PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ

FACULTAD DE CIENCIAS E INGENIERÍA

DISEÑO DE UNA ARQUITECTURA PARA LA INTERPOLACIÓN DE QUARTER-PIXEL PARA ESTIMACIÓN DE MOVIMIENTO SEGÚN EL FORMATO H.264/AVC EMPLEADO EN EL ESTÁNDAR SBTVD DE TELEVISIÓN DIGITAL TERRESTRE

Tesis para optar el Título de Ingeniero Electrónico, que presenta el bachiller:

Ernesto Cristopher Villegas Castillo

ASESORES:

Dr. Carlos Silva Cárdenas

MSc. Mario Andrés Raffo Jara

Lima, AGOSTO del 2011

RESUMEN

La reciente adopción del estándar de transmisión Japonés-Brasileño de TV Digital (SBTVD-T) por parte del gobierno peruano ha motivado a realizar investigaciones en torno a este estándar por su naturaleza de "estándar abierto" permitiendo cooperar con un aporte significativo para su desarrollo. Uno de los campos más interesantes en torno al SBTVD-T es el formato de compresión de video digital en el cual se basan los codificadores/decodificacores (CODEC's).

Los CODEC's del estándar SBTVD-T utilizan el formato de compresión H.264/AVC, desarrollado por el *Joint Video Team* (JVT), el cual posee mayor tasa de compresión en comparación con sus predecesores debido a la alta complejidad computacional que presentan sus algoritmos.

El presente trabajo de tesis trata sobre el módulo de Estimación de Movimiento que forma parte del proceso de Inter-Predicción del Codificador H.264/AVC, el cual presenta la mayor complejidad computacional de todos los procesos del Codificador H.264/AVC. Para el presente trabajo se desarrolló este módulo tomando en cuenta una de las principales innovaciones del formato H.264/AVC: el algoritmo de Estimación de Movimiento Fraccional con precisión Quarter-Pixel o 0.25 píxeles.

El objetivo del presente trabajo es aplicar este algoritmo para transmisión de video digital en tiempo real considerando que será utilizado para plataformas de dispositivos portátiles cuyas características buscan reducir el consumo de energía y el espacio de hardware.

Este algoritmo fue implementado en una aplicación en el entorno de programación MATLAB®, en base a un software de referencia disponible en el portal del grupo que los desarrolló, cuyos resultados se contrastaron con los obtenidos por la simulación de la arquitectura hardware.

Posteriormente se diseño la arquitectura en base a artículos revisados para luego plantear modificaciones que mejoren la frecuencia de procesamiento y la optimización de la cantidad de recursos lógicos requeridos. La arquitectura fue descrita en el lenguaje de descripción de hardware VHDL, sintetizada para los dispositivos FPGA de la familia Cyclone II y Stratix II de la compañía Altera® y se realizó la verificación funcional por medio de Testbenchs utilizando la herramienta ModelSim de ALTERA.

De los resultados de la síntesis de la arquitectura se obtuvo la frecuencia de operación y por simulación se verificó las cantidades de ciclos de reloj por operación, con lo que se pudo fundamentar que la arquitectura diseñada para ser implementada en un FPGA de la familia Cyclone II de la compañía ALTERA es capaz de procesar secuencias de video HDTV (1920x1080 píxeles) a una tasa de 30 cuadros por segundo, es decir en tiempo real.

ÍNDICE DE CAPÍTULOS

Agradecimientos

Quiero dedicar este trabajo de tesis de culminación de mi vida universitaria a todos los seres cuyo ejemplo, preocupación, cariño y apoyo incondicional me ayudaron a creer en mí para poder alcanzar todas mis metas. Siempre estarán en mi corazón y mente.

En primer lugar quiero agradecer a Dios y a su hijo Jesucristo por responder siempre a mis oraciones en los momentos en que necesite de su ayuda y por poner en mi camino a muchas personas que me ayudaron en momentos cruciales y que sirvieron de ejemplo para mi vida.

Asimismo también quiero agradecer a mi adorada madre Magda por todo el amor que siempre me ha dado y por inculcarme los valores que han formado a mi persona. A mí querido padre Ángel por todo su amor y cariño incondicional y a mis queridos hermanos Ángel Jesús y Alexis por su amor y preocupación hacia mi persona durante toda nuestra vida.

Un agradecimiento muy especial a mis queridos tíos Blanca Luz y Oscar y sus hijas Claudia y Karen por haberme acogido en su hogar durante estos años de vida universitaria, en donde pude encontrar comprensión, preocupación y cariño sincero. A mi mamita Sucila por ser como una madre, a mi primo Mao por ser como mi hermano cuya compañía siempre fue agradable en momentos de soledad y a mi *mama* Elvita e hijos por acogerme en su hogar durante mis primeros días en Lima.

También quiero agradecer mi asesor el Dr. Carlos Silva por su amistad, el apoyo incondicional y por la confianza brindada hacia mi persona para el desarrollo de este trabajo.

A mi asesor Mario Raffo en quien encontré un gran amigo y maestro por todos sus buenos concejos que me han ayudado a crecer como persona.

A mis amigos José Tafur, Christiam Rojas, Dino Morales, Rubén Alvarado (Chuven), Christian Chacon, Kelly Tafur, Ana Fuentes, Gisela Hurtado, Leslie Chávez y demás amigos que tuve la oportunidad de conocer por todos los buenos momentos de amistad que compartimos durante estos años.

A mis amigos del Grupo de Microelectrónica (GμE): Henry Block, José Quenta, Jorge Tonfat, José Alcántara, Joel Muñoz, César Vásquez, Edward Mitaac, Christian Cano y demás miembros del grupo por su sincera amistad, por compartir sus conocimientos para el desarrollo de este trabajo y por mostrarme el camino hacia mi verdadera vocación.

Asimismo quiero agradecer a mis queridos amigos del colegio República de Panamá José Dávila, Victor Ávalos y Franklin Cruz por los buenos momentos de amistad que pasamos juntos durante muchos años. Siempre los recordaré durante toda mi vida.

Finalmente quiero dedicar este trabajo a la memoria de mi primo hermano Ronald Ivan Matta Castillo (*Toto*) por el cariño sincero que siempre tuvo conmigo y a quien no tuve la oportunidad de despedir en su partida, Dios lo tenga en su gloria.

Loa del estudio

¡Estudia lo elemental! Para aquellos
cuya hora ha llegado
no es nunca demasiado tarde.
¡Estudia el «abc»! No basta, pero
estúdialo, ¡No te canses!
¡Empieza! ¡Tú tienes que saberlo todo!
Estás llamado a ser un dirigente.

¡Estudia, hombre en el asilo!
¡Estudia, hombre en la cárcel!
¡Estudia, mujer en la cocina!
¡Estudia, sexagenario!
Estás llamado a ser un dirigente.

¡Asiste a la escuela, desamparado!
¡Persigue el saber, muerto de frío!
empuña el libro, hambriento! ¡Es un arma!
Estás llamado a ser un dirigente.

No temas preguntar, compañero!
No te dejes convencer!
¡Compruébalo tú mismo!
Lo no sabes por ti,
No lo sabes
Repasa la cuenta,
Tu tienes que pagarla.
Apunta con tu dedo a cada cosa
Y pregunta: «Y esto, ¿de qué?»
Estás llamado a ser un dirigente.

(Bertolt Brecht)

Introducción

La Televisión Digital Terrestre (TDT) es el resultado de la revolución tecnológica que se ha producido en los últimos años, debido a grandes innovaciones en los campos de Procesamiento Digital de Señales e Imágenes y en el de Microelectrónica. Estos avances sumados a las tecnologías de transmisión de Telecomunicaciones hacen posible la transmisión de datos digitales con un ancho de banda soportado por las diversas redes de transmisión con la suficiente robustez ante errores y pérdidas de información, además de un procesamiento de datos en tiempo real.

En el Perú, la Comisión Multisectorial encargada de sugerir el nuevo estándar de transmisión de Televisión Digital Terrestre, optó por el SBTVD-T (Japonés-Brasileño) en Abril, 2009, debido a que es un estándar abierto a modificaciones, por lo cual el Perú tendría la oportunidad de contribuir con algún aporte significativo. El SBTVD-T es una modificación del estándar ISDBT-T (Japonés) por parte de un grupo de investigadores del Ministerio de Comunicaciones Brasileño. Una de las modificaciones fue el cambio del formato de compresión de video digital MPEG-2 a H.264/AVC o MPEG-4 Parte-10, el cual presenta características nuevas con respecto a sus antecesores, haciendo más eficiente la codificación, permitiendo una buena calidad de imagen y a la vez menor cantidad de bytes de información.

El presente trabajo de tesis se enfoca en el proceso de Estimación de Movimiento el cual es parte del proceso de Inter-Predicción del Codificador H.264/AVC, del mismo se pretende obtener una arquitectura *hardware* para poder ser utilizado en una aplicación en tiempo real como lo es la transmisión de video en Televisión Digital Terrestre (TDT).

La estructura del texto de esta propuesta de tesis está organizada de la siguiente manera: en el capítulo uno se presenta una introducción al formato H.264/AVC donde se explica la problemática, en el capítulo 2 se aborda la hipótesis de solución y los conceptos sobre Estimación de Movimiento. En el capítulo 3 se presentan las arquitecturas *hardware* desarrolladas. Finalmente, en el capítulo 4, se presentan los resultados obtenidos de las implementaciones en *software* y *hardware* del algoritmo, las conclusiones del presente trabajo y recomendaciones para trabajos futuros.

Capítulo 1

Compresión de Video Digital en el formato H.264/AVC

1.1 Declaración de la Problemática:

Las razones de ser de la Televisión Digital radican en los requerimientos de los gobiernos de los países sobre la reutilización del espectro UHF, así como los requerimientos de los nuevos dispositivos electrónicos receptores, y la oferta de nuevos servicios adicionales por parte de las emisoras de TV para los televidentes además de la demanda de estos por una mejor calidad de imagen. Sin embargo estas nuevas demandas requieren el uso de grandes unidades de almacenamiento y memoria así como altas tasas de transmisión de datos, lo que resulta complicado y poco eficiente para ser transmitido por los canales de comunicación actuales.

Por ejemplo un video con una resolución de 720x480 pixeles a 30 cuadros por segundo (usado en SDTV y en DVDs) que utiliza 24 bits por pixel, sin comprimir tendría una tasa de transmisión próxima a 249 millones de bits por segundo (249 Mbps). Además para almacenar una secuencia de corta duración, de 10 minutos, serían necesarios 19 billones[1] de bytes (19GB). Para videos con resolución de 1920x1080 pixeles a 30 cuadros por segundo (usados en HDTV), con 24 bits por pixel, una tasa de transmisión se eleva a 1.5 billones de bits por segundo (1.5 Gbps) y serian necesarios 112 billones[2] de bytes (112 GB) para almacenar un video de 10 minutos [1].

Fue por lo expuesto en el párrafo anterior que se desarrollaron los formatos de compresión de video, los cuales son un conjunto de técnicas que aprovechan la característica más importante de las secuencias de video: la redundancia. El objetivo del estudio de la compresión de video es desarrollar técnicas que permitan la máxima eliminación de los datos innecesarios para obtener una secuencia de video con menor tamaño que el original [1].

Se ha desarrollado distintos formatos de compresión de video como el ITU-T H.261, H.262 (MPEG-2), y el H.263 (y sus mejoras como H.263+ y el H.263++). Sin embargo aún se siguen haciendo esfuerzos por mejorar la eficiencia de la codificación, así como elevar la robustez a la pérdida de información y a los errores de transmisión, además de buscar la compatibilidad con las diversas redes de comunicación existentes [2].

1 El término billón se refiere en Castellano según RAE, a un millón de millones (10^12) y difiere de la definición en Inglés y en Portugués que es mil millones (10^9) por lo que se hace esta aclaración ya que los prefijos del SI (sistema Internacional de unidades) son acordes con el idioma Inglés y el ejemplo fue tomado de la referencia [1] que está escrita en el idioma Portugués.

El más reciente estándar de compresión de video es el H.264/AVC desarrollado por el *Joint Video Team* (JVT) conformado por el ITU-T *Video Coding Experts Groups* (VCEG) y el ISO/IEC *Moving Picture Experts Group* (MPEG) [3]. Este formato presenta nuevas características sobre sus predecesores, los formatos MPEG-1 o H.261 [4] y MPEG-2 o H.262 [5], que mejoran la tasa de compresión [6] y la eficiencia por tasa de distorsión [2]. Estas mejoras se deben al aumento de la complejidad computacional de los algoritmos utilizados en los *CODECs* bajo este formato.

Existen implementaciones en *software* de los *CODECs* H.264/AVC sin embargo el enorme interés comercial en este estándar para poder ser implementado en los nuevos equipos de uso personal como laptops, celulares, y demás, impulsa a varios equipos de investigadores a desarrollar el estándar usando optimizaciones algorítmicas y/o implementación en *hardware* para cumplir con los requerimientos de operación [1]. Uno de los más recientes trabajos sobre la implementación del *CODEC* H.264/AVC en *software* se encuentra en la referencia [7], donde utilizan procesadores de arquitectura multi-núcleos para aprovechar el paralelismo de operaciones, pero sólo logran un procesamiento en tiempo real para secuencias de resolución 4CIF.

Asimismo, hoy en día las tendencias exigen compresión y decodificación de video digital como característica primordial en los nuevos equipos portátiles debido a la alta demanda de información visual por parte de los usuarios. Como un ejemplo de estos equipos se pueden citar: computadoras personales, laptops, celulares, iPods, televisores de Alta Definición (HD), reproductores de DVDs, cámaras o filmadoras digitales entre otros [1].

En la actualidad, los Sistemas de Transmisión ATSC (EE. UU.) [8], DVB-T/H (Europa) [9], DMB-T/H (China) [10] & ISDB-T (Japón) [11] utilizan el estándar de compresión de video MPEG-2 [5] mientras que el SBTVD-T (Japonés-Brasileño) hizo una modificación en su formato de compresión de video adoptando el formato H.264/AVC MPEG-4 Part-10 [12]. El Gobierno Peruano mediante la Resolución Suprema Nº 019-2009-MTC del 24 de abril del 2009, tomó la decisión de adoptar el estándar japonés-brasileño, *Integrated Services Digital Broadcasting Terrestrial (ISDBT-T)* con adaptaciones brasileñas, con base en la propuesta de la Comisión Multisectorial encargada de recomendar al Ministerio de Transportes y Comunicaciones (MTC) el estándar a utilizarse en Perú [13]. Dentro de las propuestas que los desarrolladores del sistema ISDB-T con innovaciones brasileñas (SBTVD-T) hacían a la Comisión Multisectorial [14], estaba la cooperación internacional en el desarrollo de este estándar y la transferencia tecnológica [15], la cual se adecuaba a los intereses del gobierno peruano.

Con la característica del estándar SBTVD-T de cooperación internacional entre los países que la adopten, está la posibilidad de desarrollar un aporte significativo al estándar, es por eso que se plantea como problemática el generar un aporte en torno al formato de compresión H.264/AVC para aplicaciones que requieran procesamiento en tiempo real en equipos portátiles de uso personal donde el consumo de energía es una característica que se desea reducir, para aumentar la autonomía de los mismos.

1.2 Declaración del Marco Problemático:

En la actualidad existen muchos trabajos publicados sobre la mejora de la complejidad computacional del estándar H.264/AVC implementados en *software* y/o *hardware*, como la tesis doctoral del Dr. Agostini [1]. La variedad de aplicaciones (entre ellas TDT) y la diversidad de redes de transmisión hacen que el estudio en esta área continúe progresando ante la necesidad de flexibilidad y personalización [2].

La optimización del formato H.264/AVC se basa en el aumento de la complejidad de los algoritmos de cada uno de los módulos que lo conforman, lo que implica un aumento de la capacidad computacional. Debido a las limitaciones de un *software*, que es un sistema básicamente secuencial, no siempre es posible alcanzar los requisitos de desempeño del sistema, necesitando de aceleradores *hardware* [16].

El tener una implementación en *hardware*, la cual es un circuito dedicado que funciona como acelerador *hardware*, permite aprovechar el paralelismo de operaciones respecto a la solución *software* para poder aumentar la frecuencia de operación del sistema y cumplir con las restricciones de respuesta de tiempo del SBTVD-T.

Capítulo 2

Estudio del Algoritmo de Interpolación Quarter-Pixel para Estimación de Movimiento en el formato H.264/AVC

2.1 Formato de Compresión de Video Digital H.264/AVC:

Un archivo de video o imagen es representada digitalmente en base al espacio de colores que se está utilizando el cual es muy importante ya que la eficiencia de codificación depende de ésta [1]. Los espacios de colores más conocidos son: RGB, HSI e YCbCr. Para la compresión de video digital se utiliza el espacio de colores YCbCr, ya que el sistema visual humano es mucho más sensible a la información de luminancia (Y) que a la de crominancia (CbCr). De esa manera los formatos de compresión aprovechan esta característica humana para aumentar la eficiencia de codificación a través de la reducción de la tasa de muestreo de las componentes de crominancia en relación a las componentes de luminancia [17].

