TELEMÁTICA
REVISTA DIGITAL de las Tecnologías de la Información y las Telecomunicaciones
Revista Telemática. Vol. 17. No. 3, septiembre-diciembre, 2018, p.11- 24 ISSN 1729-3804

DISEÑO E IMPLEMENTACIÓN DE UN DECODIFICADOR CABAC SOBRE UN SISTEMA EMBEBIDO

Rufino R. Cabrera Alvarez[1], Osmany Yaunner Núñez[2], Laura Quesada del Busto[3], Gustavo Aguirre Soler[4], Yosmany Hernández Sánchez[5], Glauco Guillén Nieto[6]

[1-4]***LACETEL***, Instituto de Investigación y Desarrollo de Telecomunicaciones, Ave. Independencia No.34515, Km 14½, Reparto 1ro de Mayo, Municipio Boyeros, La Habana, Cuba.
[1]e-mail: rufino@lacetel.cu
[2]e-mail: osmany@lacetel.cu
[3]e-mail: laura@lacetel.cu
[4]e-mail: g.aguirre@lacetel.cu
[4]e-mail: yosmany@lacetel.cu
[4]e-mail: glauco@lacetel.cu

RESUMEN

Este trabajo describe el diseño por bloques del Módulo IP de un decodificador de entropía CABAC (Codificación Aritmética Binaria Adaptada al Contexto). El diseño se basa en el código de referencia JM19.0 haciendo uso del estándar H.264/AVC. El código se ejecuta sobre el procesador PowerPC440 del FPGA Virtex-5 de una tarjeta de desarrollo ML507. La implementación del módulo se realiza en VHDL. Los resultados propuestos fueron elaborados usando las herramientas de diseño ISE Design Suite y Microsoft Visual Studio. Estos resultados fueron obtenidos a partir de videos codificados H.264 con el codificador de referencia JM19.0. Con la inserción de este módulo en el sistema, se logró disminuir en un 18% las demoras de procesamiento del decodificador.

PALABRAS CLAVES: CABAC, H.264/AVC, Módulo IP, Sistemas embebidos, FPGA.

ABSTRACT

This work describes the design of an Intellectual Property module of a CABAC entropy decoder. The design is based on the H.264/AVC reference code JM 19.0 and it is implemented on the PowerPC440 processor with Virtex-5 FPGA technology in a ML507 board. The proposed results were obtained by using the design tools ISE Design Suite and Microsoft Visual Studio. Video frame used to evaluate were coded with the JM19.0 encoder with a variety of characteristics. With the insertion of this module into the system it was possible to reduce the processing delay of the decoder by 18%.

KEY WORDS: CABAC, H.264/AVC, IP module, Embedded system, FPGA.

1. INTRODUCCIÓN

Hoy día Cuba se encuentra inmersa en el proceso de despliegue de la Televisión Digital (TVD) y es LACETEL, Instituto de Investigación y Desarrollo de Telecomunicaciones, la entidad responsable de la asimilación y transferencia de esta tecnología en el territorio nacional. Como parte de este proceso se desarrolla un proyecto para asimilar la etapa de compresión de video a partir del diseño de un decodificador bajo uno de los estándares correspondientes a esta tecnología. Existen múltiples estándares internacionales que regulan la codificación/decodificación de video, pero en televisión digital destacan solo tres [1]. Los más importantes de acuerdo al alcance y a su utilización son publicados y reconocidos por organizaciones especializadas, tales como la Unión Internacional de Telecomunicaciones (UIT), Organización Internacional para la Estandarización (ISO), el Grupo de Experto en Imágenes en Movimiento (MPEG) y el Grupo de Trabajo del Estándar (chino) de Audio y Video (AVS) [1]. Estas grandes organizaciones reguladoras son los

desarrolladores de los que se conocen, hasta 2017, como los tres principales estándares de codificación/decodificación de video usados en la TVD: MPEG-2, H.264/AVC y AVS. Los tres definen la estructura y características de un sistema de este tipo. Por su eficiencia y madurez en la codificación es H.264/AVC quien más destaca en popularidad y utilización, el cual está siendo utilizado en Cuba en la actualidad. Es esta la principal razón por la que fue seleccionado para dar inicio al estudio de estos decodificadores de video [1].

El grupo responsable del desarrollo y actualización de H.264/AVC publica la última versión de un software en lenguaje C, que representa una implementación detallada del codificador y decodificador del estándar. Este se conoce como Modelo Conjunto (JM). A partir del trabajo realizado por Landrove [1] se modifica el software de referencia JM, versión 18.4, para ser insertado en el procesador PowerPC440 del FPGA Virtex-5 de una tarjeta de desarrollo ML507.

Tomando como base el trabajo desarrollado en [1], se insertó la versión 19.0 en el procesador de esta tarjeta. Para determinar las demoras de decodificación, se utilizó un video de 1s de duración codificado con distintas resoluciones como muestra la Tabla 1. Utilizando el módulo de propiedad intelectual temporizador/contador de la tarjeta ML507 se obtuvieron los tiempos de procesamiento, los que exceden entre 15 y 453 veces su tiempo de duración, lo cual no permite que sean visualizados en tiempo real.

Para que logren visualizarse correctamente, cada uno de sus campos o cuadros debe ser decodificado antes de su tiempo de presentación en pantalla. Desde la presentación de la primera imagen hasta la visualización de la última, el tiempo de decodificación no debe exceder al de duración.

Tabla 1: Tiempo de decodificación del diseño inicial para 4 videos H.264/AVC de 1s.

Videos	Resolución (píxeles)	Decodificación (s)
Video 1	176x144	15.82
Video 2	352x288	49.52
Video 3	720x480	186.82
Video 4	1280x720	453.22

Generalmente, los decodificadores de tiempo real son implementados en hardware [1-8]. Para decidir realizar la implementación en hardware de este decodificador es necesario tener en cuenta la interdependencia de cada una de sus variables. Este análisis debe realizarse de forma que puedan ser paralelizadas cada una de las variables.

2. PUNTO DE PARTIDA

El presente trabajo comenzó partiendo del diseño planteado por Landrove en [1]. La Figura 1 muestra el diagrama en bloques de los componentes de la tarjeta de desarrollo ML507 utilizados en este diseño. El software de referencia JM (en lenguaje C) fue modificado para ser ejecutado sobre el procesador PowerPC440, de forma que pueda utilizar los componentes de hardware de la tarjeta que son mostrados en la Figura 1.

