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

PROPUESTA DE OPTIMIZACIÓN DE UN DECODIFICADOR CABAC

Ing. Osmany Yaunner Núñez

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

RESUMEN

CABAC es un modo opcional de codificación de entropía utilizado en los perfiles Principal y Alto del formato de compresión de video H.264/AVC. Esta herramienta de codificación contribuye significativamente a la alta eficiencia de H.264, pero al mismo tiempo aumenta su complejidad en el diseño e implementación. ***LACETEL®,*** Instituto de Investigación y Desarrollo de Telecomunicaciones cuenta con un modelo de un decodificador de video H.264 basado en FPGA. Este modelo aunque es completamente funcional, no cumple con los requerimientos de tiempo para ser validado como una solución práctica. El decodificador CABAC aparece como uno de los bloques principales a rediseñar para optimizar el sistema. Su función es extraer del código binario proveniente del codificador H.264, los parámetros que se utilizan para la reconstrucción de la secuencia de video. La implementación de CABAC existente en ***LACETEL***, es puramente software, lo cual se traduce en una alta secuencialidad que dificulta la aceleración del proceso de decodificación. El presente trabajo propone una alternativa para aumentar la velocidad de decodificación CABAC mediante el diseño e implementación de un módulo IP del decodificador aritmético binario de "bypass", el cual integra el núcleo de CABAC.

PALABRAS CLAVES: CABAC, codificación de entropía, decodificador de video, FPGA, módulo IP.

OPTIMIZATION PROPOSAL OF A CABAC DECODER

ABSTRACT

CABAC is an optional entropy coding mode, which is used in Main and High profiles of the H.264/AVC video compression format. This coding tool contributes significantly to H.264 high efficiency, but increases its complexity of design and implementation at the same time. ***LACETEL®***, Research and Development of Telecommunications Institute has designed a model of an FPGA-based H.264 video decoder. Although this model is completely functional, it doesn't meet the time requirements to be proposed as a practical solution. CABAC decoder appears as one of the main blocks to redesign in order to optimize the system. Its function is to extract from the binary code generated by the H.264 encoder, parameters that are used for the video sequence reconstruction. The CABAC implementation that was developed by ***LACETEL,*** is completely software, which means that it is highly sequential, making difficult to accelerate the decoding process. An alternative is proposed in the present paper, in order to increase CABAC decoding speed by designing and implementing an IP module of the bypass binary arithmetic decoder which is part of CABAC core.

KEY WORDS: CABAC, entropy coding, video decoder, FPGA, IP module.

1. INTRODUCCIÓN

La transformación de la mayoría de los sistemas analógicos en digitales ha sido uno de los principales cambios tecnológicos en las últimas décadas. La televisión es una de las ramas de las Telecomunicaciones que está ligada al proceso de digitalización, debido a las numerosas ventajas que trae consigo. Los sistemas de Televisión Digital (TVD) requieren de un proceso conocido como compresión o codificación de video, ya que este permite reducir la cantidad de datos que se utilizan para representar la señal de video digital, eliminando

redundancia y convirtiéndola en un código binario que ocupa menos capacidad cuando se almacena o transmite. La compresión posibilita que las señales de video digital puedan ser enviadas a través de las redes de distribución por los sistemas de transmisión convencionales. En la actualidad, es cada vez más necesario este proceso, no solo en televisión digital, sino en todas las aplicaciones que requieran almacenamiento y transmisión de video.

Actualmente el país se encuentra en un proceso de asimilación de la Televisión Digital Terrestre (TDT). Una fase importante de este proceso es la selección de la norma de compresión de video, lo cual implica un análisis de los estándares existentes. Los que más destacan en el contexto de la TVD son: MPEG-2 parte 2, H.264/AVC y AVS-1 parte 2, los cuales superan las deficiencias de sus antecesores (MPEG-1, 2, 4 parte 2) para lograr mejores tasas de compresión.[1] Como parte de este análisis y para tributar al proceso de independencia tecnológica en el país, ***LACETEL*** define entre sus proyectos algunas soluciones teóricas y prácticas utilizando H.264/AVC por su elevada eficiencia de codificación y amplio rango de aplicaciones.

Entre las soluciones obtenidas por ***LACETEL*** se encuentra un modelo de un decodificador H264/AVC basado en FPGA. Este modelo, a pesar de no constituir una solución para tiempo real, es completamente funcional y constituye una base para el diseño y desarrollo de bloques que contribuyan a optimizar su procesamiento. De esta manera, gradualmente se tributa a un sistema H.264/AVC capaz de soportar los requerimientos para la decodificación de video en tiempo real.

Un elemento esencial que contribuye a la alta eficiencia de H.264, es la codificación aritmética binaria basada en adaptación de contexto (CABAC, por sus siglas en inglés, Context-based Adaptive Binary Arithmetic Coding). CABAC es uno de los dos modos alternativos de codificación de entropía empleados en H.264, mejora la eficiencia del códec y aumenta también su complejidad en el diseño e implementación. El decodificador de entropía CABAC aparece como la primera etapa del decodificador de video H.264 y entrega a los restantes bloques, parámetros imprescindibles para la reconstrucción del video, por lo que es importante que su velocidad de procesamiento permita la decodificación de la secuencia de video en tiempo real.