La forma en que se relacionan las componentes de luminancia con los de crominancia es a través de los formatos de submuestreo, como: 4:4:4, 4:2:2 o 4:2:0. En el primer formato las muestras de luminancia y crominancia poseen la misma resolución, en el segundo caso las muestras de crominancia posee la misma resolución vertical que las muestras de luminancia pero la mitad de resolución horizontal y en el caso del formato 4:2:0 las muestras de crominancia poseen la mitad de resolución horizontal y vertical de las muestras de luminancia, es decir por cuatro muestras de Y solo habrán una muestra de Cr y una muestra de Cb, lo cual implica una tasa de compresión del 50% con respecto al formato RGB [1]. En el caso del formato H.264/AVC [12] se utiliza el espacio de colores YCbCr con formato 4:2:0 de preferencia por las razones ya expuestas.

Otra consideración para la compresión de video es la redundancia que pueda presentar, básicamente existen 3 tipos de redundancia [1]:

- **Redundancia Espacial:** Esta correlación puede ser percibida en dominio espacial y temporal. Por ejemplo en el dominio espacial, existe redundancia cuando un conjunto de píxeles vecinos poseen valores semejantes, para lo cual los formatos de compresión utilizan la codificación Intra-Cuadro.
- **Redundancia Temporal:** Esta correlación puede ser percibida entre los cuadros que se encuentran próximos entre sí en el tiempo. Se puede llegar a conseguir altas tasas de compresión si se hace una exploración eficiente de la redundancia [1], al cual se le denomina codificación Inter-Cuadros.

- **Redundancia Entrópica:** Este tipo de redundancia está relacionada con las probabilidades de ocurrencia de los símbolos de video codificados, es decir mientras mayor sea la cantidad de información nueva, disminuirá la probabilidad de aumento de la aparición de un símbolo.

Hasta ahora se ha explicado conceptos importantes sobre compresión de video, desde ahora el enfoque será en el formato de compresión H.264/AVC, el cual fue desarrollado por el *Joint Video Team* (JVT) conformado por el ITU-T *Video Coding Experts Groups* (VCEG) y el ISO/IEC *Moving Picture Experts Group* (MPEG) [3]. El diagrama de bloques del codificador H.264/AVC se muestra en la **Figura 1**.

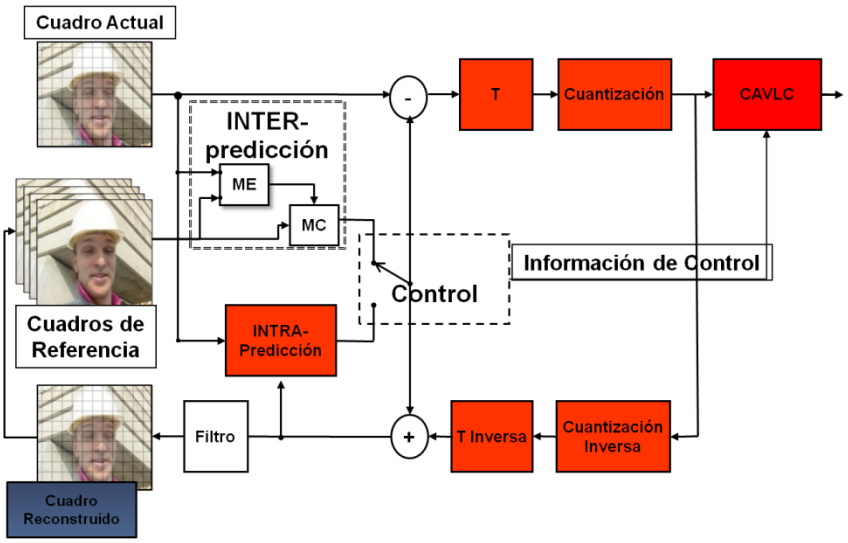


Fig. 1 Diagrama de bloques de un codificador H.264/AVC [1]

El codificador H.264/AVC tiene un módulo de Inter-Predicción, el cual está compuesto por el proceso de Estimación de Movimiento (ME, por sus siglas en inglés) y Compensación de Movimiento (MC, por sus siglas en inglés), que en conjunto se encargan de disminuir la redundancia temporal, por medio de una comparación de bloques del cuadro actual con respecto a los bloques de un cuadro de referencia.

El módulo Intra-Predicción se encarga de atenuar la redundancia espacial, utilizando solamente la información del cuadro actual. El tipo de predicción que se desee realizar es seleccionado por medio del circuito de control mostrado en la **Figura 1**, para luego ser restado de la imagen original [1].

Luego el residuo es enviado al bloque de Transformadas (T), cuya función es la de disminuir la redundancia espacial en el dominio de la frecuencia. Las transformadas que se aplican son:

la Transformada Discreta de Cosenos Bidimensional (DCT-2D) y la Transformada Hadamard. El bloque de Cuantización se encarga de atenuar la redundancia espacial. Finalmente el bloque de Codificación de Entropía (CAVLC, *Context-Adaptive Variable Length Coding*, por sus siglas en inglés) tiene por objetivo disminuir la redundancia entrópica.

Asimismo también se utilizan los bloques de Cuantización y de Transformada inversa para obtener un cuadro reconstruido después de pasar por el filtro de desbloqueo o filtro de *loop*, utilizado para atenuar el Efecto de Bloqueo[2] [1], por el mismo efecto de división en bloques. La división en bloques o macrobloques se explicará posteriormente en esta tesis.

El foco de este trabajo será el módulo de Estimación de Movimiento (ME), el cual irá desde el estudio del algoritmo, para luego validarlo por medio de una aplicación en *software* pudiendo así diseñar una arquitectura descrita en Lenguaje de Descripción de Hardware VHDL y finalmente comprobar y validar su funcionamiento por medio de herramientas de simulación.

2.2 Estimación de Movimiento:

El proceso de Estimación de Movimiento (ME) consiste en la comparación de una región de MxN pixeles del cuadro actual con algunas o todas las posibles regiones de M×N pixeles dentro de un área de búsqueda del cuadro de referencia. A la región de MxN pixeles se le denomina bloque, además se denomina bloque actual o de referencia dependiendo del tipo de cuadro al que pertenece. El área de búsqueda contiene a todos los posibles bloques de referencia que se pueden formar con MxN pixeles a los que se denomina bloques candidatos, así como está centrada en la misma posición que el bloque actual con el cual se hace la comparación. El resultado obtenido de este proceso es el bloque de referencia que mejor se aproxima al bloque actual. A este resultado se le denomina "*best match*" respetando la terminología en inglés [17]. En la **Figura 2** se puede apreciar el proceso de ME hasta obtener el "*best match*" como se explicó anteriormente.

[2] El Efecto de Bloqueo se refiere a los bordes que se generan alrededor de cada bloque de la imagen reconstruida.

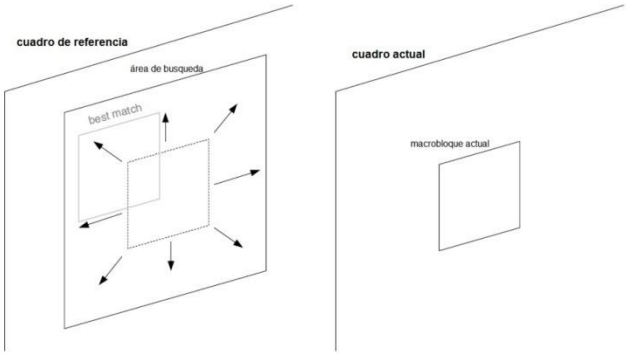


Fig. 2 Proceso de Estimación de Movimiento. Izquierda: Cuadro de Referencias. Derecha: Cuadro Actual [17].

El criterio utilizado para encontrar el bloque de referencia con el "*best match*" es la energía residual que se obtiene de restar el bloque actual con un bloque candidato del área de búsqueda. El bloque que obtenga la menor energía residual tendrá el *"best match"*, lo cual significa que se encontró al bloque con mayor similitud al bloque actual. Además de lo anterior, dentro del proceso, se calcula el desplazamiento de la posición del bloque actual con respecto a la posición del bloque candidato obteniendo un parámetro denominado vector de movimiento (MV), el cual se usará en el proceso de Compensación de Movimiento, con el cual se completa todo el proceso de Inter-Predicción. Tanto el MV como el bloque candidato con la menor energía residual, son codificados por los procesos de Transformadas Directas, Cuantización y Entropía. Luego esta información codificada puede ser recibida por cualquier equipo que contenga un decodificador bajo el mismo formato H.264/AVC. El decodificador utiliza el MV recibido para reconstruir el bloque predicho y formar el cuadro codificado [17].

En el formato H.264, un cuadro de video es dividido en bloques de un tamaño básico con $M = N = 16$, es decir de 16x16 pixeles. A esta región se le denomina macrobloque (MB) que se utiliza como unidad base para el proceso de codificación/decodificación de video [12]. Si se emplea el formato de video YCbCr con relación de 4:2:0[3], un MB estaría compuesto por un bloque de 16 × 16 pixeles representado por 256 muestras de Luminancia Y (dispuestos en cuatro bloques de 8x8 muestras), 64 muestras de Crominancia Azul Cb (un bloque de 8 × 8) y 64 muestras de Crominancia Roja Cr (8×8 muestras), dando un total de seis bloques de 8 × 8 [17]. En la **Figura 3** [17] se muestra como está compuesto un MB de 16x16 píxeles en el formato YCbCr 4:2:0.

[3] Concepto explicado en la sección 2.1

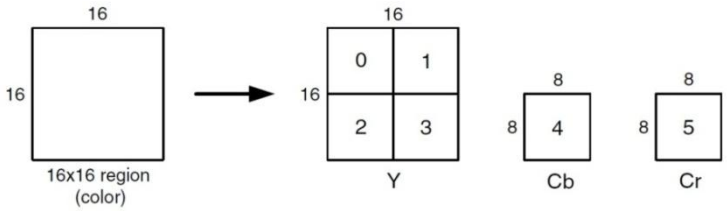


Fig. 3 Componentes de Luminancia (Y), Crominancia Azul (Cb) y Crominancia Roja (Cr) en el formato 4:2:0.

En la componente de Luminancia, cada MB de 16 × 16 muestras puede dividirse en sub-MBs, ya sea como una partición de 16 x 16, dos de 16 × 8, dos de 8 × 16 o cuatro particiones de 8 × 8. Si se elije el modo de 8x8, cada uno de los cuatro sub-MB 8 × 8 (que forman el MB de 16x16) se puede dividir en otros cuatro modos, ya sea como una sub-partición de 8x8, dos sub-particiones de 8 × 4, dos 4 × 8 o cuatro de 4 × 4 píxeles.

El uso de bloques de tamaños variables para ME es otra innovación del formato H.264/AVC, sin embargo en el presente trabajo de tesis no se tomará en cuenta esta característica y se trabajará sobre bloques de 8x8 píxeles. Se escogió esta característica ya que en trabajos previos [18] [19] se realizaron pruebas de codificación con varias secuencias de video de distinta resolución con el *software* de referencia H.264/AVC [20] y se obtuvo que el 94.75% de los bloques procesados tuvieron un tamaño mayor o igual a 8x8 píxeles [18].

El módulo de ME se caracteriza por estar presente sólo en el codificador H.264/AVC, así como poseer la mayor complejidad computacional de todos los módulos que forman parte del codificador [21]. Este alto coste computacional se debe a las innovaciones propuestas por los desarrolladores[4] de la norma internacional ITU-T H.264/AVC [12] con el objetivo de lograr altas tasas de compresión de video [2].

Las principales innovaciones del formato H.264/AVC de compresión de video con respecto al proceso de ME son: la selección del tamaño de las particiones de un MB, como se menciono anteriormente, además de la generación de MVs con valores enteros y fraccionarios con una precisión de 1/2 y 1/4 de pixel, así como el uso de múltiples cuadros de referencia previamente decodificados [12]. La complejidad computacional que presenta este módulo se debe a las características anteriormente descritas, sin embargo, debido a que este módulo sólo está presente en el codificador, se tiene la ventaja de poder realizar el proceso dejando de lado algunas de las características ya mencionadas, dependiendo de las necesidades de la aplicación en cuestión [1].

[4] Joint Video Team (JVT) formado por el Video Coding Experts Group (VCEG) ITU-T SG16 Q.6 y el Moving Picture Experts Group (MPEG) ISO/IEC JTC 1/SC 29/WG 11 [2]

De lo anteriormente expuesto el foco de este trabajo será diseñar una arquitectura *hardware* del módulo de ME para MB de 8x8 pixeles con precisión de 0.25 pixel para los MVs.

A. Estimación de Movimiento Entera (IME)

Como se ha explicado hasta el momento el proceso de ME contribuye en la compresión de datos de video, expresando la información como MV en lugar de toda la información correspondiente a los pixeles que forman parte de un MB. Asimismo se explicó que incrementando la precisión del movimiento de un MB se obtienen mejores tasas de compresión sin perder detalles en el movimiento que puede suceder entre los cuadros que forman una cadena de video.

El primer proceso que se realiza en ME es la estimación de movimiento de los pixeles enteros a lo que se denomina Estimación de Movimiento Entera (IME). El siguiente proceso es la Estimación de Movimiento Fraccional[5] (FME) donde se calcula el desplazamiento con precisión de Half-Pixel y/o Quarter-Pixel alrededor del MB con el *"best match"* obtenido del proceso anterior.

- **Algoritmos de Búsqueda para Estimación de Movimiento Entera:**

Según se explicó en el ítem anterior, para poder encontrar el "*best match*" en el área de búsqueda se debe efectuar la evaluación de la energía residual para todos los posibles bloques en el área de búsqueda del cuadro de referencia. Existen tres formas de medir la energía residual: Suma de Diferencias Absolutas (SAD, *Sum of Absolute Diferences*), Error Cuadrático Medio (MSE, *Mean Squared Error*) y Error Absoluto Medio (MAE, *Mean Absolute Error*) expresados por las ecuaciones (1), (2) y (3) respectivamente [17]. Todas las fórmulas se basan en la resta de los valores de los pixeles del MB actual (C_{ij}) con los del MB de referencia (R_{ij}) con la misma posición relativa

$$MSE = \frac{1}{N^2}\sum_{i=0}^{N-1}\sum_{j=0}^{N-1}(C_{ij} - R_{ij})^2 \qquad \textbf{(1)}$$

$$MAE = \frac{1}{N^2}\sum_{i=0}^{N-1}\sum_{j=0}^{N-1}\left|C_{ij} - R_{ij}\right| \qquad \textbf{(2)}$$

$$SAD = \sum_{i=0}^{N-1}\sum_{j=0}^{N-1}\left|C_{ij} - R_{ij}\right| \qquad \textbf{(3)}$$

SAD es la medida más ampliamente utilizada de la energía residual por la simplicidad del coste computacional que demanda y a la calidad de los MVs generados, según la referencia [17], por ello, en el presente trabajo se utilizará este criterio de búsqueda.

[5] Más adelante se abordará con mayor detalle este concepto.

El algoritmo óptimo de búsqueda *Full Search* (FS) consiste en aplicar un criterio de búsqueda como el SAD (*Sum of Abbsolute Differences*) a cada posible bloque en el área de búsqueda del cuadro de referencia. Para poder explicar mejor en qué consiste la IME utilizando el algoritmo *Full Search*, se empleará la **Figura 4**. En (a) se tiene un cuadro actual con un bloque de NxM píxeles, luego para poder encontrar el *"best match"* se tendrá que generar un área de búsqueda alrededor de la posición del bloque actual en el cuadro de referencia. El área de búsqueda se generará a partir del bloque de NxM píxeles pero expandiendo en p píxeles por cada borde como se muestra en la parte derecha de (a). Aplicando el algoritmo de búsqueda se puede encontrar el bloque con el *"best match"* cuya posición $(x+u, y+v)$ respecto a la posición del bloque actual (x, y) permite generar el MV movimiento (u, v). Las coordenadas del MV tienen el siguiente rango: $-p \leq u \leq p$ y $-p \leq v \leq p$ [22].

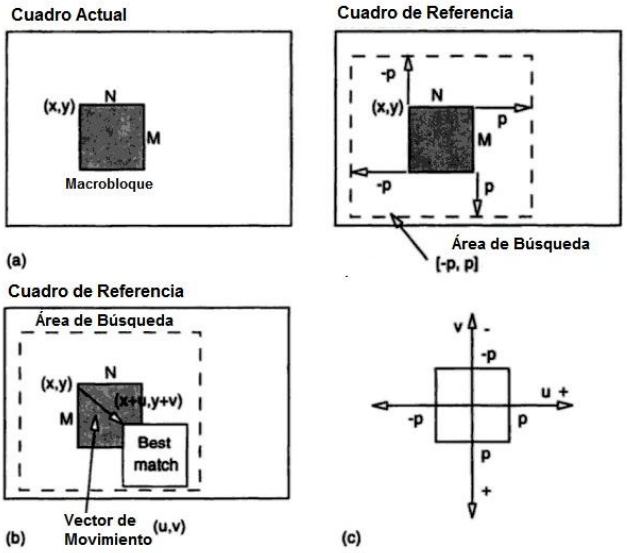


Fig. 4 Proceso de Estimación de Movimiento con algoritmo de búsqueda [22].

