Implementación de un estimador de movimiento de baja latencia para codificador HEVC sobre FPGA

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

Resumen— **El recientemente desarrollado codificador de vídeo HEVC dobla en eficiencia de compresión a su predecesor H.264/AVC. Como en los estándares anteriores, la estimación de movimiento es uno de los bloques más críticos del codificador, debido a la gran complejidad que requiere para eliminar la redundancia temporal, lo que se traduce en una ganancia de compresión bastante significativa. Con el fin de reducir el tiempo de codificación de vídeo en general, en este trabajo se propone una implementación del bloque de estimación de movimiento del HEVC en hardware sobre FPGA. La arquitectura propuesta se basa en (a) un nuevo orden de escaneo de memoria, (b) una nueva estructura del árbol de sumadores, la cual soporta todos los modos de particionado de una manera eficiente y con baja latencia. Esta arquitectura se ha desarrollado en lenguaje VHDL y se ha sintetizado e implementado en la FPGA de Xilinx, Zynq-7000 xc7z020 clg484-1. Se ha hecho una verificación funcional del diseño implementado logrando unas tasas de frame de más de 30 fps para formatos de vídeo de Full-HD y 15 fps para 2K.**

Palabras clave— **HEVC, codificación de vídeo, FPGA, estimación de movimiento.**

I. Introducción

EL nuevo estándar de codificación de vídeo llamado HEVC (High Efficiency Video Coding) ha sido desarrollado con la colaboración conjunta de ITU-T VCEG e ISO/IEC MPEG, bajo el nombre de JCT-VC (Joint Collaborative Team on Video Coding). HEVC consigue mejorar la eficiencia de compresión, logrando la misma calidad de vídeo que el anterior estándar H.264/AVC (Advanced Video Coding) con la mitad de tasa de bits, aproximadamente.

Sin embargo, el módulo de estimación de movimiento (ME) necesita más del 90% del tiempo total de codificación. Ello se debe al extenso conjunto de modos de particionado del Coding Tree Unit (CTU) , unidad básica de codificación del HEVC, y también al mayor tamaño del CTU con respecto a su predecesor H.264/AVC [1].

La estructura de codificación a nivel de frame dentro del HEVC consta de CUs (Coding Units) con un tamaño de bloque máximo de 64x64 píxeles, y cada CU se puede dividir recursivamente hasta un tamaño de bloque de 8x8 píxeles. Los CUs constan de unidades de predicción (Intra o Inter) y el tamaño de cada unidad de predicción puede variar desde el tamaño máximo del CU hasta 4x4 píxeles en predicción Intra y 4x8 ó 8x4 en Inter, soportando 8 modos de particionado para cada nivel de profundidad del CU como se muestra en la Figura 1, donde 2N=64 representa una profundidad 0, 2N=32 profundidad 1 y así sucesivamente. Las particiones de (a) a (d) se conocen como modos de particionado simétricos (SMP) y de (e) a (h) como modos de particionado asimétrico (AMP). La unidad de estimación de movimiento es la encargada de comparar todas las unidades de predicción del frame actual con los bloques de la ventana de búsqueda perteneciente al frame de referencia (pasado o futuro) de manera que la diferencia de ambos bloques contenga la menor información residual [2].

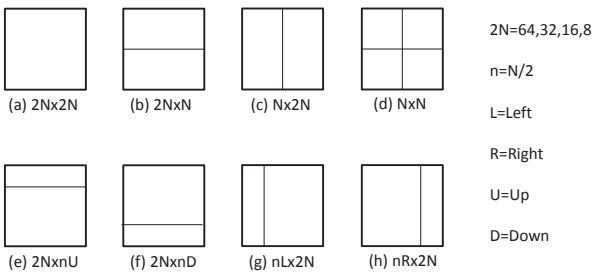


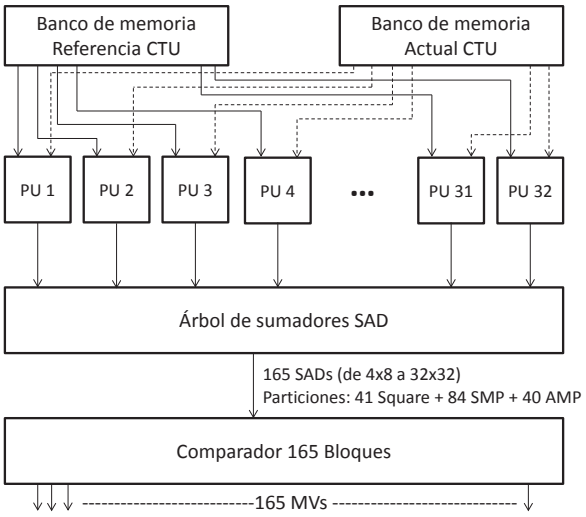
Fig. 1. Tamaños de la unidad de particionado en la predicción Inter de HEVC