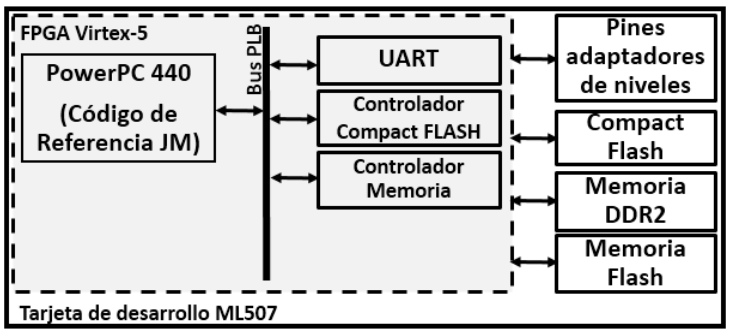


Figura 1: Diagrama en bloques del diseño tomado como punto de partida. [1]

Del diseño para FPGA Landrove hace uso del procesador PowerPC440 donde se ejecuta el software de referencia modificado y de los controladores de periféricos (componentes de la tarjeta de desarrollo ML507) siguientes:

- Controlador del puerto serie UART, utilizado para reportar los resultados de la decodificación del video. La comunicación con una PC, se realiza a través de los pines adaptadores de niveles de la tarjeta.
- Componente y controlador de memoria externa, para la ejecución de la aplicación, ya que la memoria interna no es suficiente para esta aplicación. Tanto para los datos (DDR) como para el programa (memoria Flash)
- El bus PLB v46, para la comunicación con los periféricos del sistema de procesamiento embebido.
- Controlador de memoria Compact Flash, para el almacenamiento del video codificado H.264/AVC a procesar y el almacenamiento del video decodificado resultante.

Haciendo uso de este diseño, se agruparon las distintas etapas del decodificador H.264/AVC contenido en el software de referencia. Utilizando el módulo temporizador de la tarjeta se realizó un análisis para determinar cuál es el primer bloque a implementar. Nótese en la Tabla 2, que el mayor tiempo de decodificación corresponde a la predicción, pero la salida de esta es procesada al mismo tiempo que la de transformación y cuantificación. De esta forma, si se paraleliza la predicción solo se puede reducir las demoras de procesamiento en un 7% (26% - 19%). Para lograr una mayor reducción de la demora de procesamiento también es necesario implementar en hardware la transformación y cuantificación. Realizar esto implica modificar 43 funciones software.

Tabla 2**:** Valores del decodificador H.264/AVC por bloques.

Funciones correspondientes a los bloques del decodificador H.264/AVC	**Tiempo (s)**	**%**	**Funciones Softwares**	**Implementadas en HW**
Decodificación de entropía	2,86	24%	15	1
Cuantificación y transformación inversa	2,26	19%	20	0
Predicción	3,21	26%	23	0
Filtro de desbloqueo	2,73	23%	37	0
Otros	0,93	8%	16	0
Total	11,96	100%	111	1

Las siguientes etapas críticas en cuanto a tiempo son la decodificación de entropía (24%) y el filtro de desbloqueo (23%). Implementar la primera, implica estudiar y diseñar 14 funciones (15 menos una implementada en trabajo previo), mientras el filtro de desbloqueo presenta 37 funciones a modificar. Debido a que el esfuerzo dedicado para reducir la demora de procesamiento es menor diseñando la decodificación de entropía esta fue la etapa seleccionada.

Un codificador/decodificador de video del estándar H.264 utiliza tres decodificadores de entropía, CABAC (Codificación Aritmética Binaria Adaptada al Contexto), CAVLC (Codificación Adaptativa de Códigos de Longitud Variable) y Exponencial Golomb. En Exponencial Golomb, los códigos más cortos se asignan a símbolos o coeficientes con mayor frecuencia de ocurrencia y es utilizado para codificar determinados elementos de cabecera en la trama H.264/AVC que es necesario transmitir periódicamente. CAVLC consiste en codificar los coeficientes transformados a partir de información estadística de datos recientemente procesados, emplea métodos de adaptación de contexto y es utilizado para la codificación de los arreglos de coeficientes con perfil básico. CABAC es presentado por H.264/AVC como una poderosa herramienta de codificación y es utilizado para los arreglos de coeficientes con perfiles superiores como los utilizados en los esquemas de televisión digital, por lo que es la opción seleccionada. [9]

En [5, 10-11] se explica que CABAC es un método de codificación/decodificación de entropía en la que intervienen tres sub-etapas, la Decodificación Aritmética Binaria (DAB), De-Binarización y el Modelado de Contexto. Estos tres bloques toman cada uno de los elementos de una matriz de 4x4 píxeles, correspondientes a un área (*slice*) de una imagen cualquiera en un video digital y lo codifica en cadenas de bins de longitudes variables, donde cada bin representa un bit binarizado.

La Codificación Aritmética Binaria Adaptada al Contexto se decodifica con las 15 funciones mencionadas con anterioridad, resumidas en la Tabla 3. Al inicio de la trama H.264/AVC se ejecuta la función número uno (arideco_start_decoder), la cual inicializa las variables a actualizar en la decodificación aritmética binaria utilizando las funciones dos (getword) y tres (getbyte). Además, la función cuatro (biari_init_ctx) establece el valor inicial del estado de probabilidad, haciendo un uso de las funciones cinco (imin) y seis (imax). Al inicio de cada *slice*, se ejecuta la función siete (init_slice), la cual inicializa entre otros parámetros el índice de contexto y el tipo de De-binarizador a utilizar. A partir de este momento el De-binarizador comienza a solicitar uno a uno los bins a los Decodificadores Aritméticos Binarios (DAB) correspondientes, hasta lograr generar el símbolo de salida, repitiendo este proceso hasta la finalización de un *slice*. Esto quiere decir que dentro de un mismo *slice* solo se ejecuta un mismo De-binarizador y por consecuencia un mismo DAB.

Tabla 3: Resumen de las funciones del JM 19.0

No.	Nombre de la función	Descripción
1	arideco_start_decoder	Inicializa la trama H.264/AVC con los primeros bytes a procesar.
2	getword	Llama a la función *getbyte* dos veces y ordena los valores.
3	getbyte	Accede al siguiente byte de la trama H.264/AVC a procesar.
4	biari_init_ctx	Inicializa al DAB con los valores del estado de probabilidad.
5	imin	Determina el valor mínimo entre 126 y un valor de entrada.
6	imax	Determina el valor máximo entre 64 y un valor de entrada.
7	init_slice	Inicializa todos los parámetros de los De-binarizadores al inicio de cada s*lice.*
8	int_idxctxinc	Calcula el índice de contexto incrementado a partir de una tabla.
9	unari_bin_decode	De-binarizador. Devuelve el símbolo de salida por el método unario.
10	unari_bin_max_decode	De-binarizador. Devuelve el símbolo de salida por el método unario truncado.
11	exp_golomb_decode_eq_prob	De-binarizador. Devuelve el símbolo de salida por el método exponencial golomb de orden k.
12	unari_ exp_golomb_level_decode	De-binarizador. Devuelve el símbolo de salida por el método de longitud fija.
13	biari_decode_symbol	DAB. Devuelve el bin de salida por el método regular a solicitud del De-binarizador unario o unario truncado.
14	biari_decode_symbol_eq_prob	DAB. Devuelve el bin de salida por el método regular a solicitud del De-binarizador exponencial golomb de orden k.
15	biari_decode_final	DAB. Devuelve el bin de salida por el método regular a solicitud del De-binarizador de longitud fija.

