PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ

FACULTAD DE CIENCIAS E INGENIERÍA

DISEÑO DE UNA ARQUITECTURA PARA ESTIMACIÓN DE MOVIMIENTO FRACCIONAL SEGÚN EL ESTÁNDAR DE CODIFICACIÓN HEVC PARA VIDEO DE ALTA RESOLUCIÓN EN TIEMPO REAL

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

Jorge Guillermo Martín Soto León

ASESORES:
Dr. Carlos Silva Cárdenas
MSc. Cristopher Villegas Castillo

Lima, Junio de 2016

FACULTAD DE
CIENCIAS E
INGENIERÍA

PONTIFICIA
UNIVERSIDAD
CATÓLICA
DEL PERÚ

TEMA DE TESIS PARA OPTAR EL TÍTULO DE INGENIERO ELECTRÓNICO

Título	:	Diseño de una arquitectura para Estimación de Movimiento Fraccional según el estándar de codificación HEVC para video de alta resolución en tiempo real.
Área	:	Circuitos y Sistemas #1305
Asesor	:	Carlos Silva Cárdenas – Ernesto Cristopher Villegas Castillo
Alumno	:	Jorge Guillermo Martín Soto León
Código	:	20100036
Fecha	:	24/08/2015

PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ
Facultad de Ciencias e Ingeniería
12 NOV. 2015
MIGUEL MEJIA PUENTE
DECANO

MÁXIMO 50 PÁGINAS

Descripción y Objetivos

La compresión de video digital ha evolucionado tecnológicamente de manera notoria en los últimos años gracias al trabajo, entre otros, de organizaciones especializadas como ITU-T Video Coding Experts Group e ISO/IEC Moving Picture Experts Group. Los esfuerzos en conjunto permitieron el desarrollo del actual estándar de codificación denominado H.264/AVC el cual presenta una alta de tasa compresión de video digital permitiendo el desarrollo de una variedad de aplicaciones de alta demanda tanto a nivel de software como de hardware. Sin embargo, las exigencias actuales por servicios de mayor resolución han propiciado el desarrollo de un nuevo estándar de codificación denominado "*High Efficiency Video Coding*" (HEVC) el cual presenta una mayor tasa de compresión que el estándar H.264/AVC sin afectar la calidad de imagen percibida por los usuarios. Esto fue posible gracias a la inclusión de nuevas características y cambios como por ejemplo aquellos realizados en el algoritmo de Estimación de Movimiento Fraccional (FME). A partir de esto, el presente trabajo de tesis tiene como objetivo principal diseñar una arquitectura hardware del algoritmo FME según el formato HEVC para secuencias de video con resolución Full HD (1920x1080 píxeles) considerando bloques de área cuadrada y tamaño 8x8 píxeles. Los objetivos específicos son los siguientes:

PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ
FACULTAD DE CIENCIAS E INGENIERÍA
M. Sc. Ing. MIGUEL ANGEL CATAÑO SÁNCHEZ
Coordinador de la Especialidad de Ingeniería Electrónica

1) Diseñar la arquitectura del algoritmo FME en RTL y realizar su descripción en VHDL.

2) Evaluar el comportamiento del RTL a través de verificación funcional por medio de simulaciones empleando Test Benches descritos en VHDL.

3) Sintetizar la arquitectura para los dispositivos FPGA de la familia Virtex de Xilinx.

4) Verificar y optimizar los resultados de máxima frecuencia de operación de la arquitectura así como el área ocupada del dispositivo.

FACULTAD DE
CIENCIAS E
INGENIERÍA

PONTIFICIA
UNIVERSIDAD
CATÓLICA
DEL PERÚ

TEMA DE TESIS PARA OPTAR EL TÍTULO DE INGENIERO ELECTRÓNICO

Título : Diseño de una arquitectura para Estimación de Movimiento Fraccional según el estándar de codificación HEVC para video de alta resolución en tiempo real.

Índice

PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ
FACULTAD DE CIENCIAS E INGENIERÍA

M. Sc. Ing. MIGUEL ANGEL CATAÑO SÁNCHEZ
Coordinador de la Especialidad de Ingeniería Electrónica

MÁXIMO 50 PÁGINAS

Resumen

Las labores de organizaciones especializadas como ITU-T Video Coding Experts Group e ISO/IEC Moving Picture Experts Group han permitido el desarrollo de la codificación de video a lo largo de estos años. Durante la primera década de este siglo, el trabajo de estas organizaciones estuvo centrado en el estándar H.264/AVC; sin embargo, el incremento de servicios como transmisión de video por Internet y redes móviles así como el surgimiento de mayores resoluciones como 4k u 8k llevó al desarrollo de un nuevo estándar de codificación denominado HEVC o H.265, el cual busca representar los cuadros de video con menor información sin afectar la calidad de la imagen.

El presente trabajo de tesis está centrado en el módulo de Estimación de Movimiento Fraccional el cual forma parte del codificador HEVC y presenta una elevada complejidad computacional. En este trabajo, se han tomado en cuenta las mejoras incluidas por el estándar HEVC las cuales radican en los filtros de interpolación empleados para calcular las muestras fraccionales.

Para verificar el algoritmo, se realizó la implementación del mismo utilizando el entorno de programación MATLAB®. Este programa también ha permitido contrastar los resultados obtenidos por medio de la simulación de la arquitectura.

Posteriormente, se diseñó la arquitectura teniendo como criterios principales la frecuencia de procesamiento así como optimizar la cantidad de recursos lógicos requeridos. La arquitectura fue descrita utilizando el lenguaje de descripción de hardware VHDL y fue sintetizada para los dispositivos FPGA de la familia Virtex los cuales pertenecen a la compañía Xilinx®. La verificación funcional fue realizada por medio de la herramienta ModelSim empleando Testbenchs.

Los resultados de máxima frecuencia de operación fueron obtenidos por medio de la síntesis de la arquitectura; adicionalmente, por medio de las simulaciones se verificó la cantidad de ciclos de reloj para realizar el algoritmo. Con estos datos se puede fundamentar que la arquitectura diseñada es capaz de procesar secuencias de video HDTV (1920x1080 píxeles) a una tasa de procesamiento mayor o igual a 30 cuadros por segundo.

Índice

Anexos (ver CD adjunto)

Anexo A: Diagrama de Estados Controlador Buffer
Diagrama de Estados Controlador Interpolación
Diagrama de Estados Controlador SAD ½
Diagrama de Estados Controlador Acumulador SAD ½
Diagrama de Estados Controlador SAD ¼
Diagrama de Estados Controlador Acumulador SAD ¼

Anexo B: Resultados de la simulación de la arquitectura y resultados obtenidos por medio del software desarrollado.

Anexo C: Resultados de la simulación de la arquitectura para la secuencia de video “speed_bag_1080p.y4m”.

Anexo D: Resultados de la aplicación software para las secuencias de video disponibles.

Anexo E: Código Fuente de la aplicación desarrollada en Matlab®.

Anexo F: Archivo de proyecto de la arquitectura obtenido por medio del software ISE incluyendo los archivos de extensión *.vhd los cuales contienen el código VHDL de los módulos que conforman la arquitectura.

Anexo H: Lista de Abreviaturas

A mi Mami Trini y Papi Moisés por enseñarme con el ejemplo una vida llena de sabiduría, valores e integridad y por brindarme todo su amor

A mis padres, Jorge y Carmen, por apoyarme, mostrarme grandes enseñanzas de vida y darme todo su cariño y amor

A mi hermano, Fernando por siempre alentarme a continuar

A la Sra. Chepita por acompañarme, cuidarme y ayudarme durante esta etapa maravillosa de mi vida.

A mis amigos y hermanos Victor, Maxell y Jhon quienes me han acompañado desde el colegio

Al Prof. Carlos Silva y al Ing. Cristopher Villegas por brindarme su apoyo y conocimiento en todo momento

A todos ellos, Muchas Gracias.

Introducción

En la actualidad, el tráfico de video representa una importante carga en las redes de datos así como en los sistemas de almacenamiento y procesamiento es por ello que ante requerimientos de mayor resolución en las secuencias de video, se necesitó el desarrollo de un nuevo estándar que pudiera mejorar las características del estándar H.264/AVC el cual es ampliamente utilizado.

El nuevo estándar de codificación denominado HEVC, cuyas siglas provienen de su nombre en inglés “High Efficiency Video Coding”, ha sido desarrollado teniendo como principal objetivo disminuir la tasa de transmisión de datos sin afectar la calidad de la resolución. Con la finalidad de conseguir este objetivo fundamental, se realizaron diversas mejoras en comparación con el estándar H.264/AVC en especial aquellas involucradas en el proceso de Estimación de Movimiento Fraccional, el cual presenta una elevada complejidad y emplea un importante tiempo de codificación. Los cambios realizados en el proceso de Estimación de Movimiento Fraccional involucran nuevos filtros de interpolación con lo cual es posible mejorar la etapa fraccional en comparación con la versión previa del estándar.

El presente trabajo de tesis se centra en el diseño de una arquitectura hardware para el proceso de Estimación de Movimiento Fraccional teniendo como objetivo conseguir una tasa de procesamiento mayor o igual a 30 cuadros por segundo para secuencias de video con resolución HDTV (1920x1080 píxeles).

El texto del presente trabajo de tesis está organizado de la siguiente manera: el capítulo 1 presenta una introducción al formato HEVC así como una explicación de la problemática. En el capítulo 2, se detallarán conceptos importantes sobre el estándar HEVC así como sobre Estimación de Movimiento, en el capítulo 3 se presentarán las arquitecturas diseñadas para realizar el algoritmo de Estimación Fraccional. En el capítulo 4, se presentarán los resultados obtenidos por medio de la implementación en software y hardware del algoritmo y finalmente se detallarán las conclusiones y recomendaciones del presente estudio.

Capítulo 1

Compresión de Video Digital en el formato HEVC

1.1 Declaración de la problemática

Los servicios de video digital son empleados cotidianamente en nuestra vida, en cada ambiente es posible encontrar dispositivos capaces de capturar, transmitir y mostrar video digital. Los avances tecnológicos han permitido que el usuario pueda grabar y visualizar contenido en ultra alta definición (UHD del inglés, "*Ultra High Definition*") inclusive desde el celular como sucede con el último equipo de la familia Galaxy de SAMSUNG® que puede grabar video a una resolución UHD 4K (3840 x 2160 píxeles) con un procesamiento de 30 cuadros por segundo (fps del inglés, "*Frames per second*") y además cuenta con una pantalla principal con una resolución "Quad HD" (2560 x 1440 píxeles) [1].

La televisión digital también ha mostrado avances en los últimos años, es así que en Diciembre del 2014, se logró en Alemania la transmisión del primer concierto en vivo en televisión digital utilizando una resolución 4K la cual fue posible gracias a los importantes esfuerzos y colaboración del Instituto Fraunhofer Heinrich Hertz [2]. Adicionalmente a los servicios de televisión digital, por Internet circulan datos que corresponden a video en alta definición (HD del inglés, "*High Definition*") el cual se espera aumente su resolución en los próximos años debido a los avances tecnológicos y supone también un importante reto para las redes de datos [4].

El desarrollo de las aplicaciones mencionadas ha sido posible gracias a la evolución de la codificación de video ya que ha permitido minimizar la tasa de transmisión necesaria para representar el contenido sin afectar la calidad de la imagen; adicionalmente la codificación de video permite reducir el espacio de almacenamiento necesario ya que busca optimizar la eficiencia de la codificación [4].

A lo largo de los años, se han desarrollado diversos estándares de codificación de video gracias a organizaciones especializadas como ITU-T Video Coding Experts Group (VCEG) e ISO/IEC Moving Picture Experts Group (MPEG)

quienes en conjunto formaron el Joint Colaborative Team on Video Coding (JCT-VC) para el desarrollo del estándar H.262/MPEG-2 y H.264/AVC. Actualmente, el estándar H.264 es utilizado en diversos servicios como transmisión de televisión digital HD, video a través de redes móviles, entre otras; sin embargo, debido a los avances tecnológicos y a los requerimientos actuales y futuros por mayores resoluciones de video como 4K u 8K (7680x4320 píxeles), fue necesaria la evolución a un nuevo estándar de compresión de video llamado HEVC, del inglés "*High Efficiency Video Coding*" o también conocido como H.265 [3].

El formato de codificación HEVC ha sido diseñado para satisfacer los requerimientos de las actuales aplicaciones que emplean el formato H.264 así como reducir en un 50% la tasa de transmisión sin afectar la calidad de video. Una de las herramientas más importantes empleadas por el estándar HEVC es el proceso de Estimación de Movimiento (ME del inglés, "*Motion Estimation*"). El ME es de importante relevancia para el proceso de codificación porque permite trabajar sobre la redundancia temporal presente en las secuencias de video; sin embargo, utiliza un importante tiempo del proceso de codificación debido a la elevada complejidad que implica realizar este proceso [4].

Disminuir la carga computacional y alcanzar un procesamiento en tiempo real, es decir una tasa de procesamiento mayor o igual a 30 fps, para secuencias de alta definición se vuelve una tarea difícil para soluciones basadas en software debido a su ejecución secuencial, es así que para conseguir una implementación eficiente del estándar HEVC es necesario explotar el paralelismo de operaciones [4].

En el contexto peruano, el gobierno sentó las bases para la transición de televisión analógica a digital, mediante Resolución Suprema N° 019-2009-MTC del 24 de Abril del 2009 y con base en las propuestas de la Comisión Multisectorial encargada de analizar las propuestas, se decidió adoptar el estándar Integrated Services Digital Broadcasting Terrestrial (ISDB-T) con adaptaciones brasileñas conocido como el estándar Japonés – Brasileño. Una de las ventajas del estándar mencionado es la posibilidad de cooperación internacional en el desarrollo así como la transferencia tecnológica [5].

De acuerdo a lo expuesto en los párrafos anteriores, se plantea generar el desarrollo de arquitecturas hardware destinadas al proceso de codificación según el estándar HEVC las cuales puedan aprovechar el paralelismo de operaciones y de ese modo descongestionar las cargas de procesamiento requeridas ya que se trata de un proceso de codificación complejo en especial cuando se está llevando a cabo el proceso de estimación de movimiento. Adicionalmente, el presente estudio permite realizar aportes al estándar adoptado por nuestro país logrando que se pueda transmitir video digital de una mayor resolución como 4K u 8K con una menor tasa de datos.

1.2 Declaración del Marco Problemático

Una de las características de los estándares de codificación es la flexibilidad que otorgan las organizaciones encargadas de su control para que los especialistas puedan brindar soluciones a los requerimientos del estándar, el requisito indispensable es que la trama de bits a transmitir se ajuste a los patrones determinados [4].

Gracias a la flexibilidad del estándar, existen numerosas investigaciones que buscan reducir la carga de procesamiento de la Estimación de Movimiento porque involucra una cantidad de tiempo considerable para su ejecución lo cual está estrechamente relacionado con el consumo de potencia en especial para los dispositivos portátiles en donde es necesario mantener la autonomía del equipo. Algunas de estas investigaciones afrontan las soluciones por medio de implementaciones en software como el trabajo realizado por Xiang-wen *et. al.* [24] y por otro lado existen trabajos realizados implementando arquitecturas hardware para el proceso de codificación como la investigación desarrollada por Nalluri *et. al* [6].

Las arquitecturas hardware presentan una solución óptima ya que funcionan como un acelerador hardware, es decir un circuito únicamente dedicado a la codificación de video con el cual se espera aprovechar en importante medida el paralelismo de operaciones para satisfacer los requerimientos de procesamiento en tiempo real, es decir 30 fps, para altas resoluciones de video.

Capítulo 2

Estudio del estándar de codificación HEVC y del proceso de Estimación de Movimiento Fraccional

2.1 Formato de Compresión de video digital HEVC

2.1.1 Conceptos relacionados a la codificación de video

El concepto de video digital consiste en una secuencia de imágenes representadas en bits con la finalidad de que su información pueda ser procesada por dispositivos electrónicos. Sin embargo, la gran cantidad de información encontrada en un archivo de video digital en alta definición, crea la necesidad de utilizar técnicas que ayuden a disminuir la cantidad de datos necesarios para representar los cuadros de video y así mejorar su transmisión y almacenamiento [7].

En este sentido, se puede decir que la información contenida en cada cuadro de una secuencia de video es una imagen digital compuesta por una matriz de píxeles que contienen la información de color. Existen diferentes mecanismos para representar información de color, los más representativos son los espacios de color RGB e YCbCr. El espacio RGB, cuyo nombre proviene de "*Red, Green & Blue*", representa a los diferentes colores y sus tonalidades por medio de proporciones entre el rojo, verde y azul por lo cual es necesario que las tres particiones sean representadas con la misma resolución para no afectar la calidad de la imagen; a diferencia de lo que sucede en el espacio de color RGB, la representación YCbCr aprovecha las características del sistema visual humano para reducir la cantidad de datos necesarios para representar los cuadros de video [7]. El espacio de color YCbCr representa la imagen por medio de muestras de luminancia y cromaticidad; resulta más eficiente separar la información de luminancia de la información de color con la finalidad de representar con mayor resolución a las componentes de luminancia porque es posible reducir la información sin afectar la calidad de la imagen ya que el sistema visual del ser humano es menos sensible a la información de color que a las componentes de luminancia [7].

Para disminuir la resolución de las muestras de cromaticidad, se aplican ciertos patrones de muestreo los cuales pueden ser los siguientes: 4:4:4, 4:2:2 y 4:2:0. En el caso del primer patrón de muestreo, se conserva la resolución para cada componente; en el segundo caso, se tiene la misma resolución vertical para las componentes de cromaticidad y luminancia pero se tiene la mitad de muestras horizontales para las componentes de cromaticidad; por último en el tercer caso, las componentes de cromaticidad tienen la mitad de resolución horizontal y vertical de las muestras de luminancia. Este último patrón de muestreo es ampliamente utilizado para aplicaciones como video conferencias, televisión digital y almacenamiento. En la figura 1, se puede observar los patrones de muestreo mencionados para el espacio de color YCbCr [7].

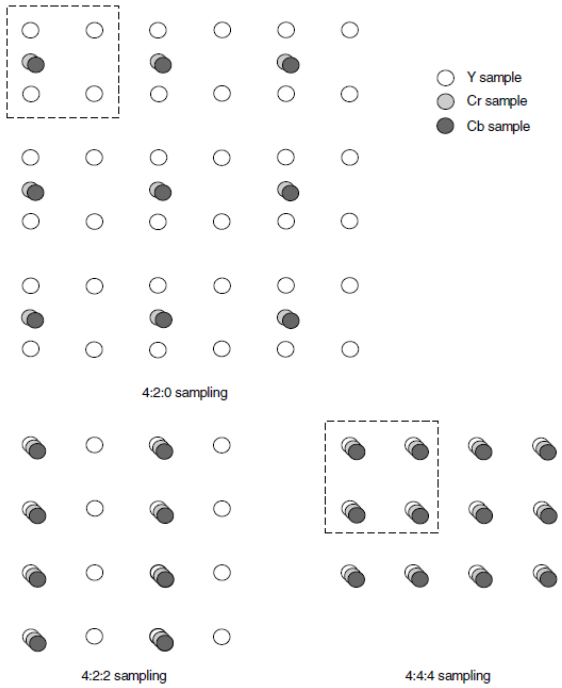


Fig. 1. Patrones de muestreo para el espacio de color YCbCr [7]

Para el observador no existirá diferencia notoria entre una imagen RGB y una imagen representada empleando algún patrón de muestreo en el espacio de color YCbCr, por lo tanto, esta técnica es una forma simple pero efectiva para ayudar a la compresión de las secuencias de video [7].