B. Estimación de Movimiento Fraccional (FME)

En el formato H.264/AVC, los movimientos que suceden de un cuadro para otro no están restringidos a posiciones enteras de pixel, se pueden utilizar MVs con valores fraccionarios de 0.5 y 0.25 pixeles [1]. El desplazamiento del MB o sub-bloque del cuadro de referencia con respecto al del cuadro actual se expresa por medio de un MV, cuya precisión es de ½ o ¼ de pixel para el componente de Luminancia (Y) y la mitad de precisión de pixel para los componentes de Crominancia (Cr ó Cb), es decir una precisión de ¼ o $\mathbf{1}/\mathbf{8}$ de pixel respectivamente. Esto se debe a que la resolución de las muestras de Cr y Cb es la mitad del valor respecto a la resolución de las muestras de Y, como fue explicado en la sección anterior. Sin embargo, las muestras en las posiciones fraccionarias o sub-muestras tanto para

Y como Cb & Cr deben de ser calculadas por medio de un algoritmo denominado Interpolación [17]. En la **Figura 5** se muestra un ejemplo de Estimación de Movimiento Fraccional (FME, por sus siglas en inglés) para un sub-bloque de 4x4 pixeles.

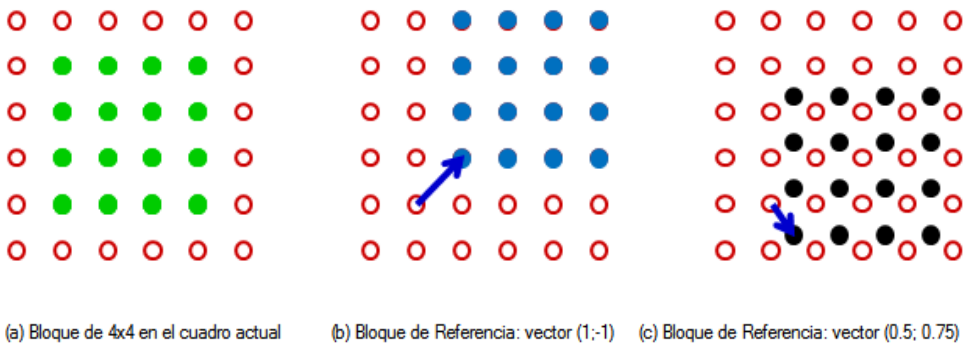


Fig. 5 Estimación de Movimiento Fraccional de un sub-bloque de 4x4 pixeles [17].

El sub-bloque del cuadro actual de 4x4 pixeles representado por los puntos verdes de la parte (a) de la **Figura 5** sirve como referencia para generar una región de búsqueda en el cuadro de referencia alrededor de la posición del bloque actual. Si las componentes verticales y horizontales del cuadro de referencia son números enteros entonces los pixeles del sub-bloque en el cuadro de referencia serán representados por los puntos azules como se muestra en la parte (b) de la **Figura 5**. Si el desplazamiento es fraccionario, se debe calcular por interpolación de los pixeles enteros (puntos blancos) los pixeles fraccionarios (puntos negros) los cuales son mostrados en la parte (c) de la **Figura 5** [17].

A continuación se profundizará más sobre los algoritmos de interpolación para precisión de 1/2 pixel (Half-Pixel) y ¼ de pixel (Quarter-Pixel) y los algoritmos de búsqueda para ME.

- **Interpolación para Precisión de Half-Pixel**

Para poder encontrar un bloque con precisión de Half-Pixel que posea el "*best match*" con respecto al macro bloque actual será necesario generar un macro bloque compuesto de Half-Pixels. La unidad de Interpolación Half-Pixels se encargará de generarlos a partir del MB compuesto por pixeles enteros. Para calcular el valor de Luminancia de un Half-Pixel adyacente a dos posiciones enteras, se aplica un Filtro FIR[6] a las muestras enteras de

6 El filtro de interpolación de 6-taps es relativamente complejo, pero produce un ajuste exacto de los datos de píxeles enteros y por lo tanto un buen rendimiento en el proceso de compensación de movimiento [17]. Asimismo la elección del tipo de filtro a utilizar fueron determinadas por los desarrolladores del formato H.264/AVC en [12].

Luminancia que forman una línea vertical u horizontal en el MB [12]. De la **Figura 6**, por ejemplo, para calcular el valor de luminancia del Half-Pixel "y", se tiene que calcular un valor intermedio "$y1$" aplicando el Filtro FIR (en este caso de 6-taps) a la línea de pixeles enteros formados desde la A hasta la F. El valor de luminancia (Y) de los pixeles varía desde 0 a 255. En **(4)** se muestra la ecuación para calcular el valor intermedio $y1$, luego en **(5)** se muestra el cálculo final para conseguir el valor y, donde se realiza una operación de "clipping"[7] para que el valor de luminancia (Y) del pixel este en el rango de 0-255 [12]. La ecuación completa del filtro FIR de 6-taps viene dado por **(6)**. El Filtro FIR de 6-taps tiene los siguientes pesos: (1/32, -5/32, 5/8, 5/8, -5/32, 1/32) [1].

$$y_1 = A - 5B + 20C + 20D - 5E + F \qquad (4)$$

$$y = Clip_{0-255}[(y_1 + 16) \gg 5] \qquad (5)$$

$$y = \frac{A-5B+20C+20D-5E+F}{32} \qquad (6)$$

En la **Figura 6** [18] se muestran a los pixeles enteros como pequeños cuadros de color gris con una letra mayúscula de referencia *(A, B ,C, D, E, F, K, L, H, N, O, P, U, S, I, J, Q, W, D, N, R & T*) y algunos de los Half-Pixel como cuadros blancos *(aa, bb, cc, dd, ee, ff, gg, hh, g, m, y, x & s*). Es importante notar que hay 3 tipos de Half-Pixels: los de Tipo H, que son aquellos que se calculan a partir de la línea de pixeles horizontales más cercanos a la posición del Half-Pixel, los de Tipo V que son aquellos que se calculan a partir de la línea de pixeles verticales más cercanos a la posición del Half-pixel y por último lo de Tipo D que son los Half-Pixels alineados diagonalmente con pixeles enteros, para lo cual se toma la línea horizontal de Half-Pixels más próxima.

En el ejemplo de la Figura 6 [18], el Half-Pixel *g* sería de Tipo H ya que es calculado a partir de la línea vertical de pixeles enteros *U, S, C, H, I & J*. Los Half-Pixels *cc, dd, g, m, ee & ff* también serían de Tipo H, *aa, bb, y, s, gg & hh* serían de Tipo V y *x* sería de Tipo D calculado a partir de la línea horizontal formada por *cc, dd, g, m, ee & ff*, los cuales son Tipo V.

[7] Clipping: operación que se le aplica a un número entero con la finalidad de limitar su valor al rango de 0-255 entero. Si el número excediera el valor de 255 o menor a 0, el resultado será 255 o 0 respectivamente.

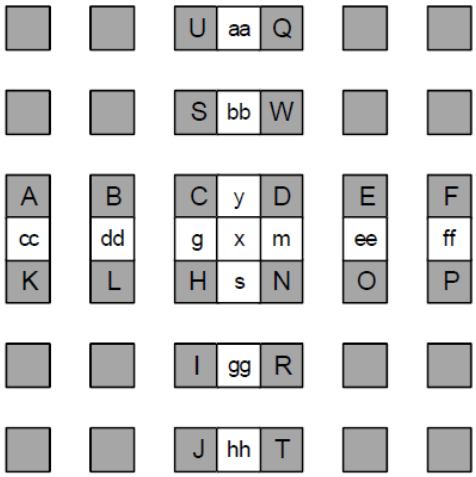


Fig. 6 MB de 6x6 píxeles. Pixeles enteros (cuadros grises). Half-Pixels (cuadros blancos) [18].

Es muy importante mencionar que para obtener una nueva área de búsqueda formada por Half-Pixels alrededor de un bloque de píxeles enteros se utilizan 3 pixeles adicionales por cada borde (superior, inferior, derecha e izquierda). En la Fig. 7 [18] se observa un área de Half-pixels obtenidos a partir del bloque de 4x4 pixeles para lo cual se utilizó 3 columnas adicionales de píxeles enteros por cada lado lateral y 3 filas en cada borde superior e inferior respectivamente, por ejemplo los Half-Pixels *h1, h6, h11 & h16* se obtienen a partir de la línea de pixeles enteros horizontal formado por 3 pixeles a la derecha (que forman parte del bloque de 4x4) y 3 pixeles a la izquierda los cuales forman parte de las 3 columnas adicionadas. Lo mismo se aplica a los Half-Pixels Tipo V y D.

En la nueva área de búsqueda hay dos posibles coincidencias con bloques del mismo tamaño formado por Half-Pixels de Tipo V, dos posibles coincidencias con el área formado por los de Tipo H y cuatro posibles coincidencias con el área formada por los de Tipo D [18].

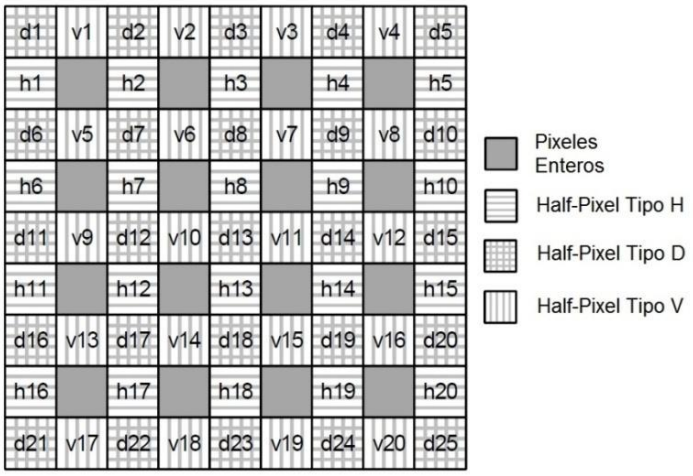


Fig. 7 Área de Half-Pixels Interpolados a partir de un bloque de 4x4 píxeles [18].

- **Interpolación con Precisión de Quarter-Pixel**

La precisión de Quarter-Pixel (0.25 píxel) es una de las nuevas características del formato H.264/AVC que permite aumentar aún más la tasa de compresión de video. Después del proceso de IME y FME con precisión Half-Pixel se compara el menor SAD obtenido de ambos procesos. Si el menor SAD obtenido por FME con precisión Half-Pixel es menor que el obtenido por IME se obtiene un MV desplazado en ±0.5 píxeles para ambas direcciones (x e y) obtenido del FME, caso contrario sólo se considera el MV obtenido del IME. Luego a partir del MB de referencia con la posición obtenida de los procesos anteriores, se aplicará un refinamiento para obtener una nueva área de búsqueda formada por Quarter-Pixels. Esta área es generada a partir del área del MB formado por píxeles enteros (encontrado por IME) y el área formada por Half-Pixels (generada por Interpolación del MB entero).

A este refinamiento se le denomina Interpolación Quarter-Pixel el cual consiste en aplicar un filtro bilineal a los píxeles y subpíxeles que forman parte del área generada por Interpolación Half-Pixel., siendo este filtro una simple media aritmética entre dos medios píxeles. En la **Figura 8** se muestran los tipos de Quarter-Pixel que se obtienen por la Interpolación y que generan una nueva área de búsqueda. Al igual que los Half-Pixels, tenemos los de Tipo H, V &[8] D. En el caso de Tipo H, sólo se utiliza los Half-Pixels adyacentes en una línea horizontal para calcular el nuevo sub-píxel, en este caso tenemos: *a, c, i & k*. En el caso de Tipo V es semejante al Tipo H pero en una línea vertical (*d, n, f & q*) y en el caso de los de tipo D, con los Half-Pixels que forman una diagonal (*e, g, p & r*).

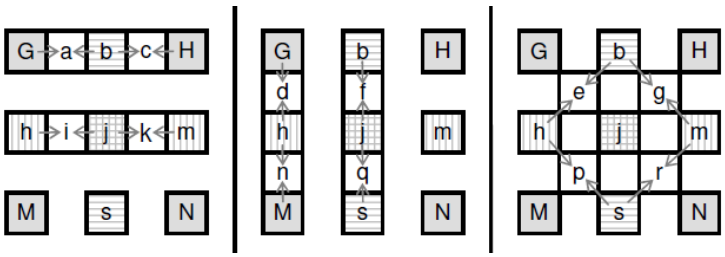


Fig. 8 Tipos de Quarter-Pixel obtenidos por Interpolación [1].

En (7) se muestra la ecuación del filtro bilineal, donde a es un Quarter-Pixel Tipo H y G & b son Half-Pixels [19]. Asimismo para los Tipo V y D, se muestra la aplicación del filtro en (8) y (9) respectivamente.

[8] Se usa "&" en vez de "y" para evitar errores de comprensión en la lectura debido a que las referencias son letras.

$$a = {}^{(G+b+1)}/_{2} \quad \textbf{(7)}$$

$$d = {}^{(G+h+1)}/_{2} \quad \textbf{(8)}$$

$$e = {}^{(h+b+1)}/_{2} \quad \textbf{(9)}$$

Finalmente en la **Figura 9** se muestra un área de búsqueda generada por Interpolación Half-Pixel y Quarter-Pixel alrededor de un bloque de 3x3 píxeles enteros.

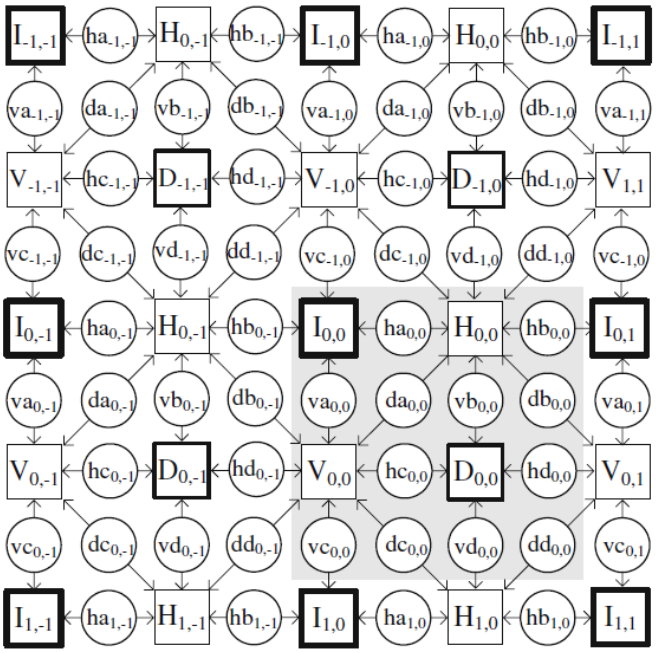


Fig. 9 Píxeles Enteros (I_{XY}, mostrado en cuadros resaltados), Half-pixels (H_{XY}, V_{XY} y D_{XY}, mostrado en cuadros) y Quarter-Pixels (ha_{XY}, hb_{XY}, hc_{XY}, hd_{XY}, va_{XY}, vb_{XY}, vc_{XY}, vd_{XY}, da_{XY}, db_{XY}, dc_{XY} y dd_{XY}) [23].

El proceso de búsqueda es similar a la efectuada en el proceso de interpolación Half-Pixel, ya que se utiliza el mismo criterio de comparación, SAD, para comparar el bloque actual con los 9 bloques candidatos formados por Quarter-Pixels en el proceso de Interpolación. El menor SAD obtenido de la búsqueda con precisión Quarter-Pixel se comparará con el menor SAD obtenido en el proceso anterior obteniendo el menor SAD de todo el proceso de ME con un MV de precisión de 0.25 píxeles, el cual será codificados junto el bloque residual obtenido por los demás procesos del codificador H.264/AVC.

Capítulo 3

Diseño de la Arquitectura Hardware del Algoritmo de Interpolación de Quarter-Pixel

3.1 Hipótesis de Solución:

- **Hipótesis Principal:**

  En base a la problemática y al marco teórico presentado en las secciones anteriores, el tener una implementación en *hardware*, el cual es un circuito dedicado que funciona como acelerador *hardware*, permite aprovechar el paralelismo de operaciones respecto a una implementación *software* para poder aumentar la frecuencia de operación del sistema y cumplir con las restricciones de respuesta de tiempo de transmisión de video digital en el SBTVD-T (30 cuadros por segundo).

  Una arquitectura hardware para la FME implementada en un FPGA tiene los recursos suficientes para una aplicación con alta complejidad computacional operando en tiempo real.

- **Hipótesis Secundarias:**
  - La arquitectura hardware obtenida por la síntesis comportamental del algoritmo, optimiza el tiempo de procesamiento aprovechando el alto grado de paralelismo presente en el flujo del algoritmo.
  - El resultado será una arquitectura flexible y adaptable a cualquier modificación, innovación y mejora del algoritmo de Estimación de Movimiento.

3.2 Descripción y Justificación de la Hipótesis:

Dentro de las soluciones hardware disponibles tenemos a los *Digital Signal Processors* (DSPs, por sus siglas en inglés) mostrado en la **Figura 10** [24] ya que la arquitectura interna que poseen está compuesta de al menos un arreglo de multiplicadores rápidos (16x16 bits hasta 24x24 bits de punto fijo o 32 bits de punto flotante) que mejora la frecuencia de procesamiento de un algoritmo DSP pero cuyos limitantes son: el ancho de bus de datos (8, 16, 32 y 64 bits) generando un cuello de botella en el procesamiento y la manera secuencial con la que realizan sus operaciones, limitando de esa manera la frecuencia de operación del sistema. Asimismo ya ha sido probado que la tecnología de los FPGAs es mejor en implementaciones de filtros FIR [25].

Fig. 10 *Digital Signal Processor* (DSP) de la compañía Texas Instruments [24].