Según [2,3], existen diferentes aspectos a tener en cuenta para decidir la implementación en hardware o software de una determinada funcionalidad. Aplicando esto a las 15 funciones softwares que presenta el decodificador CABAC, las siete funciones correspondientes a la Decodificación Aritmética Binaria y la De-binarización fueron implementadas en hardware. Esto es debido a que las variables que se actualizan en cada uno de ellos son independientes entre sí y pueden calcularse en paralelo con operaciones algebraicas. Las restantes ocho funciones son las encargadas de inicializar correctamente al decodificador CABAC.

Aunque no se haya mencionado, el Modelado de Contexto sí se realiza en este software, pero su funcionalidad está implementada en las propias líneas de código de inicialización, en el Decodificador Aritmético Binario y el De-binarizador. Es decir, la funcionalidad del Modelador de Contexto fue dividida en estos tres grupos de funciones. Este aporte fue modificado en el software de referencia a partir de que Marpe, Chen, Yu y Nunes-Yanes combinan estas soluciones en [12-15]. Los resultados de estos trabajos mejoran entre un 25% y 32% la

velocidad de procesamiento de CABAC. Se simplifica la concepción del diseño, aun cuando son complejizados los diseños de algunos componentes del DAB y del De-binarizador. Por esta razón este aporte fue utilizado en el diseño presentado durante este trabajo.

De acuerdo con [2,3], otro de los elementos a tener en cuenta es que, si una tarea interactúa estrechamente con el sistema operativo o si una determinada función no es crítica en cuanto a tiempo, es factible mantenerla implementada en software. Teniendo en cuenta el segundo criterio, se reafirma la decisión de implementar en hardware los bloques del DAB y del De-binarizador. Sin embargo, analizando ambas, implicaría que las funciones de inicialización al ejecutarse una sola vez durante todo el proceso de decodificación de un video o al inicio de cada *slice*, no son críticas en cuanto a tiempo. De las funciones de inicialización existen dos (getword y getbyte) que están estrechamente relacionadas con el acceso a memoria a partir de controladores de alto nivel proporcionados por Xilinx. Estas dos funciones son las encargadas de solicitar los próximos bytes de la trama que deben ser procesados, por lo que fueron mantenidas en software. El resto de las funciones de inicialización fueron implementadas en hardware para reducir la demora en el procesamiento y minimizar la cantidad de bits de los registros que son creados para controlar el modulo IP diseñado, lo cual se explica en la siguiente sección.

En resumen, las funciones 13, 14 y 15 fueron diseñadas en hardware en un bloque denominado "*Decodificados Aritmético Binario*" o *DAB*. Las funciones 9, 10, 11 y 12 se implementaron en VHDL en el bloque "*De-binarizador*", las funciones 1, 4, 5 y 6 se llevaron a hardware agrupadas como "*Inicializador DAB*" pero necesitan de una función que interactúe con las funciones 2 y 3 (que permanecieron en software), ordene los valores en registros y los escriba en el bus PLB. Las funciones 7 y 8 se implementaron en VHDL bajo el nombre "*Inicialización de slice HW*". Estas funciones necesitan interactuar con valores de la trama H.264/AVC (descripción del primer bloque de cada slice), por lo que se creó una función "*Inicialización de slice SW*" que busca los valores, los ordena en un registro y los escribe en el bus PLB.

La funcionalidad de la "*Inicialización de slice HW*" utiliza el acceso a cinco tablas para poder realizar el cálculo de los parámetros de entrada al "*De-binarizador*", sin embargo, en cuatro de ellas se puede realizar la búsqueda al mismo tiempo. Para ello, en [15], Yu presenta una solución para reordenar todas las descripciones de bloque necesarias en la inicialización de *slice*. En este algoritmo se simplifican cuatro de las tablas definidas en el estándar y se reduce a una sola, en la cual salen en paralelo los bits necesarios para cada uno de los parámetros que habilitan al "*De-binarizador*". Además, simplifica la quinta, lo que, combinado con lo anterior, mejora la eficiencia de decodificación en un 5%, por lo que esta variante fue utilizada en este trabajo.

Las arquitecturas basadas en predicciones se han propuesto para lograr un alto rendimiento [16]. Algunos métodos como la predicción de descripciones de bloques, los circuitos redundantes y las técnicas de reenvío, pueden adoptarse para anular los tiempos de procesamiento con las tablas, pero no están lo suficientemente documentados para realizar una implementación. Por esta razón se decidió no incluir esta propuesta en el diseño, ya que una inadecuada predicción, desencadena una serie de retardos para corregirlos, lo que genera una serie de retardos en la decodificación.

3. DISEÑO E IMPLEMENTACIÓN DEL DECODIFICADOR

La implementación del decodificador H.264/AVC presenta como núcleo el procesador PowerPC440 embebido en el FPGA Virtex-5 de la tarjeta, sobre el cual se ejecuta el software decodificador de video del código de referencia JM 19.0. Con la ejecución de este software en lenguaje C, el procesador controla los periféricos insertados por Landrove en [1], que son listados en la sección 2.

Como parte de este trabajo se insertó un nuevo módulo de propiedad intelectual no provisto por Xilinx: el decodificador CABAC diseñado. Como muestra la Figura 2, los cambios respecto al diseño utilizado como punto de partida (Figura 1) consistieron en la inserción del controlador de interrupciones y el módulo IP

diseñado. En la siguiente sección se explicará la conexión entre PowerPC440 y los dos módulos insertados (Figura 4).

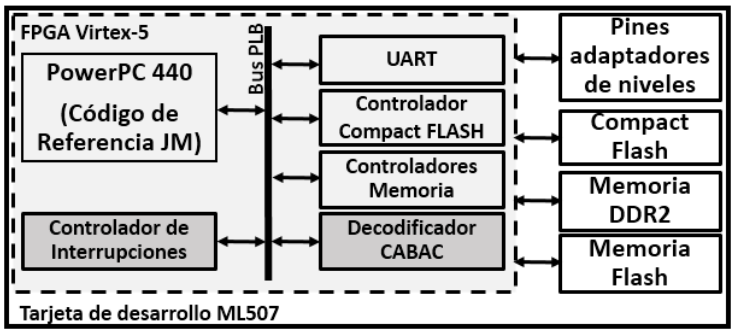


Figura 1: Diagrama en bloques del diseño realizado. [1]

Diseño del Módulo IP Decodificador CABAC

El proceso de diseño consistió en dividir la entidad de mayor jerarquía en tres bloques (Inicialización, De-binarizador y Decodificador Aritmético Binario), como se explica anteriormente a partir de la Tabla 3. Inicialmente se ejecutó paso a paso en *Microsoft Visual Studio 2008* el código de los bloques de interés. Luego se diseñaron circuitos con componentes hardware que implementan estas funcionalidades. Por ejemplo, la Figura 3 representa cómo una sentencia "if" se implementó en hardware mediante comparadores y multiplexores. Además, los ciclos "for y do-while" fueron implementados como máquinas de estados en dependencia de la funcionalidad.