De acuerdo a lo mencionado, expresar las imágenes en un determinado espacio de color ayuda a la compresión de la información; sin embargo, la compresión

de datos es verdaderamente alcanzada cuando se eliminan las redundancias presentes en los cuadros de video. Las redundancias pueden ser del tipo espaciales o temporales; en el dominio espacial, generalmente existe una alta correlación entre píxeles que se encuentran cercanos entre sí; en el dominio temporal, existe una alta relación entre los cuadros de video que son capturados muy cercanos en el tiempo porque generalmente no se producen cambios abruptos en el movimiento de los diferentes objetos representados en las secuencias de video. Los codificadores de video se encargan de crear un modelo de la imagen, trabajando sobre las redundancias y de este modo representar a las secuencias, que se desean transmitir, con la menor cantidad de bits sin afectar la calidad de la imagen de los cuadros de video [7].

2.1.2 Formato de Codificación HEVC

Actualmente el tráfico de video representa una importante carga en la redes de comunicaciones y en los sistemas de almacenamiento por lo cual el estándar de codificación HEVC se presenta como una alternativa sólida para reducir la cantidad de datos así como permitir la transmisión de secuencias de video de una mayor resolución [3].

De acuerdo a información proporcionada por JCT-VC en [3], el codificador del formato HEVC debe estar compuesto por los elementos y conexiones mostradas en la figura 2.

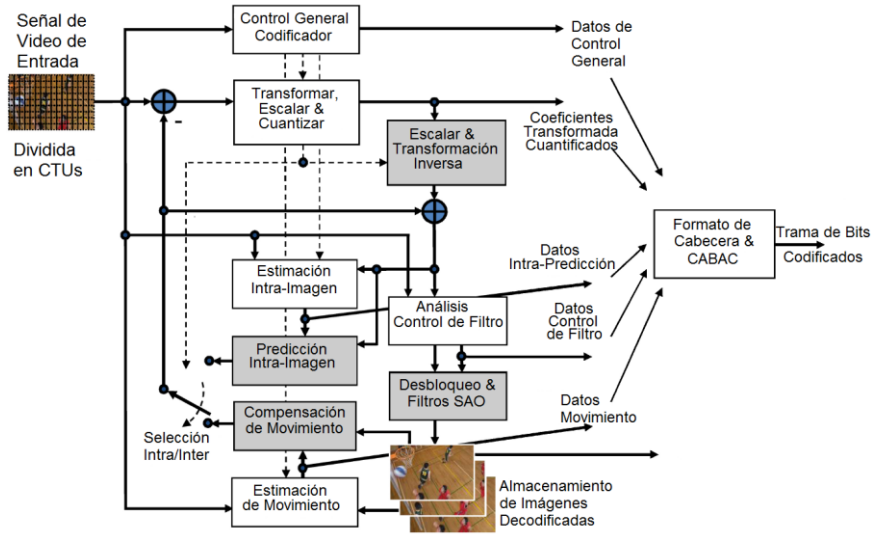


Fig. 2. Codificador de video HEVC [3].

El algoritmo de codificación debe proceder como se detalla a continuación. Cada imagen es dividida en bloques de muestras o píxeles, estos bloques deben ingresar al proceso de estimación intra-imagen o al proceso de estimación de movimiento. La estimación intra-imagen se encargará de disminuir la información trabajando únicamente sobre la redundancia espacial presente dentro del cuadro actual; por otro lado, la Estimación de Movimiento se encargará de disminuir la redundancia temporal utilizando para ello el cuadro actual y cuadros de referencia que hayan sido previamente codificados. Ambos procesos se encargarán de entregar una señal residual la cual es calculada por medio de la diferencia entre el cuadro actual y el cuadro estimado por ambos procesos mencionados. Posteriormente, la señal residual es procesada por una transformada espacial, mediante la cual se obtienen una serie de coeficientes que serán escalados, cuantificados y codificados para luego ser transmitidos junto con la información residual obtenida por medio de la predicción [3].

Para eliminar las redundancias espaciales es necesario contar con cuadros referencia es por ello que los coeficientes y la señal residual son adicionalmente procesados para reconstruir las secuencias transmitidas y que puedan ser almacenadas en un repositorio que se empleará para realizar la predicción de las nuevas secuencias a codificar [3]. Un detalle importante a destacar es que el primer cuadro de la secuencia de video es siempre proceso mediante la estimación intra-imagen ya que aún no se cuenta con información de cuadros previamente procesados por el codificador.

2.1.2.1 División de cuadros de video en el formato HEVC

El estándar de codificación HEVC ha sido diseñado bajo el principio de codificadores híbridos basados en bloques mediante los cuales las imágenes son divididas en conjuntos de muestras que a su vez forman identidades las cuales son independientemente codificables [11].

El estándar HEVC divide las imágenes mediante una estructura altamente flexible y eficiente, con diferentes unidades de acuerdo a sus funcionalidades, esta característica lo diferencia de los estándares desarrollados hasta el momento. Dentro del estándar HEVC, una imagen es dividida en bloques CTB

denominados así por su nombre en inglés "*Coding Tree Block*", cada CTB de muestras de luminancia en unión con ambos CTB de cromaticidad se reúnen en una unidad CTU denominada así por su nombre en inglés "*Coding Tree Unit*" y representan a la unidad básica de procesamiento en el estándar HEVC tal como se observa en la figura 3; los bloques CTB de las muestras de luminancia cubren un área cuadrada de la imagen de tamaño $2^n \, x \, 2^n$ en donde n puede ser 4, 5 o 6; por lo tanto, si se emplea el patrón de muestreo 4:2:0 del espacio de color YCbCr, cada CTB de cromaticidad cubriría un área de tamaño $2^{n-1} \, x \, 2^{n-1}$ muestras tal como se explicó en la sección 2.1.1 [11].

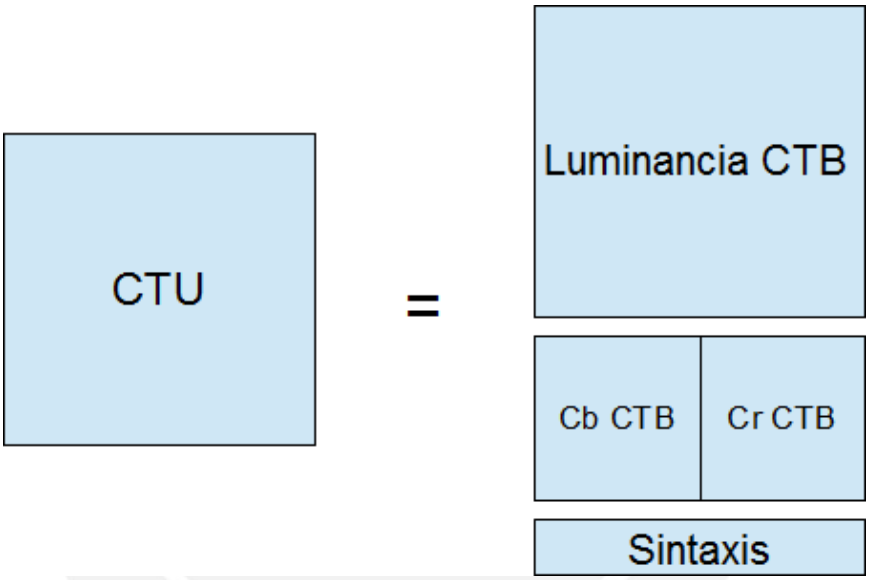


Fig.3. Combinación de Bloques CTB en una unidad CTU [9].

Cada conjunto CTU es adicionalmente divido en unidades de diversos tamaños denominadas CU por su nombre en inglés "*Coding Unit*" tal como se puede observar en la figura 4.

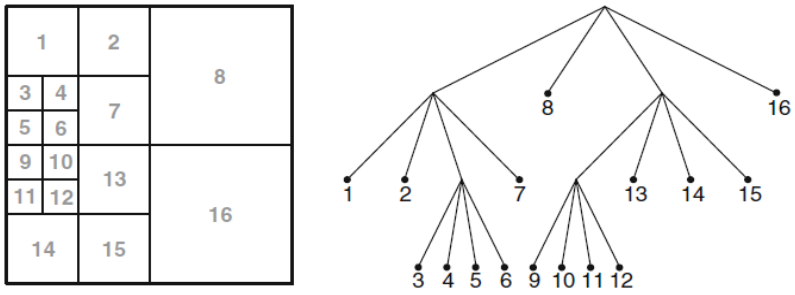


Fig. 4. División de CTU en unidades CU [3].

El tamaño de cada CU puede variar desde arreglos de 8 x 8 muestras hasta inclusive ser del tamaño de cada CTU con el objetivo de adaptarse a las características del cuadro de video a analizar. De igual modo como sucede con las unidades CTU, las componentes de luminancia y cromaticidad contenidas dentro de cada CU son denominadas CB por su nombre en inglés "*Coding Block*".

Los CUs son unidades de procesamiento a los cuales se les asigna un determinado tipo de codificación, es decir, si las muestras serán codificadas empleando la predicción intra-imagen o empleando la Estimación de Movimiento. En este último caso, las componentes de luminancia y cromaticidad son divididas en muestras denominadas PB por su nombre en inglés "*Prediction Block*". Cada bloque PB es un conjunto de muestras que utilizan los mismos parámetros de movimiento para eliminar la redundancia temporal, los parámetros a considerar deben incluir información sobre la imagen empleada como referencia del proceso de estimación.

El estándar HEVC incluye 8 diferentes modos de dividir cada CU en unidades de predicción tal como se observa en la figura 5.

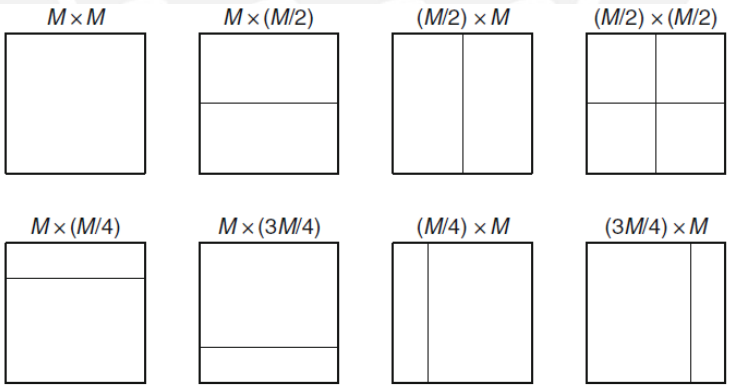


Fig. 5. Modos de partición de cada CU en unidades de predicción [11].

Adicionalmente a las divisiones mencionadas hasta ahora, los bloques CB son particionados en múltiples bloques denominados TB por su nombre en inglés "*Transform Block*". Los bloques TB son arreglos cuadrados de muestras en donde se aplica la misma transformada bi-dimensional para codificar la señal residual.

Para tener una correcta comprensión del estándar es necesario también incluir el concepto de los segmentos denominados "*slice*" los cuales permiten realizar operaciones independientes y en paralelo. Un "*slice*" puede ser una imagen completa así como divisiones de ella y considera a las unidades CTU como la mínima estructura que pueden contener. En el formato HEVC, un "*slice*" es definido como un conjunto de segmentos donde el primer segmento es independiente el cual es seguido de segmentos dependientes tal como se muestra en la figura 6 [4].

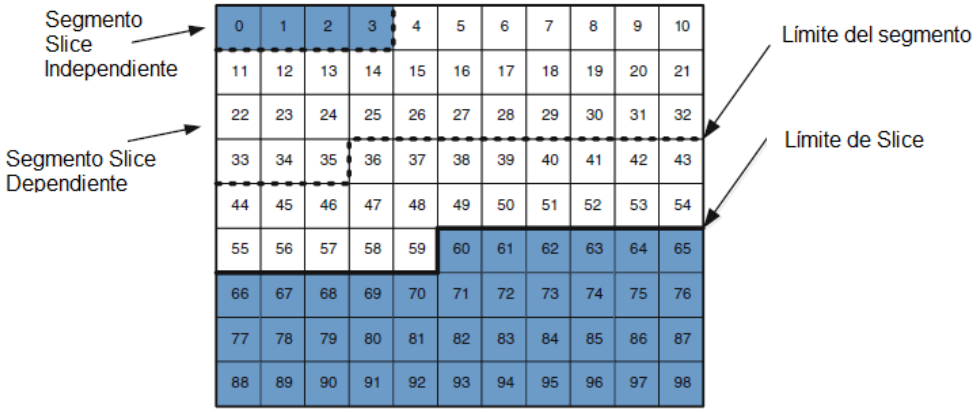


Fig. 6. Estructura de imagen en HEVC empleando slices [4].

2.2 Estimación de Movimiento

El movimiento en las secuencias de video puede ser estimado a partir de imágenes anteriormente decodificadas, el objetivo es encontrar una predicción para el bloque de muestras a codificar a través de un determinado conjunto de píxeles de una imagen de referencia con la finalidad de reducir la cantidad de datos para expresar cada cuadro ya que es posible almacenar y transmitir únicamente la diferencia entre ambos bloques en conjunto con información adicional que permitirá reconstruir cada secuencia codificada. Para cumplir con el objetivo mencionado, se analiza un conjunto de bloques de muestras dentro de una ventana de búsqueda en las imágenes empleadas como referencia. En la figura 7 se muestran al cuadro actual, al cuadro de referencia y el área de búsqueda para encontrar la mejor correlación.

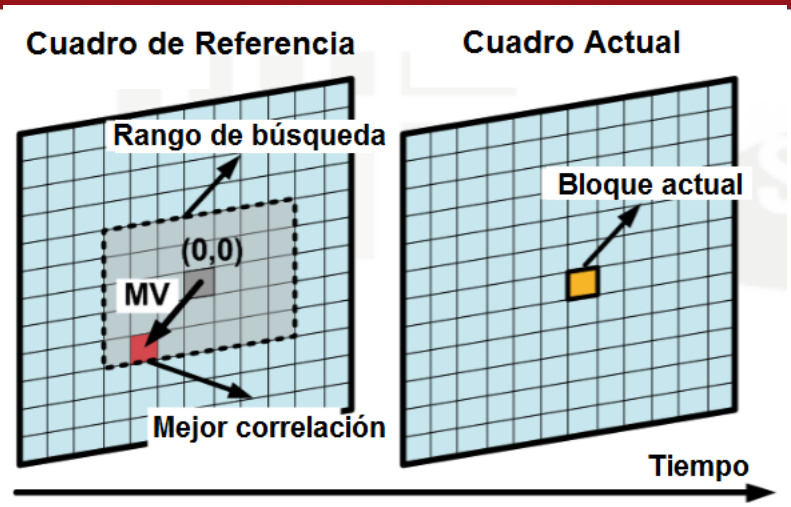


Fig. 7. Determinación de la mejor correlación entre PU del cuadro a codificar y el PU de la imagen de referencia [12].

Para determinar la mejor correlación entre los bloques se emplea el cálculo del SAD, MSE o MAE cuyas siglas provienen del inglés "*Sum of Absolute Diference*", "*Mean-Square Error*" y "*Mean Absolute Error*" y son expresados por las ecuaciones 1, 2 y 3 respectivamente. Generalmente, el SAD es el de mayor uso como criterio de determinación de correlación debido a que involucra un menor costo computacional es así que un menor valor de SAD infiere que existe una mejor correlación entre ambos cuadros [7].

$$MSE = \frac{1}{N^2} * \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} (Cij - Rij)^2 \qquad (1)$$

$$MAE = \frac{1}{N^2} * \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} |Cij - Rij| \qquad (2)$$

$$SAD = \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} |Cij - Rij| \qquad (3)$$

La posición del bloque de referencia de mayor correlación con el cuadro actual es señalada por medio de un vector de movimiento $(\Delta x, \Delta y)$ en donde los valores Δx y Δy determinarán el desplazamiento relativo horizontal y vertical tal como se

muestra en la figura 8. Adicionalmente es necesario incluir el índice Δt para ubicar la imagen a emplear dentro del conjunto de cuadros de referencia [4].

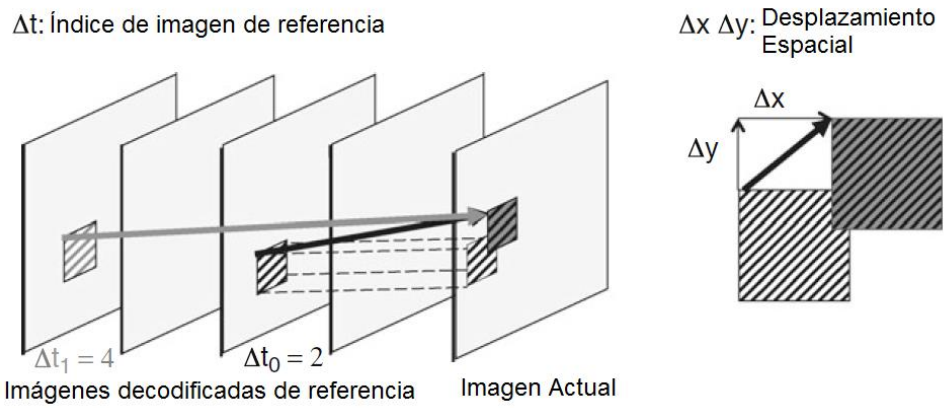


Fig. 8. Predicción inter-imagen e información de movimiento [4].

El estándar de codificación H.265 presenta la opción de determinar hasta dos vectores de referencia; en este caso se tendrá como resultado una estimación promedio la cual es producto de las estimaciones encontradas al emplear ambos vectores de movimiento es por ello que en la figura 8 se observan los índices ∆t0 y ∆t1 que hacen referencia a los cuadros empleados para determinar ambos vectores de movimiento [4].

2.2.1 Estimación de Movimiento Entera y Fraccional

El proceso descrito en la sección anterior se conoce como IME cuyas siglas provienen del inglés "*Integer Motion Estimation*" y es el primer paso dentro del proceso de Estimación de Movimiento.

Finalizado el IME, se realiza el segundo proceso denominado FME debido a su nombre en inglés "*Fractional Motion Estimation*". Este segundo proceso se realiza con el objetivo de reducir los datos transmitidos y representar con mayor exactitud y continuidad el movimiento en las secuencias de video. En la figura 9 se muestra el diagrama del proceso de Estimación de Movimiento.

El FME recibe como datos iniciales los resultados del proceso IME, es decir se tiene como variables de entrada al vector de movimiento y al SAD obtenido de la mejor correlación entre el cuadro actual y los bloques del cuadro de referencia.

El FME se encargará de realizar una búsqueda fina alrededor del resultado calculado por el IME y para ello utilizará muestras de referencia denominadas muestras fraccionales; las muestras fraccionales son determinadas aplicando filtros de interpolación a los píxeles del cuadro de referencia.

