RIELAC, Vol.XXXV 3/2014 p.39-47 Septiembre - Diciembre ISSN: 1815-5928

RIELAC
Revista de Ingeniería Electrónica, Automática y Comunicaciones

IMPLEMENTACIÓN DE LOS PROTOCOLOS DE COMUNICACIÓN PARA VoIP: RTP y RTCP, SOBRE FPGAs

Nelia R. León González[1], Mario García Montoya[2], Víctor Marín Contreras[3], René Yañez de la Rivera[4]

1 Soluciones Integrales de Telecomunicaciones (SOLINTEL S.A.), Cuba, 2 FONDON Redes y Fluidos S.L., Cuba, 3 Centro de Investigaciones en Microelectrónica CUJAE, Cuba, 4 Dpto. de Telecomunicaciones CUJAE, Cuba

RESUMEN / *ABSTRACT*

En este trabajo se propone una solución de diseño e implementación de los protocolos RTP/RTCP sobre dispositivos lógicos programables, necesarios para la puesta en marcha de la tecnología Voz sobre IP. El diseño se basa en el lenguaje C++ y se recurre a las herramientas: NiosII IDE, Quartus y SOPC Builder, del fabricante Altera. Se utiliza el procesador Nios II embebido en un FPGA, con el sistema operativo en tiempo real MicroC/OS-II. La puesta a punto de la propuesta se implementa en la tarjeta "Nios II Development Kit", comprobando los resultados tanto con la consola del Nios II IDE, como con los periféricos del Kit. Se prueba la comunicación de dos PC convencionales, en las que se ejecuta la aplicación sobre Windows XP. Finalmente, se logra el envío de paquetes RTP y RTCP sobre UDP desde el FPGA a la PC, verificando la transmisión con la herramienta Wireshark.

Palabras claves: RTP, RTCP, Nios II, MicroC/OS-II, FPGA, C++.

An RTP/RTCP protocol design and implementation solution over programmable logic devices is proposed in this paper; those protocols are needed for Voice over IP technology start up. The design is based on C++ language and it employs the Altera tools: Nios II IDE, Quartus and SOPC Builder. Nios II embedded processor is used in an FPGA, with MicroC/OS-II real time operative system. The proposal tuning-up is implemented over the "Nios II Development Kit", which results are checked with the Nios II IDE console and with the Kit peripherals. The communication between two conventional PCs running the application over Windows XP is tested. Finally, RTP and RTCP packets over UDP are sent from the FPGA to the PC, checking the transmission with Wireshark tool.

Key words: RTP, RTCP, Nios II, MicroC/OS-II, FPGA, C++.

Implementation of VoIP Communication Protocols: RTP and RTCP over FPGAs

INTRODUCCIÓN

El presente trabajo forma parte del proyecto ramal "Plataforma de Conmutación de Paquetes" que se desarrolla con la colaboración de varias entidades del Ministerio de la Informática y las Comunicaciones (MIC). Dicho proyecto tiene como objetivos fundamentales: la implementación de un *Gateway* o pasarela de Voz sobre IP (VoIP, siglas en inglés) orientada al ámbito residencial, para operar en las Redes de Nueva Generación (NGN); y la fabricación de un producto comercializable que garantice soberanía tecnológica.

El estándar VoIP utiliza el protocolo UDP *(User Datagram Protocol)* para la transmisión de voz, pues aunque no ofrece integridad en los datos, el aprovechamiento del ancho de banda es mayor que con TCP *(Transfer Control Protocol)*. Esto implica que es necesario utilizar el protocolo RTP *(Real-time Transport Protocol)* que maneja los aspectos relativos a la temporización, marcando los paquetes UDP con la información necesaria para la correcta entrega de los mismos en recepción. Paralelamente, se requiere del protocolo de control de RTP (RTCP), que garantiza la calidad de servicio y porta información sobre los participantes de la sesión. Por otro lado, para establecer, modificar y terminar las sesiones de la llamada a través de Internet el protocolo más utilizado es SIP (*Session Initiation Protocol*) [1]. La implementación de cada uno de estos protocolos, unido a su conformación en paquetes IP y su posterior transmisión a través de Ethernet, conforman el denominado *Gateway*.