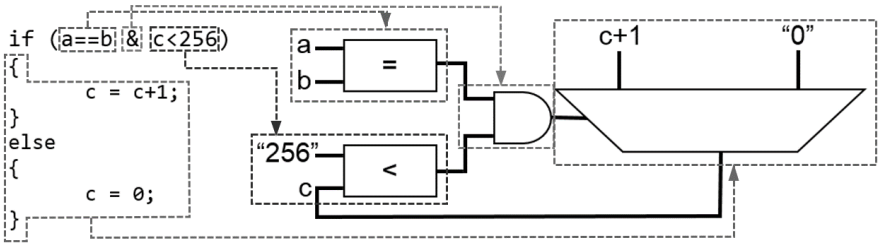


Figura 3: Interpretación a VHDL de instrucciones en lenguaje C

Existen herramientas de síntesis que realizan la descripción en hardware de un código en lenguaje de alto nivel, las cuales pudieran ser empleadas para simplificar el trabajo anterior. No obstante, se decidió obtener una descripción en hardware propia que aunque pueda resultar ineficiente, es un resultado funcional. Lo anterior permite un correcto entendimiento de la teoría estudiada, que a pesar de explicarse en numerosas bibliografías, no es tan completa cómo el propio código de referencia emitido por los desarrolladores del estándar.

La Figura 4 muestra, enmarcado con líneas discontinuas, el diagrama en bloques del módulo IP diseñado, donde se le insertaron señales de control, de forma que pueda ser controlado por el bus PLB. El "*reset*" (Rst), es el encargado de establecer condiciones iniciales al encender o reiniciar la tarjeta ML507. Cuando está activo cada una de las señales de salida toma el valor cero, las habilitaciones o señales de selección de los bloques se desactivan, "*rango*" toma valor 510 y "*contador*" valor 16. Estas dos últimas, son internas al Decodificador Aritmético Binario. El "*reloj*", es encargado de mantener el sincronismo del módulo, pero no es entrada de todos los componentes del diseño.

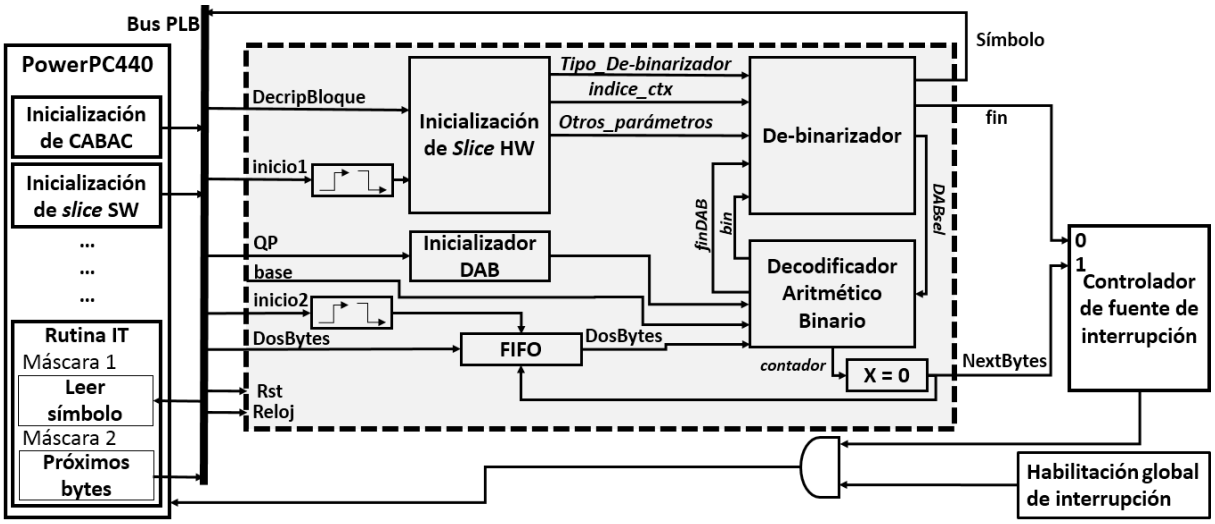


Figura 4: Diagrama en bloques del diseño del módulo IP Decodificador CABAC y su comunicación con el procesador

El funcionamiento general del Módulo IP consiste en los siguientes pasos:

- Cuando es invocada la función "Inicialización de CABAC" por el procesador, se llama a las funciones "getword" y "getbyte" para inicializar al bloque "Decodificador Aritmético Binario" con los primeros cinco bytes a procesar. Estos bytes se escriben en el bus PLB con el valor de parámetro de cuantificación ("QP").
- En este instante el bloque hardware "Inicializador DAB" calcula el estado de probabilidad y el valor del símbolo más probable e inicializa al "Decodificador Aritmético Binario".
- Al inicio de cada slice, el "Inicializador de slice SW" escribe en el bus PLB los valores de la descripción del primer bloque ("DecripBloque") y una señal de inicio ("inicio1") para poder detectar estos valores.
- En este momento el "Inicializador de slice HW" determina las salidas ("Tipo De-binarizador", "índice_ctx" y "otros_parámetros") y habilita el "De-binarizador" correspondiente con la señal "Tipo De-binarizador".
- Activado este, comienza a solicitar bins al "Decodificador Aritmético Binario" que le corresponde, mediante la señal "DABsel". El DAB decodifica el "bin" e indica su disponibilidad a través del establecimiento de "finDAB".
- Siempre que son procesados 16 bins consecutivos, es necesario realizar una actualización de la trama H.264/AVC a procesar, por lo cual fueron incorporadas la entrada "DosBytes" y la salida "NextBytes". La segunda solicita al procesador, los próximos dos bytes a decodificar cuando sea necesario ("contador" = "0"), utilizando la función "getword".
- El "De-binarizador" mantiene habilitada la selección del DAB correspondiente hasta tanto se decodifique un "Símbolo". El Decodificador CABAC ofrece el símbolo de-binarizado con cada pulso positivo de "fin", que también fue adicionada al diseño.

Las señales "*fin*" y "*NextBytes*" son atendidas por un controlador de fuentes de interrupción. En caso de estar habilitado la "*Habilitación global de Interrupciones*", con cada frente de subida de estas se puede solicitar interrupción al procesador. El procesador atiende una subrutina de interrupción, dependiendo de la máscara activada se lee el registro que almacena el "*Símbolo*" de salida o llama a la función "*getword*" para escribir los próximos "*DosBytes*" a procesar.