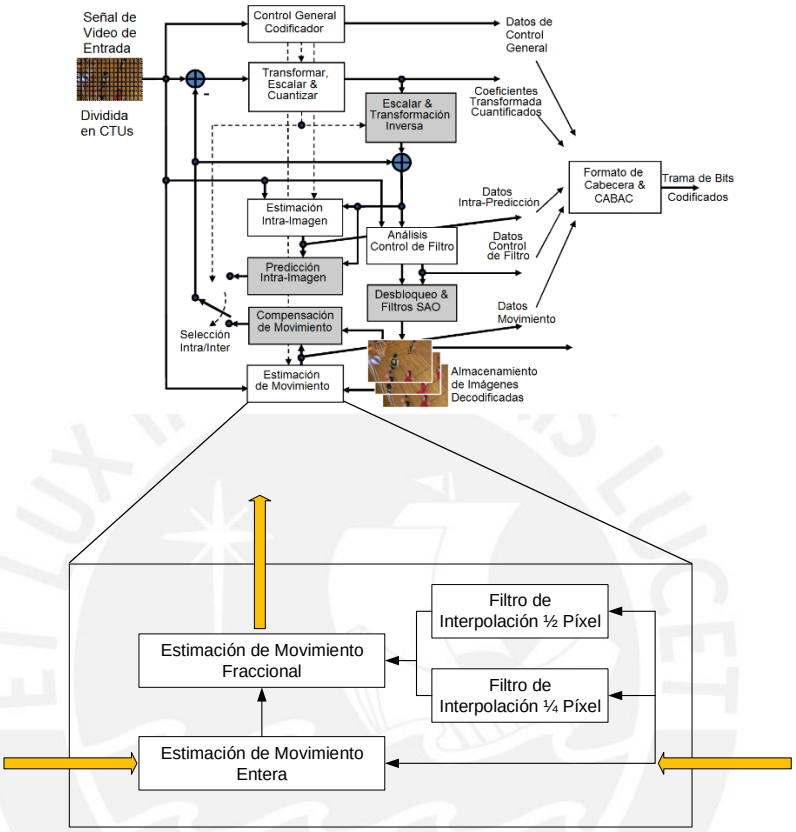


Fig. 9. Representación del proceso de Estimación de Movimiento [3].

Empleando la figura 10 se logra una mejor comprensión del concepto de muestras fraccionales; en esta figura se puede observar de color celeste a los píxeles enteros mientras que de color amarillo y verde a los píxeles fraccionales los cuales son del tipo half-pixel (½ píxel) y quarter-pixel (¼ píxel) respectivamente y serán tratados con mayor detalle en la siguiente sección [4].

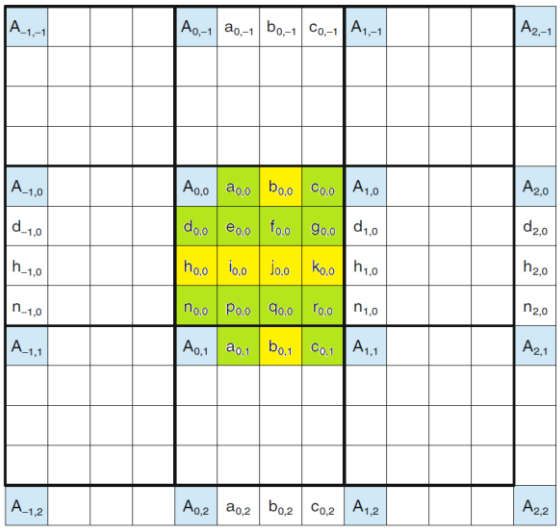


Fig. 10. Píxeles enteros y fraccionales en cuadro de referencia [3].

Finalizado el proceso fraccional se obtendrá como resultado un vector de movimiento que de acuerdo a las características de los bloques analizados podría tener componentes fraccionales. Por ejemplo, en la figura 11 se necesita encontrar la mejor correlación para el bloque mostrado en la figura (a) y para ello se aplicará el proceso IME, debido a este primer procesamiento se obtiene como mejor resultado el bloque señalizado por el vector de movimiento mostrado en la figura (b). Como última etapa del proceso de Estimación de Movimiento, se lleva acabo el FME y se obtiene como resultado que la mejor correlación para este caso corresponde a un bloque de muestras fraccionales con su respectivo vector de movimiento.

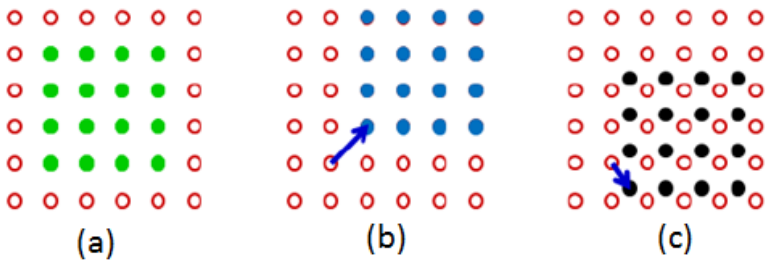


Fig. 11. Estimación de Movimiento fraccional [13].

2.2.1.1 Interpolación Half-pixel

El primer paso del proceso FME requiere calcular un grupo inicial de muestras de referencia las cuales tienen precisión Half-pixel (½ píxel). Este grupo inicial

de muestras es calculado empleando un filtro FIR simétrico de 8 coeficientes los cuales son detallados en la tabla 1.

Tabla 1. Coeficientes de filtro para muestras de luminancia Half-pixel [10].

Índice	-3	-2	-1	0	1	2	3	4
Filtro H [i]	-1	4	-11	40	40	-11	4	-1

Por ejemplo, se desea calcular la muestra fraccional horizontal mostrada en la figura 12, entonces para ello será necesario contar con 8 píxeles horizontales enteros consecutivos a los cuales se les aplicará la operación de interpolación de acuerdo a la ecuación 4 en donde A[i] representará a los píxeles enteros, Filtro H[i] a los coeficientes del filtro mostrados en la tabla 1 y b será el píxel fraccional calculado.

$$b = \left(\sum_{i=-2}^{4} A\,(i,0) * Filtro\ H[i] \right) \gg 8 \qquad (4)$$

La ecuación 4 también refleja que es necesario realizar un desplazamiento hacia la derecha de 8 posiciones lo cual equivale a realizar una división por 64. La división es necesaria para que la ganancia del filtro sea igual a 1 [10].

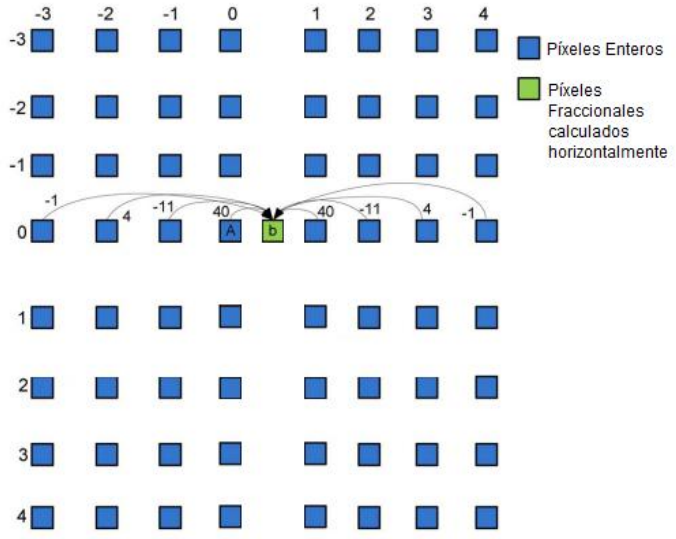


Fig. 12. Cálculo de muestra Half-pixel horizontal [14].

Si se requiere calcular muestras fraccionales verticales es necesario proceder del mismo modo pero ahora se necesitarán 8 píxeles verticales enteros

consecutivos a los cuales se les aplicará también la ecuación 4 de acuerdo a lo mostrado en la figura 13.

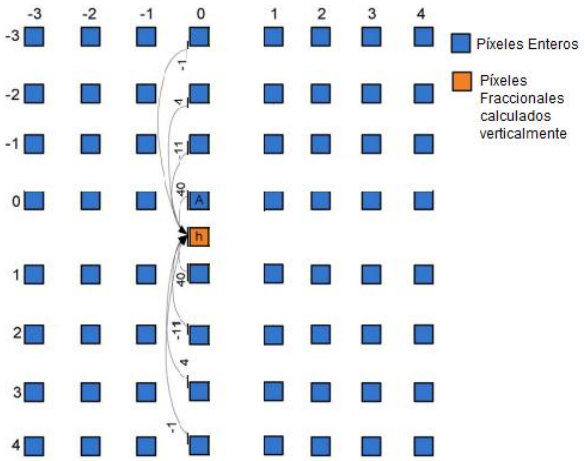


Fig. 13. Cálculo de muestra Half-pixel vertical [14].

Existe un último sub-grupo de muestras fraccionales verticales que es necesario calcular para realizar el primer paso de FME y para ello es necesario utilizar muestras fraccionales horizontales previamente calculadas de acuerdo a la figura 12. Por lo tanto, para este grupo de muestras es necesario contar con 8 píxeles fraccionales verticales consecutivos tal como se muestra en la figura 14.

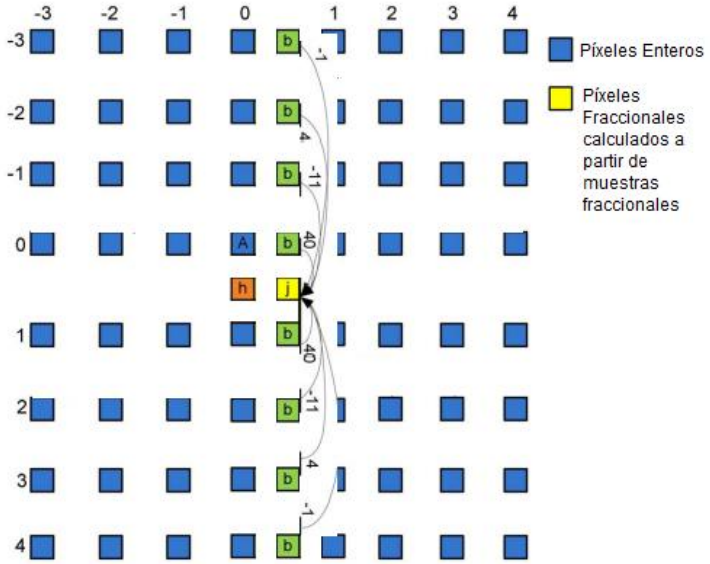


Fig. 14. Cálculo de muestra Half-pixel vertical a partir de muestras fraccionales [14].

Las muestras fraccionales calculadas permiten realizar 8 posibles comparaciones con el cuadro actual. Las 8 posibilidades permiten obtener un vector de movimiento que puede tener los siguientes valores: (0.5;0), (-0.5;0), (0;0.5), (0;-0.5), (0.5;0.5), (0.5;-0.5), (-0.5;0.5) y (-0.5;-0.5). Para determinar la mejor correlación se hará uso del cálculo del SAD entre cada conjunto de muestras y el cuadro actual. En la figura 15, es posible observar al cuadro de referencia de 8x8 píxeles junto con los grupos de píxeles que formarán las posibilidades de comparación mencionadas. Finalmente, es necesario considerar al valor de SAD obtenido por el proceso IME dentro de las posibles alternativas de respuesta.

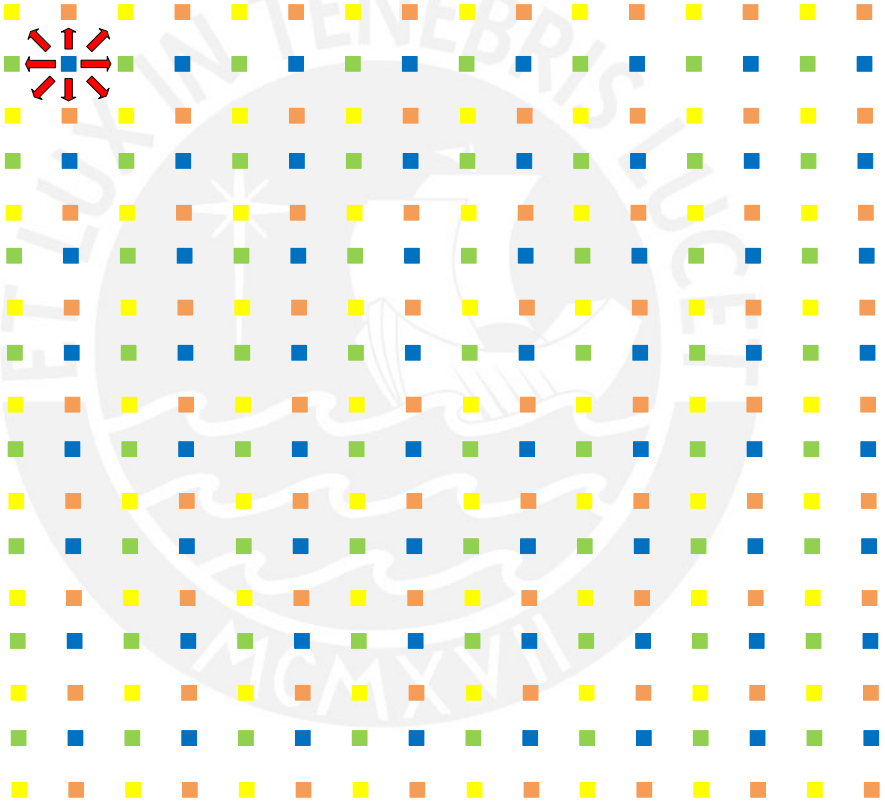


Fig. 15. Posibilidades para el vector de movimiento con precisión Half-pixel.

2.2.1.2 Interpolación Quarter-pixel

Para continuar con el proceso de compresión, es necesario realizar un segundo cálculo de muestras fraccionales que permitirán encontrar una mejor correlación con el cuadro actual. Las muestras que se calcularán tendrán precisión Quarter-pixel y son generadas empleando un filtro FIR de 7 coeficientes [3], estos coeficientes son mostrados en tabla 2.

Tabla 2. Coeficientes de filtro para muestras de luminancia Quarter-pixel [10].

Índice	-3	-2	-1	0	1	2	3
Filtro Q [i]	-1	4	-10	58	17	-5	1

Por ejemplo, se requiere calcular las muestras Quarter-pixel horizontales a y c mostradas en la figura 16. Para ello será necesario seguir la ecuación 5 y 6.

$$a = \left(\sum_{i=-3}^{3} (A\, i, 2) * Filtro\ Q[i] \right) \gg 8 \quad (5)$$

$$c = \left(\sum_{i=-2}^{4} (A\, i, 2) * Filtro\ Q[1-i] \right) \gg 8 \quad (6)$$

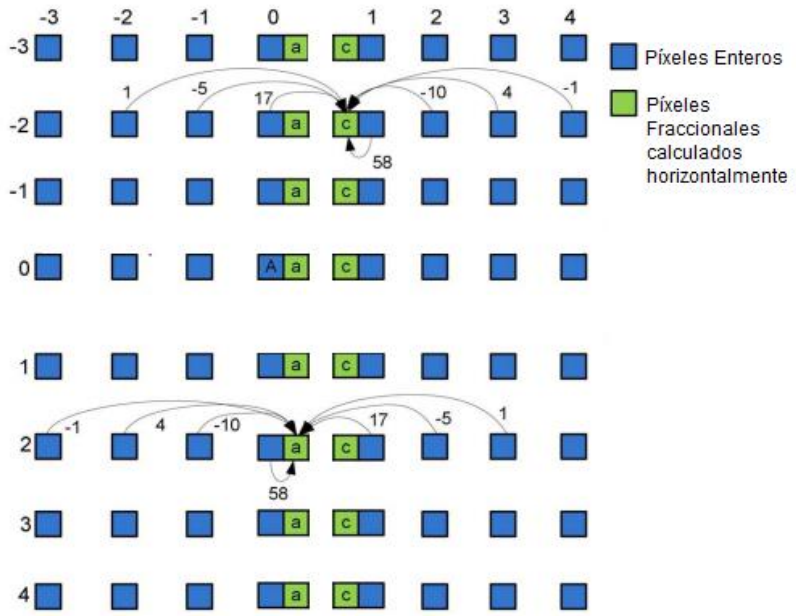


Fig. 16. Cálculo de muestras Quarter-pixel horizontales [14].

Las muestras Quarter-pixel verticales deberán de emplear 7 muestras verticales consecutivas así como se muestra en la figura 17 (a) y (b).

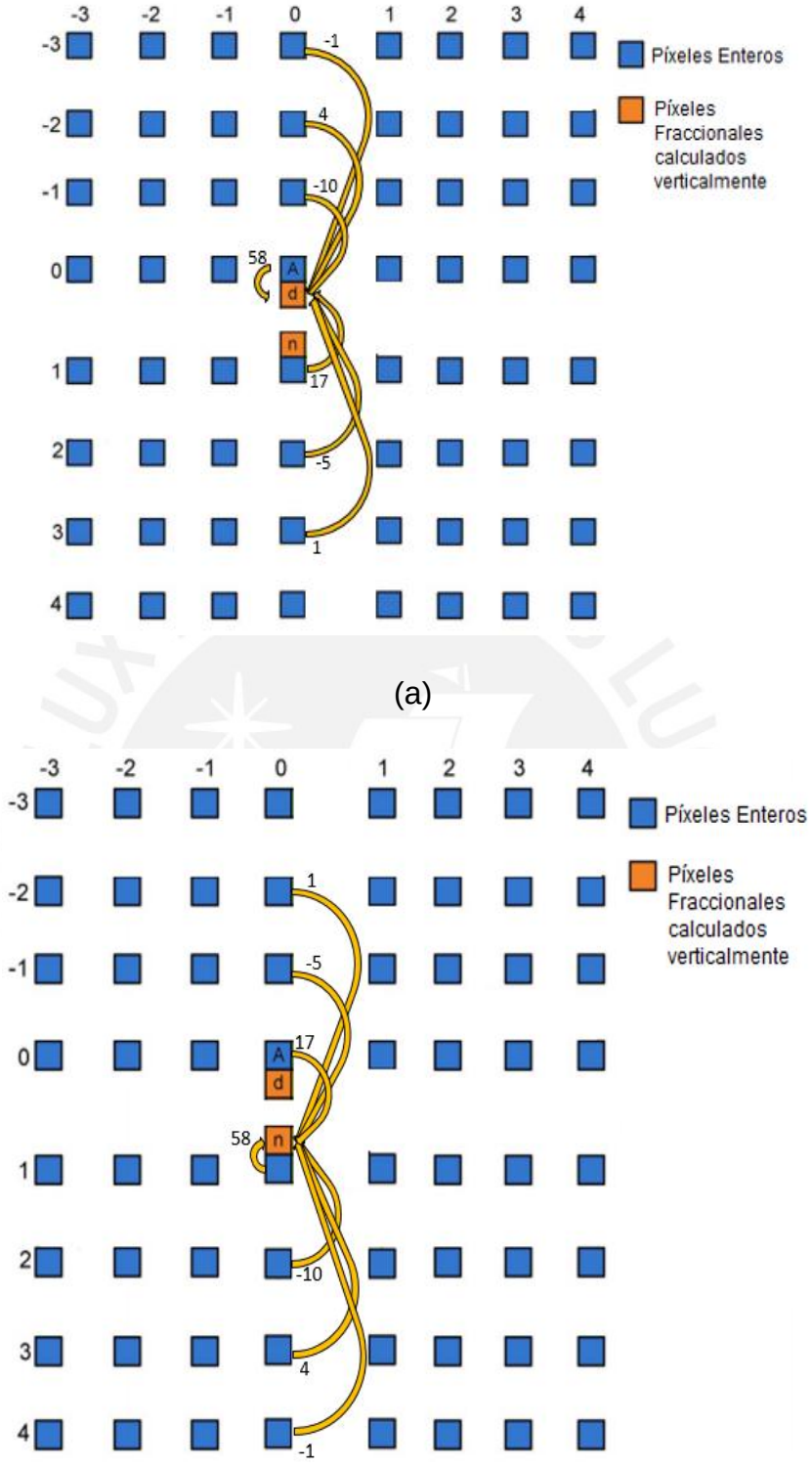


Fig. 17. Cálculo de muestras Quarter-pixel verticales [14].

