17 del 24 al 28 de noviembre de 2014
CONVENCIÓN CIENTÍFICA
DE INGENIERÍA Y ARQUITECTURA
PALACIO DE CONVENCIONES DE LA HABANA
50 años cujae

IMPLEMENTACIÓN DE LOS BLOQUES DCT, CUANTIFICACIÓN Y CAVLC DE UN CODIFICADOR H.264/AVC EN LA HERRAMIENTA SIMULINK.

Primer Autor (Ing. José Angel Espinosa Roque)[1], Segundo Autor (Ing. Dalila Garrido Mirabal)[2]

[1]***LACETEL***, Cuba, Ave. Rancho Boyeros, #34515 (19200), Zona Postal: General Peraza, La Habana
[2]***LACETEL***

RESUMEN

Actualmente el estándar de compresión de video H.264/AVC es utilizado en nuestro país para el despliegue de la Televisión Digital Terrestre. Es por esta razón que el presente artículo constituye un aporte a las investigaciones realizadas en ***LACETEL***® referente al tema, teniendo como punto fundamental el desarrollo de un modelo de la etapa de Transformación y Cuantificación del codificador H.264/AVC en la herramienta Simulink de Matlab, donde el objetivo principal radica en lograr una reducción de la información a transmitir para disminuir la memoria de almacenamiento a utilizar y los tiempos de procesamiento. Como etapa inicial del proyecto se utilizó como información de entrada imágenes con formato QCIF y se realiza la conversión de los datos según las especificaciones de Simulink. Se describen los bloques funcionales en que se divide el diseño realizado y se presentan las diferentes validaciones a través de los resultados obtenidos en las simulaciones y la bibliografía consultada.

PALABRAS CLAVES: Transformada Discreta del Coseno, cuantificación, estándar H.264/AVC, compresión de imágenes.

IMPLEMENTATION OF DCT, QUANTIZATION AND CAVLC BLOCKS FOR H.264/AVC ENCODER IN SIMULINK TOOL.

ABSTRACT

Currently the H.264/AVC video compression standard is used in our country for the deployment of Digital Terrestrial Television. For that reason this article is a contribution to the research conducted in ***LACETEL***® regarding the issue. The main point of this work is the development of a model for the transformation and quantization stage of a H.264/AVC encoder in Matlab® Simulink® tool, where the main objective is to achieve a reduction of the information to be transmitted, therefore decreasing the processing times and the amount of storage memory to be used. In the initial stage of the project were used images with QCIF format as input and data conversion was performed according to Simulink specifications. To conclude, the different functional blocks of the design are described and the different validations are presented through the achieved results and the bibliography consulted.

KEY WORDS: Discrete Cosine Transform, Quantization, H.264/AVC standard, image compression.

1. INTRODUCCIÓN

La transición de los sistemas analógicos a digitales ha sido una de las características del progreso tecnológico predominante en el sector de las telecomunicaciones durante las últimas dos décadas, lo que ha posibilitado el desarrollo de la TDT (Televisión Digital Terrestre)[1] a nivel mundial, permitiendo no solo una mejora en

[1] TDT (Es la transmisión de imágenes en movimiento y el sonido asociado, mediante una señal digital y a través del uso de una red de repetidores terrestres).

calidad de los servicios sino además un aumento considerable de los mismos. Una de las principales ventajas que aporta la TDT es la posibilidad de comprimir o codificar la señal digital de video permitiendo obtener la misma señal pero con una menor cantidad de bits, lo que implica reducir y eliminar los datos redundantes y que al transformarla en un nuevo código binario esta señal resultante pueda ser almacenada y transmitida de forma más eficiente[1], lo que se traduce en un aprovechamiento mayor del canal, sin que esto represente algún problema para la visualización de la imagen decodificada. Los estándares de compresión de video han sido desarrollados con el fin de satisfacer una amplia gama de aplicaciones, los más utilizados actualmente son los estándares H.262[2], H.264/AVC[3] y AVS1-P2[4]. Nuestro país no ha quedado detrás en este proceso de despliegue de la televisión digital por lo que en ***LACETEL***® se tributa a la obtención de módulos IP (Propiedad Intelectual) en el proceso de Transmisión de la TDT, formando parte de ellos la compresión de video que es el enfoque principal de este artículo.

2. GENERALIDADES DEL DISEÑO

A pesar que H.264/AVC no normaliza una forma específica de implementación de la codificación este sistema de compresión está basado en el modelo híbrido DPCM/DCT (*Differential Pulse Code Modulation/Discrete Cosine Transform*) también conocido como Esquema de Codificación de Video Hibrido Basado en Bloques como se muestra en la Figura. 1. El diseño propuesto está enmarcado en la etapa de salida del codificador y la entrada de la realimentación hacia el decodificar interno como muestra el cuadro seleccionado de la Figura. 1. Es importante destacar que la entrada que utilizaremos en el diseño no es un video sino dos cuadros de un video con una resolución de 176x144 (formato QCIF) con el objetivo de centrarnos en el procesamiento de la señal que sale del restador denominada residual. Esta señal residual obtenida y con la que se trabajará en cada uno de los bloques implementados es producto de una codificación diferencial realizada entre estos dos cuadros, donde un cuadro se compara con otro de referencia y solo se codifican los pixeles que han cambiado respecto al cuadro de referencia. La codificación diferencial no reducirá los datos de forma significativa si hay mucho movimiento en el video, en estos casos pueden ser utilizadas otras técnicas de predicción las cuales serán abordadas en artículos posteriores.