Sin embargo también se encuentran los dispositivos *Graphic Processing Unit – Compute Unified Device Architecture* (GPU-CUDA, por sus siglas en inglés) [26] mostrado en la **Figura 11** con su arquitectura multi-núcleos los cuales han presentado mejor performance que los FPGA en tiempo de procesamiento, en implementaciones de Filtros FIR bidimensionales *(2D-FIR Filters)* en [27] pero cuyo consumo es aproximadamente 7 veces de lo que requiere un FPGA para su funcionamiento, por lo que se descarta esta alternativa ya que lo que adicionalmente se plantea en la problemática es el menor consumo energético del dispositivo hardware con miras al uso en dispositivos portátiles, donde la optimización de la cantidad de energía consumida de las baterías es un factor fundamental. Los FPGA por el contrario pueden reducir su consumo por medio de técnicas de reconfiguración dinámica de la arquitectura [28], la cual no se realizará en el presente trabajo debido a que es un tema de líneas de investigación a nivel posgrado.

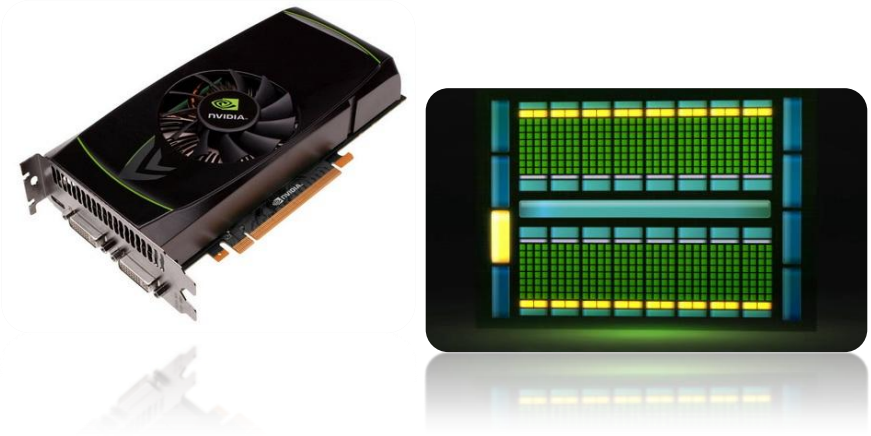


Fig. 11 *Processing Unit – Compute Unified Device Architecture* (GPU-CUDA) de la compañía NVIDIA [26].

3.3 Objetivos:

- **Objetivo Principal:**
  Diseñar una arquitectura Hardware para Interpolación de *Quarter-Pixel* según el formato H.264/AVC para HDTV (1920x1080 píxeles) a 30 cuadros/segundo.

- **Objetivos Específicos:**

- Verificación del algoritmo en software para bloques de 8x8 muestras.
- Síntesis Comportamental y descripción en VHDL.
- Verificación funcional por simulación por medio de circuitos de testeo utilizando la herramienta ModelSim de la compañía MENTHOR GRAPHICS para ALTERA® [29].
- Optimización de Recursos Hardware y Frecuencia de operación.

3.4 Descripción de la Arquitectura Genérica del Dispositivo FPGA:

Un Arreglo de Compuertas Lógicas de Campo Programable (FPGA, por sus siglas en inglés) es un circuito integrado (IC, por sus siglas en inglés) digital que contiene bloques e interconexiones configurables [30]. Las arquitecturas diseñadas para un FPGA pueden ser descritas por un lenguaje de descripción de hardware como VHDL o Verilog HDL, siendo estos lenguajes HDL independientes de la tecnología por encontrarse en el nivel de comportamiento, según los niveles de representación de un diseño digital propuestos por el Dr. Gajski en 1983 [31], tanto así que es bueno resaltar que no sólo sirven para hacer descripciones para FPGAs, CPLDs sino también para ASICs.

La estructura interna del FPGA comprende dos tipos de módulos:

- **Módulos Principales:**

  Comprenden únicamente a los Bloques Lógicos Programables los cuales se dividen en lo que los fabricantes llaman *Slices*, basados en celdas lógicas o elementos lógicos (llamados así por XILINX® [32] y ALTERA® [33] respectivamente) las cuales están formadas por un LUT de 4 entradas, un multiplexor y un Flip-Flop. El número de *Slices* de un Bloque Lógico Programable depende de la familia de FPGA [34].

- **Módulos Secundarios:**

  Se encuentran alrededor de los Bloques Lógicos Programables, embebidos dentro del chip del FPGA. Principalmente son Memorias de Acceso Aleatorio (RAM, por sus siglas en inglés) para almacenamiento temporal, Multiplicadores con acumulación (MACs, por sus siglas en inglés) y núcleos de procesadores que pueden ser programados para actuar como microprocesadores para tareas especificas que no requieran gran capacidad computacional. [34]. Adicionalmente también se tiene al Árbol de Reloj el cual conecta la señal de reloj con todos los Flip Flops del FPGA. Finalmente el Controlador de Reloj el cual produce diferentes señales de reloj a

partir de la señal del oscilador interno en base a los controladores de Fase de Lazo Cerrado (PLL, por sus siglas en inglés) que posee [34].

En la **Figura 12** se muestra el diagrama de bloques del FPGA Cyclone II de la compañía ALTERA® tomado de la hoja técnica adjunta en el **ANEXO A** conformado por lo módulos anteriormente descritos. En el caso de la Cyclone II, los bloques de RAM embebidos se conocen como bloques M4K.

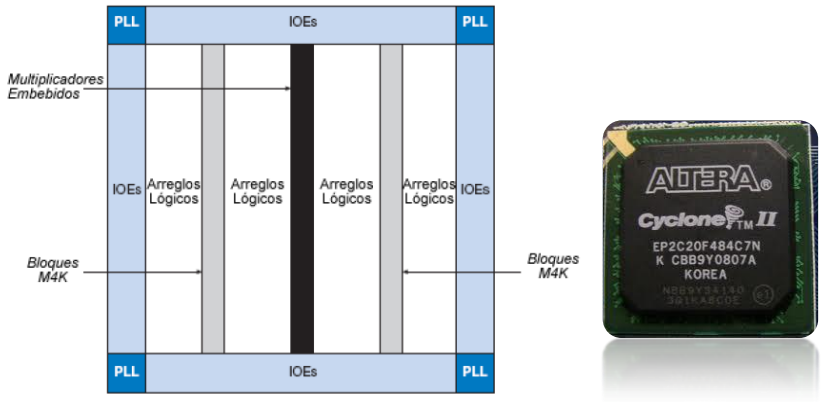


Fig. 12 Diagrama de Bloques del FPGA Cyclone II de la compañía ALTERA®.

En una implementación en *hardware*, para una etapa de desarrollo del "prototipo de verificación", el uso de FPGAs es fundamental ya que al hacer las verificaciones pueden encontrarse errores respecto de la especificación por lo que el diseño debe ser corregido, implementado y verificado nuevamente hasta ser validado.

El FPGA permite la rápida configuración de sistemas digitales lo cual acelerará la etapa de verificación del prototipo. En contraparte el uso de un ASIC para un "prototipo de verificación" no es conveniente debido a los tiempos de fabricación y a los elevados costos de producción de pocas unidades. Cuando la etapa del "prototipo de verificación" está terminada se analiza el volumen de unidades que se tendrá del diseño para luego decidir si se realizará la fabricación del sistema en ASIC de acuerdo a la masa de producción demandada.

3.5 Diseño de la Arquitectura partir de la síntesis comportamental del algoritmo:

Luego de validar el Algoritmo de Estimación de Movimiento Fraccional por medio de una aplicación en el entorno de programación MATLAB® (cuyo código fuente está en el **ANEXO B**) en base a un software de referencia disponible en el portal [36][9], se realizará la síntesis comportamental del mismo, es decir se diseñará una arquitectura *hardware* del

[9] El software de referencia fue modificado por el autor del presente trabajo de tesis, como se explica en la sección 4.1

algoritmo aprovechando el grado de paralelismo que presenta con la finalidad de optimizar el tiempo de procesamiento para cumplir con los objetivos.

La propuesta del presente trabajo está basada en el estado del arte presentado en la bibliografía [18][19][23][37][38]. Los criterios que se tomarán en cuenta para el diseño de la arquitectura serán: frecuencia de operación, número de ciclos de operación y cantidad de recursos *hardware*.

3.5.1 Arquitectura la Unidad de Estimación de Movimiento Fraccional Half-Pixel:

Para la síntesis comportamental del Algoritmo de Estimación de Movimiento Fraccional con precisión de 0.5, se tomará como base dos de los trabajos previos [18] [19], ya que cumplen con los requisitos anteriormente mencionados.

A. Arquitectura de la Unidad de Interpolación Half-Pixel:

Esta arquitectura se encarga de realizar las operaciones de Interpolación (FIR 6-Taps) para obtener los Half-Pixels que serán utilizados en el proceso de búsqueda Half-Pixel. Esta arquitectura está basada en un trabajo previo [18] con una modificación en la fórmula utilizada para el cálculo de los Half-Pixel Tipo D, el utilizar todos los valores intermedios de los Half-Pixel Tipo V (almacenados en el buffer V1) mientras que en [19] sólo se considera los valores intermedios de los dos píxeles adyacentes a la posición fraccionaria que se desea calcular y los demás Half-Pixels con sus valores finales. El formato H.264/AVC [12] permite manejar ambas opciones y no sólo utilizar los valores de los Half-Pixel Tipo V sino utilizar los valores de los Half-Pixel Tipo H para realizar la interpolación [17]. En la **Figura 13** se muestran las Interconexiones de los módulos de la unidad de Interpolación Half-Pixel.

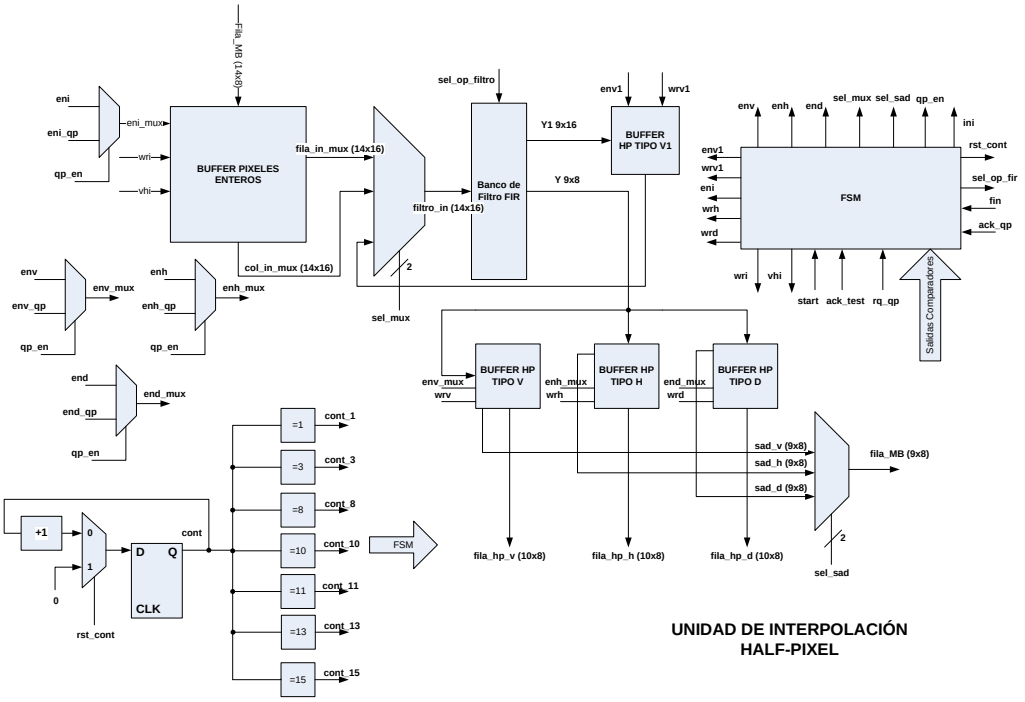


Fig. 13 Interconexiones de los módulos de la Unidad de Interpolación Half-Pixel.

Esta unidad consiste en cuatro buffers, un banco de de Filtros FIR-6taps, un contador con un conjunto de comparadores y una máquina de estados (FSM) que se encarga de realizar el control de cada uno de los módulos que conforman la unidad.

Los cuatro buffers presentes tienen la siguiente función:

- **Buffer de Píxeles Enteros:** Es el encargado de almacenar un arreglo de 14x14 píxeles del cuadro de referencia que será utilizado para la interpolación, siendo su comportamiento muy similar a la de una FIFO circular [18]. En la **Figura 14** se muestra la arquitectura de un buffer del mismo tipo que almacena un arreglo de 3x3 píxeles donde cada píxel está representado por un byte (parte de Luminancia), en la **Figura 15** se ilustra su comportamiento circular y en la **Tabla 1** se muestra una descripción de sus entradas y salidas.

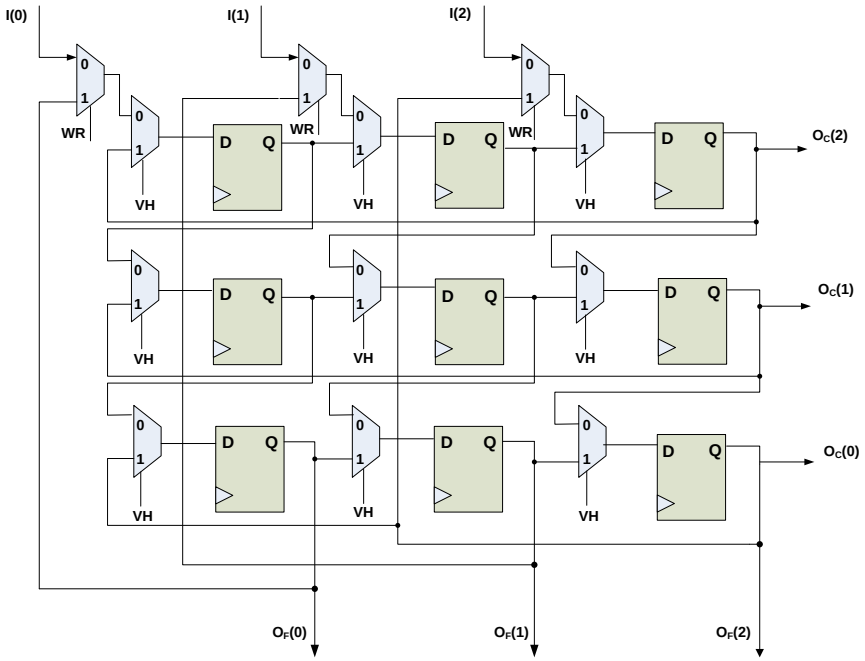


Fig. 14 Buffer de Píxeles enteros de 3x3. Los registros son de 8 bits.

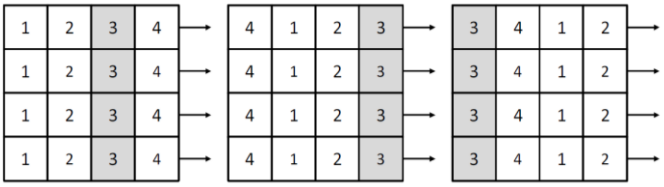


Fig. 15 Comportamiento Circular del Buffer [18].

Tabla 1: Descripción de Entradas y Salidas del Buffer de Píxeles enteros.

Parámetro	Tipo de Dato	Descripción
I(0), I(1), I(2) , I(3), … I(14)	Entrada	Arreglo de 14 bytes que representa una fila de píxeles del MB de referencia a la entrada del buffer.
O_F(0), O_F (1), O_F (2), … O_F (14)	Salida	Arreglo de 14 bytes que representa una fila de píxeles del MB de referencia a la salida del buffer.
O_C(0), O_C (1), O_C (2), … O_C (14)	Salida	Arreglo de 14 bytes que representa una columna de píxeles del MB de referencia a la salida del buffer.
WR	Entrada	Modo Escritura/Lectura
VH	Entrada	Modo de desplazamiento Vertical/Horizontal
EN	Entrada	Habilitador
CLK	Entrada	Señal de Reloj

- **Banco de Filtros:** Está conformado por 9 módulos que representan a la operación de un Filtro FIR de 6-taps como una arquitectura de pipeline. La entrada de este Banco de Filtros es la salida del multiplexor que se encarga de seleccionar entre filas o columnas para calcular los Half-Pixel Tipo H y V respectivamente o una fila de los valores intermedios de los Half-Pixel Tipo V (V1) para el cálculo de los Half-Pixel

Tipo D. En la **Figura 16** se muestra la arquitectura pipeline de uno de los 9 filtros FIR que conforman el Banco de Filtros.

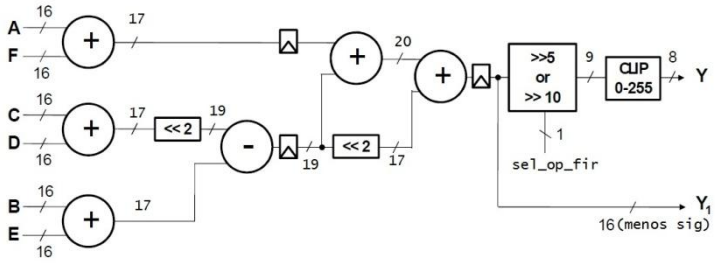


Fig. 16 Arquitectura Pipeline del Filtro FIR de 6-taps [19].

En la **Tabla 2** se muestra la descripción de los terminales de entrada y salida del Banco de Filtros. Es importante especificar que la distribución de los píxeles de la línea de entrada para cada uno de los Filtros FIR se hace dentro del Banco de Filtros.