Al igual con lo sucedido en la interpolación Half-pixel, existirán muestras con precisión Quarter-pixel que serán calculadas utilizando muestras Half-pixel horizontales o Quarter-pixel horizontales previamente calculadas.

Finalmente se obtendrán 8 posibles conjuntos de comparación formados únicamente por muestras Quarter-pixel que deberán ser analizados con el bloque actual para obtener un vector de movimiento que podría tener los siguientes valores: (0.25;0), (-0.25;0), (0;0.25), (0;-0.25), (0.25;0.25), (0.25;-0.25), (-0.25;0.25) y (-0.25;-0.25). Para concluir con el proceso de Estimación de Movimiento Fraccional, el resultado de mejor correlación, es decir SAD y vector de movimiento con precisión Quarter-pixel deberá ser comparado con el resultado del proceso de interpolación Half-pixel con el objetivo de determinar la mejor correlación de todo el proceso así como permitir la mayor compresión posible del cuadro actual analizado.

Capítulo 3

Diseño de la Arquitectura Hardware del algoritmo de Estimación de Movimiento Fraccional

3.1 Hipótesis de Solución

De acuerdo a lo presentado en las secciones anteriores, tener una implementación en hardware del algoritmo de Estimación de Movimiento Fraccional (FME) permite aprovechar el paralelismo de operaciones a una alta y adecuada frecuencia de operación para así lograr una tasa de procesamiento mayor o igual a 30 cuadros por segundo (fps del inglés, "*Frames per second*"). Dicha tasa de procesamiento es requisito necesario para lograr una codificación de video de alta definición (1920x1080 píxeles) en tiempo real.

Hipótesis Principal

Mediante la implementación en hardware del algoritmo de Estimación de Movimiento Fraccional se comprobará que es posible obtener una tasa de procesamiento mayor o igual a 30 cuadros por segundo permitiendo una codificación de video con resolución HDTV (1920x1080 píxeles) en tiempo real.

Hipótesis Secundaria

El algoritmo de Estimación de Movimiento Fraccional de acuerdo al estándar HEVC permite obtener una menor energía residual, es decir un menor valor de SAD, a medida que se utilicen muestras interpoladas de mayor precisión.

3.1.1 Descripción y Justificación de la Hipótesis

Las señales de video generan una importante cantidad de datos y cada vez se esperan mayores resoluciones, mayor tasa de procesamiento y un deseo de tener mejores accesos al contenido de video digital. En conclusión, el tráfico de datos ocasionado por la información de video representa una importante carga para las redes de comunicaciones así como para los medios de almacenamiento por lo cual ha sido fundamental la evolución hacia un nuevo estándar de codificación que mejore la compresión de datos en comparación con estándares previos como H.264/AVC o MPEG-2 [4].

Dentro de las posibles soluciones que permiten implementar el algoritmo de Estimación de Movimiento, existen dispositivos como las Unidades de Procesamiento Gráfico (GPU, del inglés "*Graphing Proccesing Unit*") los cuales constan de múltiples núcleos para procesamiento y por lo tanto están diseñados para manejar diversas tareas en simultáneo [18]. Sin embargo, realizar una implementación en un GPU requiere de un mayor consumo de potencia en comparación con lo requerido por un FPGA o por un ASIC. Por ejemplo en la tabla 3 se muestra una comparación entre diversas implementaciones del codificador HEVC [12]. Por lo tanto, realizar una implementación por medio de un GPU requiere de un consumo de potencia elevado lo cual no es óptimo debido a que los codificadores también son implementados en dispositivos portátiles en donde es de vital importancia administrar adecuadamente la energía a consumir para asegurar una mayor duración de la batería.

Tabla 3. Comparación de implementaciones del Codificador [12].

Parámetro	GPU + CPU [19]	FPGA [20]	ASIC [21]
Costo de Desarrollo	Medio	Medio	Elevado
Velocidad	Medio	Medio	Elevado
Consumo de Potencia	Elevado	Medio	Bajo

Resultaría óptimo realizar una implementación del codificador por medio de un ASIC porque se lograría una mayor velocidad y el consumo de potencia es bajo por lo cual se lograría un buen desempeño para aplicaciones portátiles; sin embargo, el costo de implementación resulta elevado. Por lo tanto, utilizar un FPGA se presenta como una alternativa viable ya que se trata de un dispositivo flexible que permite realizar el paralelismo de operaciones necesario para conseguir el objetivo deseado el cual es una tasa de procesamiento mayor o igual a 30 fps.

3.2 Objetivos:

3.2.1 Objetivo Principal

Diseñar una arquitectura hardware del algoritmo FME según el formato HEVC para secuencias de video con resolución Full HD (1920x1080 píxeles) considerando bloques de área cuadrada y tamaño 8x8 píxeles.

3.2.2 Objetivos Específicos

- Diseñar la arquitectura del algoritmo FME en RTL y realizar su descripción en VHDL.
- Evaluar el comportamiento del RTL a través de verificación funcional por medio de simulaciones empleando Test Benchs descritos en VHDL.
- Sintetizar la arquitectura para dispositivos FPGA de la familia Virtex de Xilinx.
- Verificar y optimizar los resultados de la máxima frecuencia de operación de la arquitectura así como el área ocupada del dispositivo.

3.3.1 Descripción de la arquitectura del dispositivo FPGA.

Los dispositivos FPGA, cuyas siglas provienen del inglés "*Field Programmable Gate Arrays*", son dispositivos semiconductores programables los cuales contienen conexiones configurables entre bloques lógicos que también poseen la misma característica. Los diseñadores configuran los diversos bloques a través de un lenguaje de descripción de hardware el cual puede ser VHDL (del inglés "*Very High Integrated Circuit Hardware Description Language*") o alternativamente podría tratarse de Verilog HDL. Ambos lenguajes de descripción son independientes de la tecnología es así que se pueden emplear para realizar la descripción hardware para FPGAs, CPLDs e inclusive ASICs [22].

La arquitectura de los FPGA varía entre los diversos fabricantes presentes en el mercado pero tienen componentes en común. Los Bloques Lógicos Configurables (CLB, del inglés "*Configurable Logic Block*") son las unidades lógicas básicas del FPGA y están compuestos por una matriz de conexiones con 4 o 6 entradas, circuitos de selección de entradas y flip-flops [22]. En la Figura 12, se muestra la estructura interna del FPGA.

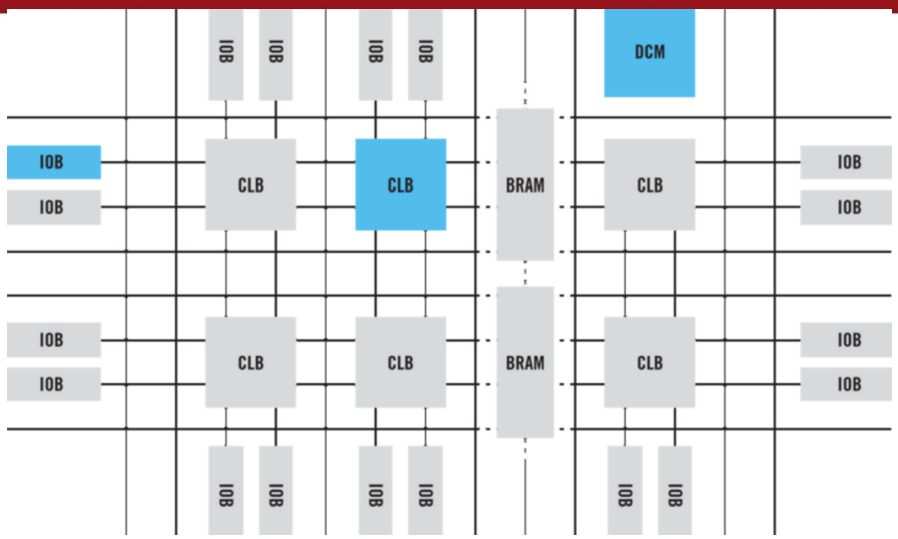


Fig. 12. Estructura del FPGA [23].

En el caso de la compañía Xilinx [23], los CLBs son denominados celdas lógicas. En la figura 13, se puede observar la estructura de una celda lógica.

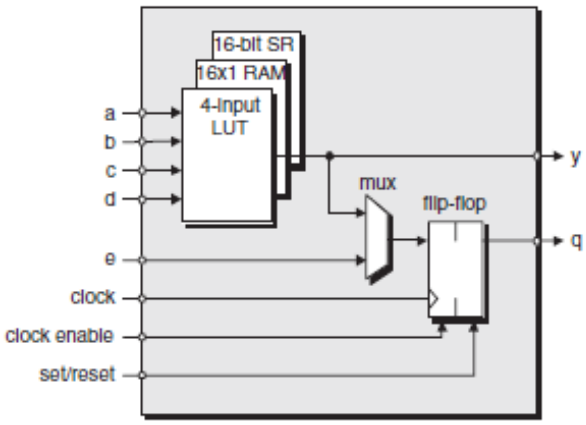


Fig. 13. Celda Lógica de un FPGA Xilinx [22].

En este caso, la lógica combinacional es implementada por medio de tablas de búsqueda (LUT, del inglés "*Look Up Table*"), es decir las entradas sirven como especie de índices para encontrar la respuesta dentro de la tabla.

Los FPGA también cuentan con bloques multiplicadores o encargados de procesamiento de señales. Adicionalmente, se tienen bloques de entradas y salidas y aquellos encargados de la administración de reloj los cuales permiten generar diferentes señales de reloj teniendo como base al oscilador interno y controladores de fase de lazo cerrado [22].

Estos dispositivos permiten acelerar la etapa de validación de los prototipos así como realizar las correcciones necesarias en el diseño. Implementar prototipos por medio de un ASIC no resulta conveniente debido a los tiempos de fabricación y a los costos que implica producir pocas unidades.

3.3.2 Estado de la Investigación

Una de las estrategias planteadas para disminuir el tiempo de codificación del proceso de Estimación de Movimiento Fraccional radica, según Agostini *et. al.* [16], en una arquitectura hardware para el proceso de Suma de Diferencias Absolutas (SAD del inglés *"Sum of Absolute Differences"*); sin embargo, gran parte de los esfuerzos realizados han tenido como base la arquitectura del estándar H.264/AVC.

De acuerdo con los experimentos realizados en [16], se resuelve contar con divisiones de imagen basadas en bloques cuadrados y de área variable ya que de este modo se logra obtener, en el proceso de codificación, pérdidas de calidad aceptables pero con mejoras en el paralelismo de operaciones y por lo tanto un menor tiempo de procesamiento.

La investigación realizada por Agostini *et. al.* propone realizar una arquitectura para el cálculo SAD que será tomada como referencia en el presente trabajo porque realiza las operaciones con mayor simplicidad e involucra una menor cantidad de área utilizada a diferencia de lo sucedido en [6] y [25] en donde las operaciones a utilizar involucran una mayor complejidad así como una mayor área de implementación.

Otro proceso importante del ME es el cálculo de los píxeles intermedios fraccionales (Half-pixel y Quarter-pixel) a cargo de los filtros de interpolación. Para mejorar el desempeño del estimador de movimiento fraccional, se diseña en [8] una arquitectura para el filtro de interpolación de 8 pixeles. Dicha propuesta implica un ahorro del 9.9% del tiempo de procesamiento en promedio a la vez que introduce una baja degradación de la calidad de la imagen. La arquitectura planteada renuncia al proceso de interpolación de los bloques de predicción de área 4 x 8, 4 x 16 y 12 x 16 lo cual guarda relación con los experimentos desarrollados por Agostini *et. al.* [16] ya que proponen considerar el procesamiento con únicamente bloques de tamaño cuadrado. Adicionalmente,

Lian *et. al*. [8] proponen una arquitectura reutilizable para la interpolación de los pixeles en posiciones fraccionarias es por ello que está arquitectura será tomada como referencia para el desarrollo de la presente investigación. También se ha tomado en cuenta que el trabajo realizado por Lian *et. al* ha sido tomado como referencia en [15] obteniendo buenos resultados.

3.4 Diseño de la arquitectura de Estimación de Movimiento Fraccional

En la presente sección se mostrarán los diseños de las unidades funcionales empleadas en la arquitectura que realizará el proceso de Estimación de Movimiento Fraccional.

3.4.1 Arquitectura de la Unidad de Estimación de Movimiento Fraccional Half-pixel

A. Arquitectura de la Unidad de Interpolación Half-Pixel

La arquitectura de interpolación Half-Pixel es la encargada de calcular las muestras horizontales y verticales con precisión de ½ píxel necesarias para realizar el primer proceso de Estimación Fraccional. En la figura 14, se muestra la arquitectura de la unidad en mención así como las respectivas señales de control. Es importante mencionar que esta unidad tendrá como datos de entrada a los píxeles del cuadro de referencia, es decir a un bloque de 16x16 muestras. El bloque irá ingresando fila por fila, las cuales serán almacenadas progresivamente en el Buffer y al mismo tiempo serán procesadas por la Unidad de Interpolación, de ese modo se podrá aprovechar los ciclos de reloj.

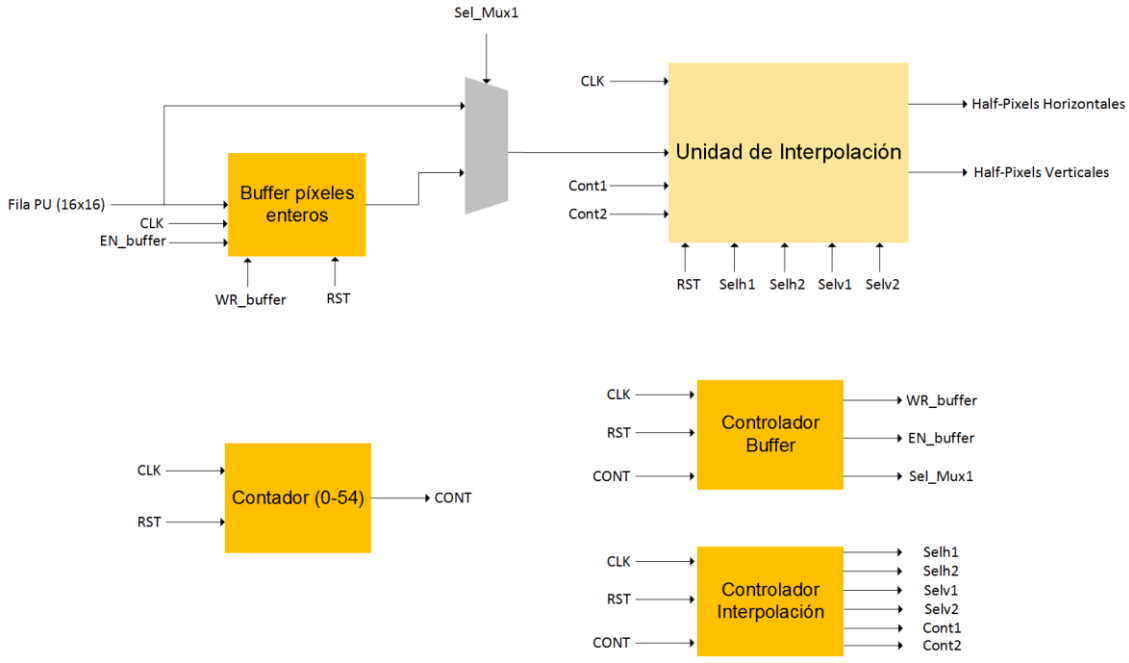


Fig. 14. Arquitectura de la unidad de Interpolación Half-Pixel

La dirección del funcionamiento de la arquitectura está distribuido en dos controladores los cuales se encargarán de enviar las señales de control

correspondientes tanto para el dispositivo de almacenamiento así como para la Unidad de Interpolación. Los controladores están basados en máquinas de estado que determinarán las señales de acuerdo al contador presente en la arquitectura y al estado presente de cada uno.

- Buffer píxeles enteros

Se encargará de almacenar las muestras de píxeles enteros del cuadro de referencia con la finalidad de reutilizar estos datos para el proceso de interpolación con precisión Quarter-pixel, este buffer es del tipo FIFO (del inglés "*First In – First Out*"). El Buffer tendrá la capacidad de almacenar a un bloque de 16x16 píxeles ya que para calcular las muestras fraccionales de un bloque de 8x8 píxeles de referencia es necesario adicionar 4 muestras vecinas en la dirección superior, inferior, lateral derecho y lateral izquierdo. Cada una de las muestras de entrada está codificada con 8 bits. En la figura 15, se muestra un buffer para un bloque de 3x3 muestras; en la tabla 4 se detallan las entradas y salidas de esta unidad de almacenamiento.

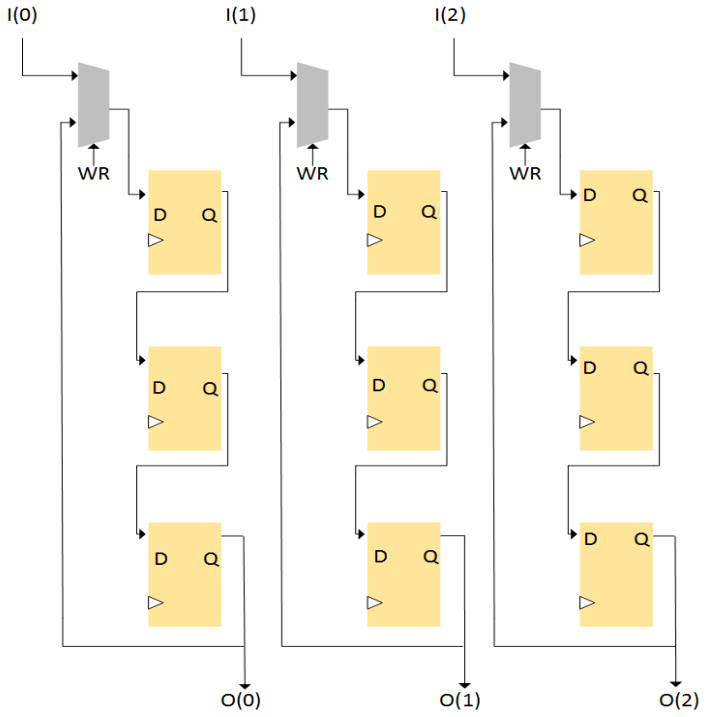


Fig. 15. Buffer píxeles enteros.