En la actualidad existen algunas implementaciones de estos protocolos desarrollados en ANSI C, C++, para aplicaciones de tiempo real sobre múltiples sistemas operativos [2, 3]. Sin embargo, para conformar un *Gateway* de VoIP se requiere contar también con los siguientes dispositivos: procesador digital de señales o procesador de paquetes, *Switch Ethernet*, procesador, memoria y dispositivos de interfaz del sistema. Estos elementos pueden tener interfaces incompatibles que necesitan ser interconectadas, por lo que requieren dispositivos adicionales, generalmente FPGAs (*Field Programmable Gate Arrays*) [4, 5].

Con el avance de los sistemas de nueva generación, los operadores de telecomunicaciones no solo buscan un mayor ancho de banda y mejor funcionalidad, sino también se persigue constantemente disminuir los costos y potencia consumida. Para los proveedores de servicio un criterio clave del éxito es adoptar plataformas de tecnología escalable, que permitan soportar mayores anchos de banda y densidad de servicio, siendo flexibles para incorporar mejoras futuras. Por estas razones se hace cada vez más ventajosa la utilización de dispositivos lógicos programables para los diseños, permitiendo, además, hacer modificaciones posteriores según los requerimientos de los operadores [6].

Por otra parte, la implementación de protocolos que requieren determinadas características de tiempo real exige mecanismos de procesamiento específicos, que no todos los CPUs pueden satisfacer. Para resolver este problema pueden utilizarse FPGAs que, además de la lógica programable, ofrecen núcleos de procesador escalable y dinámicamente cargable. Existen también núcleos de propiedad intelectual (*IP cores*) que pueden incluirse en el FPGA y enlazarse al procesador embebido [7].

Altera proporciona un diseño de referencia que desarrolla Video sobre IP, incluyendo el *hardware* que implementa UDP y encapsulamiento de datos de video en RTP de manera opcional. Utiliza el procesador embebido Nios II y la pila de protocolos IP bajo el sistema operativo en tiempo real eCos [8]. Por su lado, el fabricante Xilinx ofrece un diseño de referencia que implementa un emisor RTP de nivel muy básico [9]. Sin embargo, ninguno de estos implementa el protocolo RTCP.

Existen otros desarrollos de protocolos de comunicación en tiempo real, por ejemplo Ethernet desarrollado en VHDL (*VHSIC Hardware Description Language*) [10, 11] y RTP diseñado con Verilog [12].

Sobre la base de estos criterios, este trabajo tiene el objetivo de implementar los protocolos RTP y RTCP utilizando FPGAs, y comprobar que el diseño cumpla con los estándares establecidos, para una futura comunicación con equipos comerciales.

Para alcanzar los objetivos propuestos fue necesario realizar: la adaptación y desarrollo de un conjunto de clases que componen y verifican los diferentes paquetes RTP y RTCP, de forma tal de que fueran compatibles con el Nios II y el sistema operativo MicroC/OSII, y verificar su correcta codificación y decodificación en la consola del Nios II; establecer y comprobar la comunicación entre el procesador embebido y otros módulos del *Gateway* desarrollados en VHDL; el envío y recepción por socket de paquetes RTP y RTCP desde el Kit de desarrollo hacia la PC, en una primera fase utilizando el protocolo TCP y en una segunda fase el UDP; y el diseño de la sesión RTP, de acuerdo a la RFC 3550 [13], y su correspondiente transmisión por UDP entre dos PCs con Windows XP y entre el Kit y la PC. Los cuales son considerados los aportes fundamentales de este trabajo.

