra.

Por último, debemos tener en cuenta la resolución de la secuencia de vídeo con el fin de realizar el particionado de los frames, de tal manera que el número de CTUs en cada tile sea casi el mismo y por lo tanto la carga computacional esté más equilibrada.

Agradecimientos

Esta investigación ha sido financiada gracias al Ministerio de Economía y Competitividad TIN2015-66972-C5-4-R co-financiado con fondos FEDER.

Referencias

[1] Benjamin Bross, Woo-Jin Han, Jens-Rainer Ohm, Gary J. Sullivan, Ye-Kui Wang, and Thomas Wiegand, "High Efficiency Video Coding (HEVC) text specification draft 10," Tech. Rep. JCTVC-L1003, Joint Collaborative Team on Video Coding (JCT-VC), Geneva (Switzerland), January 2013.

[2] 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*, 2012.

[3] Minhua Zhou, "AHG10: Configurable and CU-group level parallel merge/skip," Tech. Rep., Joint Collaborative Team on Video Coding-H0082, 2012.

[4] 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.

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

[6] 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.

[7] K. Misra, A. Segall, M. Horowitz, Shilin Xu, A. Fuldseth, and Minhua Zhou, "An overview of tiles in HEVC," *Selected Topics in Signal Processing, IEEE Journal of*, vol. 7, no. 6, pp. 969–977, Dec 2013.

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

[9] HEVC Reference Software, https://hevc.hhi.fraunhofer.de/svn/svn_HEVCSoftware/tags/HM-16.3/, ," .