Como se ha mencionado anteriormente, de todas las tareas en la codificación de vídeo HEVC, la estimación de movimiento es la que tiene un mayor coste computacional, por ello, en este trabajo se propone una arquitectura VLSI para realizar el cálculo de la estimación de movimiento de manera más rápida y precisa con el fin de reducir significativamente el coste computacional de todo el codificador.

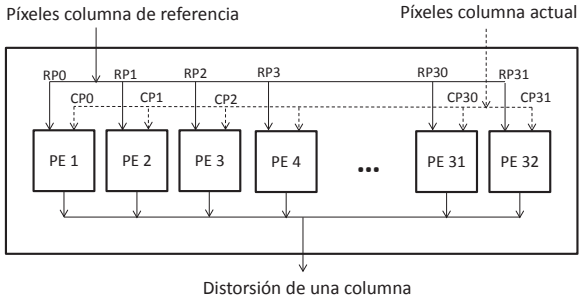
Este trabajo está estructurado de la siguiente manera: En la sección II se da una visión general del diseño propuesto, explicando cada uno de los módulos de los que consta el estimador de movimiento hardware. En las secciones III, IV y V se desarrollan con mayor nivel de detalle cada uno de los módulos del estimador de movimiento: El controlador de memoria, el árbol de sumadores SAD (Suma de diferencias absolutas) y el bloque de comparadores. Finalmente, en la sección VI se muestran los resultados obtenidos en términos de latencia, área utilizada en FPGA y tasas de frame de la arquitectura hardware propuesta, mientras que en la sección VII se presentan las conclusiones finales.

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

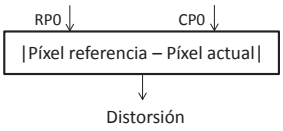
[2]Dpto. de Ingeniería de Comunicaciones, Univ. Miguel Hernández. Elche, e-mail: roberto.gutierrez@umh.es



(a) Arquitectura de la unidad SAD altamente paralela



(b) Arquitectura de un PU



(c) Arquitectura de un PE

Fig. 2. Arquitectura propuesta

II. Diseño general

El tamaño de CTU máximo en el diseño propuesto es de 32x32 píxeles, reduciendo así la complejidad del módulo ME con respecto a otros trabajos. Esta estrategia logra un buen equilibrio entre los recursos hardware utilizados y la calidad de compresión obtenida [3]. Además, la estrategia de búsqueda en el frame de referencia escogida es la de Full Search (búsqueda completa) por ser la más precisa, ya que de este modo, se buscan exhaustivamente los bloques del frame de referencia para cada unidad de predicción, comparando todas las posiciones en la ventana de búsqueda [4]. Esto tiene la ventaja de una regularidad computacional y una calidad de vídeo de salida excelente.

Esta arquitectura consta de (a) áreas de memoria asignadas a píxeles de la unidad de codificación actual (current CU) y píxeles correspondientes al área de búsqueda del frame de referencia, (b) 32 unidades de procesado (PU), (c) 1024 elementos de procesado (PE), (d) un árbol de sumadores donde se calcula la suma de diferencias absolutas (SAD) de los píxeles bajo estudio, y (e) un bloque comparador que almacena los valores de SAD mínimos y sus correspondientes vectores de movimiento (MVs) para todas las posibles particiones de un CTU.

Cada PE computa la distorsión de un píxel de referencia con un píxel actual, es decir, obtiene la diferencia en valor absoluto de ambos. Un PU consta de 32 PEs, el cual calcula los valores de distorsión de una columna de 32 píxeles. En cada ciclo de reloj, cada columna de píxeles, tanto del frame actual como el de referencia, se distribuyen en 32 PUs. De esta manera, el codificador es capaz de obtener la distorsión a nivel de píxel de un bloque de 32x32 (tamaño máximo del CTU en esta arquitectura). Por tanto, los valores de distorsión correspondientes a un bloque de 32x32 son calculados en un solo ciclo de reloj. El bloque del árbol de sumadores SAD (SATB) calcula los SADs para todos los modos de particionado del CU, utilizando las distorsiones obtenidas para un bloque de 32x32 píxeles. En la Figura 2 se muestra la estructura de la arquitectura propuesta.