Tabla. 4. Descripción de entradas y salidas para el buffer de píxeles enteros

Parámetro	Tipo de Dato	Descripción
I(0), I(1), I(2)…, I(15)	Entrada	Arreglo de 16 bytes de entrada que representan a una fila del bloque de referencia.
CLK	Entrada	Señal de reloj del sistema.
WR	Entrada	Modo Lectura / Escritura.
EN	Entrada	Habilitador.
O(0), O(1), O(2), …, O(15)	Salida	Arreglo de 16 bytes de salida que representan a una fila del bloque de referencia.

- Unidad de Interpolación

La unidad de interpolación es la encargada de calcular los pixeles en posiciones fraccionales, tanto horizontales como verticales, que serán utilizados como nuevas posibilidades de bloques de referencia en el proceso de Estimación de Movimiento. Esta unidad también se utilizará para realizar la interpolación con precisión Quarter-pixel como se verá posteriormente. En la figura 16, se muestra la arquitectura de la Unidad de interpolación la cual ha sido diseñada con base en los trabajos previos [8] y [15] tomando en consideración que las señales de control y algunos elementos lógicos han sido planteados originalmente en este trabajo. Se escogió esta arquitectura porque se trata de una unidad reutilizable tanto para el proceso Half-pixel y Quarter-pixel y adicionalmente permite utilizar adecuadamente los recursos lógicos.

La unidad de interpolación necesita como datos de entrada 16 bytes que representan a 16 píxeles correspondientes a una fila del cuadro de referencia. A partir de esta fila de 16 elementos, se generan sub-grupos de 8 muestras contiguas que servirán como datos de entrada para cada banco de filtros de interpolación quienes serán los responsables de determinar los pixeles fraccionales.

La primera etapa de la arquitectura corresponde a un banco de filtros horizontales conformado por un filtro FIR de 8-taps y 2 filtros FIR de 7-taps de acuerdo a lo mostrado en la figura 17. La primera etapa del proceso FME requiere únicamente las labores del filtro FIR de 8-taps ya que solo se trabajarán con muestras fraccionales con precisión Half-pixel. En la segunda etapa del

proceso, donde se obtendrán las muestras Quarter-pixel, será necesario el uso de los 3 filtros en simultáneo es por ello que este primer arreglo de filtros tiene la opción de seleccionar cuáles de ellos serán necesarios poner en funcionamiento.

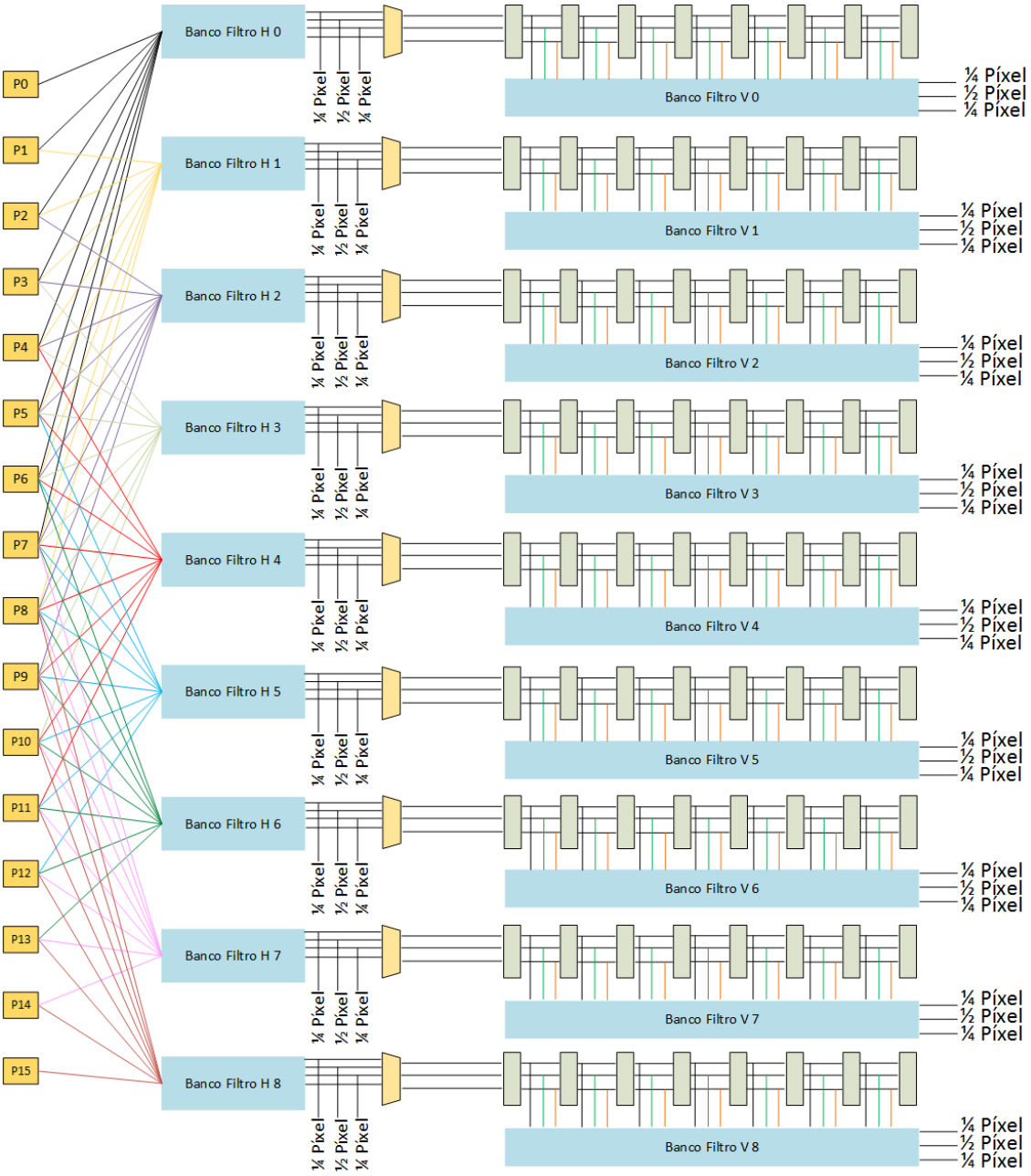


Fig. 16. Unidad de Interpolación de pixeles fraccionales.

Por lo tanto, los bancos de filtros horizontales permiten realizar el cálculo de los conjuntos de muestras horizontales mostrados tanto en la sección 2.2.1.1 como en la sección 2.2.1.2. En la tabla 5, se detallan las entradas y salidas de banco de filtro horizontal.

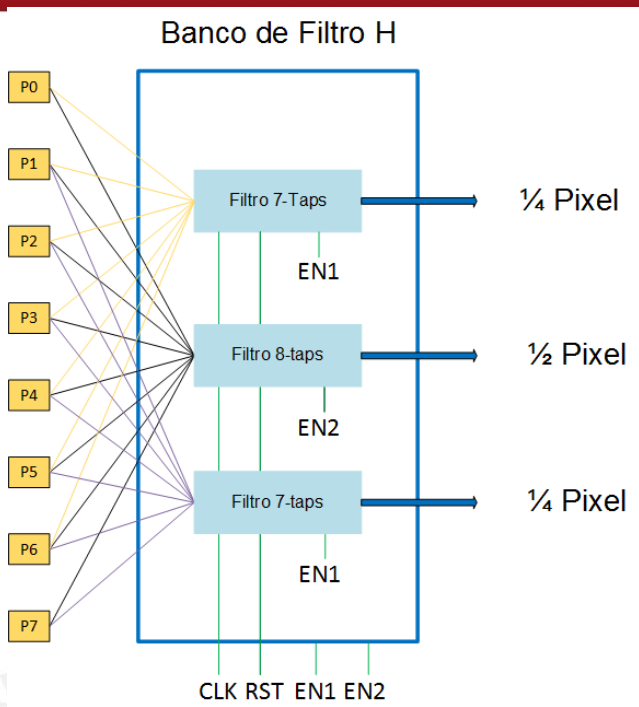


Fig. 17. Arquitectura del banco de filtro horizontal.

Tabla 5. Detalle de Entradas y Salidas para Banco de Filtro Horizontal

Parámetro	Tipo de Dato	Descripción
P0, P1, …, P7	Entrada	Píxeles Enteros de 8 bits cada uno.
CLK	Entrada	Señal de reloj del sistema.
RST	Entrada	Señal de Reset.
EN1	Entrada	Señal para habilitar los filtros de 7-taps.
EN2	Entrada	Señal para habilitar los filtros de 8-taps.
Pixel 1/2 y Pixel 1/4	Salida	Muestras interpoladas por los filtros. Cada una es de 8 bits.

A medida que se van generando las muestras fraccionales horizontales, se deberán ir procesando las muestras verticales que permitirán completar los 6 casos restantes de comparación que determinarán la mejor correlación con precisión de hasta Half-pixel. Para cumplir con este objetivo, las muestras fraccionales horizontales así como ciertos píxeles enteros son propagados hacia la etapa de cálculo vertical y para ello se cuenta con un pipeline de 8 etapas por cada banco de filtro de horizontal. La etapa de pipeline permitirá ir acumulando las muestras que serán procesadas por los bancos de filtros verticales tal como se muestra en la figura 16. Debido a que la arquitectura presenta un diseño reutilizable, las muestras que pasarán a la etapa de procesamiento vertical serán seleccionadas por multiplexores ya que para el segundo proceso FME, que

considera píxeles con precisión Quarter-pixel, se deben propagar otras muestras hacia la etapa vertical.

Los bancos de filtros verticales poseen una arquitectura similar a los bancos de filtros horizontales con la diferencia que ahora se trata de un arreglo de 6 filtros FIR de 7-taps y 2 filtros FIR de 8-taps de acuerdo a lo mostrado en la figura 18.

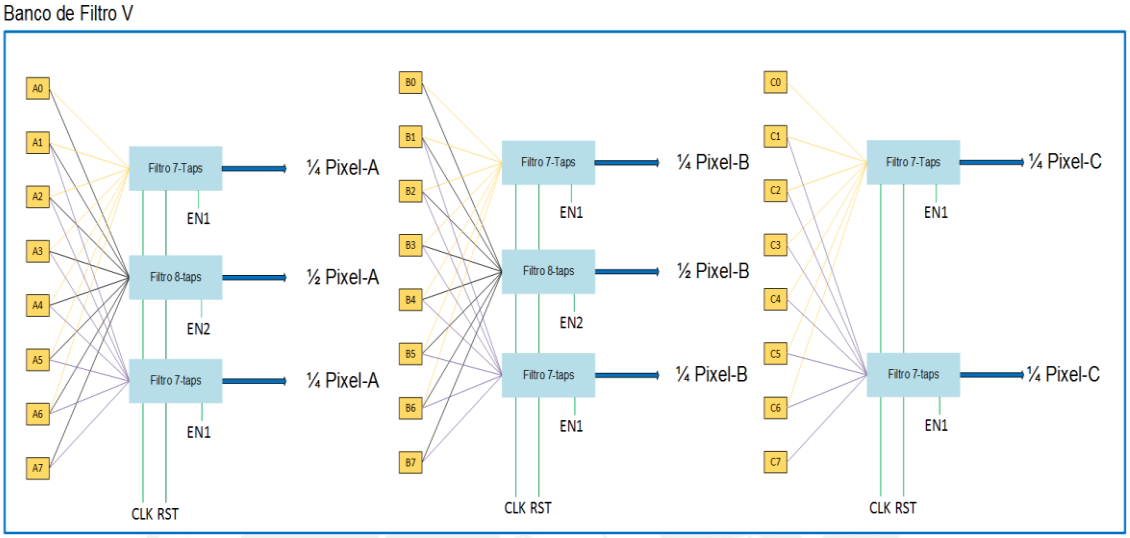


Fig. 18. Arquitectura del Banco de Filtro Vertical.

Esta arquitectura también permite seleccionar qué filtros serán necesarios poner en funcionamiento de acuerdo al proceso de estimación que se está llevando a cabo es así que para la primera parte del proceso solo será necesario utilizar los filtros de 8-taps. En la tabla 6 se detallan las señales de entrada y salida del banco de filtros verticales.

Tabla 6. Entradas y Salida del Banco de Filtros Verticales.

Parámetro	Tipo de Dato	Descripción
A0, A1,…, A7	Entrada	Píxeles Enteros de 8 bits cada uno.
B0, B1,…, B7	Entrada	Píxeles Enteros de 8 bits cada uno.
C0, C1,…, C7	Entrada	Píxeles Enteros de 8 bits cada uno.
EN1	Entrada	Señal para habilitar los filtros de 7-taps.
EN2	Entrada	Señal para habilitar los filtros de 8-taps.
CLK	Entrada	Señal de reloj del sistema.
RST	Entrada	Señal de Reset.
1/4 Pixel A, 1/2 Pixel A	Entrada	Salida de Filtros Verticales
1/4 Pixel B, 1/2 Pixel B	Entrada	Salida de Filtros Verticales
1/4 Pixel C, 1/2 Pixel C	Entrada	Salida de Filtros Verticales

El núcleo de ambos conjuntos son filtros FIR de 7-taps y 8-taps, estos filtros han sido diseñados de acuerdo al trabajo previo [15]. Las arquitecturas de ambos filtros utilizan desplazamientos de bits tanto hacia la derecha como hacia la izquierda para realizar divisiones y multiplicaciones respectivamente. En la figura 19 (a) y (b) se muestran las arquitecturas para ambos filtros implementados.

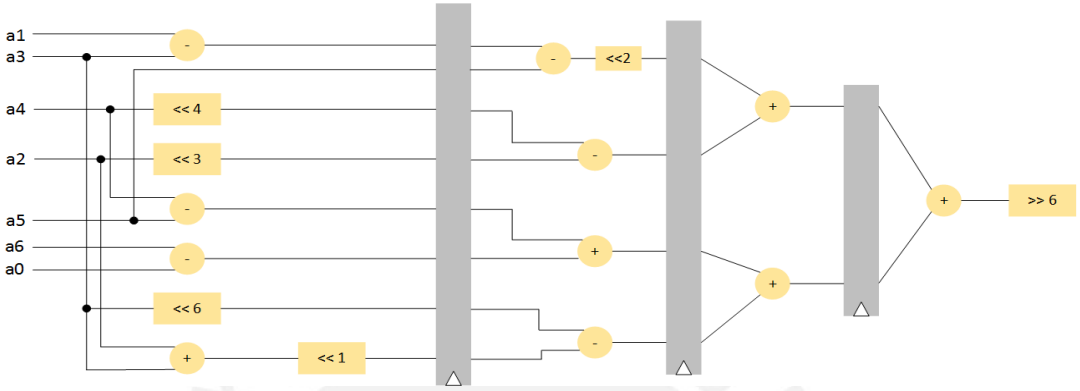


Fig. 19 (a). Filtro de 7 coeficientes.

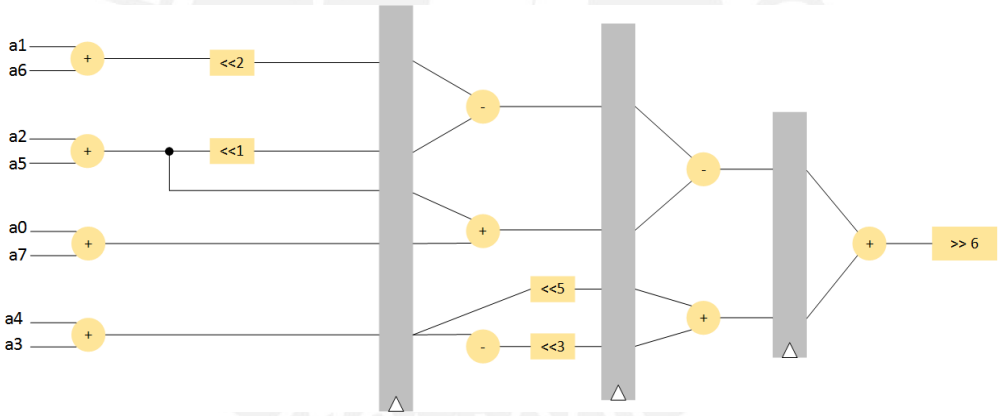


Fig. 19 (b). Filtro de 8 coeficientes.

Es importante resaltar que estas arquitecturas también incluyen secuencias de pipeline de 3 etapas las cuales permiten aprovechar el paralelismo de operaciones de la arquitectura del filtro por lo que no será necesario esperar al final del cálculo de una muestra fraccional para iniciar el siguiente proceso. Adicionalmente, permiten reducir los tiempos retardo introducidos por las puertas lógicas que efectúan las operaciones.

- Controlador Buffer

La unidad de Control Buffer se encargará de manejar los selectores de los multiplexores y los modos lectura y/o escritura del buffer presente en esta arquitectura y de todos los espacios de almacenamiento propuestos en el presente trabajo. Para mantener un orden adecuado, solo se presentan en la figura 14 las señales de control utilizadas por los bloques que intervienen en este primer proceso de estimación.

El controlador está basado en una máquina de estados que realiza los cambios necesarios en sus señales de salida de acuerdo al valor del contador presente también en la arquitectura y al estado actual en el que se encuentre. El diagrama de estados del controlador se encuentra detallado en el Anexo A.

- Controlador Interpolación

El controlador de Interpolación es el encargado de guiar a la unidad de Interpolación y de seleccionar qué filtros serán empleados para determinado caso, ya que existen diversas opciones de trabajo para los filtros de 7-taps y 8-taps como se presentó en la arquitectura de la unidad de interpolación. El controlador es una máquina de estados que tiene en cuenta el valor del contador y posteriormente analizará el valor del vector de movimiento con precisión ½ píxel para poder determinar sus señales de salida para el control del módulo de interpolación. El diagrama de estados se encuentra detallado en el Anexo A.

B. Arquitectura de la Unidad de Búsqueda Half-Pixel

La unidad de búsqueda se encargará de analizar las 8 posibilidades de bloques Half-pixel obtenidos del proceso de interpolación y determinar cuál de estos conjuntos tiene la mejor correlación con el bloque actual. El resultado final será el vector de movimiento con precisión Half-pixel que se sumará al vector de movimiento obtenido del proceso IME.

Adicionalmente a los píxeles fraccionales, es necesario tener como dato a las muestras del cuadro actual. El cuadro actual irá ingresando a la arquitectura al mismo tiempo que van ingresando las muestras de referencia y también lo harán por filas de 8 píxeles ya que los bloques actuales son arreglos de 8x8 muestras.

La arquitectura de búsqueda necesita de 3 espacios de almacenamiento los cuales tienen las siguientes capacidades: 2 unidades de 4x4 píxeles y 1 unidad de 8x8 píxeles. Todos los espacios de almacenamiento están destinados para las muestras actuales y se optó por realizar esta división como estrategia para evitar desperdiciar ciclos de reloj porque las muestras fraccionales horizontales se obtienen primero que las muestras verticales lo que permite iniciar el proceso de comparación con estas muestras paralelamente. En la figura 20 se muestra el diagrama de la arquitectura para la unidad de búsqueda.