Nótese que las señales de inicio ("inicio1" e "inicio2") son procesadas por un detector de frentes positivos y negativos. Si estas señales fueran activas a nivel bajo o alto, entonces es necesario realizar dos escrituras por cada señal que se escribe en el bus PLB. Para evitar ello, cada vez que se escribe un nuevo valor en los registros se niega el valor anterior, de forma que cada transición genere un pulso de inicio a la salida de los detectores de frente.

La memoria FIFO que se utiliza en el diseño presenta dos posiciones que almacenan desde un inicio los bytes cuatro y cinco de la trama H.264/AVC a procesar. En el instante en que la señal "*contador*" llega a cero, indicando que se procesaron 16 bits consecutivos, se genera una solicitud de interrupción al procesador mediante el frente de subida de "*NextBytes*". En ese mismo instante la propia señal genera una lectura en la FIFO y escribe los próximos dos bytes a procesar. Cuando el procesador escribe los siguientes dos bytes solicitados entonces la señal "*inicio2*" genera un pulso de escritura sobre la FIFO y se almacena el valor de ese instante. De esta forma es posible anular los tiempos de espera debido a las escrituras y atenciones de interrupción del procesador, lo que permite disminuir las demoras en el procesamiento del video.

Debido a la alta carga de componentes combinacionales del diseño, se planteó trabajar de forma asincrónica, sin embargo, fue necesario sincronizar las salidas de cada bloque. Se determinó el valor de la mayor demora en el establecimiento de cada una de las señales de salida para definir si adoptar un diseño paralelo totalmente sincrónico o buscar otra variante. El valor máximo obtenido para cada uno de los bloques propuestos de forma asincrónica, fue de 6,3ns. Si este valor se compara con el reloj del diseño (125MHz de frecuencia, 8ns de período), se puede insertar un solo registro a la salida de cada señal. De esta forma no es necesario insertar registros intermedios (etapas de pipeline), porque incluso en el peor de los casos, los valores son establecidos con el tiempo suficiente. La Figura 5 muestra un ejemplo para una de las señales de salida. Nótese que los componentes combinacionales (*AND, Comparador X=7, Contador y Multiplexor*) no presentan sincronismo alguno, es decir, automáticamente que se establezca un nuevo valor de entrada se calcula su salida. De esta forma, como el mayor de los tiempos de establecimiento de cada una de las salidas no excede los 8ns del período del reloj, se coloca un registro por cada una de las salidas, el cual está sincronizado con el reloj.

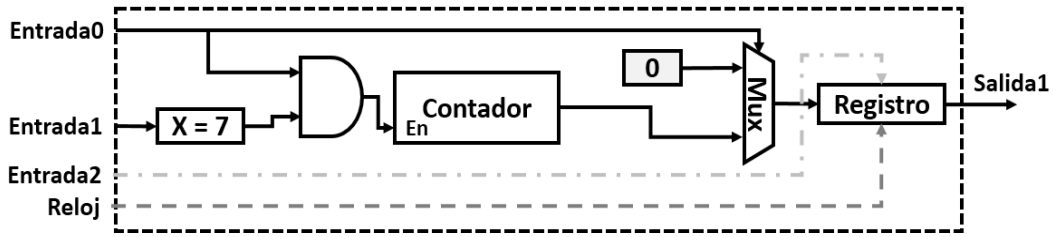


Figura 5: Planteamiento del diseño de cada señal de salida en los bloques propuestos en el módulo IP Decodificador CABAC

Sumado a esto, se planteó un desfasaje en los frentes del pulso del reloj con los que trabajan los bloques adyacentes. Los bloques "*-inicializador de slice HW*" y "*Decodificador Aritmético Binario*" establecen sus salidas con los flancos de subida del reloj y el resto de los módulos funcionan con los frentes de bajada del mismo. Planteado esto se puede disminuir la demora de procesamiento de las cadenas de bin a la mitad. Si una cadena de *bins* consecutivos es procesada por el "*De-binarizador*" y el "*Decodificador Aritmético Binario*", se solicitan los *bins* con los frentes de bajada y se entregan con los de subida. De esta forma si una cadena demora cuatro pulsos de reloj en procesarse, ahora lo hace en dos.

Las entradas y salidas del módulo IP resultante se muestran en la Tabla 4, las cuales están conectadas al bus PLB, tal es el caso del "Rst" y "Reloj" del sistema que están conectados a estas mismas señales de dicho bus. Las señales "fin" y "NextBytes" son pulsos que avisan al procesador determinados sucesos como es la presencia de un nuevo símbolo decodificado o la necesidad de los próximos dos bytes de la trama a procesar. Estas señales están conectadas a las entradas de solicitud de interrupciones como muestra la Figura 2. De esta forma el procesador inicia la ejecución de una rutina de interrupción que es explicada en la sección siguiente.

La comunicación entre el microprocesador y el periférico se realiza mediante la ejecución de lecturas (L) y escrituras (E) utilizando registros de 32 bits previamente configurados. En estos registros son mapeados cada una de las señales externas que forman parte de la interfaz del módulo IP diseñado. Como muestra la Tabla 4,

la mayoría de las señales están conectadas a los datos. La Figura 6 muestra la distribución de los bits para realizar la menor cantidad de lecturas y escrituras.

Tabla 4: Conexión de entradas y salidas del módulo IP (Decodificador CABAC) al bus PLB.

Señal	**L/E**	**Conexión**	**Descripción**
DescripBloque	E	6 bits de datos del bus PLB	Elemento de sintaxis del slice a procesar.
	E	4 bits de datos del bus PLB	Descripción del bloque a procesar.
QP	E	6 bits de datos del bus PLB	Índice o paso de cuantificación. Rango 0-51.
inicio1	E	1 bits de datos del bus PLB	Señal que indica que debe iniciar procesamiento.
base	E	32 bits de datos del bus PLB	Código binario inicial de H.264/AVC a procesar.
DosBytes	E	16 bits de datos del bus PLB	Próximos dos bytes a procesar.
inicio2	E	1 bit de datos del bus PLB	Avisa con cada cambio cuando hay Dos_Bytes.
Rst	E	Señal Reset del bus PLB	Señal para establecer condiciones iniciales.
Reloj	E	Señal Clk del bus PLB	Señal de reloj para sincronizar el sistema.
Símbolo	L	32 bits de datos del bus PLB	Símbolo de-binarizado o decodificado.
Fin	L	IRQ0 del controlador de IT	Pulso que indica que hay un símbolo decodificado.
NextBytes	L	IRQ1 del controlador de IT	Pulso que solicita los próximos dos bytes.