GENERALIDADES DEL DISEÑO

La realización de un *Gateway* tiene como propósito la integración de todas las funcionalidades que se exigen en un solo equipo, de manera tal que este sea capaz de funcionar con independencia de la PC. Para la implementación del diseño se cuenta con la tarjeta *"Nios II Development Kit"* de Altera, la cual tiene un FPGA EP2C35F672C5N, que contiene 33216 elementos lógicos y 483840 bits de memoria *"on-chip"*, entre otros recursos; que tentativamente puede abarcar el producto final. También se utilizó el procesador embebido Nios II, que posee una arquitectura RISC (Reduced Instruction Set Computing) de 32 bits [14].

Para elaborar la aplicación se utilizó el lenguaje C++, debido a su compatibilidad con C. Lenguaje en el que la especificación RTP/RTCP propone algunas implementaciones de algoritmos relacionados con funciones básicas del protocolo [13]. Este motivo, unido a que C++ es el lenguaje más utilizado para implementar protocolos como éste y que además existen compiladores para todas las plataformas, conduce a la decisión de su utilización.

El diseño de este programa está elaborado sobre la base del paradigma de programación orientado a objetos (POO). Lo que posibilitó que durante su realización se utilizaran conceptos avanzados de programación, tales como: encapsulamiento, herencia y polimorfismo. Vale destacar que estos dan una mayor complejidad al proyecto, pero brindan a su vez, facilidades asociadas con la optimización de los recursos y la disminución de los tiempos de modificación del código por su nivel de organización [15]. Contar con estas potencialidades fue posible gracias a la utilización del compilador para lenguaje C/C++ que acompaña a la plataforma

de desarrollo Altera *Nios II Integrated Development Enviroment* (IDE). Esta sólida herramienta de programación contiene el sistema operativo MicroC/OS-II, que proporciona a los diseñadores la capacidad de construir rápidamente aplicaciones que precisen de procesamiento en tiempo real [16].

Es válido destacar que el fabricante Altera proporciona una herramienta, conocida como Quartus, muy amigable para los diseños en lenguaje de descripción de hardware y captura esquemática. Conjuntamente, utilizando estas herramientas, es posible la realización híbrida de diseños *hardware* y *software*, para luego programarlos directamente en el FPGA.

MicroC/OS-II, por su parte, constituye un *kernel* de sistema operativo completamente portable, escalable, determinista y multitarea, escrito en ANSI C, que contiene una pequeña porción de código en lenguaje ensamblador, para adaptarse a diferentes arquitecturas de procesadores. Tiene la posibilidad de manejar hasta 63 tareas y ofrece la opción de uso de semáforos, eventos, exclusión mutua, mensajes, gestión de tareas y de memoria, según se necesite. Con todas estas condiciones, e incluyendo la pila TCP/IP que proporciona el Nios II Development Kit llamada *NicheStack*, se pudieron garantizar las condiciones requeridas para el establecimiento de la conexión del sistema elaborado con cada uno de los destinos a través de la red *Ethernet* [17].

Además, se utilizó la herramienta *SOPC Builder*, que permite conformar el procesador a la medida, especificando la cantidad de memoria, núcleos y periféricos requeridos, así como adicionar los módulos de hardware (implementación en VHDL) que serán parte del *Gateway*. Se utilizaron, por ejemplo, interfaces paralelas de entrada/salida (PIO), para realizar pruebas durante la etapa de diseño, por ejemplo: leds, lámparas 7 segmentos y botones [18, 19]. Se implementaron bloques para acceder a la interfaz de abonados mediante hardware, de la cual se adquiere la voz del abonado local y hacia la que se envía la del abonado remoto [20].

DESCRIPCION DE LA SOLUCION IMPLEMENTADA

El procesamiento de los diferentes canales de multimedia desde su entrada al sistema propuesto, hasta la salida hacia su destino en forma de flujo único, siguiendo con las normas que imponen los protocolos UDP, IP y Ethernet, puede ser subdividido en varios procesos, como se muestra en la Figura 1. Algunos de estos son ejecutados por la aplicación programada utilizando librerías de funciones que se facilitan por Altera y por los creadores del sistema operativo y de *NicheStack* [17].