El sistema se ha segmentado completamente, tal y como se observa en la Figura 3. El proceso de lectura de memoria y la propagación de los registros de desplazamiento requieren solamente un ciclo de reloj, una vez se ha realizado una precarga en los registros de desplazamiento de las 31 primeras filas de los píxeles de referencia correspondientes al primer CU de 32x32. Otro ciclo de reloj es necesario en el módulo de los PUs donde se obtienen las distorsiones de cada CU. El bloque SATB requiere diez ciclos más, y el comparador que almacena los SADs mínimos necesita 1 ciclo más. Finalmente, puesto que los procedimientos anteriores se realizan para cada CU dentro del área de búsqueda de referencia (64x64), la arquitectura propuesta necesita 4140 ciclos de reloj para realizar la estimación de movimiento en enteros (IME) para todos los tamaños de CUs desde 32x32 a 8x8 y todas sus posibles particiones.

III. Controlador de memoria

El área de búsqueda es la región del frame de referencia donde se realiza una búsqueda completa del CU actual, evaluando la distorsión en cada uno de los puntos de esa área [5]. En esta arquitectura, el tamaño del área de búsqueda es la definida por el estándar HEVC, es decir, el doble del tamaño del CU, siendo en este caso 64x64 píxeles. El área de búsqueda se encuentra centrada alrededor del CU actual (ver Figura 4).

Con el fin de reutilizar los datos al máximo, se ha adoptado un sistema de escaneo denominado escaneo de serpiente y un camino reconfigurable de datos con 32 registros de desplazamiento, tantos como número de filas del CU máximo [6]. En la primera etapa, (a), un registro de desplazamiento almacena una fila de 32 píxeles proporcionados por el sistema de alma-

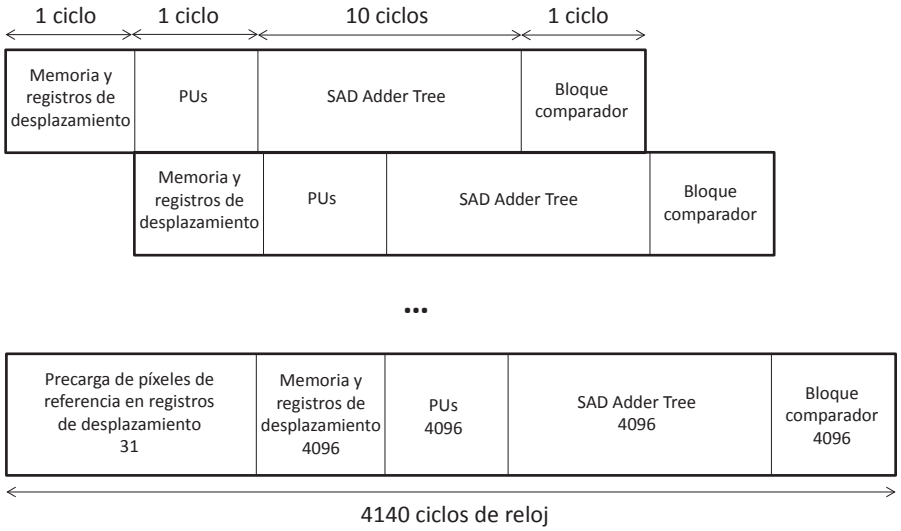


Fig. 3. Proceso pipeline del sistema propuesto

cenamiento de Block-RAMs, estableciendo la propagación de los registros en sentido de arriba hacia abajo. Tras cargar 32 filas en los registros (32x32 CU), una columna de 32 píxeles se envía a cada PU desde cada uno de los dos bancos de memoria, uno de los píxeles del frame de referencia y otro de los píxeles del CU actual, calculando así el SAD correspondiente. Tras ello, una nueva fila de 32 píxeles se carga en los registros de desplazamiento descartando la primera fila y así sucesivamente. De esta forma se procede al cómputo de un nuevo CU solamente desplazado un píxel en la misma dirección. Cuando se ha procesado el último CU en la dirección de escaneo de arriba hacia abajo, el siguiente CU a estudiar será el resultante de desplazar el registro un píxel hacia la derecha, etapa (b). Tras el procesado de este último CU, el controlador de memoria procederá a realizar el mismo cálculo de los CUs en dirección de abajo hacia arriba, etapa (c). Este procedimiento se llevará a cabo de manera iterativa hasta que todas las posiciones del CU en el área de búsqueda hayan sido procesadas. La Figura 4 muestra el proceso en el orden de escaneado del rango de búsqueda descrito anteriormente.