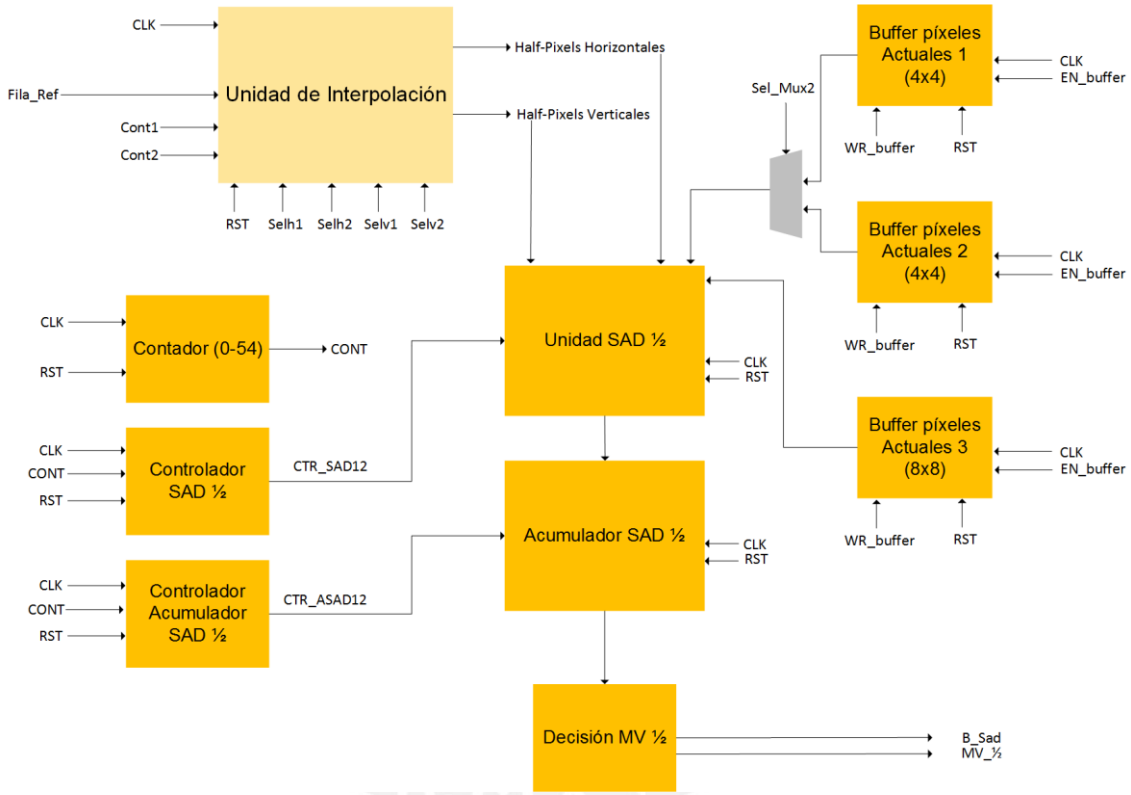


Fig. 20. Unidad de Búsqueda Half-Pixel.

Los datos de salida de la arquitectura serán la menor Suma de Diferencias Absolutas (SAD) y el vector de movimiento correspondiente a aquella mejor correlación.

- Unidad SAD ½ y Unidad de Acumulación SAD ½

De acuerdo a lo señalado en la sección 2.2.1, la mejor correlación estará determinada por el cálculo de la Suma de Diferencias Absolutas o SAD ya que involucra un menor costo computacional en comparación con los restantes métodos de comparación como son MSE y MAE.

La unidad SAD ½ está formada por 8 bloques encargados de realizar el SAD ya que se tienen 8 posibilidades formadas por los conjuntos de muestras fraccionales con precisión Half-pixel. Cada uno de estos bloques iniciará su funcionamiento de manera independiente tomando en cuenta únicamente las señales de control proporcionadas por el Controlador SAD ½.

Cada uno de los 8 bloques encargados de realizar el cálculo del SAD es capaz de implementar la ecuación 3 mostrada en el capítulo 2 y para cumplir con este objetivo se ha tomado como referencia la arquitectura mostrada en el trabajo previo [16]. Del mismo modo que la arquitectura de los filtros FIR, se cuenta con una sección de pipeline de 3 etapas, lo que permite reducir el tiempo de retardo de los datos que son procesados por esta unidad, obteniendo una mayor frecuencia de operación a cambio de adicionar 3 ciclos de reloj para su procesamiento. En la figura 21 se muestra la arquitectura para el proceso SAD.

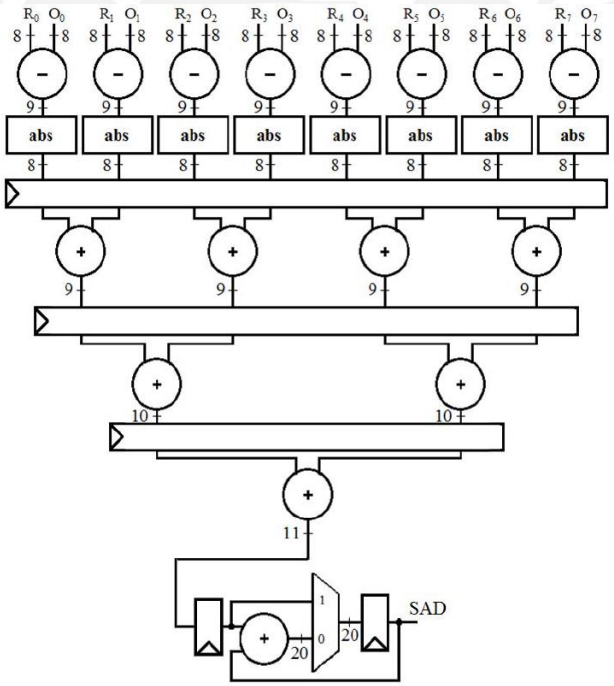


Fig. 21. Arquitectura para el cálculo de SAD [16].

Los datos de entrada están formados por 2 filas en donde una de ellas corresponde a la fila de píxeles que conforman el bloque actual y la otra a la fila de píxeles fraccionales generados en la etapa de interpolación; cada fila estará formada por 8 píxeles tal como se muestra en la figura 21.

Para obtener el resultado final, los valores de SAD parciales de cada fila deberán ser acumulados para obtener el valor correspondiente a la comparación de las 8 filas de cada bloque. Cada bloque SAD cuenta con su propia unidad de acumulación por lo cual se tienen 8 bloques acumuladores que son controlados independientemente de acuerdo a las señales generadas por el controlador acumulador SAD ½ como se muestra en la figura 20.

- Buffer de Píxeles Actuales

Las unidades de almacenamiento también son del tipo FIFO y siguen la misma arquitectura mostrada en la figura 15 pero difieren en la capacidad de almacenamiento. Para esta etapa del proceso, será necesario tener 2 buffers de 4x4 muestras y 1 buffer de 8x8 muestras. El objetivo de estas unidades es almacenar las muestras actuales para realizar el cálculo del SAD tanto en la etapa Half-pixel como en la etapa Quarter-pixel. La opción de lectura y/o escritura de estos espacios de almacenamiento es manejado por el controlador buffer

- Unidad de Decisión de Vector de Movimiento (Decisión MV ½)

Los valores de SAD acumulados permitirán tomar la decisión de cuál de las 8 posibilidades exhibe la mejor correlación con el cuadro actual tomando en cuenta el menor valor obtenido. Adicionalmente, es necesario asignar un vector de movimiento de acuerdo al mejor resultado y para ello se debe tener en cuenta la codificación binaria mostrada en la tabla 7.

La arquitectura de la unidad de decisión fue diseñada tomando como referencia la arquitectura mostrada en [13]. En la figura 22 se muestra a la arquitectura en mención.

Tabla 7. Codificación de Vectores de Movimiento

Vector de Movimiento	Codificación
(+X; 0)	0001
(-X; 0)	0011
(+Y; 0)	0100
(-Y; 0)	1100
(+X; +Y)	0101
(+X; -Y)	1101
(-X; +Y)	0111
(-X; -Y)	1111

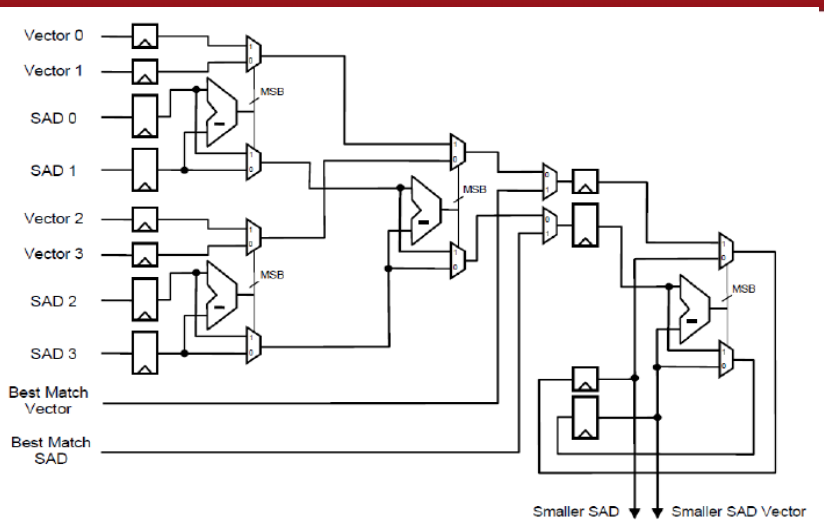


Fig. 22. Arquitectura de la Unidad de Decisión de Vector de Movimiento [13].

En este caso no se incluyó las etapas de pipeline presentes en la figura 22 para reducir el número de ciclos del proceso de búsqueda Half-pixel.

- Controlador SAD ½

El controlador SAD ½ está basado en una máquina de estados que permite habilitar el funcionamiento de cada uno de los 8 bloques de cálculo presentes en la unidad SAD ½. Las señales de control serán generadas a partir del contador y el estado presente. En el Anexo A, se detalla el diagrama de estados correspondiente para este controlador.

- Controlador Acumulador SAD ½

El Controlador Acumulador SAD ½ permite habilitar el funcionamiento de las unidades de acumulación que se encargarán de sumar los valores parciales obtenidos para cada uno de los 8 bloques de cálculo SAD. Las señales de control serán generadas a partir del valor del contador así como del estado presente. En el Anexo A, se detalla el diagrama de estados correspondientes para este controlador.

3.4.2 Arquitectura de la Unidad de Estimación de Movimiento Fraccional Quarter-Pixel

A. Arquitectura de la Unidad de Interpolación Quarter-Pixel

Finalizado el proceso de interpolación y búsqueda con precisión Half-pixel, se debe continuar con la segunda etapa del proceso de Estimación de Movimiento Fraccional teniendo como base el bloque de referencia con la mejor correlación obtenida en el proceso anterior de búsqueda Half-pixel porque las muestras fraccionales con precisión Quarter-pixel serán generadas y agrupadas alrededor de este bloque. En la figura 23, se muestra la arquitectura de la Unidad de Interpolación Quarter-Pixel.

En este caso, se reutilizará la arquitectura mostrada en la figura 14 porque el diseño permite que el filtro de interpolación sea capaz de generar muestras Half-pixel y Quarter-pixel para reducir el área utilizada en el dispositivo. Sin embargo, para esta etapa del proceso, los datos de entrada al filtro serán tomados del Buffer píxeles enteros señalizado por una conexión roja en el diagrama presentado.

La unidad de Interpolación deberá ser capaz de generar píxeles con precisión Quarter-pixel así como aquellos con precisión Half-pixel que en este caso son considerados auxiliares y son necesarios para calcular ciertas muestras verticales Quarter-pixel. Tomando en consideración el diseño de la unidad de Interpolación mostrado en las figuras 16, 17 y 18, es necesario que el controlador de interpolación determine el modo de funcionamiento de la unidad de interpolación y para ello sus señales de salida serán modificadas de acuerdo al valor del contador y al vector de movimiento obtenido en la primera etapa del proceso FME.

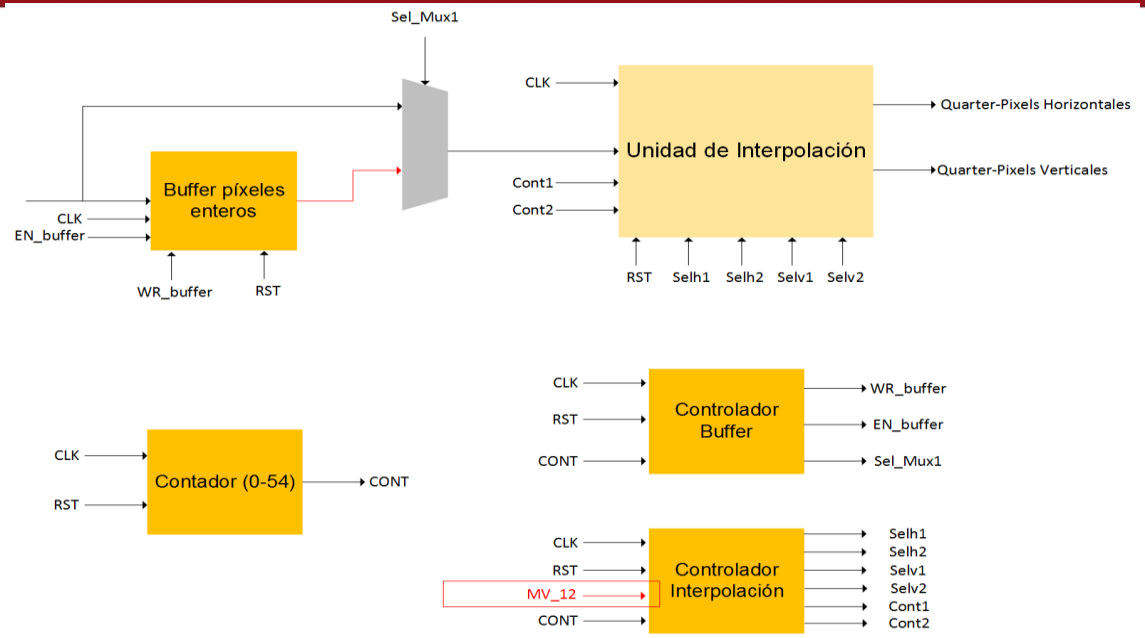


Fig. 23. Arquitectura Unidad de Interpolación Quarter-Pixel.

B. Arquitectura Unidad de Búsqueda Quarter-Pixel

Las muestras fraccionales determinadas en el proceso de interpolación permitirán generar una búsqueda alrededor de la mejor correlación obtenida con precisión Half-pixel, para ello se empleará la arquitectura mostrada en la figura 24.

Las muestras interpoladas serán agrupadas utilizando los multiplexores mostrados en la arquitectura; estos multiplexores tienen en cuenta al vector de movimiento con precisión Half-pixel.

Luego de la etapa de multiplexores, las muestras interpoladas ingresarán a la unidad SAD ¼ y Acumulador SAD ¼ del mismo modo como sucede en la unidad de búsqueda Half-Pixel. Ambos controladores están basados en máquinas de estado que se encargarán de modificar las señales de acuerdo al contador, al vector de movimiento con precisión Half-pixel y al estado presente. Adicionalmente, se reutilizarán los Buffers de píxeles actuales ya que contienen los datos necesarios para realizar la comparación.

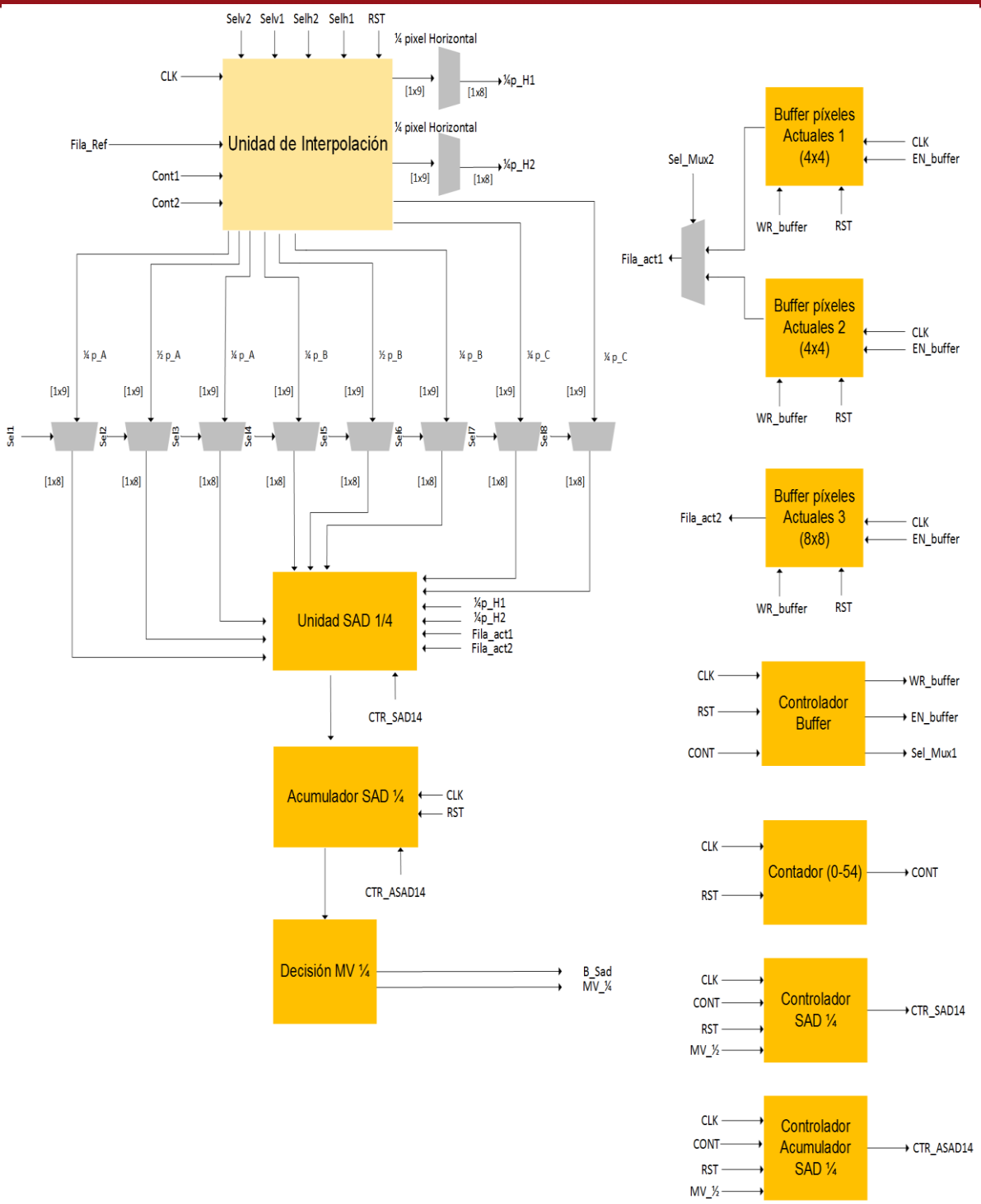


Fig. 24. Arquitectura Unidad de Búsqueda Quarter-Pixel.

Capítulo 4

Simulación, Validación y Resultados

4.1 Verificación del algoritmo de Estimación de Movimiento Fraccional

El algoritmo de Estimación de Movimiento Fraccional fue verificado por medio de una aplicación realizada en el entorno de programación MATLAB®. Los resultados de la aplicación han permitido comparar y verificar los datos obtenidos por medio de la arquitectura diseñada.