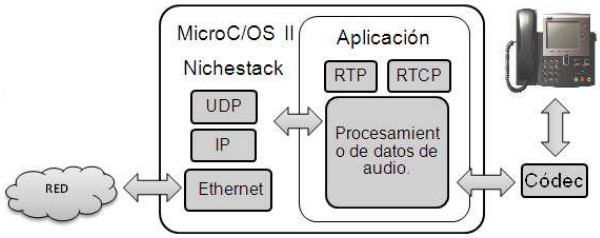


Figura 1. Utilización de las librerías de programa para el procesamiento de los datos desde la aplicación.

Una de las tareas más importantes en el diseño consiste en el procesamiento desde la entrada de la información de audio, sometida previamente a algún algoritmo de compresión de datos (como G729, G711 o LPC), hasta la obtención de un paquete RTP/RTCP. Para ello en el programa se utiliza una jerarquía de clases obtenidas de diversas fuentes que, como parte del trabajo, fue necesario adaptarlas a la plataforma Nios II y al MicroC/OS-II, para posibilitar la validación y procesamiento de los datos obtenidos, culminando sus funciones con la entrega de un paquete de carga útil con su respectivo encabezado, según lo que establece la RFC 3550 para el transporte de señales en tiempo real [13].

La segunda tarea de mayor peso, ocurre desde la formación del paquete UDP a partir de RTP/RTCP como carga útil, hasta la obtención de un paquete IP que pueda ser enviado al destino mediante una red *Ethernet* [21]. La Figura 2 muestra las etapas para conformar dichos paquetes. Esta labor se realiza con la vinculación al programa de funciones del sistema operativo en tiempo real y otras proporcionadas por la jerarquía antes mencionada, para agilizar el trabajo con la pila TCP/IP y hacer posible la conexión física del sistema a la red.

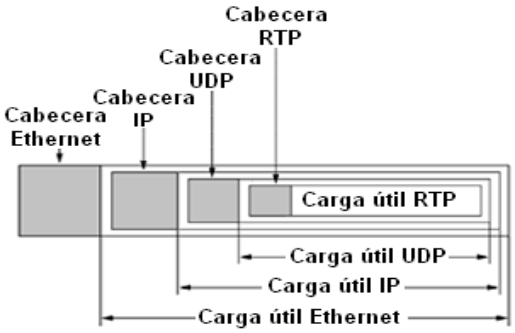


Figura 2. Estructura del paquete *Ethernet* partiendo de RTP.

FORMACIÓN DEL PAQUETE RTP/RTCP

Los protocolos RTP/RTCP, según propone la variante más actual de sus estándares, permite la transmisión de datos multimedia con características de tiempo real. Es por ello que necesita para su puesta a punto de un conjunto de operaciones dinámicas que requieren de recursos específicos, tanto del sistema operativo como del procesador.

Analizando el encabezado establecido para RTP se puede tener una idea general de algunas de estas operaciones que son implementadas en el programa propuesto. Dicho encabezado está compuesto por tres palabras de 32 bits y según corresponda, algunas extensiones formadas igualmente por 4 octetos, como se muestra en la Figura 3. En esta, el campo V (*Version*) de dos bits, indica la versión del protocolo RTP que se usa, la última definida en la RFC3550 es la número dos. El bit P (*Padding*) indica si hay relleno o no en el paquete. El bit X (Extensión) muestra si existe luego de la cabecera del paquete, una extensión de cabecera. Los cuatro bits que se representan como CC (Contadores de contribuyentes) muestran la cantidad de campos que presenta el paquete como identificadores de fuente de contribuyente.

<table>
<tr><td colspan="4">0..1 2 3 4..7</td><td colspan="2">8 9..15</td><td>16..31</td></tr>
<tr><td>V</td><td>P</td><td>X</td><td>CC</td><td>M</td><td>PT</td><td>Número de Secuencia</td></tr>
<tr><td colspan="7">Marca de Tiempo</td></tr>
<tr><td colspan="7">Identificador de fuente de sincronización</td></tr>
<tr><td colspan="7">Identificador de fuente de contribuyente
…</td></tr>
</table>

Figura 3. Encabezado del paquete RTP.