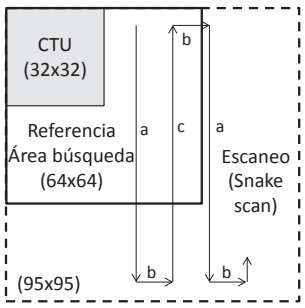


Fig. 4. Orden de escaneado del área de búsqueda

Para llevar a cabo el proceso de escaneado anterior, en cada ciclo de reloj se necesita acceder a 32 píxeles de una fila. Como se observa en la Figura 5 el banco de memoria donde se almacenan los píxeles del frame de referencia, está formado por 32 Block-RAMs donde se almacenan las columnas del área de referencia consecutivamente en cada una de ellas.

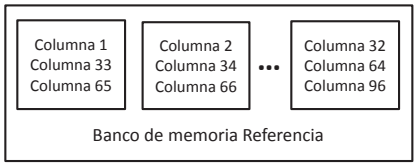


Fig. 5. Configuración de área de referencia

IV. Árbol de sumadores SAD

En este módulo se obtienen los valores SAD del CTU para cada partición posible a partir del tamaño del CU máximo, siendo en este caso desde 32x32 a 4x8 y 8x4 como se determina en la predicción inter del estándar HEVC (incluyendo los modos de particionado asimétricos). A partir de las 32x32 distorsiones (diferencias absolutas) proporcionadas por los PUs, las cuales están asociadas a la posición actual en el área de búsqueda, se realiza una sucesión de sumas, como se muestra en la Figura 6, con el fin de obtener los valores SAD correspondientes a todas las particiones del CU. En este caso, el número total de SADs correspondientes a los CUs y sus siete particiones dentro del CTU es de 165, de los cuales 41 SAD son del modo de particionado cuadrado (2Nx2N, NxN), 84 SADs del modo de particionado simétrico (2NxN, Nx2N) y 40 SADs del modo de particionado asimétrico (2NxnU, 2NxnD, nLx2N, nRx2N). En la Tabla I se muestra un resumen del número de SADs calculados por cada CU candidato en una posición fija del área de búsqueda.

En cada etapa de este árbol de sumadores, dos filas o columnas consecutivas se suman, reduciendo a la mitad la anchura o altura de la partición resultante. Este proceso de agregación de distorsiones continúa hasta que el último tamaño de partición sea 1x1, es decir, hasta que se obtiene el SAD correspondiente al CU de 32x32. En etapas intermedias, los SADs del resto de las particiones se almacenan como se

TABLA I
NÚMERO TOTAL DE SADS PARA CADA PARTICIÓN DE UN CTU DE 32X32

Tamaño de bloque	**No. SADs**	**Tamaño de bloque**	**No. SADs**
32x32(2Nx2N)	1	16x16(Nx2N)	8
32x32(2NxN)	2	16x16(NxN)	16
32x32(Nx2N)	2	16x16(nLx2N)	8
32x32(NxN)	4	16x16(nRx2N)	8
32x32(nLx2N)	2	16x16(2NxnU)	8
32x32(nRx2N)	2	16x16(2NxnD)	8
32x32(2NxnU)	2	8x8(2Nx2N)	16
32x32(2NxnD)	2	8x8(2NxN)	32
16x16(2Nx2N)	4	8x8(Nx2N)	32
16x16(2NxN)	8	**TOTAL**	**165**

muestra en la Figura 6. Por ejemplo, en el bloque 2x2, se tiene el resultado de 4 sumas de diferencias absolutas correspondientes a los valores SAD de las particiones NxN del CU de 32x32 y 2Nx2N del CU de 16x16.

Como se ha mencionado anteriormente en la Sección II, toda la arquitectura se ha segmentado completamente, y en el caso de este bloque, con solo 10 ciclos de retardo, este módulo entregará cada ciclo de reloj un registro con 165 SADs al siguiente bloque comparador.