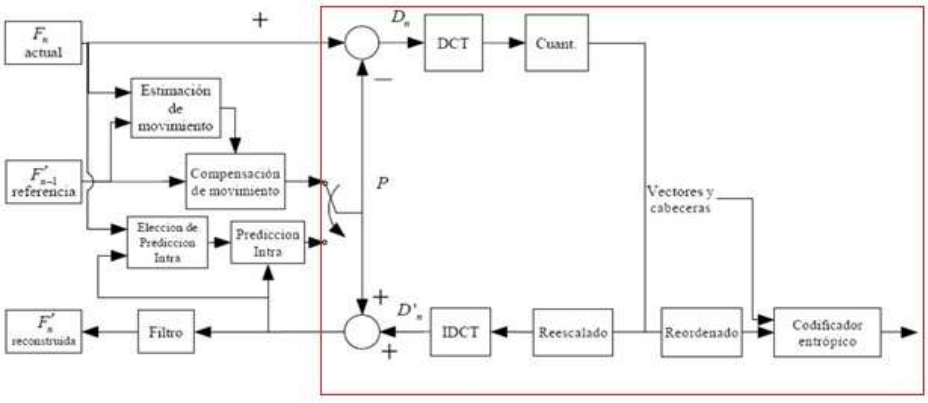

Figura. 1 Etapa seleccionada para el diseño del Codificador de Video H.264/AVC

3. PROPUESTA DE DISEÑO

2 H.262 (ISO/IEC 13818-2 o MPEG-2 Parte 2, también conocido como MPEG-2 Video).
3 ISO/IEC 14496-10 o MPEG-4 Parte 10, más conocido como H.264/AVC (Advanced Video Coding)
4 AVS1-P2 (Audio Video Coding Standard 1 Parte 2) es la parte de AVS que norma el proceso de decodificación de video y tiene como objetivo lograr una eficiencia de codificación similar a H.264 pero con una menor complejidad computacional.

El diseño se puede dividir en seis bloques fundamentales como muestra la Figura. 2, los bloques utilizados para conformar la señal residual, los bloques encargados de aplicar la transformada discreta del coseno y la cuantificación directa e inversa a la imagen y la etapa de conformación del flujo binario de salida mediante el uso del codificador adaptativo de longitud variable basado en el contexto (CAVLC). Se quiere destacar que H.264/AVC posee dos codificadores de entropía, también existe el codificador binario adaptable al contexto (CABAC) el cual brinda una mayor eficiencia que CAVLC pero su nivel de complejidad es mucho mayor.

Figura. 2 Diseño planteado

En la ventana principal del diseño en Simulink como se muestra en la Figura. 3 aparecen los bloques para la obtención de la señal residual, los bloques de procesamiento, la visualización y la obtención de las señales de entrada y salida de la etapa de procesamiento. Todos los elementos antes mencionados son justificados por el diagrama anterior excepto la conversión de los datos. Algunas de las preguntas que surgieron de esta problemática fueron ¿Por qué convertir la información residual? ¿Podría incidir el tipo de conversión en los resultados finales? Para dar respuesta a estas interrogantes se realizaron un grupo de pruebas arrojando las siguientes conclusiones: Los elementos que constituyen una imagen en Matlab tienen el formato entero uint8, que es un tipo de dato que puede variar de 0 a 255, sin poder soportar decimales y valores que salgan fuera de ese rango. Lo anterior resulta una desventaja principalmente en aquellos casos donde se implementan algoritmos que trabajan con este tipo de datos para realizar operaciones de división o multiplicación por tipo de dato flotante. En estos casos es necesario trasformar la imagen de tipo de dato uint8 a double.