Dentro de las fases de diseño de un decodificador H.264 en que se encuentra ***LACETEL***, resulta importante incrementar en la práctica la velocidad de procesamiento del decodificador CABAC. En este contexto se define el problema como la elevada dependencia entre los bloques internos que conforman a CABAC, con un algoritmo altamente secuencial, resultando engorroso la sustitución de módulos software por otros concurrentes, y trayendo como consecuencia que CABAC aparezca como uno de los bloques importantes a rediseñar para lograr optimizar el sistema y a su vez acercarlo a una solución práctica para tiempo real.

En este trabajo se pretende resolver el problema planteado anteriormente mediante el diseño e implementación de un módulo IP para un decodificador de "bypass" que forma parte del núcleo CABAC, así como su inserción dentro de un sistema basado en el bus PLB con el microprocesador PowerPC embebido.

2. DESCRIPCIÓN TEÓRICA DE CABAC

Decodificación de entropía

La figura 1 muestra el diagrama en bloques simplificado de un decodificador de video H.264. La primera etapa, denominada decodificación de entropía, se encarga de extraer parámetros de video a partir del código binario H.264 proveniente del codificador. Estos parámetros se denominan elementos de sintaxis y son utilizados por los restantes bloques (transformada inversa, cuantificación inversa y reconstrucción) para reconstruir los cuadros o campos que conforman una secuencia de video.

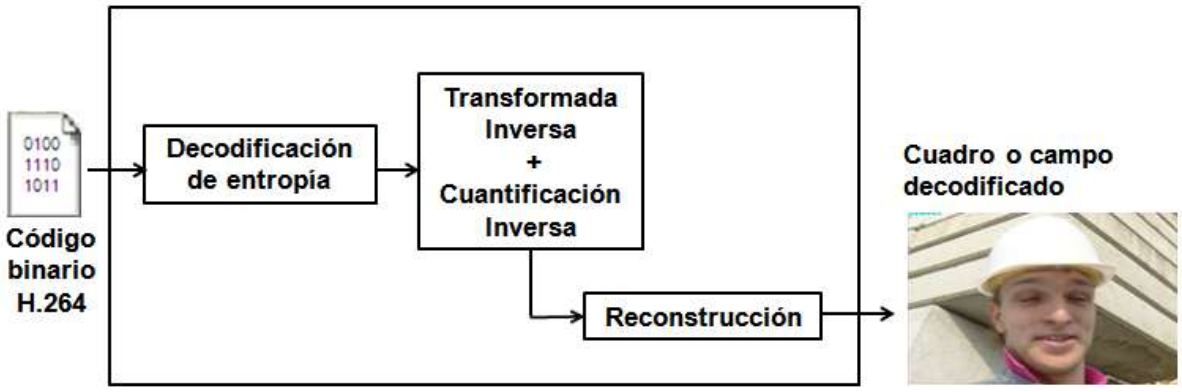

Figura 1 Decodificador H.264

H.264 emplea dos modos fundamentales de codificación/decodificación de entropía: CABAC y CAVLC (codificación de longitud variable basada en adaptación de contexto, en inglés, Context-based Adaptive Variable Length Coding). El segundo se basa en la asignación de códigos de longitud variable (VLC, por sus siglas en inglés, variable length coding) a los símbolos en dependencia de su probabilidad de ocurrencia. A los símbolos con altas probabilidades les corresponden palabras de código cortas y a los de bajas probabilidades, palabras de código largas. Para obtener las probabilidades se seleccionan tablas VLC en dependencia del contexto del símbolo, o sea en función de los valores que ha tomado previamente.

Aunque CAVLC utiliza adaptación de contexto para la selección de las tablas VLC, este método y todos los que se basan en códigos de longitud variable, tienen como principal deficiencia que no representan los símbolos con probabilidades mayores que 0.5 eficientemente por el siguiente motivo. Cuando se trabaja con estos códigos existe un parámetro denominado información de contenido (log2 (1/p) donde p es la probabilidad de ocurrencia del símbolo), el cual indica la cantidad óptima de bits con que se debe representar la palabra de código. La expresión anterior permite observar que para símbolos con probabilidades mayores que 0.5 la información de contenido es menor que 1 y sus palabras de código deben tener un bit de longitud. Esto hace que la codificación de esos símbolos sea ineficiente. [2]

CABAC supera estas deficiencias y logra una reducción de la tasa de bits entre un 5 y 15 % superior con respecto a CAVLC, [3] ya que selecciona modelos de probabilidad para cada elemento de sintaxis según su contexto, los cuales son actualizados cada vez que se codifica un símbolo y además emplea codificación aritmética binaria.

Proceso de decodificación CABAC

La adaptación de contexto en CABAC se basa en que a cada elemento de sintaxis se le selecciona una tabla de probabilidad, en dependencia de los valores que ha tomado ese elemento en bloques previamente codificados. Por otro lado, la codificación aritmética permite representar una secuencia de símbolos con un solo número basándose en sus probabilidades de ocurrencia mediante una subdivisión sucesiva de intervalos [4]. En el caso de CABAC que emplea codificación aritmética binaria sólo se codifican dos símbolos, el "0" y el "1", por lo que a cada símbolo antes de ser codificado se le realiza un proceso denominado binarización.