V. Bloque comparador

La función final de este bloque de comparadores es la de almacenar los valores mínimos de SAD para cada partición del CU con sus correspondientes MVs (posiciones en el área de búsqueda).

En cada ciclo de reloj se comparan todos los SADs procedentes del árbol de sumadores, con los SADs anteriormente encontrados en ciclos de reloj previos, quedándose con los valores de SAD mínimos (los que minimizan la cantidad de información residual) y actualizando en los registros correspondientes el MV. Tras recibir y procesar el último registro de 165 SADs del árbol de sumadores, el comparador tendrá los SADs mínimos de cada partición del CTU con los MVs correspondientes.

VI. Resultados

Este trabajo ha sido diseñado en VHDL y ha sido sintetizado, simulado e implementado en la FPGA Zynq-7000 de la familia Xilinx, XC7Z020-1CLG484C [7]. Las herramientas utilizadas para obtener los resultados de frecuencia máxima, latencia del diseño y área ocupada en el dispositivo han sido las herramientas de Xilinx: ISE Design Suite 14.7, ISIM y Vivado 2013.4.

El objetivo ha sido lograr la menor latencia posible en la obtención de los SADs mínimos y MVs asociados, dando uso al gran potencial de un dispositivo reconfigurable y utilizando al máximo los recursos disponibles en esta FPGA. Por ello, como se ha comentado en la Sección II, tras los ciclos necesarios de carga previa del área de búsqueda en memoria, en un solo ciclo de reloj, obtenemos 165 SADs, correspondientes a todos los CUs y sus 7 particiones, englobadas en un CTU para una posición de píxel concreta del área de búsqueda. En el siguiente ciclo, se vuelven a obtener 165 SADs correspondientes a la siguiente posición en el rango de búsqueda y se realiza una comparación entre estos SADs y los del ciclo anterior para quedarse con los mínimos y proporcionar los MVs asociados. Este procedimiento se repite para todas las posiciones dentro del área de búsqueda. Puesto que el tamaño del CTU en esta arquitectura es de 32x32 y se define el rango como el doble del tamaño del CTU, para recorrer todos los píxeles de la ventana de búsqueda (64x64), necesitamos 4096 ciclos para obtener los SADs mínimos y los MVs. Por tanto, se logra una latencia mínima de 4096, al ser capaz de realizar en un solo ciclo de reloj el cálculo de los todos los SADs centrados en una posición a nivel de píxel.

Los resultados de la arquitectura implementada en la FPGA, nos muestran una utilización eficiente del área disponible. Se hace uso de 40,455 F.F. (Flip-flop) de los 106,400 disponibles y de 49,450 LUTs de de las 53,200 disponibles. Además, la frecuencia de operación es de 140 MHz. Por tanto, teniendo en cuenta la frecuencia a la que puede trabajar la FPGA, los ciclos que se necesitan para obtener la estimación de movimiento y los píxeles procesados correspondientes a un CTU, la arquitectura presentada es capaz de procesar 32 fps a una resolución de vídeo Full-HD y 16 fps en 2K.

VII. Conclusiones

En este trabajo se ha presentado una arquitectura hardware para el cálculo paralelo de SADs en el estimador de movimiento de HEVC y ha sido prototipado en la FPGA de Xilinx ZC702 (XC7Z020-1CLG484CES). Esta arquitectura es altamente eficiente en términos de paralelismo y complejidad computacional. El sistema propuesto cálcula 165 valores SAD para tamaños de bloque desde 4x8 o 8x4 hasta 32x32 soportando todos los modos de particionado, incluidos los asimétricos, con una baja latencia, tan solo 4140 ciclos de reloj.