Tabla 2: Descripción de Entradas y Salidas del Banco de Filtros FIR de 6-taps.

Parámetro	Tipo de Dato	Descripción
filtro_in[0 … 14]	Entrada	Arreglo de 14 elementos de 16 bits que representa una fila o columna de píxeles.
Y1[0 … 8]	Salida	Arreglo de 9 elementos de 16 bits que representa una fila o columna con los valores intermedios de los Half-Pixel interpolados.
Y[0 … 8]	Salida	Arreglo de 9 elementos de 8 bits que representa una fila o columna con los valores de los Half-Pixel interpolados.
sel_op_fir	Entrada	Modo de operación (>>5 ó >>10) a la salida del Filtro.
CLK	Entrada	Señal de Reloj

- **Buffer Tipo V1:** Es similar al buffer de píxeles enteros con la diferencia que los datos se guardan por columnas y se acceden por filas. Este buffer almacena un arreglo de 14x9 valores intermedios de los Half-Pixels Tipo V utilizados para calcular el valor de los Half-Pixel Tipo D por lo que cada registro será de 16 bits. En la **Tabla 3** la descripción de sus entradas y salidas.

Tabla 3: Descripción de Entradas y Salidas del Buffer Tipo V1.

Parámetro	Tipo de Dato	Descripción
I[0 … 9]	Entrada	Arreglo de 9 elementos de 16 bits que representa una columna de los valores intermedios de los Half-Pixels Tipo V.
O_F[0 … 14]	Salida	Arreglo de 14 elementos de 16 bits que representa una fila con los valores intermedios de los Half-Pixel Tipo V, los cuales pasarán por el banco de filtros para calcular los Half-Pixel Tipo D.
WRV1	Entrada	Modo Escritura/Lectura.
ENV1	Entrada	Habilitador
CLK	Entrada	Señal de Reloj

- **Buffer Tipo V:** Tiene la misma arquitectura que el buffer anterior con la diferencia que almacena los valores finales de los Half-Pixel Tipo V, por lo que sólo almacenará un arreglo de 10x9 Half-Pixels donde cada registro será de 8 bits.

- **Buffer Tipo H:** Este buffer se encarga de almacenar los valores de los Half-Pixels Tipo H. Tiene el mismo de los anteriores buffers pero con la diferencia que los datos ingresan por filas y salen en filas. Almacena un arreglo de 9x10 valores donde cada registro es de 8 bits.

- **Buffer Tipo D:** Tiene la misma arquitectura que el buffer anterior con la diferencia que almacena un arreglo de 9x9 Half-Pixels Tipo D donde cada registro también es de 8 bits.

- **Máquina de Estados (FSM):** Es la que se encarga de realizar el control de todos los módulos anteriormente descritos. Esta máquina de estados se basa en un contador, es decir las señales de control provienen del valor actual del contador para lo cual se utilizan comparadores. En el **ANEXO C** – Unidad de Interpolación Half-Pixel se adjunta el diagrama de estados y la tabla de descripción de entradas y salidas.

B. Arquitectura de la Unidad de Búsqueda Half-Pixel:

Se diseñará una arquitectura capaz de realizar el cálculo del SAD (*Sum of Abssolute Differences*) de cada uno de los 9 bloques de 8x8 obtenidos del proceso anterior y luego compararlos junto con el SAD obtenido en la IME, para obtener el menor SAD posible junto con su MV. Esta arquitectura está basada en los módulos que conforman la arquitectura de un trabajo previo [19] pero cuyas señales de control y algunos elementos lógicos tuvieron que ser planteados originalmente en este trabajo. Se escogió esta arquitectura debido al paralelismo de operaciones entre el proceso de interpolación Half-Pixel, el cálculo de los valores de SAD y la comparación de los mismos. En la **Figura 17** se muestra la arquitectura de la Unidad de Búsqueda Half-Pixel, donde también se especifica las señales de control de la Máquina de Estados.

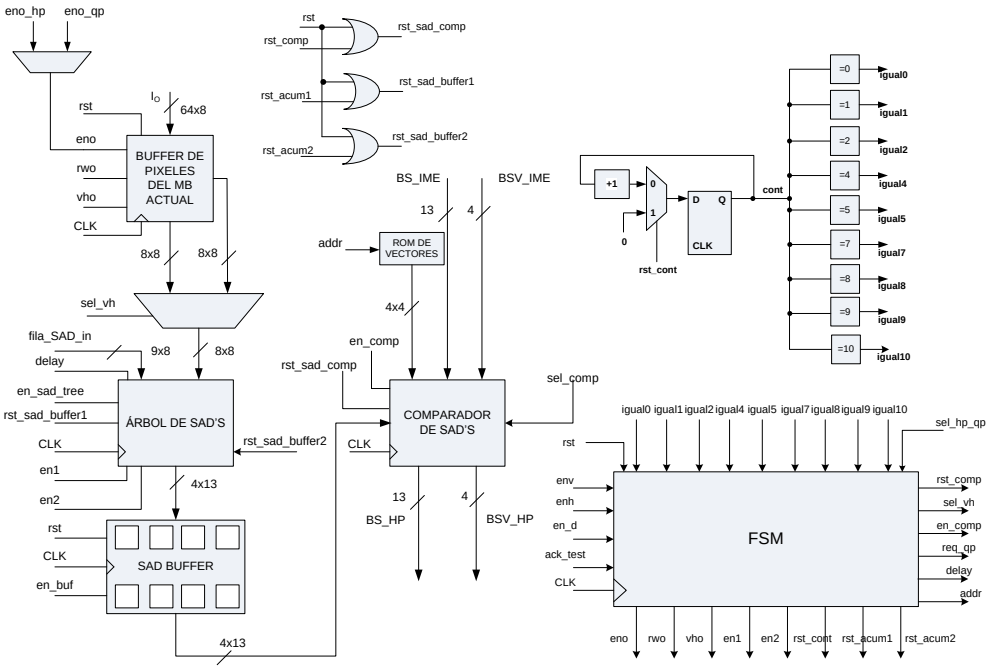


Fig. 17 Interconexiones de los módulos de la Unidad de Búsqueda Half-Pixel.

La Unidad de Búsqueda Half-Pixel está compuesta de los siguientes bloques:

- **Buffer de Píxeles Enteros del MB actual**: Tiene la misma arquitectura que el Buffer de Píxeles enteros utilizado en la Unidad de Interpolación Half-Pixel con la diferencia que almacena una matriz 8x8 muestras.

- **Árbol de SADs**: está compuesto por cuatro módulos que se encargan de realizar la operación de cálculo de SAD. Cada uno de los 4 módulos está basado en una arquitectura *pipeline* de dos estados y un circuito acumulador donde se almacena el valor final del SAD calculado, ya que posee como entradas dos arreglos de 8 elementos que representan a los valores de una fila (o columna) de los Half-Pixels y el bloque actual respectivamente. Como el proceso se realiza línea por línea de 8 píxeles, entonces se necesitarán 10 ciclos de reloj para poder calcular un valor de SAD. En la **Figura 18** se muestra la arquitectura del módulo encargado de realizar el cálculo del SAD.

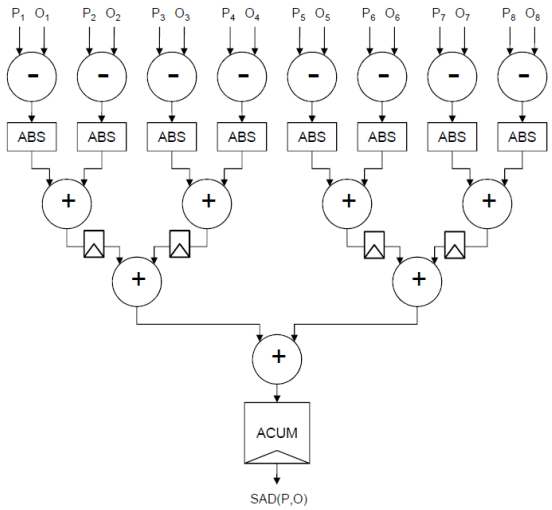


Fig. 18 Arquitectura de uno de los 4 módulos que conforman el Árbol de SADs [19].

El uso de los 4 módulos de SAD se justifica en el paralelismo que se desea lograr entre el cálculo de SAD y el proceso de Interpolación. En una primera etapa cuando se obtienen los Half-Pixel Tipo V, como los valores se obtienen por columnas, se cambia la entrada de la línea de entrada del bloque actual por medio del multiplexor conectado a la entrada del Árbol de SADs. Se utilizan dos de los cuatro módulos, ya que la columna de 9 valores de Half-Pixel Tipo V presenta dos posibilidades para obtener el *"best matching"* (0, -0.5) y (0, 0.5). En una segunda etapa cuando se obtienen los Half-Pixel Tipo H se utilizan los dos módulos restantes ya que también se tiene dos posibilidades presentes (-0.5, 0) y (0.5, 0). Para la tercera etapa los valores de SAD obtenidos hasta entonces se guardarán y mientras que se obtienen los valores de los Half-Pixel Tipo D para lo cual se utilizarán los 4 módulos presentes para el cálculo de SADs debido a las 4 posibilidades presentes: (-0.5, -0.5), (0.5, -0.5), (-0.5, 0.5) y (0.5, 0.5).

- **SAD Buffer:** Es el encargado de almacenar los valores de SAD obtenidos en la salida del Árbol de SADs. La arquitectura de este buffer está compuesto por dos filas de 4 registros de 13 bits cada una para almacenar los 8 valores de SAD obtenidos.

- **Comparador de SADs:** Es el encargado de realizar la comparación de los 4 valores de SAD obtenidos en la salida del SAD Buffer. Tal como se mencionó anteriormente, en una primera etapa se comparan los valores de SAD obtenidos a partir de la diferencia entre el bloque Actual y los Half-Pixel Tipo H y V mientras se realiza la interpolación de los Half-Pixel Tipo D y el cálculo del SAD por lo que no son necesarios más ciclos de reloj [19]. Luego en una segunda etapa se comparan los valores de SAD obtenidos a partir de los Half-Pixel Tipo D y se compara con el valor del menor SAD obtenido en la primera etapa. Finalmente el menor SAD

obtenido hasta entonces se comparará con el menor valor de SAD obtenido en el proceso de IME. Los valores de los MVs son transmitidos a la vez que se hace la comparación de los valores de SAD correspondientes a cada vector. En la **Figura 19** se muestra la arquitectura del módulo Comparador de SADs.

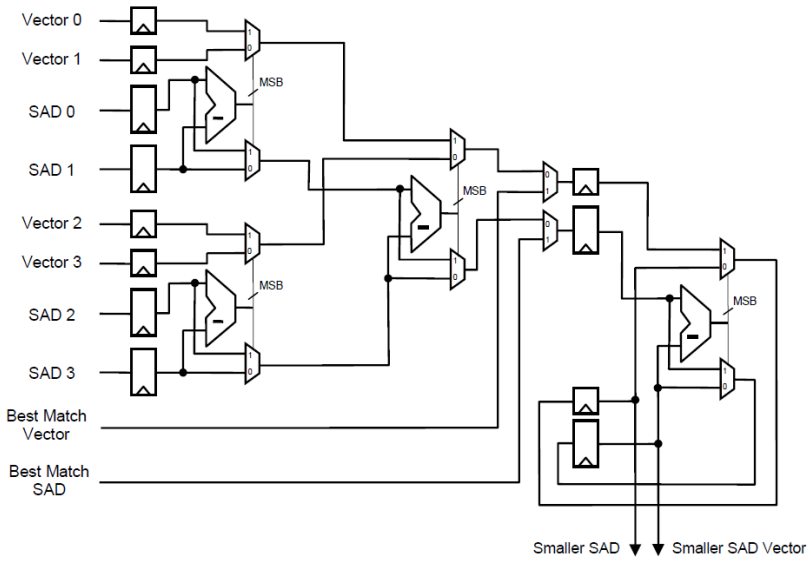


Fig. 19 Arquitectura del Comparador de SADs [19]

- **ROM de vectores:** es una memoria ROM que contiene los valores de los Mbps expresados en 4 bits según la **Tabla 4**.

Tabla 4: Representación en binario de los MVs con precisión 0.5.

Vector de Movimiento Fraccionario	Representación en binario (4 bits)
(0.0, -0.5)	0011
(0.0, 0.5)	0001
(-0.5, 0.0)	1100
(0.5, 0.0)	0100
(-0.5, -0.5)	1111
(0.5, -0.5)	0111
(-0.5, 0.5)	1101
(0.5, 0.5)	0101

- **Máquina de Estados (FSM)**: Esta máquina de estados se caracteriza por tener una comunicación con la Máquina de Estados de la Unidad de Interpolación Half-Pixel y la Unidad de Estimación de Movimiento Quarter-Pixel en base al Protocolo Handshake [39] por medio de señales de requerimiento (*request*) de la Unidad de Búsqueda Half-Pixel y señales de confirmación de pedido (*acknowledge*) de las máquinas de estado de las demás unidades. El diagrama de estados y la tabla de descripción de parámetros se adjuntan en el **ANEXO C** – Unidad de Búsqueda Half-Pixel.

3.5.2 Arquitectura de la Unidad de Estimación de Movimiento Fraccional Quarter-Pixel:

Para el diseño de la arquitectura de Estimación de Movimiento Fraccional Quarter-Pixel se tomará como base la arquitectura propuesta en un trabajo previo [23], ya que presenta alto grado de paralelismo en las operaciones de interpolación y se adecúa muy bien a la arquitectura de Estimación de Movimiento Fraccional Half-Pixel que se diseñada hasta el momento.

A. Arquitectura de la Unidad de Interpolación Quarter-Pixel:

Posee un alto grado de paralelismo en sus operaciones, ya que en un sólo ciclo de reloj calcula 3 filas de píxeles correspondientes a las posiciones fraccionarias de precisión 0.25. El principal aporte con respecto a [23], está en el empleo de los buffers que se utilizaron en la arquitectura de la Unidad de Interpolación Half-Pixel en lugar de utilizar una memoria RAM para los 4 tipos de datos que se procesarán (Píxeles I, Tipo H, V o D), en la **Figura 20** se muestra la arquitectura general propuesta en el trabajo previo [23].

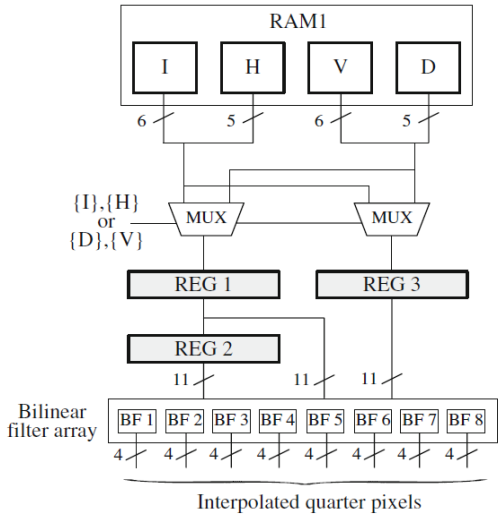


Fig. 20 Arquitectura propuesta que se utilizará como base [23].

La arquitectura desarrollada en el presente trabajo se muestra en la **Figura 21** donde también se incluyen las señales de control de la FSM.

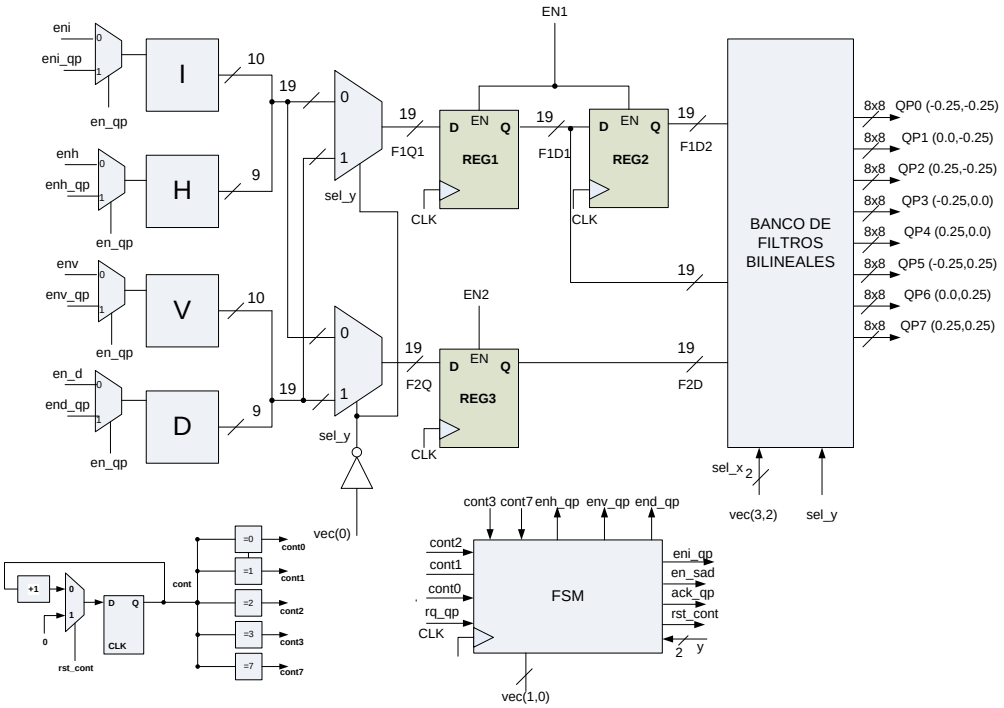


Fig. 21 Interconexiones de los módulos de la Arquitectura de la Unidad de Interpolación Quarter-Pixel.