El bit M (*Marker*) es un marcador que se utiliza para propósitos muy variados y que puede tener usos definidos por el programador. Por último el campo PT (*Payload Type*) o tipo de carga útil es un número de siete bits que se utiliza para determinar el formato de la carga útil del paquete RTP. Entre estos formatos se encuentran la clasificación de audio y de video.
El resto de los campos resaltan sobre los anteriormente mencionados, si tenemos en cuenta el consumo de recursos de procesamiento utilizado en su obtención. Por este motivo se explican a continuación con un mayor detalle:

- Número de Secuencia: Contador de 16 bits que es incrementado con cada envío de un paquete RTP. Este número de secuencia es utilizado para permitirle al receptor de un flujo RTP detectar paquetes perdidos o que no lleguen en orden. El valor inicial de cada secuencia debe ser un número aleatorio impredecible y de esto depende en alto grado la seguridad en la transmisión de los datos.

- Marca de tiempo RTP: Valor de 32 bits que refleja el instante de tiempo en que fue muestreado el primer octeto del paquete RTP. La generación de este valor debe ser garantizado por un reloj que lo incremente de manera lineal. Este campo permite al receptor reproducir las muestras en los intervalos de tiempo apropiados y posibilita que diferentes flujos de datos de multimedia puedan ser sincronizados. También hace posible el cálculo de parámetros asociados con la calidad de servicio. El primer valor de la marca de tiempo de un flujo de datos de RTP debe ser generado también de forma aleatoria al igual que en el caso de los números de secuencia.
- El identificador de fuente de sincronización es un campo de 32 bits que identifica de manera única la fuente de un flujo RTP. Es diferente de la dirección IP o del número de puerto, que permite, por ejemplo, que un nodo con múltiples fuentes distinga cada una de ellas. Este identificador debe ser seleccionado aleatoriamente, con el objetivo de que dos fuentes de sincronización dentro de la misma sesión RTP no tengan el mismo SSRC (*Synchronization Source Identifier*). Aunque la probabilidad de que se repita el SSRC es muy baja, la implementación debe estar preparada para detectar y solucionar colisiones.

Por su parte, el protocolo de control del transporte en tiempo real (RTCP) tiene como función principal proporcionar la calidad de la distribución de los datos lograda por RTP, así como el control de flujo y congestión de la red. Permite supervisar la calidad de una sesión de llamada siguiendo la pérdida de paquetes, latencia (retraso) y otras preocupaciones claves de VoIP. Las especificaciones recomiendan que la fracción del ancho de banda de la sesión asignada a RTCP se fije en un cinco por ciento del tráfico de RTP, lo cual garantiza el diseño.
Existen diferentes tipos de paquetes RTCP para garantizar la variedad en el control de la información, estos son: RR (Reporte del Receptor), SR (Reporte del Emisor), SDES (Descripción de la Fuente), APP (funciones específicas de la aplicación) y BYE (fin de la participación).
A continuación se muestra un fragmento de la clase base para diferentes tipos de paquetes RTCP, de la cual heredan 6 clases que describen dichos tipos.

```
class RTCPPacket
{
public:
  /** Identifica los tipos específicos de paquetes RTCP. */
  enum PacketType
  {
      SR,    /**< Sender Report. */
      RR,    /**< Receiver Report. */
      SDES,  /**< Source Description. */
      BYE,   /**< Paquete BYE. */
      APP,   /**< Paquete RTCP que contiene datos específicos de la aplicación. */
      Unknown /**< No se reconoce el tipo de paquete RTCP. */
  };
```

Múltiples paquetes RTCP, de diferentes tipos, se concatenan en uno solo (compuesto) que se envía en un paquete acorde al protocolo de nivel más bajo (UDP). Los campos de mayor complejidad de algunos tipos de encabezado RTCP son:

- *Jitter* entre arribos: Estimado de la varianza estadística del tiempo entre arribos de paquetes RTP, medidos en unidades de marca de tiempo y expresados como un entero sin signo de 32 bits. Permite conocer la variación del retardo que existe entre la llegada de los paquetes y la duración de los mismos. La variación de este valor es un índice de la calidad del servicio con que se está estableciendo la comunicación.
- Fracción de paquetes perdidos: Fracción de paquetes de datos RTP perdidos, de una determinada fuente, desde que fue enviado el paquete RR o SR previo. Se expresa como número real y se define por el número de paquetes perdidos dividido por el número de paquetes esperados. Si la pérdida es negativa debido a duplicados, la fracción toma valor cero [13].