Para obtener el valor de un elemento de sintaxis dado, el decodificador realiza el siguiente procedimiento que se muestra en la figura 2. Primero el bloque denominado de-binarización obtiene una tabla de búsqueda que contiene todos los posibles valores que puede tomar el elemento de sintaxis cuyo valor se quiere determinar y cada posible valor tiene asociada una secuencia de bits (denominados bins). Para determinar algunos bins se requiere de un proceso denominado modelado de contexto, el cual selecciona una tabla de probabilidad para

cada bin. En una misma secuencia pueden existir bins que no requieran del modelado de contexto, puesto que se asume que su probabilidad de ocurrencia es de 0.5. Luego se realiza la decodificación aritmética binaria (DAB), la cual es considerada el núcleo del decodificador CABAC. La DAB lee una variable del código H.264 y la compara con una variable interna. Como resultado de esta comparación se genera un bin a la salida de la DAB, si este no coincide con ninguna de las posibles secuencias de bins que tiene la de-binarización, entonces se repite el proceso explicado anteriormente hasta que se encuentre una secuencia válida. Cuando se encuentra la secuencia se obtiene a la salida del decodificador CABAC el valor del elemento de sintaxis. [4]

CABAC utiliza tres métodos principales de DAB: regular, "bypass" y terminación. [5] El primero es el que requiere de la tabla de probabilidad (denominada modelo de contexto), el segundo omite el proceso de modelado de contexto. El tercero tampoco utiliza el modelo de contexto y codifica sólo dos elementos de sintaxis, uno de los cuales es una bandera que indica cuando llega el último macrobloque de una imagen codificada. El método seleccionado para diseñar e implementar en este trabajo fue la decodificación de "bypass" debido a que representa un modo simplificado con respecto al regular que permite acelerar el proceso de decodificación ya que omite el modelado de contexto asumiendo que los símbolos son equiprobables.

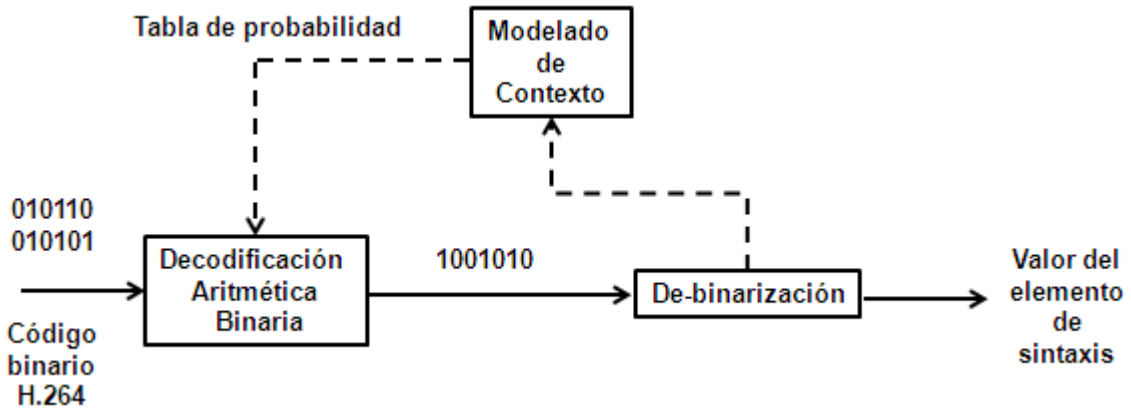

Figura 2 Proceso de decodificación CABAC

3. ESPECIFICACIONES DEL DISEÑO DEL MÓDULO IP DE DECODIFICACIÓN DE BYPASS

Esta sección se divide en dos partes fundamentales. La primera parte describe el proceso de diseño de un módulo hardware que implementa el algoritmo de decodificación de "bypass" teniendo en cuenta el software de referencia JM 18.6[1]. La segunda parte aborda el proceso de adición del módulo diseñado en un sistema embebido, de forma tal que pueda ser manejado por un microprocesador.

Implementación de la decodificación aritmética binaria de "bypass" en VHDL

Para el diseño en VHDL de la decodificación de "bypass" se utilizó el software ISE 14.2 de Xilinx®. La figura 3 muestra el esquema general del circuito. La metodología de diseño empleada consistió en dividir la entidad

[1] El Equipo de Video Conjunto (JVT, por sus siglas en inglés, Joint Video Team), el grupo responsable del desarrollo y mantenimiento de H-264/AVC , publica cada cierto tiempo un software escrito en el lenguaje de alto nivel C que representa una implementación de referencia del estándar, conocido como Modelo Conjunto (JM, por sus siglas en inglés, Joint Model).[2]

de mayor jerarquía (decodificador de "bypass") en varios componentes o sub-bloques que realizan una función específica.

Entradas

Las dos señales de entrada principales son offset y rango, que determinan el bin de salida. La señal cont_desp permite que cuando se vaya a realizar una comparación entre offset y rango, estos tengan la misma cantidad de bits significativos. Esto se hace porque en el software de referencia el offset se inicializa en un valor que resulta de leer 3 bytes de entrada (24 bits), mientras que el rango inicialmente es 510 (9 bits), entonces cuando se comparen estos dos valores para decodificar el bin de salida siempre quedará que offset es mayor.