El orden en que serán accedidas cada una de las filas del área mostrada dependerá del valor del MV obtenido de la Unidad de Búsqueda Half-Pixel. Es por eso que el valor de **vec(0)** representa si el valor del Half-Pixel en su coordenada vertical apunta a una línea formada por píxeles Tipo V & D o Tipo I & H.

Los multiplexores se encargan de seleccionar el orden de las líneas que serán procesadas, por ejemplo en el caso que la coordenada vertical del MV tenga un valor de -0.5 ó 0.5, se seleccionará como línea central a la fila conformada por los Half-Pixels Tipo V & D con dos filas adyacentes conformadas por los píxeles enteros (I) y Half-Pixels Tipo H. En el caso que la coordenada vertical hubiese tomado el valor de cero, se seleccionará como línea central a la fila conformada por píxeles enteros (I) y Half-Pixels Tipo H con dos filas adyacentes conformadas por los Half-Pixels Tipo V & D [23].

Los registros conectados a las salidas de los multiplexores se encargan de almacenar las 3 filas adyacentes que serán procesadas en el banco de filtros bilineales donde se obtendrán los valores de las muestras Quarter-Pixel. La FSM de este circuito se encarga de habilitar los registros en el momento en que se desea guardar las muestras.

- **Banco de Filtros Bilineales:** Es el que realiza las operaciones necesarias para el cálculo de las muestras Quarter-Pixel, las 57 muestras son seleccionadas a partir del valor de la coordenada horizontal del MV Half-Pixel, calculado en el proceso

anterior, luego las muestras obtenidas de la interpolación son agrupadas en 8 bloques de 8 muestras cada uno, los cuales representan una fila del mismo tipo de Quarter-Pixel, tal y como se muestra en la **Figura 21**. En la **Tabla 5** se muestran los grupos de píxeles a la salida del Banco de Filtros indicando la posición fraccional de precisión 0.25 a la que pertenecen.

Tabla 5: Representación en binario de los MVs con precisión 0.5

Vector de Movimiento Fraccionario	Salida del Filtro Bilineal
(-0.25, -0.25)	QP0
(0.0, -0.25)	QP1
(0.25, -0.25)	QP2
(-0.25, 0.0)	QP3
(0.25, 0.0)	QP4
(-0.25, 0.25)	QP5
(0.0, 0.25)	QP6
(0.25, 0.25)	QP7

- **Máquina de Estados (FSM):** Esta compuesta por 9 estados que se encargan de controlar el acceso a las filas de la región conformada por los píxeles enteros y Half-Pixels. Se utiliza como parámetro de entrada a las coordenadas del MV obtenido de la Unidad de Búsqueda Half-Pixel para determinar la fila central que se almacenará en REG3 y las filas adyacentes en REG1 y REG2. Se realiza este procesamiento fila por fila hasta que se consiga en la salida todos los posibles valores de las muestras de Quarter-Pixel para cada posición. Los valores de Quarter-Pixel posteriormente se enviarán a la Unidad de Búsqueda Quarter-Pixel para realizar el cálculo de SAD como se realizó en la Unidad de Búsqueda Half-Pixel.

  La comunicación entre estos módulos se hace por medio del protocolo Handshake. El diagrama de estados y la tabla de descripción de parámetros se adjuntan en el **ANEXO C** – Unidad de Interpolación Quarter-Pixel.

B. Arquitectura de la Unidad de Búsqueda Quarter-Pixel:

Se basa en el trabajo previo [23] de la cual se obtuvo la arquitectura de la unidad anterior. Esta arquitectura presenta un alto grado de paralelismo, ya que realiza el cálculo de SAD de los 8 posibles casos de Quarter-Pixel al mismo tiempo. Esta arquitectura es similar a la de la Unidad de Búsqueda Half-Pixel con la diferencia que se utilizan 8 módulos para el cálculo de SAD.

Otra diferencia es que en la presente arquitectura se obvia el uso de un Buffer para almacenar los valores de SAD obtenidos, pues resulta innecesario debido a que no es requerido guardar los valores obtenidos ya que los acumuladores de los módulos del árbol de SADs realizan esta tarea.

La última diferencia sería la comparación de 8 valores de SAD, en lugar de 4 como en la Unidad de Búsqueda Half-Pixel. Asimismo el módulo de comparación de SADs Quarter-Pixel cuenta con una arquitectura pipeline con la finalidad de procesar a una mayor frecuencia de operación a costa de algunos ciclos. En la **Figura 22** se muestra la arquitectura de la Unidad de Búsqueda Quarter-Pixel en donde también se especifican las señales de control de la Máquina de Estados.

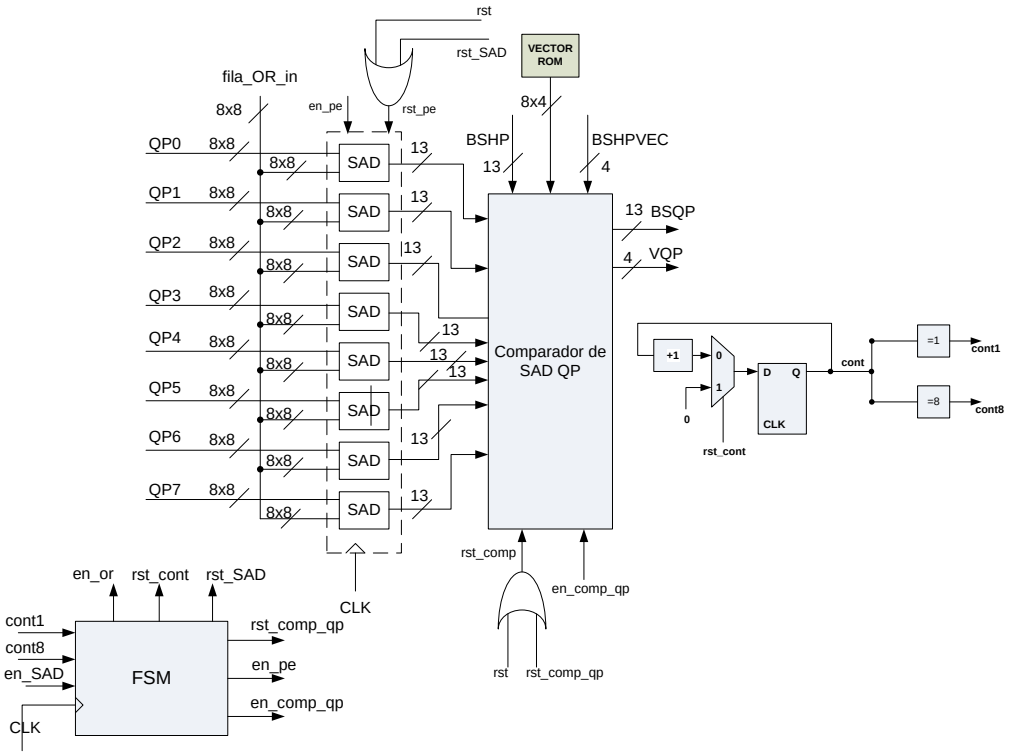


Fig. 22 Interconexiones de los módulos de la Arquitectura de la Unidad de Interpolación Quarter-Pixel.

La Unidad de Búsqueda Quarter-Pixel está conformada por los siguientes módulos:

- **Línea de SADs:** Está compuesta por 8 módulos que se encargan de realizar el cálculo de SAD para todas las posiciones de Quarter-Pixel posibles. La arquitectura de cada módulo es la misma que se mostro en la Unidad de Búsqueda Half-Pixel. El contador externo se encarga de llevar la cuenta de la cantidad de filas que se deben procesar, en este caso cuando se han procesado 8 filas (1fila por ciclo de reloj) será necesario que este módulo este activo por un ciclo adicional, debido a las dos etapas

de pipeline. Cuando el contador externo tenga el valor de 9, se deshabilitará este módulo.

- **Comparador de SADs QP**: Inmediatamente se tenga los valores de SAD en la salida del módulo de Línea de SADs se empezará a hacer la comparación de los 8 valores disponibles. Este módulo cuenta con una arquitectura pipeline de dos estados para mejorar su frecuencia de operación, por lo que será necesario que el módulo esté habilitado por un ciclo adicional para poder hacer una comparación exitosa. En la **Figura 23** se muestra la arquitectura interna del módulo.

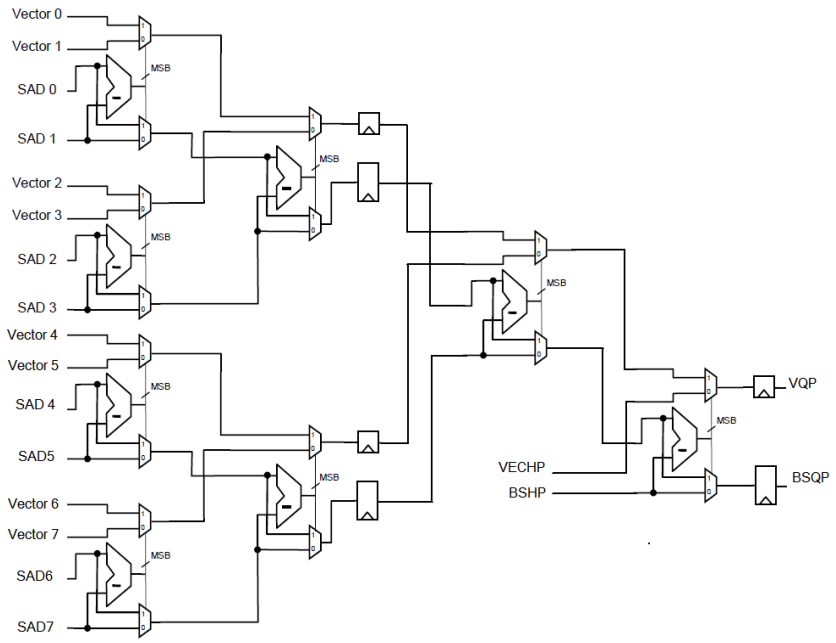


Fig. 23 Arquitectura Interna del Comparador de SADs QP.

- **Máquina de Estados (FSM)**: Esta compuesta sólo por 4 estados, la cual recibe una señal de aviso de la Unidad de Interpolación Quarter-Pixel para empezar a enviar las señales de control para los demás módulos. Al finalizar el proceso de búsqueda, la FSM envía una señal de aviso (**ack_qp_sr**) a las unidades externas (Interpolación HP, Búsqueda HP e Inteprolación QP) para que se reinicie el proceso de Estimación de Movimiento Fraccional para el próximo bloque de 8x8 del cuadro de video. El diagrama de estados y la tabla de descripción de parámetros se adjuntan en el **ANEXO C** – Unidad de Búsqueda Quarter-Pixel.

Capítulo 4

Simulación, Validación y Resultados

4.1 Verificación del Algoritmo de Estimación de Movimiento Fraccional con Interpolación Quarter-Pixel por medio de una aplicación Software:

Para verificar el funcionamiento de las características que se están tomando en cuenta para el diseño de la arquitectura de FME Quarter-Pixel, se implementó el algoritmo en una aplicación *software* en el entorno de programación de MATLAB® (cuyo código fuente se anexa en el **ANEXO B** – *fractionalme.m*). Los resultados obtenidos a partir de esta aplicación servirán para contrastar con los resultados obtenidos por medio de la simulación de la arquitectura diseñada.

Para la implementación del algoritmo, se tomó como base un *software* de referencia de código abierto del formato de compresión de video MPEG-2 disponible en el portal [40]. Una de las principales diferencias que presenta el MPEG-2 [37] con el H.264/AVC (MPEG-4 Part 10) [12] es el algoritmo de FME, la cual es foco de este trabajo, por lo que se realizó la modificación del código abierto para poder incluir esta característica.

El código de referencia se modificó de tal manera que el resultado sea una secuencia de video de cuadros residuales y una estructura[10] que contenga la información de todos los cuadros procesados, es decir el MV con precisión de 0.25 píxeles y el mínimo valor de SAD de todos los bloques de 8x8 que forman parte de los cuadros de la secuencia de video. Es necesario recalcar que el foco del presente trabajo es diseñar una arquitectura del algoritmo de ME, por lo que el objetivo de la validación en *software* es sólo para contrastar resultados y no para ser utilizado como aplicación.

La modificación consistió en agregar las funciones de Interpolación y Búsqueda Half-Pixel así como de Quarter-Pixel. El algoritmo utilizado para la implementación en *software* de estas funciones fue tomado de la bibliografía [37].

Los archivos de video utilizados para la validación, disponibles en el portal [41] [42], son de resolución QCIF (176x144 píxeles), CIF (352x288 o 352x240 píxeles) y HDTV (1920x1080

[10] Estructura (struct) es un tipo de dato del entorno de programación MATLAB® muy útil para agrupar un conjunto de variables que estén relacionadas entre sí.

píxeles) en formato descomprimido YUV4MPEG [43] utilizado por MJPEG TOOLS[11] [44], cuyas extensiones son *.yuv o *.y4m con sub-muestreo 4:2:0.

En la **Figura 24** se muestra el diagrama de flujo del algoritmo de FME, basado en la referencia [37], el cual servirá para modificar el *software* de referencia cuyos resultados serán utilizados para la validación del funcionamiento de la arquitectura como ya fue explicado.

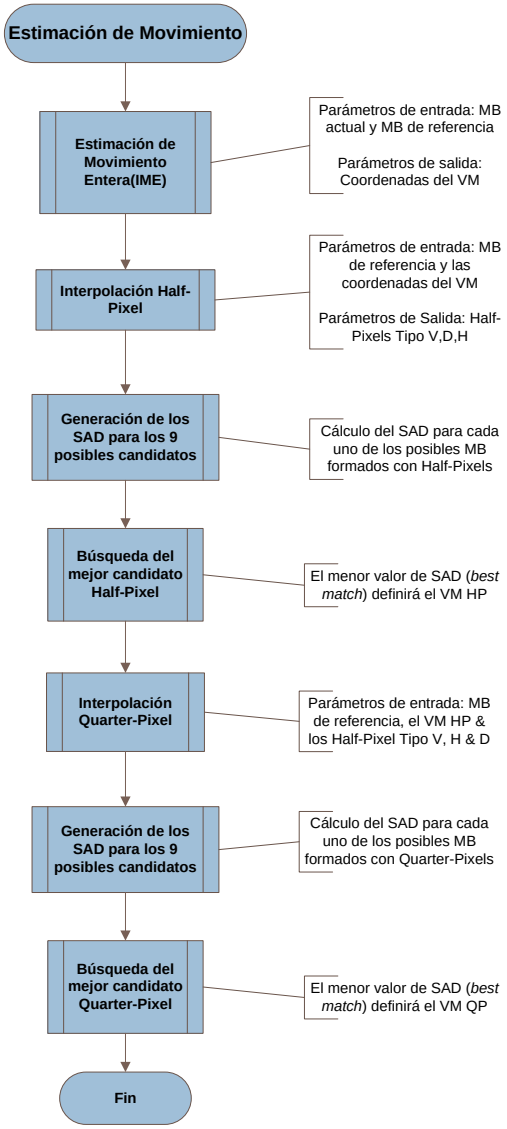


Fig. 24 Diagrama de Flujo del Algoritmo de Estimación de Movimiento.

[11] Los programas de MJPEG son un conjunto de herramientas que realizan operaciones de grabación y reproducción de videos, la edición sencilla de cortar y pegar cuadros y la compresión MPEG de audio y video en Linux.

4.2 Resultados de la Aplicación en Software:

Se realizaron las pruebas del software de aplicación con 14 secuencias distintas de diferente resolución cuyos resultados se adjuntan en el **ANEXO D**. Las secuencias utilizadas fueron de resolución QCIF (176x144), CIF (342x288 ó 352x240) y HDTV (1920x1080). Las secuencias procesadas fueron las siguientes:

- HDTV (1920x1080): *"tractor.y4m"*.
- QCIF (176x144): *"ice_qcif_15fps.y4m"*, *"suzie_qcif.y4m"* y *"akiyo_qcif 420 176_144.yuv"*
- CIF (342x288 ó 352x240): *"stefan_sif.y4m", "bus_cif.y4m", "akiyo_cif.y4m", "tennis_sif 420 352_240.y4m", "sample_int 420 352_288.yuv" & "foreman_cif 420 352_288.yuv".*

Los resultados evaluados fueron el SAD mínimo encontrado y el MV por cada bloque de 8x8 en todos los cuadros considerados en el procesamiento. Se realizaron tres pruebas para cada secuencia: una prueba considerando sólo IME, otra considerando FME Half-Pixel y finalmente FME Quarter-Pixel. Estas pruebas se realizaron con la finalidad de demostrar que la FME obtiene menor energía residual por MB en comparación con el IME y de esa manera obtener menor tasa de compresión al momento de la codificación de los cuadros residuales.