La implementación del algoritmo está basada en el software de código abierto proporcionado en [17]. En la figura 25 se detalla el diagrama de flujo del algoritmo implementado.

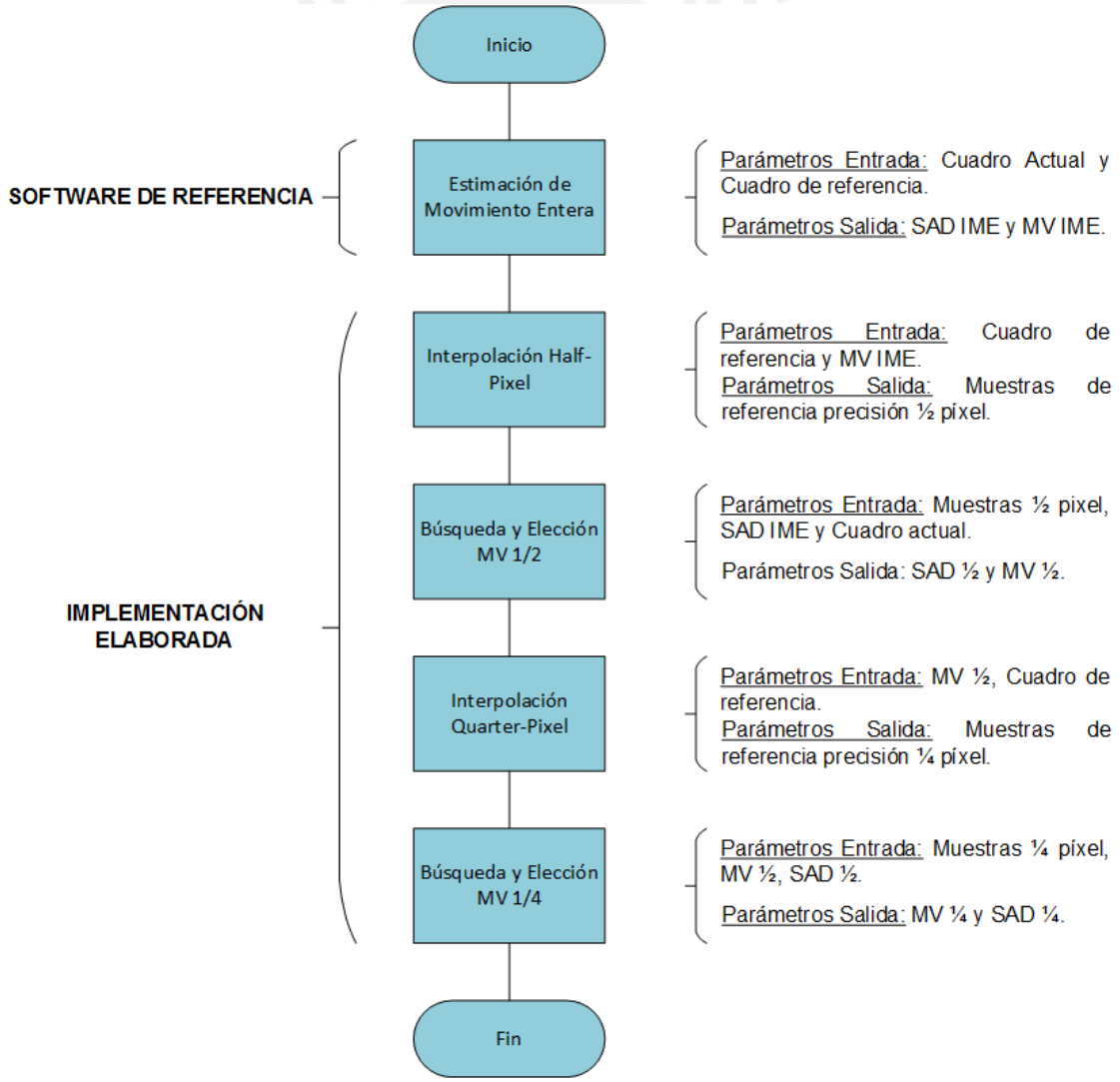


Fig. 25. Diagrama de Flujo del Algoritmo de Estimación de Movimiento.

Mediante el software de referencia se logra realizar la etapa de Estimación de Movimiento Entera la cual permite ubicar al bloque de referencia con el cual se debe realizar la Estimación de Movimiento Fraccional. Se debe resaltar que el programa desarrollado tiene como fin únicamente verificar los resultados así como generar los estímulos necesarios para realizar la simulación de la arquitectura.

La etapa de Estimación de Movimiento Fraccional desarrollada incluye las funciones de interpolación con precisión Half-pixel y Quarter-pixel; adicionalmente, se encargará de la búsqueda y determinación del Vector de Movimiento resultante.

4.2 Resultados de la Aplicación de Verificación

El software realizado fue probado utilizando 8 secuencias de video con diferentes resoluciones. Las secuencias de video que se emplearon se encuentran disponibles en [26] y fueron las siguientes:

- Resolución QCIF (176 x 144): “ice_qcif_15fps.y4m”, “suzie_qcif.y4m” y “akiyo_qcif.y4m”.
- Resolución CIF (352 x 288): “bus_cif.y4m”, “foreman_cif.y4m”, “Stefan_cif.y4m”, “akiyo_cif.y4m” y “tennis_sif.y4m”.

En cada secuencia de video se utilizó al cuadro 5 como cuadro actual mientras que el cuadro 4 fue tomado como cuadro de referencia. Empleando los cuadros mencionados en cada secuencia de video, se realizaron las siguientes pruebas de funcionamiento: considerando únicamente IME, considerando FME con precisión Half-pixel y considerando FME con precisión Quarter-pixel. Estas tres pruebas se realizaron con la finalidad de demostrar que el proceso de Estimación de Movimiento Fraccional consigue menor energía residual mientras mayor sea la precisión de las muestras interpoladas con lo cual será posible conseguir una mayor tasa de compresión al momento de realizar la codificación completa del cuadro analizado [3][4].

En la tabla 8, 9 y 10 se muestran los resultados de SAD mínimos para cada bloque de 8x8 píxeles de la secuencia “Ice” considerando las tres pruebas mencionadas.

Tabla 8. Resultados SAD mínimos de la secuencia “Ice” por medio de IME para cada bloque de 8x8 píxeles.

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 9. Resultados SAD mínimos de la secuencia “Ice” por medio de FME con precisión Half-pixel para cada bloque de 8x8 píxeles.

32	35	45	1134	1703	1018	37	28	309	66	63	17	11	20	21	39	15	31	35	24	20	16
26	31	459	332	684	640	15	14	196	466	96	16	10	38	40	21	34	38	67	26	12	16
15	225	955	163	254	574	20	23	209	937	152	19	23	1251	1730	93	847	1096	1004	601	351	137
53	321	319	245	300	69	38	24	92	378	61	17	31	142	405	2134	730	428	333	123	136	123
47	255	389	392	189	34	29	23	17	19	18	17	61	152	354	103	955	26	242	79	98	296
16	33	251	638	156	13	20	24	17	16	12	39	158	211	698	2099	122	68	605	338	133	437
25	30	496	516	158	17	17	13	11	12	14	46	165	203	1122	2198	1420	1124	1570	257	204	455
29	61	584	1059	245	29	44	33	15	23	33	43	756	27	17	1131	562	214	142	141	125	273
38	48	150	549	243	26	16	17	20	15	37	59	904	35	395	1035	93	40	116	119	395	538
37	60	47	15	17	16	12	21	13	19	609	398	993	50	34	1224	450	97	186	1289	345	531
48	22	17	13	17	19	21	19	27	54	409	681	1560	353	231	909	1912	209	86	690	766	426
93	15	18	20	13	18	20	22	31	51	57	83	315	392	592	91	99	55	122	452	183	199
199	26	24	19	16	11	19	43	54	57	38	40	115	142	230	182	65	73	164	65	28	18
212	29	24	16	20	20	12	16	19	20	33	21	181	171	153	175	94	114	196	116	18	8
288	39	17	16	19	23	21	22	23	25	30	39	278	248	164	409	128	354	163	173	20	20
377	32	15	18	24	20	20	15	22	28	25	28	58	46	58	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 10. Resultados SAD mínimos de la secuencia “Ice” por medio de FME con precisión Quarter-pixel para cada bloque de 8x8 píxeles.

32	35	45	1134	1703	982	37	28	206	66	63	17	11	20	21	39	15	31	35	24	20	16
26	31	444	332	678	607	15	14	159	431	96	16	10	38	40	21	34	38	67	26	12	16
15	225	955	145	244	530	20	23	147	937	152	19	23	1238	1730	93	761	1096	1004	574	196	122
53	159	203	238	260	69	38	24	75	331	61	17	31	142	399	2083	730	428	301	121	136	123
47	222	368	386	189	34	29	23	17	19	18	17	61	147	284	94	941	26	242	79	98	296
16	33	169	498	156	13	20	24	17	16	12	39	151	211	697	2053	111	68	587	338	131	437
25	30	496	515	153	17	17	13	11	12	14	46	134	179	1122	2186	1398	1120	1570	257	204	455
29	61	457	1048	239	29	44	33	15	23	33	43	756	27	17	1131	539	125	101	134	125	273
38	48	121	549	213	26	16	17	20	15	37	59	878	35	395	1035	93	40	116	106	334	433
37	60	47	15	17	16	12	21	13	19	571	303	963	50	34	1194	367	91	146	1289	345	531
48	22	17	13	17	19	21	19	27	54	409	667	1549	295	231	891	1912	209	86	687	676	359
93	15	18	20	13	18	20	22	31	51	57	80	315	389	563	70	92	55	122	445	158	145
139	26	24	19	16	11	19	43	54	57	38	40	106	112	227	180	65	73	154	65	28	17
111	29	21	16	20	20	12	16	19	20	33	21	115	154	153	175	50	111	151	50	18	8
111	38	17	16	19	23	21	22	23	25	30	39	278	248	109	354	79	255	94	88	20	20
377	32	15	18	24	20	20	15	22	28	25	28	58	45	58	89	68	77	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

Para reconocer la diferencia entre los tres resultados se sumaron los valores de SAD mínimo para cada prueba realizada los cuales son presentados en la tabla 11.

Tabla 11. Comparación de Energía residual entre procesos y entre Estándares de Codificación.

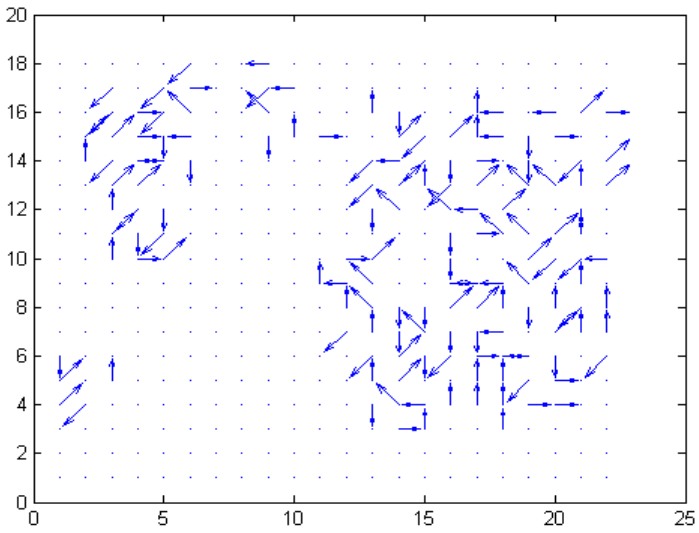
Proceso	Estándar H.264 [13]	Estándar HEVC
IME	87634	87634
FME Half-pixel	81582	81352
FME Quarter-pixel	79491	77257

Se observa que la prueba considerando únicamente IME presenta una mayor energía residual, esta energía es disminuida al utilizar el proceso FME con precisión Half-pixel. Utilizando la segunda etapa del proceso, es decir usando muestras con precisión Quarter-pixel, se consigue disminuir aún más la energía residual. Por lo tanto, se comprueba que los procesos de Estimación de Movimiento Fraccional permiten disminuir la energía residual de los cuadros de video a medida que se empleen píxeles fraccionales de mayor precisión. Adicionalmente, en la tabla 11 se muestran los resultados del estándar H.264

utilizando los mismos cuadros de la secuencia “Ice”. De acuerdo a esto, el estándar HEVC permite mejorar los resultados obtenidos por medio del estándar H.264/AVC ya que se obtienen menores valores de SAD mínimo tanto para el proceso con precisión Half-pixel así como para el proceso Quarter-pixel. Estas mejoras corresponden al rediseño de los filtros de interpolación en el estándar HEVC ya que las muestras Half-pixel son calculadas utilizando un filtro FIR de 8-taps a diferencia del filtro FIR de 6-taps utilizado por el estándar H.264/AVC. De igual modo, las muestras Quarter-pixel son interpoladas por un filtro FIR de 7-taps; por el contrario, el estándar H.264/AVC utiliza la semisuma de muestras adyacentes. Estos cambios fueron realizados con la finalidad de encontrar una mejor correlación con el cuadro actual para que el codificador pueda disminuir la tasa de transmisión [3] [4].

Mediante la aplicación desarrollada también es posible mostrar los vectores de movimiento para cada bloque de 8x8 píxeles que conforman el cuadro actual. La figura 26 (a) muestra únicamente al cuadro 5 de la secuencia “Ice” y la figura 26 (b) muestra los vectores de movimiento resultantes del proceso FME (Quarter-pixel) para el mismo cuadro.

(a)



(b)

Fig. 26. (a) Muestras de Luminancia del cuadro 5 de la secuencia “Ice”. (b) Vector de Movimiento resultante del proceso FME para cada bloque de 8x8 píxeles.

Aquellas zonas de la figura 26 (b) en donde no existen vectores de movimiento corresponden a áreas del cuadro actual con redundancia temporal y por ello son representadas como puntos que indican un gradiente de (0; 0).

La aplicación desarrollada permite generar tres archivos de estímulo que generarán las señales de entrada para la arquitectura. En un archivo se tienen los píxeles del cuadro actual divididos en bloques de 8x8 muestras, en otro archivo se tienen los píxeles del cuadro de referencia divididos en bloques de 16x16 píxeles y será necesario un tercer archivo con la información del SAD obtenido mediante el proceso IME.

4.3 Simulación de la arquitectura

La verificación del funcionamiento de la arquitectura se realizará empleando la secuencia “ice_qcif_15fps.y4m” de resolución 178x144 píxeles, específicamente se utilizará el cuadro 5 como cuadro actual y al cuadro 4 como cuadro de referencia. Los resultados de la simulación de la arquitectura así como aquellos obtenidos por medio de la aplicación de verificación son presentados en el Anexo B.

Del mismo modo, se adjunta en el Anexo C los resultados de simulación para la secuencia “speed_bag_1080p.y4m” de resolución 1920x1080 píxeles en donde el cuadro actual es el cuadro 5 de la secuencia en mención y el cuadro 4 se utiliza como cuadro de referencia.

Los resultados de la simulación demuestran que la arquitectura diseñada demora 56 ciclos de reloj en procesar un bloque de 8x8 píxeles del cuadro actual.

4.4 Resultados de Síntesis de la arquitectura

La síntesis de la arquitectura diseñada se realizó para los dispositivos de la familia Virtex utilizando el software ISE v14.7 de la compañía Xilinx.

En la figura 27 se muestran los resultados para el dispositivo Virtex-4. En la figura (a) se muestra los resultados de cantidad de elementos lógicos utilizados por la arquitectura y en la figura (b) se muestran las restricciones de tiempo analizadas para este dispositivo.

Device Utilization Summary

Logic Utilization	Used	Available	Utilization	Note(s)
Number of Slice Flip Flops	18,179	178,176	10%	
Number of 4 input LUTs	19,274	178,176	10%	
Number of occupied Slices	11,986	89,088	13%	
Number of Slices containing only related logic	11,986	11,986	100%	
Number of Slices containing unrelated logic	0	11,986	0%	
Total Number of 4 input LUTs	19,396	178,176	10%	
Number used as logic	19,082			
Number used as a route-thru	122			
Number used as Shift registers	192			
Number of bonded IOBs	212	960	22%	
Number of BUFG/BUFGCTRLs	1	32	3%	
Number used as BUFGs	1			
Average Fanout of Non-Clock Nets	3.10			

Design Summary

(a)

Constraint	Check	Worst Case Slack	Best Case Achievable	Timing Errors	Timing Score
TS_CLK = PERIOD TIMEGRP "CLK" 16.67 ns HI GH 50%	SETUP	0.379ns	16.291ns	0	0
	HOLD	0.377ns		0	0

(b)

Fig. 27. Resultados de síntesis de la arquitectura diseñada para el FPGA Virtex-4. (a) Cantidad de elementos utilizados. (b) Restricciones de tiempo.

A partir del mínimo período de reloj (“*Best Case Achievable*”) mostrado en la figura 27 (b) se obtiene una frecuencia máxima de operación de 61.38 MHz. De acuerdo a este resultado, es posible alcanzar la siguiente tasa de procesamiento para las secuencias de video con resolución HDTV (1920x1080 píxeles):

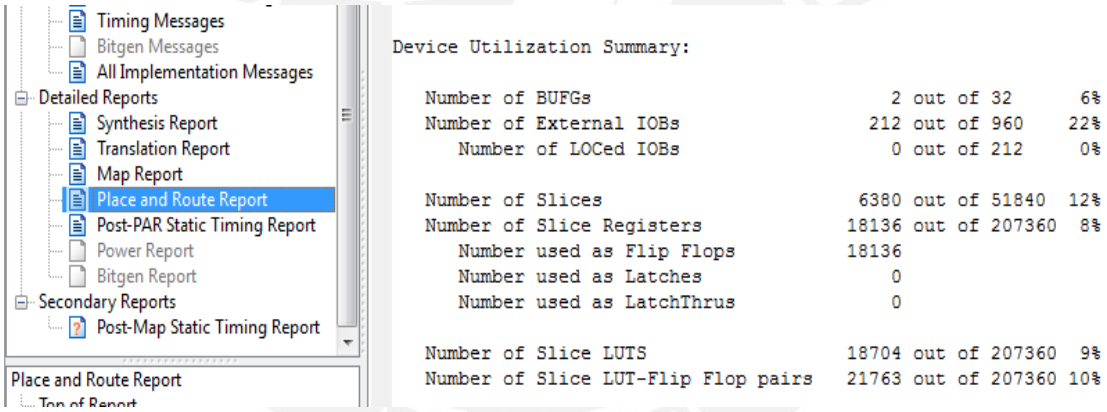
- 1 Cuadro HDTV = 32400 bloques de 8x8 píxeles.
- Frecuencia Máxima de reloj = 61.38 MHz.
- N° de ciclos de reloj para procesar 1 bloque de 8x8 píxeles = 56 ciclos.

$$56\left[\frac{ciclos}{bloque}\right] * 32400\left[\frac{bloques}{cuadro}\right] * 16.291\ \left[\frac{ns}{ciclo}\right] = 0.029\ [\frac{ns}{cuadro}]$$

$$Tasa\ de\ Procesamiento = 34.48\ fps$$

Por lo tanto, la síntesis de la arquitectura en un FPGA Virtex-4 demuestra que es capaz de cumplir con un requerimiento de procesamiento mayor o igual a 30 fps para secuencias de resolución de 1920x1080 píxeles.