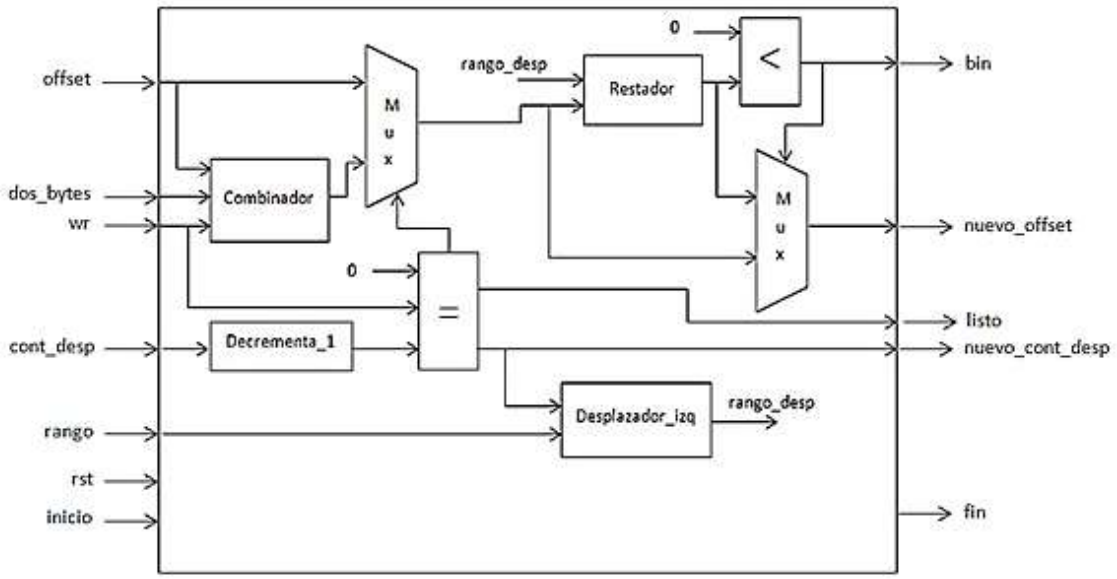

Figura 3 Diagrama en bloques del módulo hardware diseñado en VHDL de la decodificación de "bypass"

Para solucionar este problema se define la señal cont_desp que inicialmente es 15 y cada vez que se hace una operación que involucra a offset y rango, este último es desplazado a la izquierda en una cantidad de veces indicada por cont_desp lo cual equivale a multiplicar por 2^(cont_desp), de esta forma estas dos señales se comparan con la misma cantidad de bits. Por ejemplo si rango = 485, offset = 15376066 y contdesp = 15, la comparación se hace entre rango = 15892480 = 485 * 2^15 y offset = 15376066. La señal cont_desp se decrementa cada vez que se hace cualquiera de las tres decodificaciones aritméticas y cuando llega a 0 se reinicializa en 16. Es importante destacar que el rango desplazado por cont_desp sólo se usa internamente para efectuar la comparación.

Otra señal de entrada importante es dos_bytes la cual como su nombre indica representa dos bytes del flujo codificado de entrada que son leídos cuando cont_desp llega a 0.

Las entradas wr, rst, e inicio son señales de control que permiten que el circuito diseñado pueda insertarse como un periférico y ser manejado por un sistema de mayor complejidad como un procesador el cual contenga el algoritmo de decodificación CABAC y cada vez que este necesite realizar la decodificación de "bypass" le enviaría la señal de inicio al módulo hardware de la figura 3, después de haberle entregado las correspondientes señales de datos. La señal rst hace la misma función que realiza todo reset en un circuito digital, o sea

establecer condiciones iniciales, específicamente pone todas las salidas a 0 cuando se activa. La señal wr indica al circuito que los dos bytes están disponibles a la entrada para ser leídos.

Principio de funcionamiento

El funcionamiento del circuito a grandes rasgos es el siguiente. A la salida del bloque decrementa_1 se obtiene la señal cont_desp decrementada en 1, y si esta no es 0, el multiplexor de entrada deja pasar el offset de entrada, de lo contrario (si cont_desp decrementado es 0) la salida del multiplexor deja pasar la salida del combinador, El multiplexor de entrada es controlado por el comparador con símbolo de igual. El combinador inserta dos bytes del flujo codificado de entrada en los 16 bits menos significativos de la señal offset. A la salida del multiplexor, ya sea el offset de entrada o el combinado se le resta el rango desplazado según cont_desp.

Salidas

El resultado de la resta determina el valor del bin de salida y el nuevo offset que se corresponden con las señales de salida bin y nuevo_offset respectivamente. La señal de salida nuevo_cont_desp toma el valor de cont_desp de entrada menos 1, y cuando el valor decrementado es 0, a nuevo_cont_desp se le asigna 16. La señal de salida listo se activa cuando cont_desp llega 0 e indica que el circuito está listo para recibir los dos bytes que se le insertan al offset. La señal fin indica que se terminó la decodificación de "bypass" y que las señales de salida bin, nuevo_offset y nuevo_cont_desp contienen datos válidos.

La tabla 1 resume las señales de entrada/salida del decodificador de "bypass" y además especifica la cantidad de bits que contiene cada una.

Tabla 1 Resumen de señales de entrada/salida del decodificador de "bypass"

Señal	E/S	Descripción
offset	E	Desplazamiento del rango de decodificación aritmética (32 bits)
rango	E	Ancho del rango de decodificación aritmética (32 bits)
dos_bytes	E	Bits leídos del código H.264 de entrada (16 bits)
cont_desp	E	Cuando llega a 0 indica que se deben leer dos_bytes (5 bits)
wr	E	Indica que se escribieron los dos_bytes y el módulo hardware puede leerlos (activo en '1', 1 bit)
rst	E	Reset, establece condiciones iniciales (activo en '1', 1 bit)
inicio	E	Indica al módulo que opere y genere las salidas (activo en '1', 1 bit)
nuevo_offset	S	Valor actualizado de offset de entrada (32 bits)
nuevo_cont_desp	S	Valor actualizado de cont_desp de entrada (5 bits)
bin	S	Bit decodificado del elemento de sintaxis binarizado (1 bit)
fin	S	Indica al procesador que el módulo generó las salidas (activo en '1', 1 bit)
listo	S	Indica que cont_desp llegó a 0 y el módulo está listo para recibir dos_bytes (activo en '1', 1 bit)

Validación del código VHDL

Para la verificación del principio de funcionamiento del circuito, se utilizó la herramienta de simulación ISim, integrada en el ISE. Para trabajar con el ISim se creó un fichero denominado banco de prueba o "test bench". En un banco de prueba el diseñador especifica las señales de estímulo que se le inyectarán al circuito que se desea comprobar. Estos estímulos deben asemejarse a las señales reales en condiciones extremas.