Se tomó el cuadro 5 de cada secuencia de video residual obtenida de la aplicación de FME Quarter-Pixel para mostrar los valores de MV y SAD mínimo por cada sub-MB de 8x8 que conforma el cuadro. En la **Tabla 6** se muestra una representación del cuadro 5 de la secuencia "Ice" de resolución QCIF, cada casilla representa a un sub-MB de 8x8 y el número que contiene al valor del SAD mínimo obtenido de la aplicación de FME Quarter-Pixel. En la **Tabla 7** se muestra los valores de SAD mínimos al igual en la tabla anterior pero obtenida a partir de la aplicación *software* con IME y en la **Tabla 8** se muestra los valores de SAD mínimos para el mismo cuadro pero obtenidos a partir de la aplicación *software* de FME Half-Pixel.

Tabla 6: Cuadro Residual dividido en MB de 8x8 píxeles con los valores de SAD mínimos de la secuencia "Ice".

32	35	45	1134	1660	1064	33	28	279	66	63	17	11	20	21	39	15	31	35	24	20	16
26	31	443	332	680	589	15	14	200	406	90	16	10	38	40	21	34	38	67	26	12	16
15	225	973	158	247	472	20	23	199	937	127	19	25	1294	1730	93	848	1085	1004	601	214	137
53	310	217	245	326	105	38	24	92	376	58	17	31	129	394	2122	744	455	320	129	133	123
47	246	368	386	181	37	29	23	17	19	18	17	64	178	341	103	955	26	227	79	89	301
16	33	290	624	136	13	20	24	17	16	12	39	158	211	666	2155	107	68	605	339	127	430
25	30	485	516	158	17	17	13	11	12	14	41	151	189	1121	2198	1399	1093	1570	257	225	455
29	61	545	1059	253	29	39	33	15	23	33	43	756	27	17	1123	562	130	142	168	110	270
38	48	102	551	232	26	16	17	20	15	37	65	904	35	395	1027	93	40	133	119	399	468
37	60	47	15	17	16	12	21	12	19	591	342	1008	50	34	1151	382	97	200	1256	352	521
48	22	17	13	17	19	21	19	27	54	400	646	1560	331	220	928	1843	206	104	662	653	372
93	15	18	20	13	18	20	22	31	51	57	62	324	392	564	107	138	41	91	482	180	178
198	26	24	19	16	11	19	43	54	57	38	40	124	142	237	193	57	62	159	75	28	29
174	29	30	16	20	20	12	16	19	20	33	21	163	149	153	163	78	88	186	74	18	8
234	41	17	16	19	23	21	22	23	25	30	39	271	224	142	414	81	300	124	80	20	20
377	32	15	18	24	20	20	15	22	28	25	28	58	48	41	89	68	82	31	25	22	25
39	18	17	24	16	23	24	31	26	30	32	18	25	25	19	16	21	16	15	19	16	14
41	13	24	18	25	18	13	16	13	15	27	26	19	16	16	15	12	12	25	18	12	14

Tabla 7: Cuadro Residual dividido en MB de 8x8 píxeles con los valores de SAD mínimos obtenidos por IME de la secuencia "Ice".

32	35	45	1134	1703	1103	37	28	335	66	63	17	11	20	21	39	15	31	35	24	20	16
26	31	467	332	684	640	15	14	261	466	96	16	10	38	40	21	34	38	67	26	12	16
15	225	983	178	254	574	20	23	228	937	152	19	25	1294	1730	93	922	1253	1004	601	351	137
53	368	319	245	632	257	38	24	92	378	103	17	31	498	418	2141	834	513	401	129	136	123
47	280	389	392	458	37	29	23	17	19	18	17	64	246	449	103	955	26	338	79	116	365
16	33	313	699	156	13	20	24	17	16	12	39	158	338	698	2170	122	68	605	604	133	440
25	30	523	516	158	17	17	13	11	12	14	46	165	214	1122	2198	1420	1142	1570	257	585	455
29	61	700	1059	253	29	44	33	15	23	33	43	756	27	17	1137	592	296	142	168	333	273
38	48	150	749	249	26	16	17	20	15	37	65	968	35	395	1055	93	40	156	119	493	538
37	60	47	15	17	16	12	21	13	19	609	407	1024	50	34	1224	524	97	253	1289	429	531
48	22	17	13	17	19	21	19	27	54	409	755	1560	353	370	980	1915	296	124	698	766	439
93	15	18	20	13	18	20	22	31	51	57	83	324	392	592	107	163	98	122	482	273	212
211	26	24	19	16	11	19	43	54	57	38	40	124	142	237	193	163	130	184	75	28	29
212	29	30	16	20	20	12	16	19	20	33	21	189	171	153	175	94	181	223	116	18	8
339	41	17	16	19	23	21	22	23	25	30	39	469	308	244	596	128	370	163	173	20	20
377	32	15	18	24	20	20	15	22	28	25	28	58	48	64	89	68	82	31	25	22	25
39	18	17	24	16	23	24	32	26	30	32	18	25	25	19	16	21	16	15	19	16	14
41	13	24	18	25	18	13	16	13	15	27	26	19	16	16	15	12	12	25	18	12	14

Tabla 8: Cuadro Residual dividido en MB de 8x8 píxeles con los valores de SAD mínimos obtenidos por FME con precisión Half-Pixel de la secuencia "Ice".

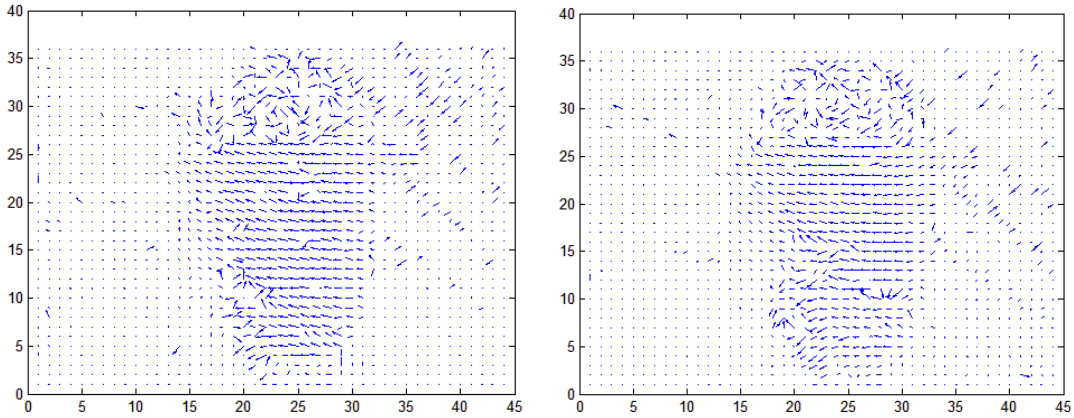
32	35	45	1134	1703	1064	33	28	279	66	63	17	11	20	21	39	15	31	35	24	20	16
26	31	467	332	684	640	15	14	200	466	90	16	10	38	40	21	34	38	67	26	12	16
15	225	973	175	254	574	20	23	199	937	127	19	25	1294	1730	93	848	1085	1004	601	351	137
53	310	319	245	326	105	38	24	92	378	58	17	31	129	394	2122	744	455	320	129	136	123
47	246	389	392	181	37	29	23	17	19	18	17	64	178	341	103	955	26	227	79	89	301
16	33	290	663	136	13	20	24	17	16	12	39	158	211	672	2155	107	68	605	339	127	430
25	30	515	516	158	17	17	13	11	12	14	41	165	189	1121	2198	1399	1113	1570	257	225	455
29	61	590	1059	253	29	39	33	15	23	33	43	756	27	17	1123	562	223	142	168	110	273
38	48	132	551	247	26	16	17	20	15	37	65	904	35	395	1035	93	40	133	119	399	538
37	60	47	15	17	16	12	21	12	19	609	407	1008	50	34	1224	464	97	200	1289	352	531
48	22	17	13	17	19	21	19	27	54	409	646	1560	353	220	928	1915	206	104	681	766	439
93	15	18	20	13	18	20	22	31	51	57	62	324	392	583	107	138	41	122	482	212	212
198	26	24	19	16	11	19	43	54	57	38	40	124	142	237	193	57	67	159	75	28	29
212	29	30	16	20	20	12	16	19	20	33	21	189	171	153	174	94	119	186	116	18	8
234	41	17	16	19	23	21	22	23	25	30	39	271	224	142	414	128	370	163	173	20	20
377	32	15	18	24	20	20	15	22	28	25	28	58	48	41	89	68	82	31	25	22	25
39	18	17	24	16	23	24	31	26	30	32	18	25	25	19	16	21	16	15	19	16	14
41	13	24	18	25	18	13	16	13	15	27	26	19	16	16	15	12	12	25	18	12	14

Los valores dentro de los cuadros son muy similares y aparentemente no hay mucha diferencia, pero si se suman todos los valores obtenidos se observará que el proceso con mayor energía residual es el IME, seguido del FME con precisión Half-Pixel y finalmente la FME con precisión Quarter-Pixel. En la **Tabla 9** se muestra una comparativa de los 3 procesos para las dos secuencias antes mencionadas.

Tabla 9: Cuadro comparativo de la cantidad de energía residual obtenida de los Procesos: IME, FME con precisión Half-Pixel y Quarter-Pixel.

Suma de los SAD mínimos del cuadro residual		
	"Ice" QCIF (176x144)	"Foreman" QCIF (352x288)
IME	87634	261587
FME con precisión Half-Pixel	81582	234389
FME con precisión Quarter-Pixel	79491	227044

De los resultados obtenidos se ha demostrado que aplicando FME Quarter-Pixel se obtiene una menor cantidad de energía residual, lo que generará mayores tasas de compresión al momento de realizar la codificación de los cuadros residuales. En la **Figura 25** se muestra los resultados de los MVs obtenidos para diferentes cuadros de la secuencia "Foreman".

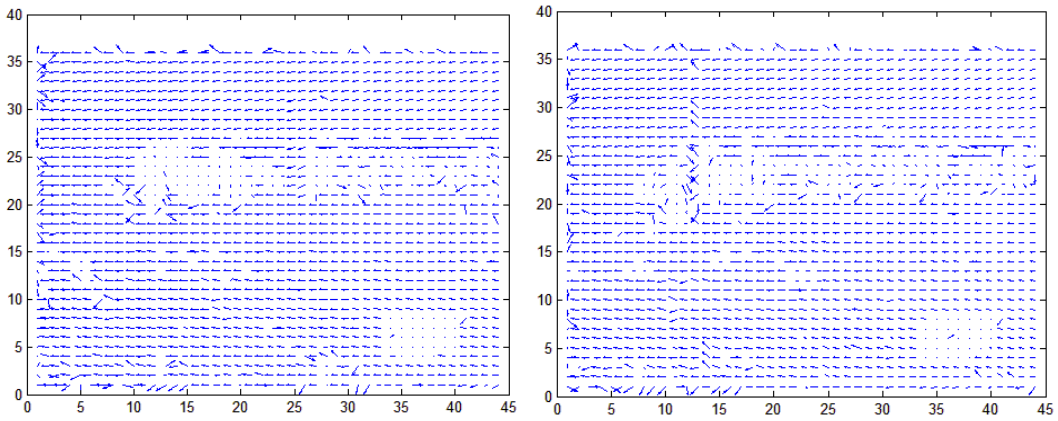


(a) **(b)**

Fig. 25 Secuencia "Foreman" (a) MV's del Cuadro 3 (b) MVs del Cuadro 5.

Lo primero que se puede observar es que los MVs de los sub-Mbps que son adyacentes entre sí tienen casi el mismo sentido, además en las zonas de la imagen donde está presente la redundancia espacial, los MVs están representados por puntos.

En la **Figura 26** se muestra los MVs para la secuencia "Bus" de resolución CIF, donde el sentido de los MVs se puede percibir mejor.



(a) **(b)**

Fig. 26 Secuencia "Bus" (a) MVs del Cuadro 2 (b) MVs del Cuadro 5.

Estas secuencias se utilizarán para probar el diseño de la Arquitectura de Estimación de Movimiento Fraccional con precisión 0.25, empleando la herramienta de simulación ModelSim [29]. Los archivos de estímulo que serán usados para la simulación serán obtenidos a partir de un script de MATLAB® (*gentext.m & gentextsad.m*) cuyo código fuente se adjunta en el **ANEXO E**.

Se requerirán tres archivos de estímulo para generar las señales de entrada: un archivo que contenga la imagen del cuadro actual dividida en sub-MB de 8x8 muestras (parte de Luminancia), un archivo que contenga la imagen del cuadro de referencia en bloques de 14x14 (parte de Luminancia) a partir de la cual se realizará la interpolación y otro archivo con los valores de SAD mínimos obtenidos del proceso de IME con los cuales se realizará la comparación.

4.3 Simulación de los módulos de la Arquitectura:

Para poder verificar la simulación de todas las etapas de los circuitos se tomará como referencia el procesamiento del cuadro 3 de la secuencia "Foreman" de resolución (342x288) empleando como referencia al cuadro 2 (cuadro de referencia). Los resultados de la simulación de cada módulo se adjuntan en el **ANEXO F** junto con los resultados obtenidos de la aplicación software para verificar el correcto funcionamiento.

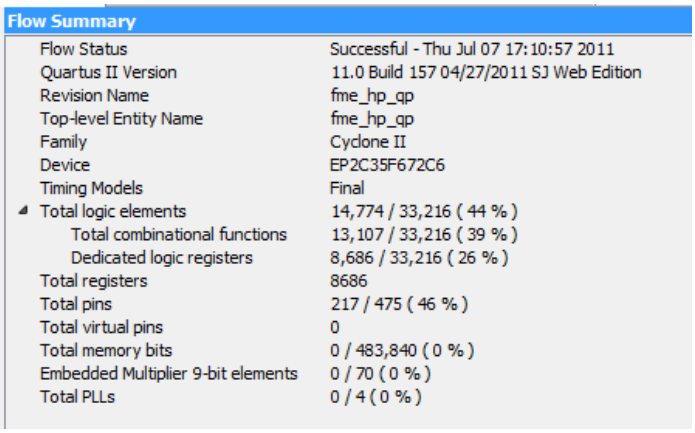
Asimismo se adjunta los archivos de proyecto de la arquitectura obtenidos del software **Quartus II v11.0** de la compañía ALTERA® junto con los resultados de la simulación para el cuadro 3 de la secuencia "Tractor" de resolución HDTV (1920x1080) en el **ANEXO G**. El nombre del archivo de texto obtenido de la simulación por *Testbench* tiene el nombre de "*Resultados_Simulacion_SAD_QP.txt*" los cuales pueden ser contrastados por los resultados obtenidos del *software* de aplicación guardados en el archivo de texto *"sad_fr3_qp_dec.txt"*.

De los resultados de la simulación se obtuvo que la arquitectura diseñada tarda 77 ciclos de reloj en procesar un sub-MB de 8x8 píxeles por lo que tomará 308 ciclos de reloj en procesar un MB (16x16 muestras)

4.4 Resultados de Síntesis de la Arquitectura:

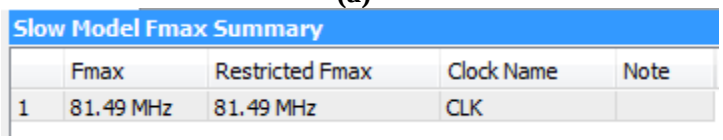
La arquitectura diseñada fue sintetizada utilizando el programa **Quartus II V 11.0** para el FPGA Cyclone II[12] EP2C35F672C6 de la compañía ALTERA® [33]. Los resultados obtenidos se muestran en la **Figura 27**. En la parte (a) se muestra la cantidad de elementos lógicos utilizados en la arquitectura y en la parte (b) la frecuencia máxima de operación.

[12] El dispositivo FPGA Cyclone II de ALTERA es de una tecnologia de fabricación de 90nm.



Flow Summary	
Flow Status	Successful - Thu Jul 07 17:10:57 2011
Quartus II Version	11.0 Build 157 04/27/2011 SJ Web Edition
Revision Name	fme_hp_qp
Top-level Entity Name	fme_hp_qp
Family	Cyclone II
Device	EP2C35F672C6
Timing Models	Final
Total logic elements	14,774 / 33,216 (44 %)
Total combinational functions	13,107 / 33,216 (39 %)
Dedicated logic registers	8,686 / 33,216 (26 %)
Total registers	8686
Total pins	217 / 475 (46 %)
Total virtual pins	0
Total memory bits	0 / 483,840 (0 %)
Embedded Multiplier 9-bit elements	0 / 70 (0 %)
Total PLLs	0 / 4 (0 %)

(a)

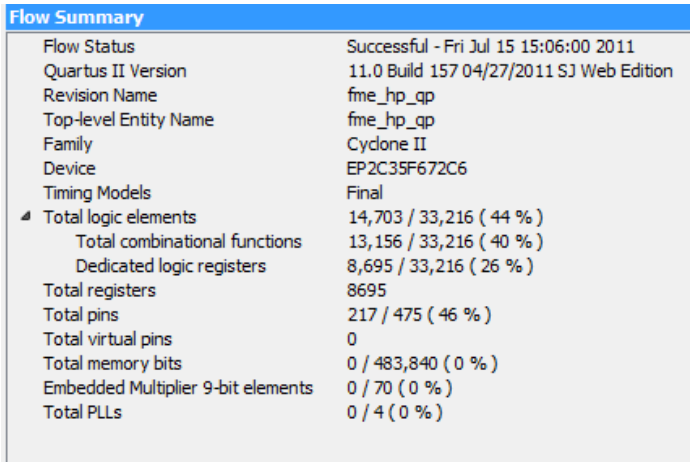


Slow Model Fmax Summary				
	Fmax	Restricted Fmax	Clock Name	Note
1	81.49 MHz	81.49 MHz	CLK	

(b)

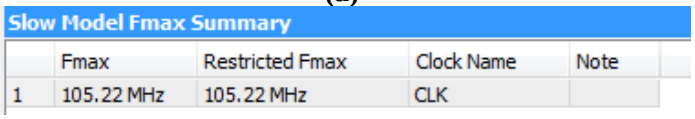
Fig. 27 Resultados de síntesis de la arquitectura diseñada para el FPGA Cyclone II (a) Frecuencia Máxima de Operación (b).