Cada una de estas operaciones se lleva a cabo por métodos independientes asociados a diferentes instancias de clases, que pueden ser reutilizados según se necesite.
Se tiene, por ejemplo, que la clase encargada de la generación de los números aleatorios facilita un conjunto de métodos, los cuales, acorde a la plataforma en la que esté corriendo la aplicación, garantiza la obtención de una semilla válida y por ende de una generación completamente caótica de números de 16 y 32 bits.
Por otra parte, tal y como se vio en algunos de los campos asociados a las cabeceras de RTP y RTCP, es imprescindible que exista una estricta atención a las variables de temporización. De su procesamiento depende, justamente, la efectividad de transmisión de

datos en tiempo real. Por este motivo, existe en la aplicación una clase que implementa el protocolo NTP (*Network Time Protocol*) y otras que permiten la atención a las variables de tiempo y la conversión entre diferentes formatos que describen este tipo de variables. Dichas clases se obtuvieron a partir de la modificación de otras con funciones similares que han sido desarrolladas por la comunidad de software libre y que fue necesario transformarlas teniendo en cuenta las particularidades de la plataforma donde se ejecuta la aplicación.

El proceso más importante dentro de la implementación de RTP/RTCP es la conformación de los paquetes en sí. Esta funcionalidad se hace posible por la colaboración de las clases que manejan los contenedores de memoria y las clases conformadoras de los paquetes.

Los contenedores de memoria son creados y procesados por una clase específica: *RTPMemoryObject*, que es la clase base abstracta de una jerarquía compuesta por dos niveles de herencia y un total de 19 clases. La misma permite que cada uno de los objetos instanciados se almacene de manera consistente en espacios de memoria hasta tanto dejen de ser necesarios. De esta forma, cada uno de los parámetros a utilizar en la creación de los paquetes forma parte independiente de un objeto almacenado y, una vez que se complete el proceso de elaboración de la información, se crea en memoria una instancia del objeto paquete RTP/RTCP y se llena cada uno de sus campos partiendo de los objetos previamente almacenados, los cuales posteriormente se destruyen.

Cada una de estas operaciones de asignación y liberación de espacios de memoria se realiza de manera dinámica por una clase cuyo objetivo fundamental es la gestión de los mismos. Dicha clase se concibió de manera diferenciada para adaptarla a cada una de las plataformas que puedan soportar a la aplicación. Esta gestión se ha realizado de manera eficiente para evitar la incorrecta utilización de las limitadas capacidades de memoria, que puedan aparecer en los entornos donde deberá correr el programa.

Una vez terminado el procesamiento de la carga útil y del resto de los parámetros que deben ser agregados a la cabecera, se conforma el paquete RTP/RTCP. Por último, se realiza un análisis del mismo para determinar otros valores a insertar como miembros de esa estructura y que dependen de la conformación específica. Con esto entonces ya se obtiene un resultado de acuerdo al estándar que pasará a ser completado con los requerimientos que imponen los protocolos de más bajo nivel.

Este proceso descrito en el sentido que tiene como dato de entrada el audio proveniente de la interfaz de abonados y como salida la interfaz a la red IP, funciona de manera análoga en la dirección inversa, gracias a que dichas opciones fueron concebidas en cada una de las clases antes mencionadas.