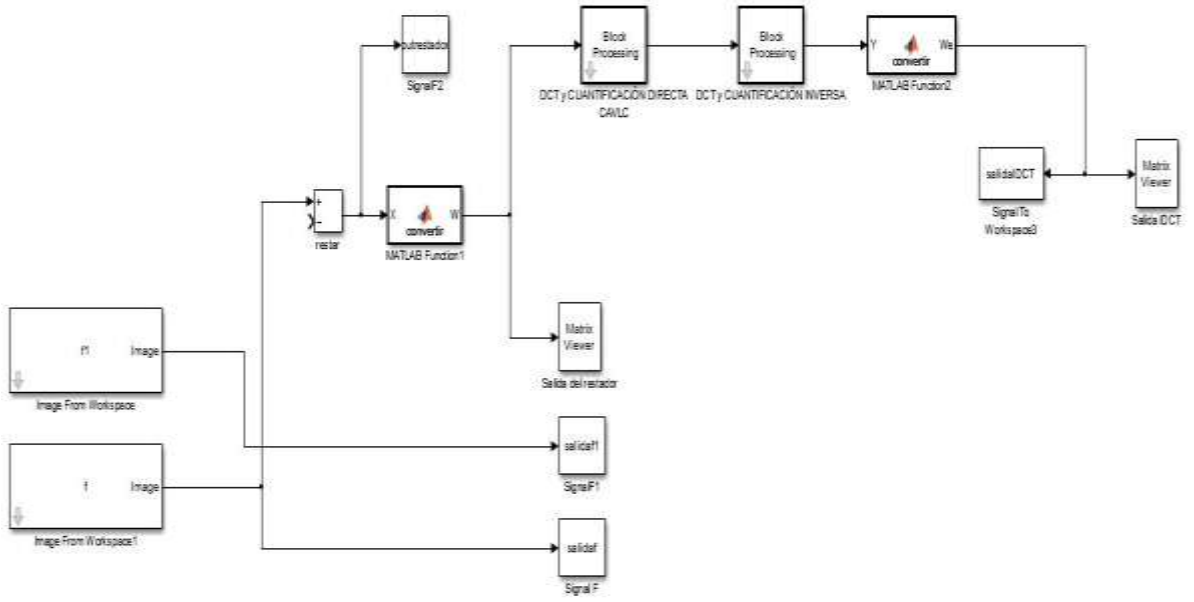

Figura. 3 Diseño General en Simulink

El diseño no se limitó a procesar información de luminancia solamente, sino que también se utilizó una imagen a color es decir con sus tres componentes (RGB), la cual fue convertida a formato YCbCr dividiéndola en la componente de Luminancia más dos componentes de color, para realizar el posterior procesamiento a cada una por separado, lográndose la recuperación de la imagen reconstruida a la salida del proceso para volver a convertir a formato RGB y obtener la imagen de entrada reconstruida. El esquema que corresponde a esta descripción se muestra en la Figura. 4.

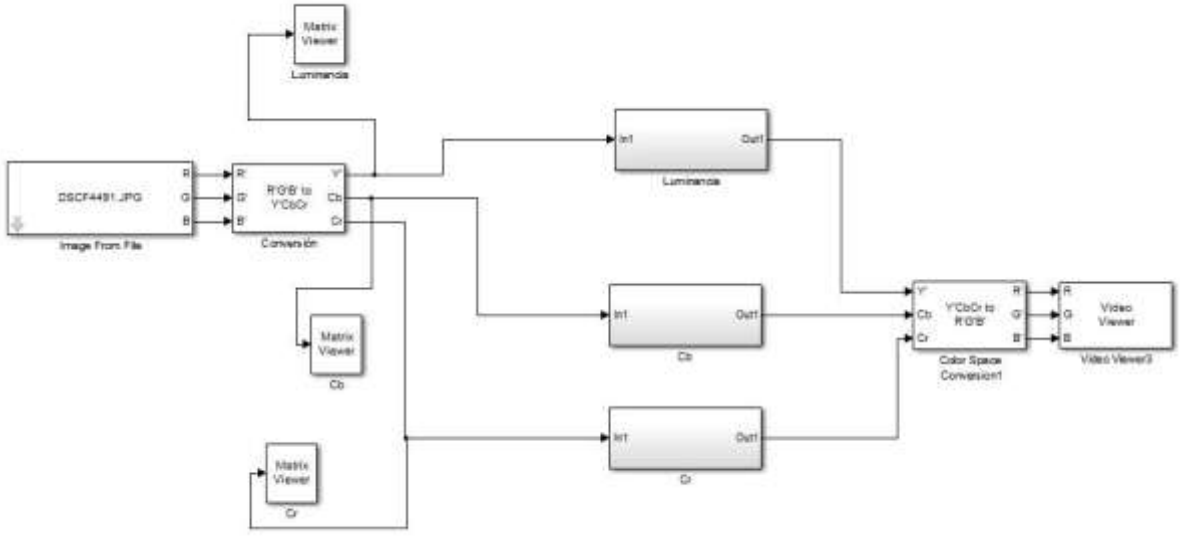

Figura. 4 Diseño para codificación de una imagen RGB

4. TRANSFORMADAS Y CUANTIFICACIÓN

La recomendación H.264/AVC define tres transformadas, dependiendo del tipo de datos residuales a codificar. Antes de entrar en el proceso de transformación la imagen es dividida en bloques de 16x16 los cuales son procesados en arreglos de tamaños 4x4 u 8x8, este último es utilizado solamente en altos perfiles. Es utilizada la Transformada de Hadamard para los coeficientes de directa (DC) de la luminancia de cada arreglo de 4x4 que han sido predichos en el modo INTRA en macrobloques (MB) de 16x16, para los arreglos de 2x2 de coeficientes DC de la crominancia también se utilizará la transformada Hadamard. Finalmente, para los demás bloques residuales de 4x4 se usará la transformada basada en la Transformada Discreta del Coseno (DCT).[2]

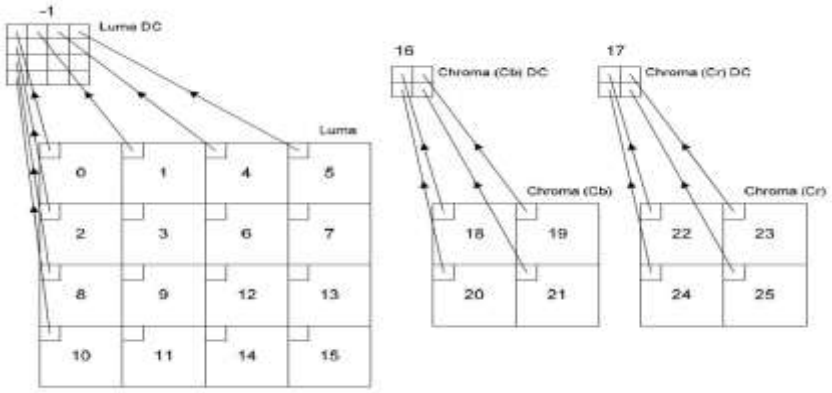

Figura. 5 Mapeo de la componente DC de un MB de 16x16

En la Figura. 5 se muestra el proceso de separado de la componente DC de la luminancia y la crominancia al ser aplicada la transformada Hadamard a los bloques "-1,16 y 17". Mientras que a los bloques del 0-15 y 18-25 se le aplicará la transformada basada en la DCT.