Antes de especificar los estímulos en el banco de prueba, se ejecutó paso a paso el software de referencia JM 18.6 en el Visual Studio 2008, y se invocó la función "unsigned int biari_decode_symbol_eq_prob (DecodingEnvironmentPtr dep)", que realiza la decodificación de "bypass" y se anotaron los valores de los argumentos como se muestra en la figura 4a. Además se apuntaron los argumentos modificados después de ejecutarse dicha función y su valor de retorno (figura 4b) para poder compararlos con las señales de salida del circuito diseñado. Como se puede observar estos números representan valores reales resultantes de la decodificación de un código H.264 en un video reconstruido. Una vez creado el banco de prueba, se incluyeron los estímulos anotados y se realizaron tres simulaciones. Las formas de onda de la figura 4c[2] comprueban que los valores de salida obtenidos coincidieron con los esperados.

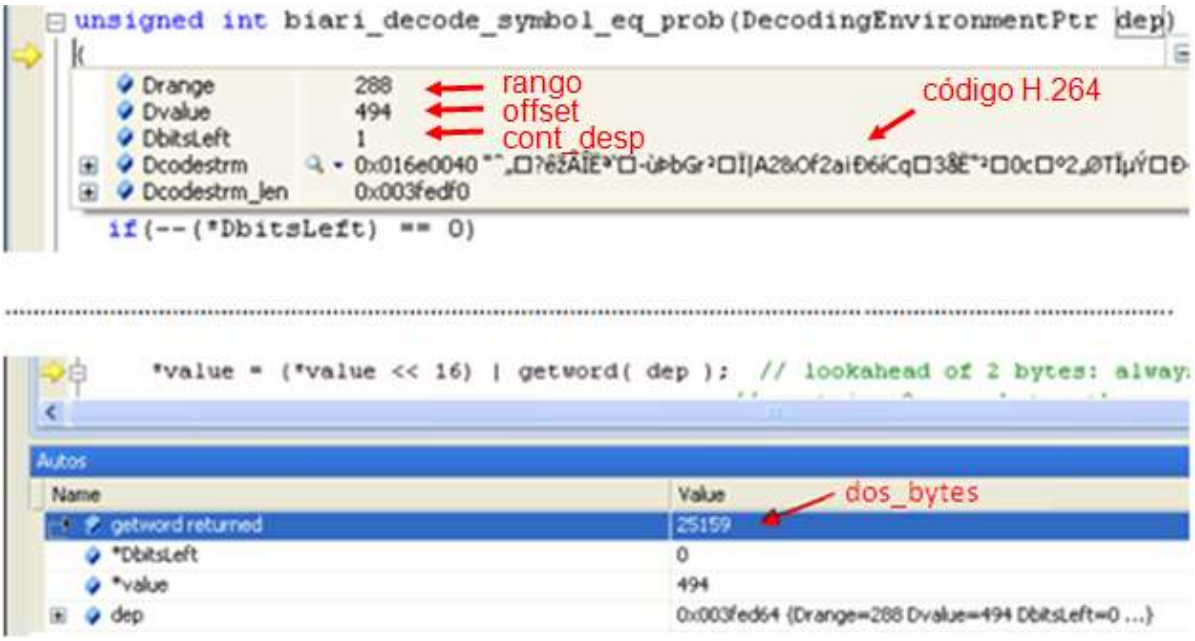

Figura 4a Estímulos para la simulación del módulo hardware de decodificación de "bypass"

Figura 4b Salidas esperadas para la simulación del módulo hardware de decodificación de "bypass"

[2] Las señales encerradas en cuadros rojos discontinuos son las de mayor importancia para la validación del código.

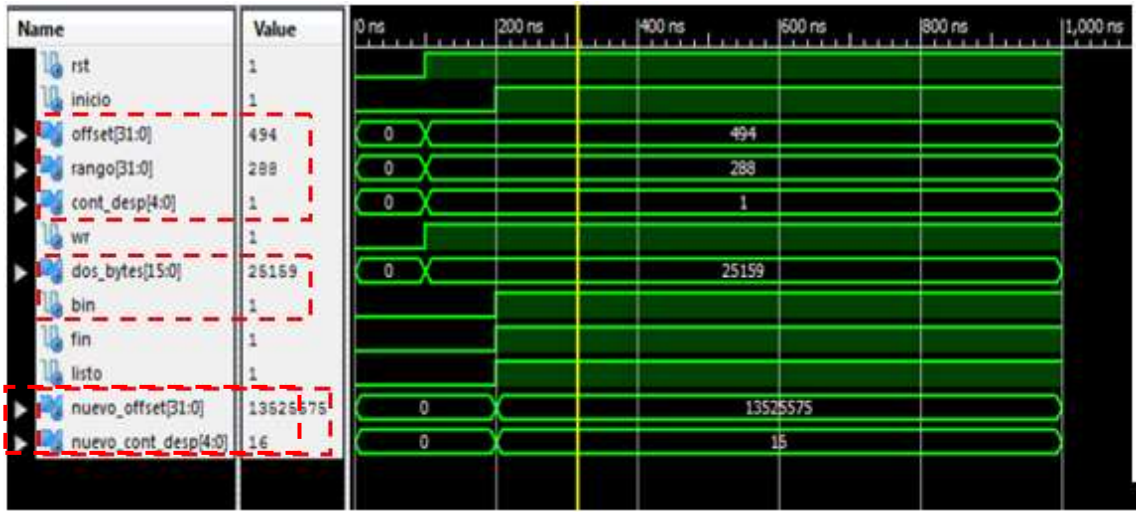

Figura 4c Formas de onda para señales de entrada y salida del circuito diseñado

Inserción del módulo hardware del decodificador de "bypass" en un sistema embebido