FORMACIÓN DEL PAQUETE *ETHERNET* Y CONEXIÓN A LA RED

La última fase a ejecutar por la aplicación se encarga de conformar con el paquete RTP/RTCP obtenido, un bloque de información que pueda ser enviado a través de la red *Ethernet* hacia su destino, cumpliendo además con los protocolos UDP e IP. Para ello, primeramente, se instancia una clase cuya función es construir la estructura de cada uno de los encabezados que se van a agregar. Esta misma clase controla, además, otros parámetros que deben tenerse en cuenta para establecer la conexión y que se agregan en la estructura antes mencionada.

Paralelamente existe una segunda clase cuya función principal es garantizar que se establezca una conexión física entre los terminales fuente y destino y además que se establezca un canal de comunicación entre dos puertos pertenecientes a cada uno de estos terminales, cuya identificación estará dada por una dirección IP. Esta clase está desarrollada, teniendo en cuenta el uso de las librerías de funciones para establecer comunicación entre terminales que pertenecen a una red, brindadas por el sistema operativo y que son conocidas como BSD *socket* API (*Berkeley Sockets Application Programming Interface*).

Entre las modificaciones realizadas vale destacar que fue agregado en las clases el soporte a aquellas conexiones que sean establecidas por los estándares IPV4 e IPV6, de forma tal que garantice la compatibilidad de la aplicación en actuales y futuras implementaciones.

Otro de los recursos que se utilizan del *NicheStack*, y que influye en el establecimiento directo de la conexión física, consiste en la posibilidad de que la aplicación adquiera su dirección IP a partir de un servidor DHCP (*Dynamic Host Configuration Protocol*) o en su defecto, que defina una dirección estática.

Todas estas funcionalidades posibilitan la interacción entre el diseño y la red existente, lo que hace posible que el flujo de datos transite por los diversos canales de comunicación y permita el establecimiento de la llamada con una calidad de servicio aceptable.

RESULTADOS Y DISCUSIÓN

Se utilizó el FPGA EP2C35F672C5N presente en el *Nios II Development Kit, Cyclone II Edition*, de Altera, empleándose un 37% de los recursos disponibles, de forma que existe virtualmente una reserva de espacio para otros bloques del *Gateway*, superior a la mitad del dispositivo.

Para comprobar el funcionamiento del diseño, se llevaron a cabo una serie de pruebas. Las primeras se realizaron con la consola del *Nios II IDE*, partiendo de paquetes tanto RTP como RTCP cuyos campos fueron confeccionados manualmente. Los mismos fueron procesados por las clases de validación de las estructuras correspondientes, y luego se mostraron los campos identificados, de manera independiente, en la consola de la herramienta, como se muestra en la Figura 4a y b.