¿Por qué no emplear mejor la Transformada Discreta de Fourier (DFT)?

Además de una mejor compactación de energía, la DCT posee otra ventaja interesante respecto a la DFT. Cuando se lleva a cabo la transformada discreta de Fourier del bloque, se asume la periodicidad del mismo, por lo que se repite a lo largo de todo el plano bidimensional que contiene la imagen. En otras palabras la ventaja principal de la DCT frente a la DFT es que los pixeles de los bordes de la imagen original se convierten en adyacentes de sí mismos al realizar las reflexiones, eliminando los cambios abruptos que aparecían en la DFT y por tanto no aparecen nuevas componentes en frecuencias altas. A esto hay que añadir que la DCT tiene todas las ventajas de la DFT y siempre es real, lo cual la convierte en una buena opción como fase de transformación en un algoritmo de compresión.[3]

Empleo de la transformada discreta del coseno

El proceso directo de transformación y cuantificación entero para bloques de 4x4 es desarrollado comenzando desde la obtención de una DCT de 4x4, la cual es una aproximación entera redondeada de la misma DCT, adicionar un paso de normalización para mantener la propiedad de ortonormalidad a la DCT e integrar el paso de normalización con el proceso de cuantificación[4]. En el diseño planteado debido al último paso explicado anteriormente no se pudo utilizar los bloques 2D-DCT y 2D-IDCT que ofrece la biblioteca de Simulink ya que no cumplían con las especificaciones de H.264/AVC, por lo que para implementar cada proceso se utilizó el bloque Matlab Function también de la biblioteca de Simulink el cual permite introducir código generado desde Matlab para lograr las especificaciones requeridas. En la
Figura. **6** se muestran los bloques funcionales de la implementación.

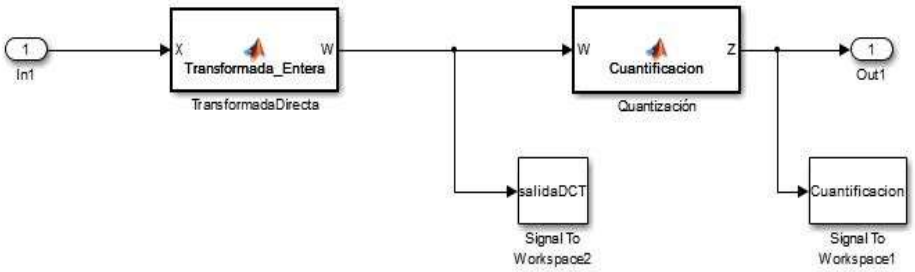

Figura. 6 Implementación de la DCT y Cuantificación

Empleo de la Transformada de Hadamard

La transformada de Hadamard está íntimamente relacionada con la transformada de Walsh, hasta tal punto que en muchas ocasiones encontramos un único tratamiento para ambas, al que se hace referencia como transformada de Walsh-Hadamard (WHT). Algunas de sus propiedades son: es real, simétrica y ortogonal. Es una transformada rápida ya que contiene únicamente valores +/-1, por lo que no se requieren multiplicaciones para su cálculo. De esta forma, el número de sumas y restas necesario puede reducirse desde N^2 hasta aproximadamente $N * \log_2 N$.
Como habíamos visto anteriormente cuando utilizamos el modo INTRA de 16x16 los coeficientes de DC para los bloques de luma y croma son transformados empleando Hadamard. La manera en que se hace es la siguiente. Para la transformación directa e inversa de los bloques de luma se utiliza matrices 4x4 (las mismas para el proceso directo e inverso) y para croma las matrices varían en dependencia del muestreo utilizado.[2]

$$Y = \left(\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \\ 1 & -1 & 1 & -1 \end{bmatrix} [X] \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \\ 1 & -1 & 1 & -1 \end{bmatrix} \right) /2 \quad (1)$$

Para el muestreo 4:2:0 hay cuatro bloques de cada coeficiente de croma por lo que los coeficientes de DC forman una matriz de 2x2. De la misma manera que en luma si multiplicamos por estas mismas matrices obtenemos la transformación inversa.

$$Y = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} [X] \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \quad (2)$$

Para el formato de video 4:2:2 hay ocho bloques de croma en cada componente de croma por lo que las matrices utilizadas son de 4x4 y 2x2.

$$Y = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \\ 1 & -1 & 1 & -1 \end{bmatrix} [X] \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \quad (3)$$

5. CODIFICACIÓN DE LONGITUD VARIABLE ADAPTIVA BASADA EN EL CONTEXTO (CAVLC)

Antes que los datos sean codificados en CAVLC se hace necesario ordenarlos; para ello existen dos métodos de reordenamiento utilizados por el estándar H.264\AVC el escaneo por campos y el escaneo progresivo o en zigzag que es el utilizado en el diseño, específicamente en bloques de 4x4 como se muestra en la Figura. 7.