Para la implementación del diseño se utilizó la tarjeta de desarrollo ML507 que contiene el FPGA XC5VFX70T-1FFG1136 de la familia Virtex-5 de Xilinx®. Esta tarjeta presenta además, un conector serie JTAG para la conexión a la PC, una entrada de alimentación de corriente alterna (AC) de 5 V, un chip generador de reloj programable de hasta 550 MHz, etc. La tarjeta ML507 también tiene entradas y salidas de video, lo cual es importante para la realización de pruebas reales con un decodificador de video, una vez que el diseño alcance un estado más avanzado.

Esta tarjeta se seleccionó por dos motivos: primero, porque la aplicación a la que tributa el diseño realizado, requiere de un alto nivel computacional y el FPGA contenido satisface este requerimiento dado que presenta numerosas potencialidades relacionadas en la hoja de datos de Xilinx® "Virtex-5 Family Overview"; segundo porque ***LACETEL*** cuenta esta tarjeta de desarrollo.

La tarjeta ML507 incluye dentro del FPGA, un microprocesador embebido PowerPC 440 en el cual se ejecutará un programa de prueba escrito en C para comprobar la comunicación entre dicho procesador y el módulo hardware. Entre las características generales del PowerPC se puede mencionar que tiene una velocidad de operación máxima de 550 MHz, un pipeline (estructura tubular) de 7 etapas y puede procesar múltiples instrucciones por ciclo de reloj. La conexión entre el PowerPC y el módulo, se realiza mediante el bus local de procesador versión 4.6 (PLBv46, por sus siglas en inglés, *Processor Local Bus version 4.6*). Este bus de 128 bits proporciona una interfaz de alta velocidad entre el PowerPC y periféricos de alto desempeño. El hardware es manejado mediante dos funciones especiales de escritura y lectura.

Subsistema hardware

La plataforma hardware del sistema basado en el bus PLB fue configurada en el ambiente de desarrollo XPS. Haciendo uso del asistente BSB (en inglés, *Base System Builder*, Constructor de Sistema Base) se configuraron una serie de especificaciones como el tipo de bus (PLB), la tarjeta de desarrollo a utilizar (ML507), la cantidad de procesadores (1), las frecuencias de reloj del procesador (125 MHz) y del bus (125 MHz) y los periféricos internos que se muestran en la figura 5. Una vez creada la plataforma, se inserta el módulo IP del decodificador de "bypass" mediante un asistente que permite crearlo e importarlo.

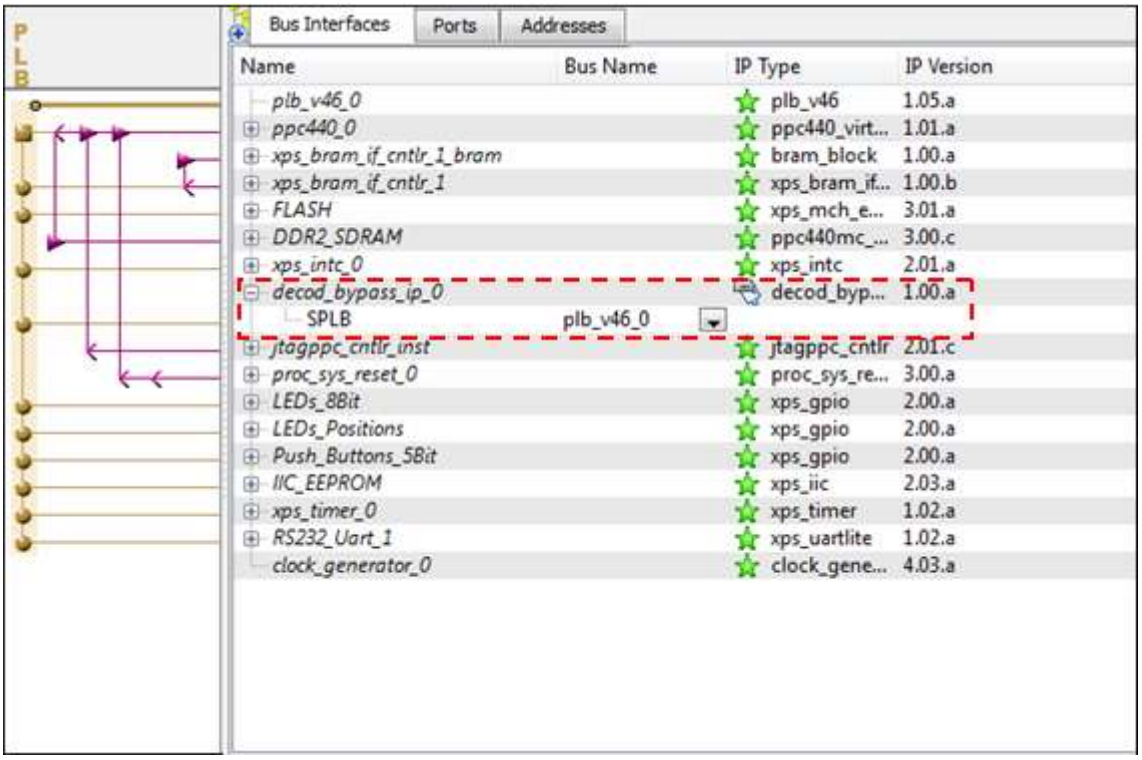

Figura 5 Configuración de la plataforma hardware con el módulo IP "decod_bypass_ip_0" conectado al procesador PowerPC a través del bus PLBv46