Asimismo esta tasa de procesamiento aún puede mejorar si se insertan más etapas de pipeline lo que en teoría permitiría incrementar el valor de la frecuencia máxima de operación de la arquitectura diseñada. A partir de esta hipótesis se adiciono registros para la salida del Banco de Filtros Bilineales y un registro para la señal de aviso proveniente de la Unidad de Interpolación Quarter-Pixel. Los resultados de la síntesis después de esta modificación se muestran en la **Figura 28**.



Flow Summary	
Flow Status	Successful - Fri Jul 15 15:06:00 2011
Quartus II Version	11.0 Build 157 04/27/2011 SJ Web Edition
Revision Name	fme_hp_qp
Top-level Entity Name	fme_hp_qp
Family	Cyclone II
Device	EP2C35F672C6
Timing Models	Final
Total logic elements	14,703 / 33,216 (44 %)
Total combinational functions	13,156 / 33,216 (40 %)
Dedicated logic registers	8,695 / 33,216 (26 %)
Total registers	8695
Total pins	217 / 475 (46 %)
Total virtual pins	0
Total memory bits	0 / 483,840 (0 %)
Embedded Multiplier 9-bit elements	0 / 70 (0 %)
Total PLLs	0 / 4 (0 %)

(a)



Slow Model Fmax Summary				
	Fmax	Restricted Fmax	Clock Name	Note
1	105.22 MHz	105.22 MHz	CLK	

(b)

Fig. 28 Resultados de síntesis de la arquitectura optimizada para el FPGA Cyclone II (a) Frecuencia Máxima de Operación (b).

El resultado refleja un aumento de la frecuencia máxima de operación probando de esa manera nuestra hipótesis. Luego a partir de este resultado obtenido se puede deducir si la arquitectura es capaz de procesar secuencias HDTV con el requerimiento de 30 cuadros por segundo. Considerando las variaciones de proceso en la implementación de la arquitectura diseñada en el FPGA se tomará un valor de 90% de la frecuencia máxima de operación para determinar la tasa de procesamiento de secuencias de video por parte de la arquitectura. Tomado los datos mostrados a continuación se podrá hacer el cálculo correspondiente.

- 78 ciclos/bloque 8x8
- 32400 bloques de 8x8/ Cuadro HDTV
- Frecuencia de Reloj Máxima: (105.22 MHz)* 90% = 94.7MHz

(78 Ciclos/bloque)*(32400 bloque/Cuadro HDTV)*(1/94.7MHz) = 26.69mseg/Cuadro HDTV

(1/26.69mseg/Cuadro HDTV) = 37.46 cuadros/seg (fps)

De este resultado se puede concluir que se está cumpliendo con el objetivo principal del presente trabajo de tesis: diseñar una arquitectura hardware del algoritmo de FME Quarter-Pixel para secuencias HDTV a 30 cuadros por segundo.

Adicionalmente se sintetizo la arquitectura en un dispositivo de mejores características tecnológicas como el Stratix II[13] EP2S15F484C3 de la compañía ALTERA® (**ANEXO A**) con la finalidad de obtener una tasa de procesamiento mayor. Los resultados de la síntesis de la arquitectura otorgaban una frecuencia máxima de operación de 127.49 MHz, con la cual restringiendo su valor a un 90%, se obtenía una frecuencia de operación de 114.71 MHz. Con este valor se obtuvo una tasa de procesamiento de 45.39 cuadros por segundo para una secuencia de video con resolución HDTV.

4.5 Comparación con trabajos de la bibliografía:

Para poder realizar una comparación en las misma condiciones que los trabajos previos [18][38][45], la arquitectura diseñada fue sintetizada para el FPGA Virtex II XC2V8000 (hoja técnica adjunta en el **ANEXO A**) de la compañía XILINX®, el cual fue utilizado en los trabajos [18][38][45] pero no se logró alcanzar la frecuencia de operación máxima requerida para procesamiento en tiempo real. Asimismo se sintetizó la arquitectura para el FPGA Virtex 4 XC4VLX15 (hoja técnica adjunta en el **ANEXO A**) utilizada en [18] pero la cantidad de elementos lógicos del dispositivo era menor a los requeridos por la arquitectura

[13] El dispositivo FPGA Stratix II de Altera es de una tecnologia de fabricación de 90nm.

diseñada. El objetivo de sintetizar la arquitectura diseñada en los dispositivos FPGA mencionados fue el de realizar una comparación bajo las mismas condiciones que los trabajos previos, sin embargo eso el uso de un dispositivo de tecnología superior como el FPGA Cyclone II permite lograr el objetivo principal.

En la **Tabla 10** se muestra los resultados obtenidos en los trabajos mencionados junto con los del presente trabajo, los cuales resultan siendo favorables con respecto a los demás debido a que sólo se ha procesado secuencias de resolución VGA (640x480) en tiempo real en [38] mientras que el presente trabajo alcanza una frecuencia de operación máxima de 94.7 MHz lo cual permite procesar secuencias HDTV (1920x1080) en tiempo real. Esto se debe principalmente a la tecnología de 90nm del FPGA Cyclone II donde los tiempos de retrasos son menores en las conexiones internas del dispositivo en comparación con la tecnología más antigua de 150nm del FPGA Virtex II XC2V8000 en donde los retrasos son mayores.

Tabla 10: Cuadro Comparativo del presente trabajo con trabajos previos.

	Yalcin [45]	**Oktem [38]**	**Correa [18]**	**Trabajo Presente**
Frecuencia Máxima (MHz)	85	60	140	**94.7**
Ciclos/MB	768	1472	148	**312**
Resolución	HDTV (1920x1080)	VGA (640x480)	QHDTV (3840x2048)	**HDTV (1920x1080)**
Dispositivo FPGA	Xilinx Virtex II (150 nm)	Xilinx Virtex II (150 nm)	Xilinx Virtex 4 (90 nm)	**Altera Cyclone II (90 nm)**
FME Quarter-Pixel	No	Si	No	**Si**

Conclusiones:

- El alto grado de paralelismo de operaciones del algoritmo y la inserción de etapas de pipeline permitieron diseñar una arquitectura hardware con una Frecuencia de Operación Máxima de 105.22 MHz para el dispositivo FPGA Cyclone II de la compañía ALTERA®. Se restringe a un 90% de la frecuencia máxima de operación para garantizar el funcionamiento de la arquitectura ante variaciones de proceso en el momento de la implementación, lo cual permite procesar una secuencia de resolución HDTV (1920x1080) a 37.46 cuadros/segundo. Con este resultado el objetivo principal del presente trabajo de tesis ha sido alcanzado.

- Los resultados obtenidos por medio de la aplicación en el entorno de programación MATLAB® permitieron demostrar que al aplicar el algoritmo de Estimación de Movimiento Fraccional Quarter-Pixel a una secuencia de video, la suma de SADs mínimos obtenidos por cada cuadro es menor que en los casos de Estimación de Movimiento Entera o Fraccional con precisión Half-Pixel, con lo cual se puede alcanzar mayores tasas de compresión al momento de realizar la codificación de los cuadros residuales.

- La metodología de Verificación Funcional por Testbench es adecuada para diseños de mediana a gran envergadura a ser validados, como el del presente trabajo, debido a los grados de libertad que permite respecto de la simulación por vector de onda. Un ejemplo de esto fue la validación del protocolo Handshake mediante el uso de la instrucción WAIT UNTIL.

- La arquitectura diseñada tiene la característica de portabilidad, lo cual permite en un futuro formar parte de todo un sistema digital de mayor complejidad como el CODEC H.264/AVC implementado en *Hardware*.

- El desarrollo de equipos basados en FPGAs o ASIC para un CODEC de formato H.264/AVC con la finalidad de procesar secuencias HDTV en tiempo real es perfectamente factible, como se demostró, sin tener la necesidad de utilizar dispositivos de última generación del mercado como los de tecnología de 28 nm como los Stratix V y los Cyclone V de la compañía ALTERA®.

Recomendaciones

- Implementar el diseño propuesto sobre un FPGA de la familia ALTERA® o XILINX® para validar el funcionamiento interno de la arquitectura ante variaciones de procesos. Para ello se usarán herramientas analizadoras lógicas como el SignalTap o el Chipscope en el caso de ALTERA® y de XILINX® respectivamente.

- Debe usarse la metodología de diseño bajo verificación (DUV por sus siglas en inglés) para los trabajos de tesis en el área de microelectrónica, que es la metodología que se usa a nivel industrial para la verificación funcional del sistema descrito. En el presente trabajo de tesis se usó la Verificación Funcional por Testbench, siendo esta un caso particular de la DUV.

- Modificar la arquitectura diseñada aprovechando otros procesos que pueden ser ejecutados en paralelo y que no se tomaron en cuenta en el presente trabajo. Asimismo la cantidad de recursos hardware también puede ser optimizada a partir de procesos que puedan ser ejecutados de manera secuencial a cambio algunos ciclos de reloj adicionales.

- Implementar otras características del estándar H.264/AVC para Estimación de Movimiento tomando como base la arquitectura diseñada, para el uso de bloques de tamaños variables, uso de múltiples cuadros de referencia, etc.

- Se propone el diseño de los demás módulos hardware que conforman la arquitectura de un Codificador o un Decodificador según el formato H.264/AVC, que posteriormente a la etapa de implementación en prototipo en un FPGA, pueda ser llevado a un ASIC utilizando las herramientas de CADENCE bajo una tecnología de 90nm o superior cuya finalidad sea el uso en equipos electrónicos portátiles y de bajo consumo de energía.

Bibliografía:

[1] Agostini, L. V. Desenvolvimento de Arquiteturas de Alto Desempenho Dedicadas à Compressão de Vídeo Segundo o Padrão H.264/AVC. *Universidade Federal do Rio Grande do Sul. Instituto de Informática. Programa de Pós-Graduaçaõ em Computaçaõ,* 2007.

[2] Thomas Wiegand, Gary J. Sullivan, G. B. A. L. Overview of the H.264/AVC Video Coding Standard. *IEEE Transactions on Circuits and Systems for Video Technology,* 2003, *13*, 570-576.

[3] "Draft ITU-T recommendation and final draft international standard of joint video specification (ITU-T Rec. H.264/ISO/IEC 14 496-10 AVC," in *Joint Video Team (JVT) of ISO/IEC MPEG and ITU-T VCEG*, JVTG050, 2003.

[4] INTERNATIONAL TELECOMMUNICATION UNION. ITU-T Recommendation H.261 v1 (11/90): video codec for audiovisual services at px64kbit/s. [S.1.], 1990.

[5] INTERNATIONAL TELECOMMUNICATION UNION. ITU-T Recommendation H.262 (11/90): generic coding of moving pictures and associated audio information - part 2: video. [S.1.] 1990.

[6] Gary J. Sullivan, T. W. Video Compression - From Concepts to the H.264/AVC Standard. *Proceedings of the IEEE,* 2005, *93*, 18-31

[7] A. Rodrigues, N. Roma, and L. Sousa, "p264: Open Platform for Designing Parallel H.264/AVC Video Encoders on Multi-core Systems". *International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV 2010)*, Junio 2010, pp. 81–86.

[8] http://www.atsc.org/cms/. *Último Acceso 19/02/2011.*

[9] http://www.dvb.org/. *Último Acceso 01/03/2011.*

[10] http://spectrum.ieee.org/consumer-electronics/standards/does-china-have-the-best-digital-television-standard-on-the-planet. *Último Acceso 19/02/2011.*

[11] http://www.dibeg.org/. *Último Acceso 19/02/2011*.

[12] ITU-T Recommendation ITU-T H.264 (03/2010): Advanced video coding for generic audiovisual services. *International Telecommunication Union.*

[13] http://tvdigitalperu.mtc.gob.pe/index2.html. *Último Acceso 19/02/2011.*

[14] http://www.mtc.gob.pe/portal/tdt/comunicado.html. *Último Acceso 22/02/2011.*

[15] Mayorga, Marco. Propuestas y avances de la Televisión Digital en la PUCP. *Seminario Técnico – Universidad Ricardo Palma*. Febrero, 2009.

[16] Henkel, J. & Parameswaran, S. Designing Embedded Processors: A Low Power Perspective. *Springer*. 2007.

[17] Richardson, I. E. G. H.264 and MPEG-4 Video Compression: Video Coding for Next-generation Multimedia. *John Wiley & Sons Ltd.*, 2003

[18] Marcel Moscarelli Corrêa, Mateus Thurow Schoenknecht, L. V. A. "A High Performance *Hardware* Architecture for the H.264/AVC Half-Pixel Motion Estimation Refinement". *SBCCI '10 Proceedings of the 23rd symposium on Integrated circuits and system design*, 2010, 151-156.

[19] Marcel Corrêa, Mateus Schoenknecht, R. D. L. A. Desenvolvimento de uma Arquitetura para Interpolação de Half-Pixels segundo o padrão H.264/AVC. *XVI Workshop Iberchip 2010*, 2010.

[20] JM15.1. "H.264/AVC JM Reference Software." Mar, 2010; http://iphome.hhi.de/suehring/tml.

[21] Puri, A. Video coding using the H.264/MPEG-4 AVC compression standard *Signal Processing: Image Communication 19,* 2004.

[22] Vasudev Bhaskaran, K. K. Springer *(Ed.)* Image and video compression standards: algorithms and architecture. *Kluger Academic Publisher,* 1997.

[23] Gustavo A. Ruiz, J. A. M. "An Efficient VLSI Architecture of Fractional Motion Estimation in H.264 for HDTV". *Journal of Signal Processing Systems- Springer*, 2010, *62*, 443-457.

[24] http://www.ti.com/ww/mx/prod_dsp.html. *Último Acceso 14/07/2011*.

[25] Meyer-Baese, U. Digital Signal Processing with Field Programmable Gate Arrays. 3ra Edición. *Springer* (Ed.) 2004.

[26] http://developer.nvidia.com/what-cuda. *Último Acceso 14/07/2011*.

[27] Llamocca D., Carranza C. & Pattichis, M. "Separable FIR Filtering in FPGA and GPU Implementations: Energy, Performance, and Accuracy Considerations". *21st International Conference on Field Programmable Logic and Applications FPL 2011,* 2011

[28] Raffo, M. "Desenvolvimento de um Sistema Dinamicamente Reconfigurável Baseado em Redes Intra-Chip e Ferramenta para Posicionamento de Módulos". *Escola Politécnica da Universidade de Sao Paulo,* 2010.

[29] http://model.com/. *Último Acceso 14/07/2011*.

[30] Maxfiled, C. "M. FPGAs: World Class Designs. *ELSEVIER* (Ed.). Newnes, 2009.

[31] Gajski, D. D. & Kuhn, R. H. New VLSI Tools-Guest Editor's Introduction. *IEEE Computer,* 1983, 11-14.

[32] http://www.xilinx.com/. *Último Acceso 14/07/2011*.

[33] http://www.altera.com/. *Último Acceso 14/07/2011*.

[34] Maxfiled, C. "M. The Design Warrior's Guide to FPGAs. *ELSEVIER* (Ed.), *Newnes*, 2004.

[35] http://www.sigmadesigns.com/. *Último Acceso 01/03/2011.*

[36] Hoelzer, Steve. MPEG-2 overview and MATLAB codec project. *Universidad de Illinois*. Software de Referencia Disponible en: http://www.cs.cf.ac.uk/Dave/Multimedia/Lecture_Examples/Compression/mpegproj/#ipb.

[37] Youn-Long Steve Lin, Chau-Yang Kao, H.-C. K. & Chen, J.-W. VLSI Design for Video Coding: H.264/AVC Encoding from Standard Specification to Chip. Springer *(Ed.)*, 2010.

[38] Oktem, S. & Hamzaoglu, I. "An Efficient Hardware Architecture for Quarter-Pixel Accurate H.264 Motion Estimation". *DSD '07 Proceedings of the 10th Euromicro Conference on Digital System Design Architectures, Methods and Tools,* 2007, 444-447.

[39] Vahid, F. & Givargis, T. Embedded System Design: A Unified Hardware/Software Approach. *Wiley & sons* (Ed.), 1999.

[40] http://www.cs.cf.ac.uk/Dave/Multimedia/Lecture_Examples/Compression/mpegproj/#ipb *Último Acceso 10/05/2011.*

[41] http://media.xiph.org/video/derf/ *Último Acceso 14/05/2011.*

[42] http://www.cipr.rpi.edu/resource/sequences/ *Último Acceso 14/05/2011.*

[43] http://wiki.xiph.org/OggYUV4MPEG *Último Acceso 13/05/2011.*

[44] http://mjpeg.sourceforge.net/ *Último Acceso 14/05/2011.*

[45] S. Yalcin, I. Hamzaoglu, "A High Performance Hardware Architecture for Half-Pixel Accurate H.264 Motion Estimation", 14th International Conference on VLSI-SoC, October 2010.