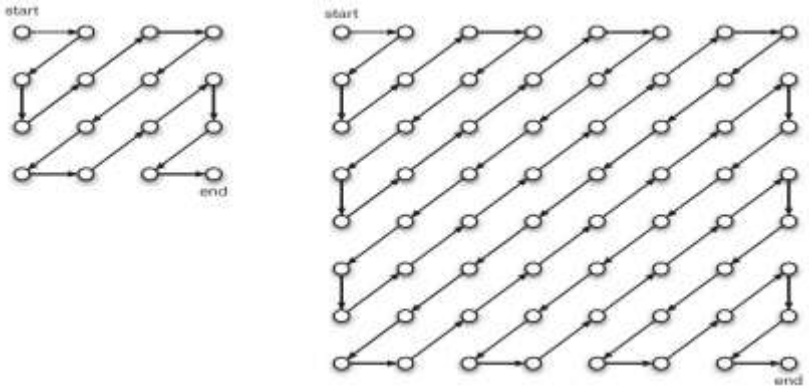

Figura. 7 Escaneo progresivo para bloques de 4x4 y 8x8

El escaneo o reordenamiento es utilizado para agrupar la mayor cantidad de coeficientes no nulos de manera que puedan ser separados de la mayoría de los coeficientes ceros existentes en cada bloque de 4x4 que ha sido transformado y cuantificado. En teoría se explica que cuando se emplea la transformada discreta del coseno se obtiene una concentración de la información en las bajas frecuencias agrupándose en el borde superior izquierdo de cada bloque la información relevante, no obstante pueden existir coeficientes aislados con valores considerables y hay que tenerlos en cuenta. En las simulaciones realizadas se pudo observar que muchos de estos bloques no poseían información en el coeficiente DC, es por eso que H.264/AVC tiene en cuenta este tipo de comportamiento.[5]

CAVLC ha sido diseñado para aprovechar las ventajas que brindan los bloques de 4x4 después de ser cuantificados. Después de la predicción, transformación y cuantificación los bloques poseen una gran cantidad de ceros por lo que CAVLC representa cadenas de ceros de manera más compacta. Además debido a que la mayor cantidad de coeficientes distintos a cero después del escaneo frecuentemente son ±1 este codificador marca a los coeficientes ±1 como Cadenas de uno de manera compacta. Otras de las características que explota CAVLC es referido a la correlación existente entre los bloques vecinos y la magnitud de cada coeficiente[6]. Para esto el codificador utiliza tres tablas de códigos de longitud variable y una de códigos de longitud fija que son utilizadas en dependencia de la información de los bloques anteriormente codificados. En la Figura. 8 se muestra el subsistema donde está implementado CAVLC. Con el objetivo de estudiar el comportamiento de los datos obtenidos se utilizaron bloques que muestran los resultados de salida en cada parte del proceso.

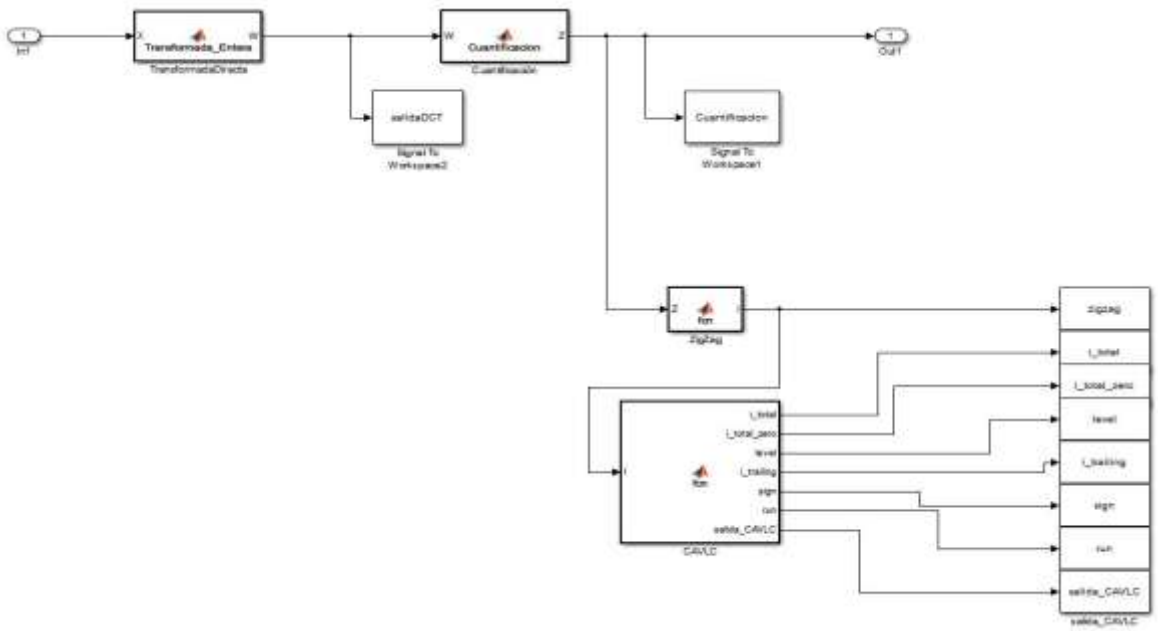

Figura. 8 Subsistema de implementación de CAVLC

6. RESULTADOS OBTENIDOS

Obtener en Simulink la señal residual y la señal de salida de la transformación y cuantificación indirecta.

Para realizar el análisis de los resultados obtenidos se parte del proceso de DCT y Cuantificación para dos cuadros de la secuencia de video analizada y de una imagen a color como se explica en apartados anteriores. La Figura. 9a es la que representa al cuadro 0 o como le llamamos imagen de referencia a partir de la cual se realiza la codificación diferencial entre esta y la Figura. 9b que representa al cuadro 1 que no es más que el cuadro actual o que se quiere codificar en la secuencia de video.

a) b)

Figura. 9 a) Imagen de referencia (Cuadro 0), b) Imagen siguiente en la secuencia de video

En la Figura. 10 se muestra la señal residual obtenida a la salida del restador. En esta imagen podemos obtener la visualización de la codificación diferencial antes mencionada.