Durante el proceso de creación se configuran las siguientes características: el módulo IP no utiliza puertos externos puesto que no se comunica con componentes fuera del FPGA y la atención al periférico es por encuesta. Otra característica importante que se configuró en esta etapa es que el IP es accesible mediante registros software. Estos registros son desde los cuales el microprocesador escribe o lee para comunicarse con el periférico. En este caso se definieron 12 registros de 32 bits cada uno (un registro para cada señal).

Después que se crea el IP sólo se tiene una "caja negra", a la cual se le añade la implementación VHDL y el mapeo de las señales del módulo hardware con los registros software como se muestra en la figura 6. Finalmente se importa el IP a la plataforma hardware del XPS.

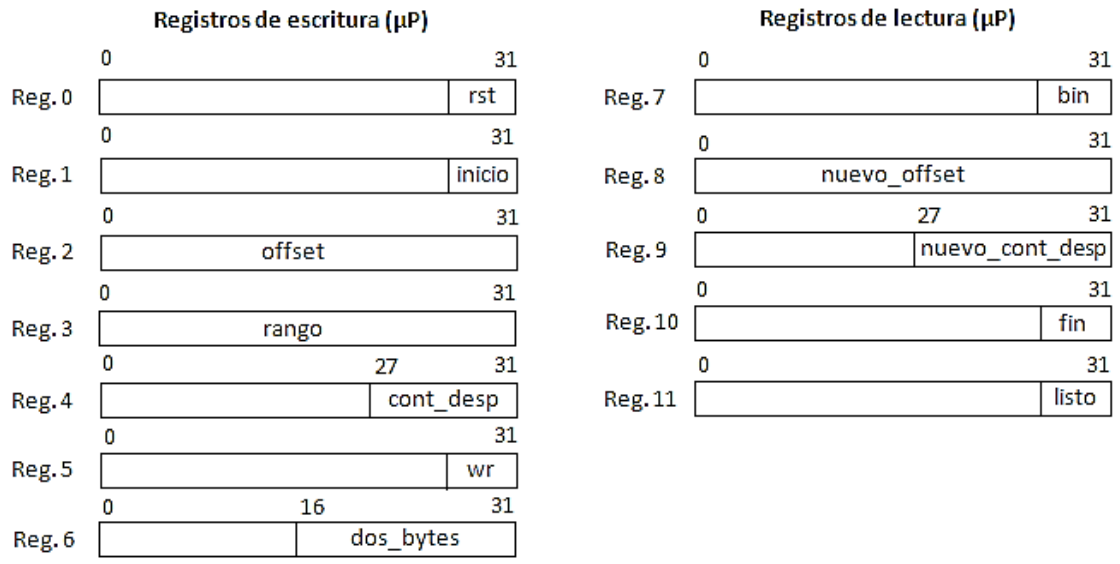

Figura 6 Distribución de registros y señales internas del módulo IP

La tabla 2 resume la cantidad de recursos del FPGA utilizados en el diseño. Como se puede observar los porcentajes de utilización mostrados son relativamente bajos. Esto permite que en un futuro se puedan insertar más módulos hardware dentro del mismo FPGA.

Tabla 2 Resumen de utilización del FPGA

Lógica utilizada	Usada	Disponible	Utilización
Número de FFs de slice	4 284	44 800	9%
Número de LUTs de slice	3 873	44 800	8%
Número de slices ocupados	2 628	11 200	23%
Número de IOBs	182	640	28%
Memoria total usada (KB)	216	5 328	4%
Número de BUFIOs	8	80	10%
Número de PPC 440s	1	1	100%

Subsistema software

La porción software del sistema se desarrolló en el SDK. La comunicación entre el procesador y los módulos hardware se realiza mediante el empleo de los "drivers". Existen "drivers" de bajo y de alto nivel. En este caso se emplearán los "drivers" básicos de bajo nivel que consisten en funciones de leer y escribir en un registro determinado. Estas funciones están definidas en un fichero de cabecera denominado "decod_bypass_ip.h" que se genera cuando se crea el BSP. El BSP es la forma de vincular los elementos incluidos en la plataforma hardware con la nueva aplicación software creada en el SDK.

Para la comunicación desde el procesador hacia el módulo hardware se usa la función DECOD_BYPASS_IP_mWriteReg(dirección_base_ip, registro, dato), la cual escribe el dato en el registro especificado dentro del módulo hardware. Para la comunicación en sentido contrario (desde el módulo hacia el

procesador) se usa la función DECOD_BYPASS_IP_mReadReg(dirección_base_ip, registro), la cual lee y devuelve el dato almacenado en el registro especificado del IP. Como se puede observar para que estas funciones escriban o lean en el periférico correctamente, se debe especificar la dirección base del IP que se almacena en forma de máscara en un fichero denominado xparameters.h que también se incluye en el BSP, lo cual permite que el microprocesador lo habilite.