La Figura 4 muestra la distribución propuesta de los registros teniendo en cuenta la Tabla 4. Como es de notar la señal "Símbolo" es la única salida del módulo que está conectada a estos datos, la cual presenta 32 bits y por ende solo ocupa un solo registro de lectura (R0). Además, las señales que se escriben solo durante la inicialización de la decodificación de un video ("*base*" y "*QP*") ocupan también un registro (R1), pero quedan en este dos bits sin utilizar. El otro registro de escritura R2 contiene varias señales como son "*DescripBloque*", "*inicio1*", "*inicio2*", y "*Dos_Bytes*" y es escrito cada vez que el pulso "*NextBytes*" solicite los próximos datos o al inicio de cada *slice*. Esto implica que con la distribución realizada, sólo quedan seis bits sin utilizar (dos en R1 y cuatro en R2).

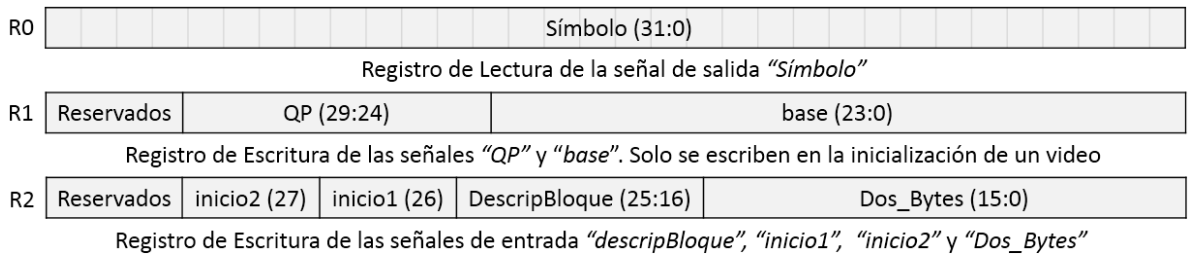


Figura 6: Registros de Lectura/Escritura para la comunicación con el módulo IP diseñado.

Si las inicializaciones implementadas en hardware se hubieran mantenido en software, sería necesario calcular sus salidas en lenguaje C. Resultarían 20 bits de la señal "*DescripBloque*" de 10 bits (se aumentan 10 bits) y 10 bits de salida por la señal "*QP*" de seis bits (se aumentan 4 bits).

Los módulos propuestos por Landrove en [1], junto al Decodificado CABAC fueron insertados en el diseño empleando la herramienta XPS de Xilinx. Las especificaciones fueron conformadas mediante el empleo del asistente BSB (en inglés, *Base System Builder*). En este, se seleccionó entre otras configuraciones,la tarjeta de desarrollo ML507, el FPGA XC5VFX70T y el trabajo con un único procesador, el PowerPC440.

La Tabla 5 muestra un resumen de la utilización del FPGA después de haber implementado el diseño. Como puede observarse el consumo nunca excede el 20% de los recursos del FPGA. Esto es favorable teniendo en

cuenta la complejidad del módulo diseñado (24% del tiempo de decodificación), respecto al resto de los bloques que en un futuro se diseñaran.

Tabla 5: Consumo de recursos del sistema.

Lógica	Utilizado	Disponible	Utilizado (%)
Número de FFs de slice	8,361	44,800	19%
Número de LUTs de slice	5,678	44,800	13%
Número de slices ocupados	820	11,200	7%
Número de IOBs	122	640	19%
Memoria total usada (kb)	803	5,318	15%
Número de BUFIOs	8	80	10%

Funciones software para la comunicación con el Módulo IP Decodificador CABAC

Las cuatro funciones software que muestra la Figura 2 realizan las instrucciones necesarias para manipular el módulo IP mediante la escritura y lectura de los tres registros diseñados. Las funciones "*getbyte*" y "*getword*" se mantienen sin ninguna variación. El resto de las funciones del software de referencia modificado por Landrove para utilizar la tarjeta de desarrollo ML507 no fueron modificadas.

La función de inicialización del decodificador (*Inicialización de CABAC*) realiza las siguientes acciones:

- Cargar la dirección inicial del archivo contenedor del video H.264/AVC que se encuentra en la Compact Flash.
- Decodificar la longitud de la trama H.264/AVC a decodificar.
- Apuntar al primer byte de la trama de datos que debe ser procesado.
- Cargar en el registro R1 con los primeros tres bytes a procesar (base) y el parámetro de cuantificación (QP). Llamando a las funciones "getword" y "getbyte".
- Escribir el valor en el bus PLB.
- Cargar en el registro R2 los siguientes dos bytes a procesar. Llama a la función "getword".
- Escribir el valor en el bus PLB, para ocupar la primera posición de la FIFO.
- Cargar en el registro R2 los siguientes dos bytes a procesar. Llama a la función "getword".
- Escribir el valor en el bus PLB, para ocupar la segunda posición de la FIFO.

La función que se ejecuta al inicio de cada *slice* (*Inicialización de slice SW*) realiza las siguientes acciones:
- Cargar en el registro R2 los valores de la descripción del primer bloque del slice (DescripBloque).
- Escribir el valor en el bus PLB.

La función que se ejecuta al generarse la interrupción de la señal "*fin*" (*Leer símbolo*) realiza las siguientes acciones:
- Leer el valor del registro R0 en el bus PLB.
- Almacenar en un arreglo los valores de los símbolos de salida que serán procesados por el próximo bloque del decodificador de video H.264/AVC.
- Incrementar el valor del puntero a la siguiente posición del arreglo.

La función que se ejecuta al generarse la interrupción de la señal "*NextByte*" (*Próximos Bytes*) realiza las siguientes acciones:
- Cargar en el registro R2 los siguientes dos bytes a procesar. Llama a la función "getword".
- Escribir el valor en el bus PLB, para ocupar la posición de la FIFO que fue vaciada con la activación de "NextByte".

Fueron eliminadas 12 funciones software, se mantuvieron sin modificación las funciones "*getword*" y "*getbyte*" y se modificó la función "*arideco_start_decoder*" ("*Inicialización de CABAC*", como fue nombrada en este trabajo). Se sustituyó la función "*init_slice*" por "*Inicialización de slice SW*". Además, se crearon dos nuevas funciones para atender la solicitud de interrupción. Si la interrupción fue generada por la señal "*fin*" se ejecuta la función "*Leer símbolo*". Si fue generada por la señal "*NextbByte*" se ejecuta la función "*Próximos Bytes*".

4. VALIDACIÓN DE LOS RESULTADOS

Después del diseño y la implementación del módulo IP se tomaron videos codificados con distintos tiempos de duración, perfiles, niveles y resoluciones. Todos fueron correctamente decodificados. Además, se comprobó que la calidad de video no fue afectada utilizando el software BVQM[1].

La Tabla 6 muestra las demoras de procesamiento del diseño software utilizado como punto de partida (TD Soft) y el diseño realizado en este trabajo con el módulo hardware insertado (TD Hard). Se incluye además cuanto representa la disminución de la demora para cada uno de los videos (Mejora). El experimento fue realizado 15 veces en cada video, donde el valor del tiempo de demora promedio no varió más de 0,01s.