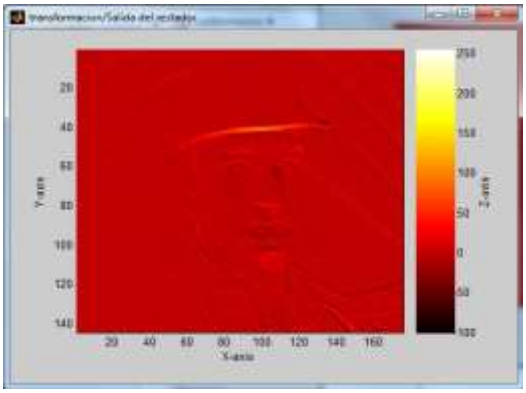

Figura. 10 Señal residual obtenida a la salida del restador

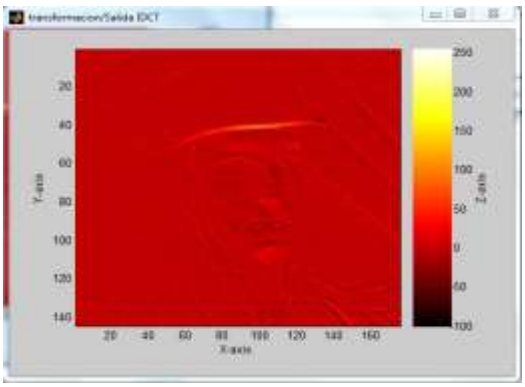

Figura. 11 Señal obtenida a la Salida de la IDCT

Una vez realizado todo el procesamiento, obtenemos la salida de la IDCT como se muestra en la Figura. 11 donde no existen cambios visibles con respecto a la imagen original. Al igual que sucede con las evaluaciones de los sistemas de compresión se hace necesario realizar un análisis cuantitativo también. En este caso se realizó una substracción de las dos imágenes (imagen inicial y recuperada) para poder obtener la diferencia real en cada posición de la misma. Al final la mayoría de los coeficientes obtenidos fueron cero y los demás tuvieron una variación de +/- 1 demostrándose que las diferencias fueron mínimas.

Obtener en Simulink la recuperación de un cuadro completo.

Para obtener las visualizaciones que se muestran a continuación utilizamos el cuadro de la Figura. 12, el cual fue convertido al formato YCbCr y procesada cada componente por separado.

Figura. 12 Cuadro de referencia de la imagen RGB

Como podemos observar en las Figura. 13 y Figura. 14 la información transmitida y recuperada presentan una gran similitud lográndose al final mostrar el cuadro recuperado en la Figura. 15.

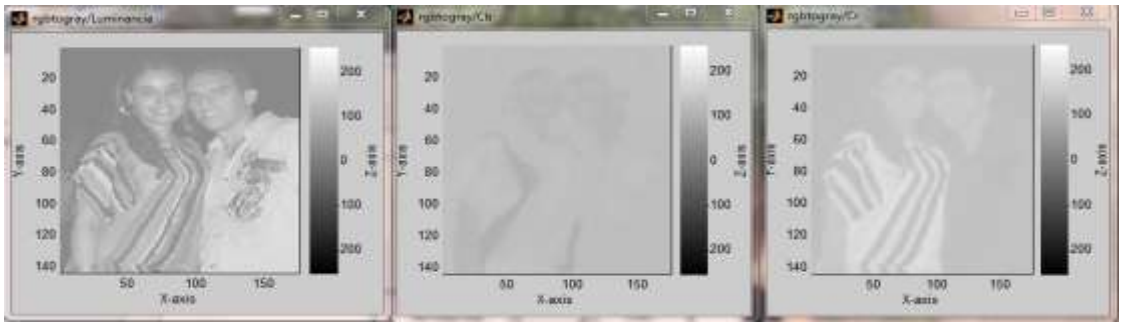

Figura. 13 Resultante de la conversión al formato YCbCr

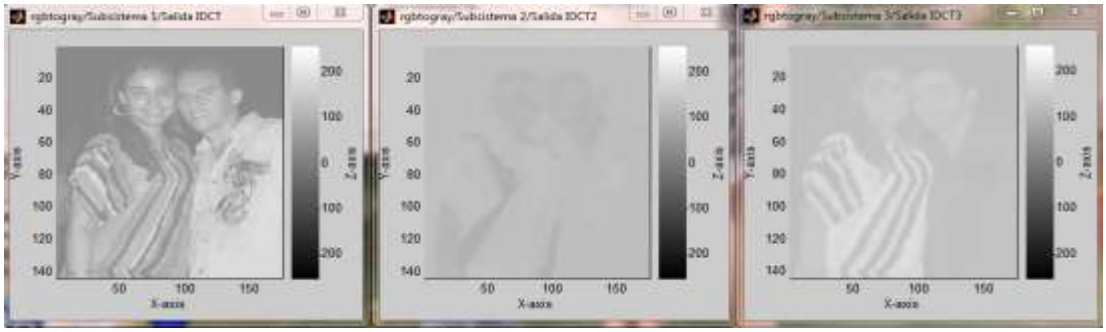

Figura. 14 Obtención de la IDCT (Imagen recuperada) para cada componente respectivamente