La figura 7 ilustra el diagrama de flujo del programa escrito en C para comprobar la comunicación entre el procesador y el módulo IP utilizando fundamentalmente las funciones de escritura/lectura mencionadas anteriormente. Como se puede notar, la lectura de los resultados del IP se hace encuestando la señal fin. También se puede observar que falta la encuesta de la señal wr, y el motivo es que este pequeño programa sólo pretende probar que la conexión entre el módulo IP y el procesador a través del bus PLB está funcionando. Además, se desea en un trabajo posterior sustituir la encuesta por la solicitud de interrupción para lograr un diseño más eficiente y optimizado.

Después de compilar la aplicación de prueba en el SDK y ejecutarla paso a paso se obtuvieron los resultados que se muestran en la figura 8. Como se observa los valores leídos por el procesador mediante la función de lectura y almacenados en las variables bin, nuevo_offset y nuevo_cont_desp coinciden con los valores esperados de las salidas de la primera simulación realizada para el módulo de decodificación de "bypass". Con este resultado se demuestra que el algoritmo de decodificación de "bypass" es transparente para el procesador, éste sólo tiene que enviar los datos necesarios y recoger los resultados cuando los necesite.

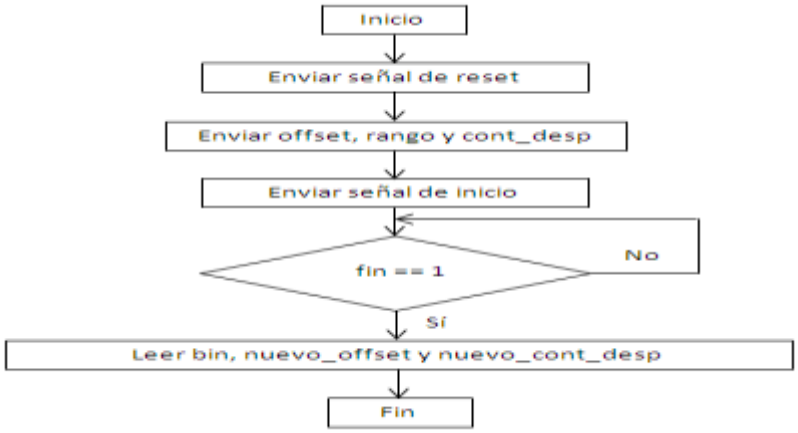

Figura 7 Diagrama en bloques del programa de prueba del módulo IP escrito en C

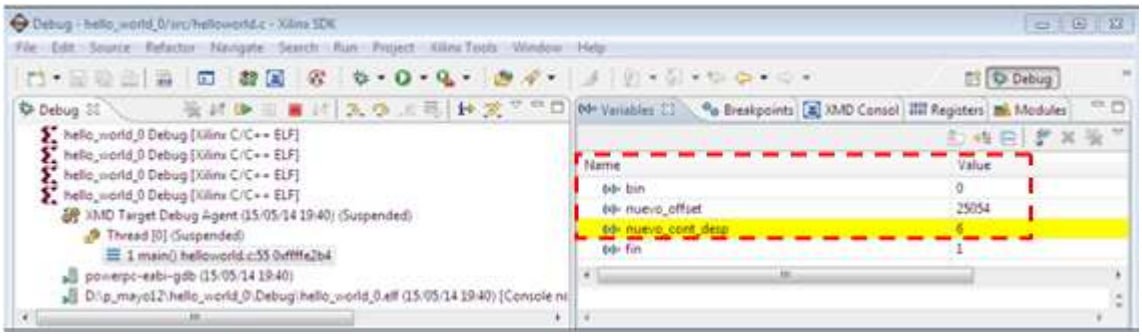

Figura 8 Resultados devueltos por el módulo hardware de decodificación de "bypass" y leídos por el procesador PowerPC

4. CONCLUSIONES

El diseño VHDL implementado contribuye a minimizar el tiempo de ejecución del algoritmo CABAC, puesto que se logró transformar las funcionalidades secuenciales de la función software de "bypass" a un modelo concurrente. Finalmente, se logró implementar un módulo IP que puede ser empleado en un sistema embebido por el microprocesador PowerPC para la decodificación de video H.264.

RECONOCIMIENTOS

El autor desea agradecer a los ingenieros Yosmany Sánchez, Yadira Alcantú Sampera y Orlando Landrove Gámez por su colaboración con los resultados obtenidos.

REFERENCIAS

1. Luthra, A., *MPEG-4 AVC/H.264 Digital Video Compression Standard.* Adv. Tech., Motorola Inc., 2006: p. 41
2. Richardson, I.E., *THE H.264 ADVANCED VIDEO COMPRESSION STANDARD,* 2010: p. 316.
3. Nasiopoulos, P., *H.264 - Overview.* 2010: p. 84.
4. Marpe, D., Heiko Schwarz y Thomas Wiegand *Context-Based Adaptive Binary Arithmetic Coding in the H.264/AVC Video Compression Standard.* IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY, 2003. **13**: p. 17.
5. UIT-T, *Recommendation ITU-T H.264, Advanced video coding for generic audiovisual services*, in *CABAC parsing process for slice data*,2012. p. 657.

SOBRE EL AUTOR

Graduado en el Instituto Superior Politécnico José Antonio Echeverría (ISPJAE) como ingeniero en Telecomunicaciones y Electrónica.