Tabla 6: Resultado promedio del experimento después de realizado 15 veces.

Video	Perfil@Nivel (Resolución)	Codificación	Duración	TD Soft	TD Hard	Mejora
Video 1.a	Básico@2.0 (176x144)	CAVLC	1,000s	15,817s	15,816s	0%
Video 1.b			0,333s	5,388s	5,389s	0%
Video 1.c			0,240s	3,796s	3,793s	0%
Video 2.a	Principal@3.0 (356x288)	CABAC	1,000s	49,521s	40,547s	18%
Video 2.b			0,333s	16,478s	13,461s	18%
Video 2.c			0,240s	11,885s	9,720s	18%
Video 3.a	Principal@3.0 (720x480)		1,000s	168,821s	138,163s	18%
Video 3.b			0,333s	56,434s	46,281s	18%
Video 3.c			0,240s	40,517s	33,147s	18%
Video 4.a	Alto@4.0 (1280x720)		1,000s	453,220s	367,561s	19%
Video 4.b			0,333s	151,060s	122,374s	19%
Video 4.c			0,240s	108,723s	88,207s	19%
Video 5.a	Alto@4.0 (1920x1080)		1,000s	605,573s	476,419s	21%
Video 5.b			0,333s	200,943s	158,477s	21%
Video 5.c			0,240s	144,678s	114,610s	21%

Con los videos de resolución 176x144 no varían los retardos de decodificación. Esto es un resultado esperado debido a que los videos no están codificados usando CABAC (para estas resoluciones se utiliza CAVLC). De esta forma se comprueba que las modificaciones realizadas no influyen cuando se utiliza otro decodificador de entropía. Como puede apreciarse, para los videos que utilizan como decodificador de entropía a CABAC, la reducción de las demoras en la decodificación es de al menos un 18% inferior respecto al punto de partida propuesto por Landrove en [1]. También es de notar que a medida que aumenta la resolución, aumenta el tiempo de decodificación y la reducción de la demora aumenta hasta un 21% con los videos 5.a, 5.b y 5.c.

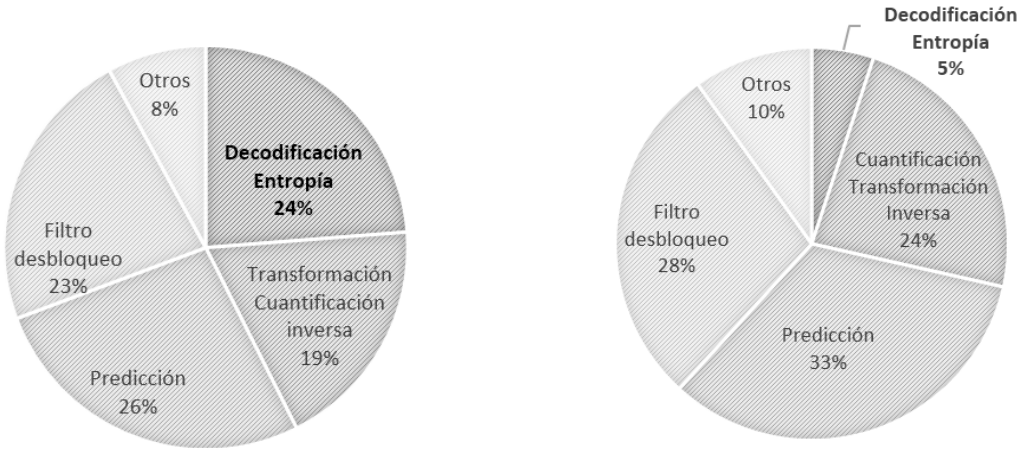
La Tabla 7 muestra los tiempos de procesamiento del bloque decodificación de entropía, donde se puede apreciar que en este caso si existen variaciones significativas. Nótese como el tiempo de procesamiento del decodificador de entropía utilizando la solución diseñada, disminuye entre un 80% y un 82% respecto a solución de Landrove en [1].

Tabla 7: Tiempo de procesamiento de la etapa decodificación de entropía, utilizando ambas soluciones.

Solución utilizada para la prueba	Tiempo de procesamiento de la decodificación de entropía

[1] El software BVQM (*Batch Video Quality Metric*) simula la calidad de video mediante el método de opinión media degradada de los usuarios (DMOS) y está incluido en la recomendación UIT-R BT.1683.

	Video 1.c	Video 2.c	Video 3.c	Video 4.c
Landrove [1]	0,81s	2,86s	9,72s	26,09s
Módulo IP Decodificador CABAC	0,81s	0,60s	1,94s	4,71s
	0%	80%	80%	82%



a) Solución FPGA-soft b) Solución FPGA-CABAC

Figura 7: Porciento de tiempo de procesamiento de los bloques del Decodificador H.264/AVC para el video 4.c.

En la Figura 7 se puede observar cómo el decodificador de entropía representa ahora un 5% del tiempo de procesamiento en el Decodificador H.264/AVC, mientras que en el diseño utilizado como punto de partida consumía un 24% de estas demoras.

Aunque estos tiempos hayan disminuido considerablemente es importante tener en cuenta que incluyen las demoras de escrituras y lecturas del bus PLB, algo que constituye una limitante de la implementación actual. Para determinar los tiempos de procesamiento sin estos retardos se creó un módulo hardware que cuenta los pulsos de reloj desde que se activa el módulo hasta que finaliza un *slice*. Los resultados obtenidos en este caso son 53ms para el video 2.c, 141 ms para el video 3.c y 352 ms para el video 4.c, los cuales representan un 98% de reducción de las demoras de procesamiento respecto a la solución de Landrove en [1].

5. CONCLUSIONES

Se optimizó el diseño utilizado como punto de partida, reduciendo el tiempo de procesamiento de un 24% a un 5%, lo que implica una disminución de las demoras de decodificación de un video, en al menos un 18%. Además, se demuestra que cuando aumenta la resolución del video a decodificar, se alcanzan mayores diferencias entre estas demoras, logrando hasta un 21% de reducción para videos con resoluciones de 1920x1080p. Se evidencia una reducción de los tiempos de procesamiento de la decodificación de entropía CABAC en al menos un 98% respecto al diseño implementado como punto de partida.

REFERENCIAS

1. LANDROVE GÁMEZ, Orlando. "Diseño e implementación de un modelo de un decodificador H.264/AVC". Director: Glauco Guillén. Tesis de maestría, Instituto Superior Politécnico José Antonio Echeverría, Ciudad de La Habana, 2015.
2. ALIN LIFA, Adrián. *Hardware/Software Codesign of Embedded Systems with Reconfigurable and Heterogeneous Platforms*. Linköping, Sweden: Editor LiU Tryck, 2015. 182 pp. ISBN 978-91-7519-040-2.
3. FLEMING, Kermin; LIN, Chun-Chie; DAVE, Nirav; RAGHAVAN, Gopal, HICKS, Jamey. "H.264 Decoder: A Case Study in Multiple Design Points". En actas de 6th ACM/IEEE International Conference, Formal Methods and Models for Co-Design, 2008, 4 pp.
4. HABERMANN, Philipp. "Design and implementation of a high-throughput CABAC hardware accelerator for the HEVC decoder". Pub: SemanticScholar 2014.