En la figura 28, se muestran los resultados para el dispositivo Virtex-5 en donde la figura (a) muestra la cantidad de elementos lógicos empleados por la arquitectura y la figura (b) muestra las restricciones de tiempo analizadas para este dispositivo.



(a)

Constraint	Check	Worst Case Slack	Best Case Achievable	Timing Errors	Timing Score
TS_CLK = PERIOD TIMEGRP "CLK" 16.67 ns HIGH 50%	SETUP	0.041ns	16.629ns	0	0
	HOLD	0.217ns		0	0

(b)

Fig. 28. Resultados de síntesis de la arquitectura diseñada para el FPGA Virtex-5. (a) Cantidad de elementos utilizados. (b) Restricciones de tiempo

A partir del mínimo período de reloj ("*Best Case Achievable*") mostrado en la figura 28 (b), se obtiene una frecuencia máxima de operación de 60.13 MHz. De acuerdo a este resultado, es posible alcanzar la siguiente tasa de procesamiento para las secuencias de video con resolución HDTV (1920x1080 píxeles):

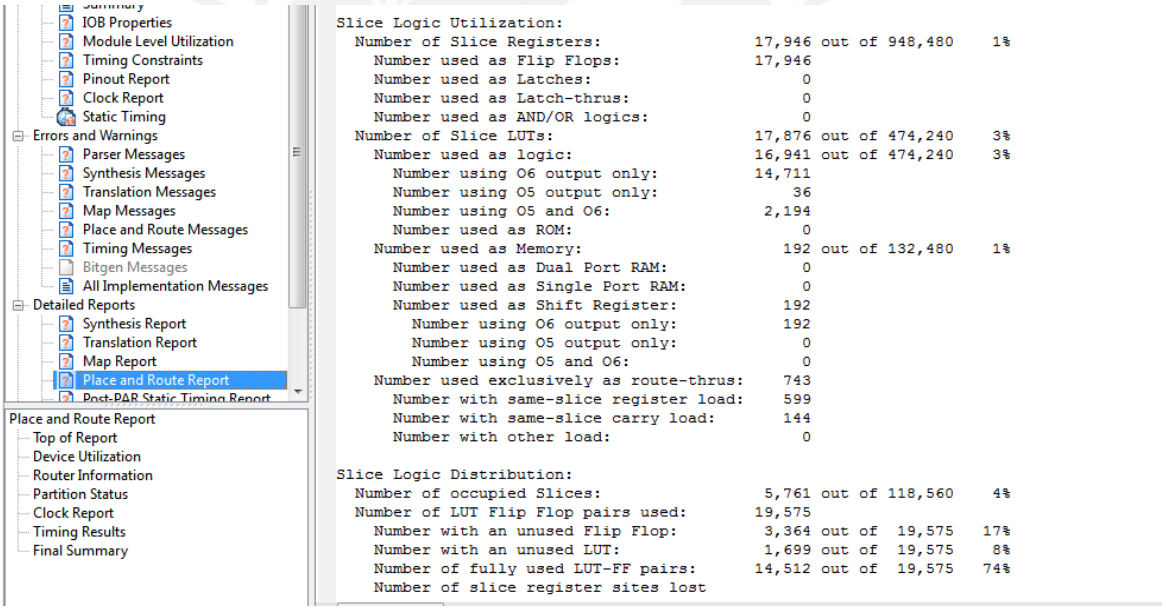
- 1 Cuadro HDTV = 32400 bloques de 8x8 píxeles.
- Frecuencia Máxima de reloj = 60.13 MHz.
- N° de ciclos de reloj para procesar 1 bloque de 8x8 píxeles = 56 ciclos.

$$56\left[\frac{ciclos}{bloque}\right] * 32400\left[\frac{bloques}{cuadro}\right] * 16.629\ \left[\frac{ns}{ciclo}\right] = 0.030\ [\frac{ns}{cuadro}]$$

$$Tasa\ de\ Procesamiento = 33.33\ fps$$

Por lo tanto, la síntesis de la arquitectura en un FPGA Virtex-5 demuestra que es capaz de cumplir con un requerimiento de procesamiento mayor o igual a 30 fps para secuencias de resolución de 1920x1080 píxeles.

En la figura 29, se muestran los resultados para el dispositivo Virtex-6 en donde la figura (a) muestra la cantidad de elementos lógicos empleados por la arquitectura y la figura (b) muestra las restricciones de tiempo analizadas para este dispositivo.



(a)

Constraint	Check	Worst Case Slack	Best Case Achievable	Timing Errors	Timing Score
TS_CLK = PERIOD TIMEGRP "CLK" 16.67 ns HI GH 50%	SETUP	4.688ns	11.982ns	0	0
	HOLD	0.028ns		0	0

(b)

Fig. 29. Resultados de síntesis de la arquitectura diseñada para el FPGA Virtex-6. (a) Cantidad de elementos utilizados. (b) Restricciones de tiempo.

A partir del mínimo período de reloj ("*Best Case Achievable*") mostrado en la figura 29 (b), se obtiene una frecuencia máxima de operación de 83.46 MHz. De acuerdo a este resultado, es posible alcanzar la siguiente tasa de procesamiento para las secuencias de video con resolución HDTV (1920x1080 píxeles):

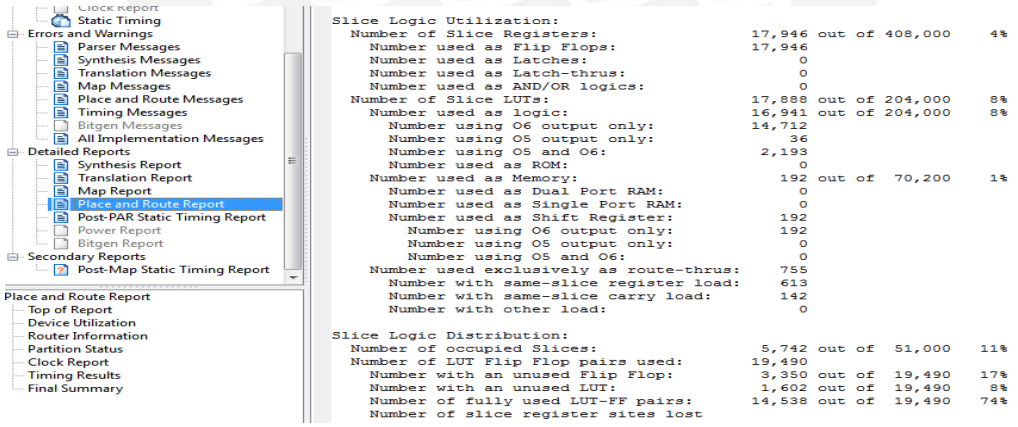
- 1 Cuadro HDTV = 32400 bloques de 8x8 píxeles.
- Frecuencia Máxima de reloj = 83.46 MHz.
- N° de ciclos de reloj para procesar 1 bloque de 8x8 píxeles = 56 ciclos.

$$56\left[\frac{ciclos}{bloque}\right] * 32400\left[\frac{bloques}{cuadro}\right] * 11.982\ \left[\frac{ns}{ciclo}\right] = 0.021\ [\frac{ns}{cuadro}]$$

$$Tasa\ de\ Procesamiento = 47.62\ fps$$

Por lo tanto, la síntesis de la arquitectura en el FPGA Virtex-6 demuestra que es capaz de cumplir con un requerimiento de procesamiento mayor o igual a 30 fps para secuencias de resolución de 1920x1080 píxeles.

En la figura 30, se muestran los resultados para el dispositivo Virtex-7 en donde la figura (a) muestra la cantidad de elementos lógicos empleados por la arquitectura y la figura (b) muestra las restricciones de tiempo analizadas para este dispositivo.



(a)

Constraint	Check	Worst Case Slack	Best Case Achievable	Timing Errors	Timing Score
TS_CLK = PERIOD TIMEGRP "CLK" 16.67 ns HI GH 50%	SETUP	6.430ns	10.240ns	0	0
	HOLD	0.000ns		0	0

(b)

Fig. 30. Resultados de síntesis de la arquitectura diseñada para el FPGA Virtex-7. (a) Cantidad de elementos utilizados. (b) Restricciones de tiempo.

A partir del mínimo período de reloj ("*Best Case Achievable*") mostrado en la figura 30 (b), se obtiene una frecuencia máxima de operación de 97.65 MHz. De acuerdo a este resultado, es posible alcanzar la siguiente tasa de procesamiento para las secuencias de video con resolución HDTV (1920x1080 píxeles):

- 1 Cuadro HDTV = 32400 bloques de 8x8 píxeles.
- Frecuencia Máxima de reloj = 97.65 MHz.
- N° de ciclos de reloj para procesar 1 bloque de 8x8 píxeles = 56 ciclos.

$$56\left[\frac{ciclos}{bloque}\right] * 32400\left[\frac{bloques}{cuadro}\right] * 10.240\ \left[\frac{ns}{ciclo}\right] = 0.018\ [\frac{ns}{cuadro}]$$

$$Tasa\ de\ Procesamiento = 55.55\ fps$$

Por lo tanto, la síntesis de la arquitectura en el FPGA Virtex-7 demuestra que es capaz de cumplir con un requerimiento de procesamiento mayor o igual a 30 fps para secuencias de resolución de 1920x1080 píxeles.

En resumen, la tabla 12 muestra los resultados de máxima frecuencia de operación y tasa de procesamiento obtenidos para cada FPGA analizado en la presente sección.

Tabla 12. Resultados de máxima frecuencia de operación y tasa de procesamiento.

Dispositivo	Máxima Frecuencia de Operación	Tasa de procesamiento
Virtex-4	61.38 MHz	34.48 fps
Virtex-5	60.13 MHz	33.33 fps
Virtex-6	83.46 MHz	47.62 fps
Virtex-7	97.65 MHz	55.55 fps

Conclusiones

- Mediante el paralelismo de operaciones de la implementación en hardware del algoritmo y considerando los resultados de síntesis, se ha logrado obtener una frecuencia de operación suficiente para procesar una secuencia de 1920x1080 píxeles a una tasa de procesamiento mayor a 30 fps en los dispositivos de la familia Virtex analizados. Por lo tanto, el objetivo principal del presente trabajo de tesis ha sido alcanzado.

- Los resultados obtenidos por medio de la implementación del algoritmo utilizando el software MATLAB® ha permitido demostrar que implementar el algoritmo de Estimación de Movimiento Fraccional con precisión Quarter-pixel permite obtener un menor valor de SAD para las secuencias de video con lo cual es posible alcanzar una mayor compresión al momento de realizar el proceso completo de codificación exigido por el estándar HEVC.

- Mediante los resultados de la aplicación de verificación desarrollada en MATLAB® del algoritmo de Estimación de Movimiento Fraccional se ha demostrado que las mejoras incluidas permitirán mejorar la compresión de datos en comparación con el estándar H.264/AVC.

Recomendaciones

- Tomar en consideración estudios futuros sobre mejoras en el algoritmo de Estimación de Movimiento Fraccional los cuales permitirían reducir las posibilidades de comparación tanto para precisión Half-pixel como Quarter-pixel. Estas mejoras se reflejarían en una menor área de implementación del algoritmo así como una menor cantidad de ciclos de reloj para un procesamiento de bloques de 8x8 píxeles.

- Implementar el algoritmo de Estimación de Movimiento Fraccional tomando en consideración bloques de tamaños variables teniendo en cuenta el diseño de la arquitectura mostrado en el presente trabajo de tesis.

- Se recomienda diseñar los módulos adicionales del codificador HEVC para realizar su posterior implementación en prototipo sobre un FPGA y tener una versión final en un ASIC.

Bibliografía

[1] Samsung

2015 Galaxy S6 [en línea] [Revisado el 26/07/2015]

<http://www.samsung.com/es/consumer/mobile-devices/smartphones/galaxy-s/SM-G920FZWFPHE>

[2] Fraunhofer Heinrich Hertz Institute

2015 “Fraunhofer HHI technology enables broadcasting of the first live convert in 4k on televisión. [en línea] [Revisado el 30/07/2015]

<http://www.hhi.fraunhofer.de/en/press-media/news/fraunhofer-hhi-technology-enables-broadcasting-of-the-first-live-concert-in-4k-on-television.html>

[3] Sullivan, G.; Ohm, J.; Han, W.; Wiegand, T.

2012 "Overview of the High Efficiency Video Coding (HEVC) Standard", Circuits and Systems for Video Technology, IEEE Transactions, Vol.22, No.12, pp. 1649-1668. [en línea] [Revisado el 24/03/2015].

[4] Sullivan, G.; Budagavi, M.; Sze, M.

2014 “High Effiency Video Coding (HEVC) Algorithms and Architectures”. New York: Springer.

[5] Ministerio de Transportes y Comunicaciones

2011 “Comunicado” [en línea] [Revisado el 10/04/2015]

<http://www.mtc.gob.pe/portal/tdt/comunicado.html.>

[6] Nalluri, P.; Alves, L.; Navarro, A.

2014 "High speed SAD architectures for variable block size motion estimation in HEVC video coding", Image Processing (ICIP), 2014 IEEE International Conference, pp.1233-1237. [en línea] [Revisado el 12/04/2015]

[7] Iain, G.; Richardson, G.

2003 “H.264 and MPEG-4 Video Compression: Video Coding for Next Generation”. New York: John Wiley & Sons, Inc.

[8] Xiaocong, L.; Zhou, X.; Zhou, W.; Liu, W.; Liu, X.

2014 "An efficient interpolation filter VLSI architecture for HEVC standard", Signal and Information Processing (ChinaSIP), 2014 IEEE China Summit & International, pp.384-388. [en línea] [Revisado el 01/04/2015].

[9] Code: Sequoia

2015 “HEVC – What are CTU, CU, CTB, CB, PB and TB?” [en línea] [Revisado el 29/06/2015].

<https://codesequoia.wordpress.com/2012/10/28/hevc-ctu-cu-ctb-cb-pb-and-tb/>

[10] Ugul, K.; Alshin, A.; Alshina, E.; Bossen, F.; Woo-Jin, H.; Jeong-Hoon, P.; Lainema, J.

2013 “Motion Compensated Prediction and Interpolation Filter Design in H.265/HEVC”, IEEE Journal of selected topics in Signal Processing, Vol.7, No. 6, pp. 946-956. [en línea] [Revisado el 04/05/2015].

[11] Il-Koo, K.; Junghye, M.; Lee, T.; Woo-Jin, H.; JeongHoon, P.

2012 "Block Partitioning Structure in the HEVC Standard", Circuits and Systems for Video Technology, IEEE Transactions, Vol.22, No.12, pp.1697-1706. [en línea] [Revisado el 01/04/2015].

[12] Sze, V.; Budagavi, M.

2014 “Design and Implementation of Next Generation Video Coding Systems” [en línea] [Revisado el 20/09/2015]

<http://www.rle.mit.edu/eems/prof-vivienne-sze-and-dr-madhukar-budagavi-to-give-tutorial-on-design-and-implementation-of-next-generation-video-coding-systems-at-iscas-2014/>

[13] Villegas, C.

2011 “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. Lima: Pontificia Universidad Católica del Perú, Facultad de Ciencias e Ingeniería.

[14] Diniz, C.; Shafique, M.; Bampi, S.; Henkel, J.

2015 “A Reconfigurable Hardware Architecture for Fractional Pixel Interpolation in High Efficiency Video Coding”, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Vol. 34, No. 2, pp. 238-251. [en línea] [Revisado el 04/08/2015]

[15] Penny, W.; Paim, G.; Porto, M.; Agostini, L.; Zatt, B.

2015 “A Hardware Architecture for an HEVC Motion Compensation Luminance Interpolator”, XXX South Symposium on Microelectronics. [en línea] [Revisado el 01/08/2015].

[16] Maich, H.; Afonso, V.; Zatt, B.; Agostini, L.; Porto, M.

2014 "HEVC Fractional Motion Estimation complexity reduction for real-time applications", Circuits and Systems (LASCAS), 2014 IEEE 5th Latin American Symposium, pp.1,4, 25-28. [en línea] [Revisado el 01/04/2015].

[17] Hoelzer, S.

2005 “Matlab Codec Project”. [En línea]. [Revisado el 10/10/2015].

<http://www.cs.cf.ac.uk/Dave/Multimedia/Lecture_Examples/Compression/mpegproj/>

[18] Sanders, J.; Kandrot, E.

2014 “CUDA by Example: An Introduction to General-Purpose GPU Programming”. New York: Addison-Wesley, pp. 2-11. [en línea] [Revisado el 20/04/2015].

<http://www.physics.drexel.edu/~valliere/PHYS405/GPU_Story/CUDA_by_Example_Addison_Wesley_Jul_2010.pdf>

[19] Ittanim Systems

2013 “Compute accelerated HEVC decoder on ARM MaliTM-T600 GPUs”. [en línea] [Revisado el 20/09/2015].

<http://www.arm.com/files/event/Mali_Partner_Track_4_GPU_Compute_accelerated_HEVC_decoder_on_ARM_Mali-T600_GPUs.pdf

[20] Cho, S.; Hyunmi, K.; Kyungjin, B.; and Nak-Woong, E.

2013 "Multi-core based HEVC hardware decoding system," Multimedia and Expo Workshops (ICMEW), 2014 IEEE International Conference on, Chengdu, 2014, pp. 1-2. [en línea] [Revisado el 20/09/2015]

[21] Huang, T.; Tikekar, M.; Juvekar, C.; Sze, V.; Chandrakasan, A.

2013 “A 249Mpixel/s HEVC video-decoder chip for Quad Full HD applications”. Solid-State Circuits Conference Digest of Technical Papers (ISSCC), pp-162-163. [en línea]. [Revisado el 20/09/2015].

[22] Thomas, D.; Moorby, P.

2008 “The Verilog Hardware Description Language”. Springer

[23] XILINX

2014 “Field Programmable Gate Array” [en línea] [Revisado el 16/04/2015].

<http://www.xilinx.com/training/fpga/fpga-field-programmable-gate-array.htm>

[24] Xiang-Wen, W.; Li, S.; Min, C.; Jun-jie, Y.

2013 "Paralleling variable block size motion estimation of HEVC on multi-core CPU plus GPU platform", Image Processing (ICIP), 2013 20th IEEE International Conference, pp.1836-1839. [en línea] [Revisado el 10/04/2015]

[25] Medhat, A.; Shalaby, A.; Sayed, M.S.; Elsabrouty, M.; Mehdipour, F.

2014 "A highly parallel SAD architecture for motion estimation in HEVC encoder", Circuits and Systems (APCCAS), 2014 IEEE Asia Pacific Conference, pp.280-283. [en línea] [Revisado el 10/04/2015].

[26] Xiph.org

2010 “Index of /video/derf/y4m”. [en línea] [Revisado el 08/09/2015]

<https://media.xiph.org/video/derf/y4m/>

[27] Rabaey, J

2002 “Digital integrated circuits: a design perspective”. New Jersey: Prentice Hall, Inc, pp. 296-300.

[28] Chu, P.

2006 “RTL Hardware Design Using VHDL: Coding for Efficiency, Portability and Scalability”. Wiley-IEEE Press.

[29] Maxfield Clive

2009 “FPGA World Class Designs”. Oxford: Elsevier, pp. 2-43.