A la frecuencia de 140 MHz, el proceso hardware es capaz de codificar formatos de vídeo Full-HD a

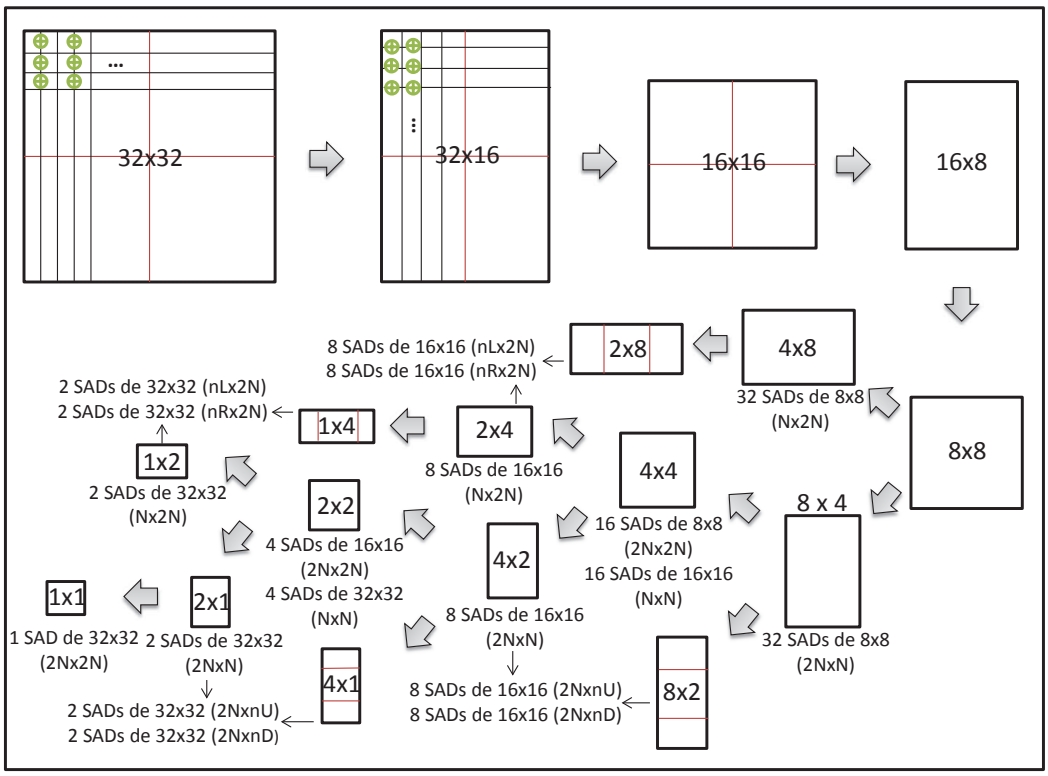


Fig. 6. Estructura del árbol de sumadores SAD

32 frames por segundo y 2K a 16, lo que representa una alta reducción de complejidad en el proceso de codificación de vídeo HEVC, logrando una codificación a tiempo real para contenidos de vídeo de alta definición.

Agradecimientos

Este trabajo ha sido financiado por el Ministerio de Economía y Competitividad a través del proyecto de I+D con referencia TIN2011-27543-CO3-03.

Referencias

[1] Ahmed Medhat, Ahmed Shalaby, Mohammed S.Sayed and Maha Elsabrouty, *A highly parallel sad architecture for motion estimation in hevc encoder*, in Consumer Electronics (ICCE) 2014 IEEE International Conference on, January 2014, pp. 187-188.

[2] Purnachand Nalluri, Luis Nero Alves and Antonio Navarro, *High speed sad architectures for variable block size motion estimation in hevc video coding*, in Image Processing (ICIP) 2014 IEEE International Conference on, October 2014, p. 12331237.

[3] Xu Yuan, Liu Jinsong, Gong Liwei, Zhang Zhi and Robert K.F.Teng, *A high performance vlsi architecture for Ïnteger motion estimation in hevc*, in IEEE 10th International Conference on ASIC (ASICON), October 2013, pp. 1-4.

[4] Wajdi Elhamzi, Julien Dubois, Johel Miteran and Mohamed Atri, *An efficient low-cost fpga implementation of a configurable motion estimation for h.264 video coding*, Journal of real-time image processing, vol. 9, no. 1, pp. 1930, 2014.

[5] J.Byun, Y.Jung and J.Kim, *Design of integer motion estimator of hevc for asymmetric motion-partitioning mode and 4k-uhd*, Electronics Letters, vol. 49, no. 18, pp. 1142-1143, 2013.

[6] Ching-Yeh Chen, Shao-Yi Chien, Yu-Wen Huang, Tung-Chien Chen, Tu-Chih Wang and Liang-Gee Chen, *Analysis and architecture design of variable block-size motion estimation for h.264/avc*, Circuits and Systems I: Regular Papers IEEE Transactions on, vol. 53, no. 3, pp. 578593, 2006.

[7] 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