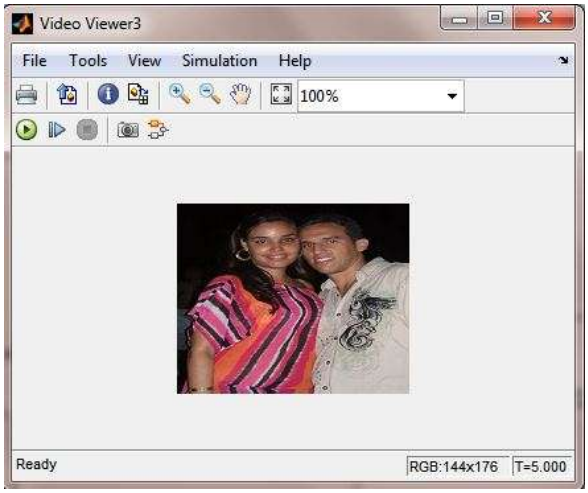

Figura. 15 Cuadro reconstruido de la imagen RGB

Recordemos que la transformación se realiza en bloques de 4x4 para el caso que estamos discutiendo por lo que la información que contiene mayor energía (en las bajas frecuencias) se concentrará en el borde superior izquierdo de cada uno de estos bloques. Decidimos evidenciar este comportamiento en este apéndice ya que la

información de un cuadro completo es mayor y gráficamente es más visible como podemos observar en la Figura. 16.

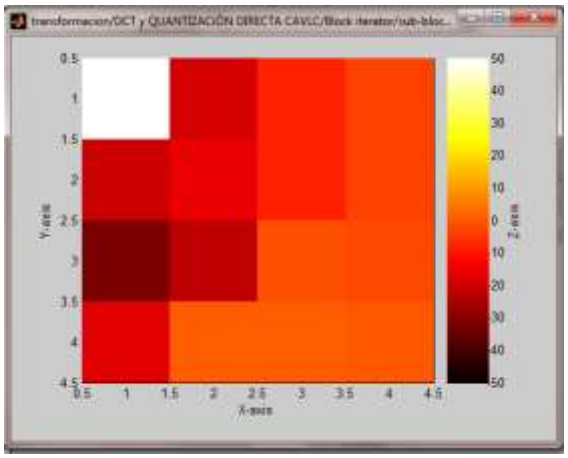

Figura. 16 Salida del proceso de DCT para un bloque de 4x4

A medida que cada bloque de 4x4 es procesado estos son enviados al codificador de entropía CAVLC. Una vez obtenido todos los datos necesarios para trabajar con las tablas de pasamos a la codificación de la información. En la Figura. 17 podemos ver la codificación de los primeros 21 bloques pero en realidad son procesados 1584 debido a la división de la imagen (176x144) en bloques de (4x4). Después de realizada la codificación final pasamos a calcular la razón de compresión obtenida con el codificador de entropía para la señal residual utilizada, con el objetivo de demostrar las potencialidades que ofrece CAVLC principalmente en perfiles bajos donde la información que se transmite no necesita usar codificación aritmética que conllevaría a un nivel mayor de complejidad. La razón de compresión calculada se basó en dividir la cantidad de bits obtenidos luego de codificar con CAVLC y los bits totales si no codificáramos la información. El resultado final fue de 202752 bit sin codificar y 54417 bits con codificación para una razón de 3.7258 lo que demuestra lo antes planteado.

SalidaCAVLC

SalidaCAVLC <1584x1 cell>

	1	2
1	'0000100011100101111101101'	
2	'0000000110100010010000010111001100'	
3	'0001110001110010'	
4	'0000000011001011001001100111101011010'	
5	'0000000101010110011101100110001010011101011'	
6	'0000010001011001111101010101001110101'	
7	'000100010110011111010010011101'	
8	'00011100011100101001'	
9	'0000100010110011111011011011110010'	
10	'0000000011001011001111101101001001101010100000001'	
11	'0000011111010001'	
12	'0010101100111110110101010001'	
13	'000001111111011'	
14	'000001111101011'	
15	'0010101100111110110101011100'	
16	'001010110011111011010101100111'	
17	'0000000101010110011111011010101100000011010011110010'	
18	'000001000101100111111011010101100000000110111110010'	
19	'00010111'	
20	'00010111'	
21	'0000100010110011111011010101100000000111111011000110'	

Figura. 17 Codificación final de los primeros 21 bloques obtenidos

7. VALIDACIÓN DEL DISEÑO

En la validación de cada etapa del diseño se utilizaron varios ejemplos encontrados en la bibliografía consultada. El ejemplo de referencia mostrado a continuación fue extraído de una de las publicaciones del autor Iain E. Richardson[5], "The H.264 Advanced Video Compression Standard".
Para verificar el desempeño del algoritmo de CAVLC se introdujeron los códigos en el editor de Matlab donde se codificó por separado dicho ejemplo. Los coeficientes del bloque de referencia son mostrados a continuación, este simula uno de los arreglos de 4x4 obtenidos a la salida de la Cuantificación:

$$B_1 = \begin{bmatrix} 0 & 3 & -1 & 0 \\ 0 & -1 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

La codificación de referencia para el ejemplo es mostrada en la Figura. 18. A la derecha de esta figura podemos apreciar el flujo binario resultante, la salida final no es más que la concatenación de cada flujo independiente. A la izquierda aparecen las variables utilizadas en la codificación. Es importante señalar que los nombres de las variables obtenidas en Matlab no coinciden con las del ejemplo, los equivalentes de TotalCoeffs, T1s, TotalZeros, ZerosLeft serán en este caso i_total, i_trailing, i_total_zero e i_zero_left respectivamente. La información de signo es guardada en una sola variable denominada sign.
Para B_1 los coeficientes reordenados son: [0, 3, 0, 1,-1,-1, 0, 1,0,…]
Los datos resultantes de la codificación en Matlab son mostrados en la Figura. 19.