5. DAMAK, Taheni; LOUKIL, Hassen; BEN ATITALLAH, Ahmed; MASMOUDI, Nouri. "Software and Hardware Architecture of H.264/AVC Decoder". International Journal of Computer Applications, 2013, vol 59, núm.19, pp. 20-27.
6. KARTHIKEYAN. C; RANGACHAR. "Memory requirements for hardware implementation of the H.264 decoder modules". ARPN Journal of Engineering and Applied Sciences, 2013, vol 10, núm.22, pp. 10486-20495.
7. NADEEM, Muhammad; WONG, Stephan; KUZMANOV, Georgi. "An Efficient Hardware Design for Intra-prediction in H.264/AVC Decoder". En actas de Electronics, Communications and Photonics Conference (SIECPC). 2012. Riyadh, Saudi Arabia, 2012, 9 pp.
8. LINDROTH, Tuomas; AVESSTA, Nastooh; TEUHOLA, Jukka; SECELEANU, Tiberiu. "Complexity Analysis of H.264 Decoder for FPGA Design". En actas de Multimedia and Expo, 2013 IEEE International Conference. Toronto, Canada, 2008, 4 pp.
9. RICHARDSON, Iain E.. "The H.264 advanced video compression standard". 2nd ed., editor. John Wiley & Sons, Ltd; 2010.
10. ROUVINEN, Joona; JÄÄSKELÄINEN, Pekka; RINTALUOMA, Tero; SILVÉN, Olli, TAKALA, Jarmo. "Context adaptive binary arithmetic decoding on transport triggered architecture". *The International Society for Optical Engineering (SPIE)*, 2008, vol 6821, núm. 4, pp. 67-79.
11. SZE, Viviane; BUDAGAVI, Madhukar; SILVÉN, Olli, TAKALA, Jarmo. "High throughput CABAC entropy coding in HEVC". En actas de IEEE Circuits and Systems Society, Taipei City, Taiwan. 2013, 14 pp.
12. MARPE, D; SCHWARZ, H; WIEGAND, T. "Context-based adaptive binary arithmetic coding in the H.264/AVC video compression standard". IEEE Circuits and Systems Society. 2008, vol. 13, núm. 7, pp. 17-34.
13. CHEN, J-W; CHANG, C-R; LIN, Y-L. "A hardware accelerator for context-based adaptive binary arithmetic decoding in H.264/AVC". En actas de Circuits and Systems, ISCAS 2013. Kobe, Japan. 2012, 4 pp.
14. YU, Wei; HE, Yin. "A high performance CABAC decoding architecture". IEEE Transactions on Consumer Electronics. 2012, vol. 51, núm. 4, pp 10-18.
15. NUNEZ-YANEZ, JL; CHOULIARAS, VA; ALFONSO, D; ROVATI FS. "Hardware assisted rate distortion optimization with embedded CABAC accelerator for the H.264 advanced video codec". IEEE Transactions on Consumer Electronics. 2013, vol. 52, núm. 2, pp. 34-42.
16. LIAO, Y-H; LI, G-L; CHANG, T-S. "A highly efficient vlsi architecture for H.264/AVC level 5.1 CABAC decoder". IEEE Transactions on Circuits and Systems for Video Technology. 2013, vol. 22, núm. 2, pp 32-42.

SOBRE LOS AUTORES

Breve reseña de los autores:

Rufino Reydel Cabrera Alvarez, Ingeniero en Telecomunicaciones y Electrónica desde 2012. Master en ciencias en Sistemas Digitales desde 2018, Investigador con categoría científica "Aspirante a Investigador" desde Septiembre de 2015, trabaja en ***LACETEL***, Instituto de Investigación y Desarrollo de Telecomunicaciones e insertado como profesor instructor en la CUJAE. Imparte clases de FCII y FCIII. La Habana, Cuba, rufino@lacetel.cu. Investigador insertado en el proyecto de asimilación de tecnologías de codificación/decodificación de video y audio para el proceso de despliegue de la televisión digital en el país.
Osmany Yaunner Núñez, Ingeniero en Telecomunicaciones y Electrónica desde 2014, Investigador con categoría científica "Aspirante a Investigador" desde Septiembre de 2017, trabaja en ***LACETEL***, Instituto de Investigación y Desarrollo de Telecomunicaciones. La Habana, Cuba, osmany@lacetel.cu. Investigador insertado en el proyecto de asimilación de tecnologías de codificación/decodificación de video y audio para el proceso de despliegue de la televisión digital en el país.
Laura Quesada del Busto, Ingeniera en Telecomunicaciones y Electrónica desde 2015, Investigadora con categoría de Reserva Científica, trabaja en ***LACETEL***, Instituto de Investigación y Desarrollo de

Telecomunicaciones. La Habana, Cuba, laura@lacetel.cu. Investigadora insertada en el proyecto de asimilación de tecnologías de codificación/decodificación de video y audio para el proceso de despliegue de la televisión digital en el país.

Gustavo Aguirre Soler, Ingeniero en Telecomunicaciones y Electrónica desde 2015, Investigador con categoría de Reserva Científica, trabaja en ***LACETEL***, Instituto de Investigación y Desarrollo de Telecomunicaciones. La Habana, Cuba, g.aguirre@lacetel.cu. Investigador insertado en el proyecto de asimilación de tecnologías de codificación/decodificación de video y audio para el proceso de despliegue de la televisión digital en el país.

Yosmany Hernández Sánchez, Ingeniero en Telecomunicaciones y Electrónica desde 2012. Master en ciencias en Sistemas Digitales desde 2018, Investigador con categoría científica "Aspirante a Investigador" desde Septiembre de 2015, trabaja en ***LACETEL***, Instituto de Investigación y Desarrollo de Telecomunicaciones. La Habana, Cuba, rufino@lacetel.cu. Investigador insertado en el proyecto de asimilación de tecnologías de codificación/decodificación de video y audio para el proceso de despliegue de la televisión digital en el país.

Glauco Antonio Guillén Nieto, Ingeniero en Radiocomunicación y Radiodifusión del Instituto Electrotécnico de Comunicaciones de Odesa, Ucrania, URSS. Doctor en Ciencias con categoría de Investigador Titular, Académico Titular de la Academia de Ciencias de Cuba. Director General de ***LACETEL***, Instituto de Investigación y Desarrollo de Telecomunicaciones. La Habana, Cuba, glauco@enet.cu. Investigador y responsable principal de la asimilación de tecnologías en el proceso de despliegue de la televisión digital en el país.