Element	Value	Code
coeff_token	TotalCoeffs = 5, T1s = 3 (use Num_VLC0)	0000100
T1 sign (4)	+	0
T1 sign (3)	−	1
T1 sign (2)	−	1
Level (1)	+1 (level_prefix = 1; suffixLength = 0)	1
Level (0)	+3 (level_prefix = 001, suffixLength = 1)	0010
TotalZeros	3	111
run_before(4)	ZerosLeft = 3; run_before = 1	10
run_before(3)	ZerosLeft = 2; run_before = 0	1
run_before(2)	ZerosLeft = 2; run_before = 0	1
run_before(1)	ZerosLeft = 2; run_before = 1	01
run_before(0)	ZerosLeft = 1; run_before = 1	No code required; last coefficient.

Figura. 18 Resultado de la codificación en el ejemplo de referencia B_1

La codificación final sería: 000010001110010111101101. Como podemos observar coincide con la cadena binaria obtenida.

Realizando el mismo procedimiento trabajamos con los otros ejemplos de referencia. Para el caso de la DCT y la cuantificación se contó con los resultados esperados en cada parte del proceso por lo que se comprobó en base a la variación que sufrían los coeficientes en cada etapa. Se pudo observar que muy pocos coeficientes sufrieron variaciones, oscilando en un rango de +/- 1.

[5] Dicho autor es considerado internacionalmente un experto en compresión de video. Director de Vcodex Ltd OneCodec Ltd, además de ser profesor honorario en RGU (Universidad Robert Gordon), Aberdeen, Inglaterra. Entre sus otras publicaciones podemos encontrar ''H.264 y MPEG-4 Compresión de Video: Codificación de video para la nueva generación de multimedia'' y ''Diseño del Códec de video: Desarrollando Sistemas de Compresión de video e imágenes''.

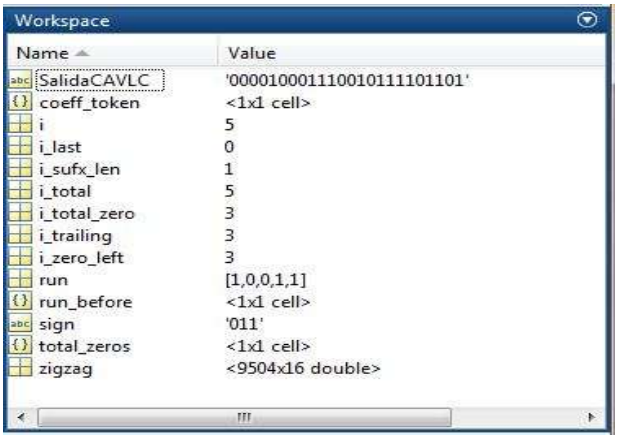

Figura. 19 Equivalente obtenido en Matlab para el ejemplo de referencia B_1

8. CONCLUSIONES

La implementación de estos algoritmos en Simulink es un primer paso a la continuidad de esquemas más avanzados que permitan la compresión de la señal de video acorde a las características que impone H.264/AVC. Los sistemas de compresión son creados por grupos de expertos en codificación con años de experiencia por lo que para poder avanzar en una materia tan compleja se necesita dividir lo más posibles los diseños a realizar de cada bloque, de manera que podamos obtener en cada uno la experiencia necesaria para conformar nuestro propio sistema.

.

RECONOCIMIENTOS

"Los autores desean agradecer al colectivo de investigadores de ***LACETEL*** en especial al Ing. Rufino Reydel Cabrera Álvarez, por la ayuda brindada en la realización de esta investigación."

REFERENCIAS

1. RUIZ, F.T., *SISTEMAS AUDIOVISUALES. TELEVISIÓN ANALÓGICA Y DIGITAL*, ED. EDICIONS UPC. VOL. II. 2000.
2. RICHARDSON, I.E., *THE H.264 ADVANCED VIDEO COMPRESSION STANDARD*. 2 ED. 2010: A JOHN WILEY AND SONS, LTD., PUBLICATION. 348.
3. ORTEGA, D.A., *COMPRESIÓN DE IMÁGENES OPTIMIZADA EN CONSUMO ENERGÉTICO PARA REDES INALÁMBRICAS*. 2013: P. 91.
4. WIEGAND, T., ET AL., *OVERVIEW OF THE H.264/AVC VIDEO CODING STANDARD*. IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY, 2003. 13(7): P. 560-576.
5. KESHAVENI, N., S. RAMACHANDRAN, AND K.S. GURUMURTHY, *IMPLEMENTATION OF CONTEXT ADAPTIVE VARIABLE LENGTH CODER FOR H.264 VIDEO ENCODER*. INTERNATIONAL JOURNAL OF RECENT TRENDS IN ENGINEERING, 2009. 2(5): P. 341-345.
6. TIAN, X., T.M. LE, AND Y. LIAN, *ENTROPY CODERS OF THE H.264/AVC STANDARD: ALGORITHMS AND VLSI ARCHITECTURES*. 1 ED. SIGNALS AND COMMUNICATIONS TECHNOLOGY. 2011: SPRINGER. 205.

SOBRE LOS AUTORES

José Angel Espinosa Roque: Ing. Telecomunicaciones y Electrónica, graduado del Instituto Superior Politécnico José Antonio Echeverría.

Dalila Garrido Mirabal: Ing. Telecomunicaciones y Electrónica, graduada del Instituto Superior Politécnico José Antonio Echeverría.