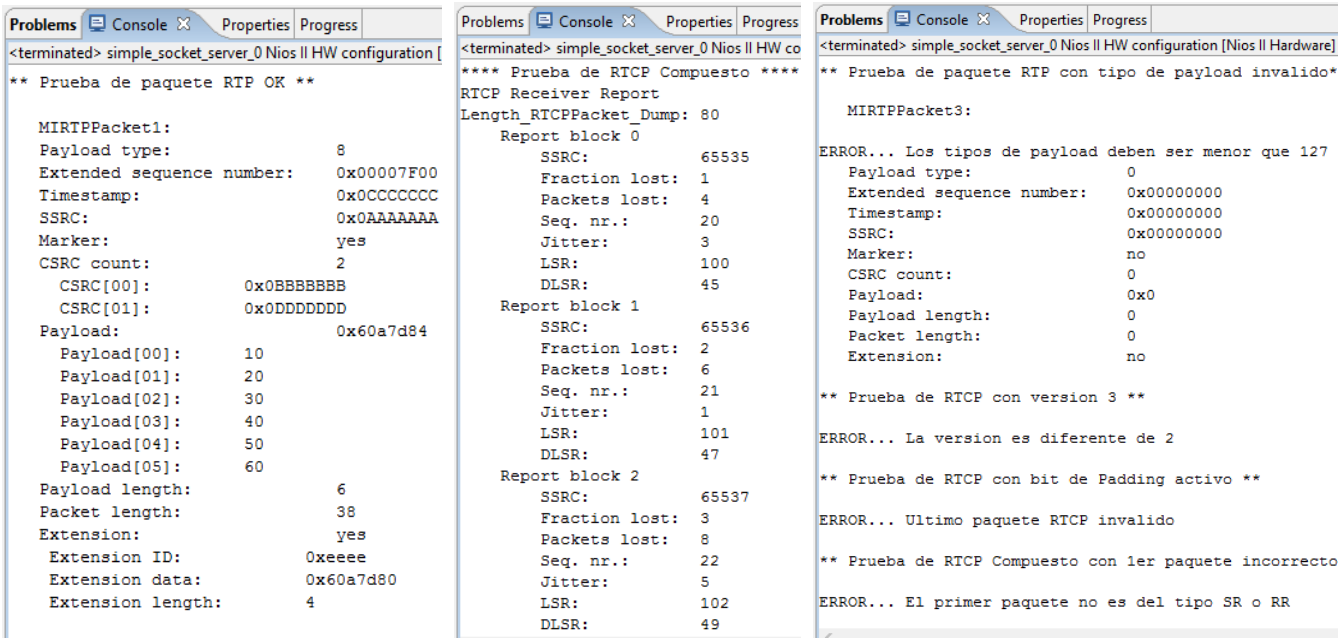


Figura 4 **a** **b** **c**

Validación de los distintos campos de paquetes RTP y RTCP.

De esta misma forma, se probaron disímiles paquetes con errores (elementos no permitidos por el estándar). Por ejemplo: con el tipo de carga útil inválido, con una versión diferente de dos, con bits de relleno (*Padding*) (activando el bit que lo indica y sin activarlo, colocando en el último octeto un número diferente de la cantidad que deben ser ignorados), como se observa en la Figura 4c. Se colocó el campo de longitud mayor o menor que la cantidad de bytes que posee, con un número de octetos que no es múltiplo de cuatro, y otros.

En el caso de RTCP, se tuvieron en cuenta casos como: un paquete compuesto cuyo primer tipo difiere de RR o SR (Figura 4c), un paquete RR con el contador de reportes menor o mayor que la cantidad que porta, un paquete BYE con el contador de SSRC diferente del número real que presenta el mismo, un paquete del tipo SDES sin el indicador de fin de la lista, entre otros [13].

Además, se sometieron a prueba del diseño paquetes reales provenientes de capturas de tráfico en la red.

En todos los casos anteriores se comprobó la eficacia del programa, el cual responde correctamente ante casos que no cumplen con lo establecido en el estándar y procesa adecuadamente los que sí están acorde con lo planteado en la RFC3550.

En una segunda etapa de pruebas, se analizó la interacción con los periféricos del *Kit* de desarrollo, comenzando con los LEDs, lámparas 7 segmentos y otros, como forma de indicar si la recepción de paquetes fue correcta, y qué tipo de error se encontró.

Por otra parte, se tuvo en cuenta cómo establecer la comunicación con la interfaz de abonados que suministra los flujos de audio ya codificados. Esto se hará posible mediante un conector físico que tiene el *Kit* de desarrollo y que posibilita el acceso al FPGA. Se comprobó el envío y recepción de datos entre el Nios II y los módulos desarrollados en VHDL independientes del procesador.

Unido a esto, y con el objetivo de comprobar cada uno de los módulos de programa que se elaboraron, así como la interacción entre las clases diseñadas, se probó una transmisión de un flujo RTP entre dos terminales interconectados por un elemento activo de la red. Luego se compiló la aplicación en cada una de estas PC que utilizaban Windows XP y se estableció una comunicación utilizando dicho protocolo. La aplicación además recibía como parámetros de entrada la IP asignada para el origen y el destino y los puertos a utilizar. Una vez establecida la conexión se mostraban los paquetes intercambiados. En todo momento se mantuvo una supervisión del canal utilizando *WireShark*, donde fue posible observar a fondo los paquetes de las distintas capas de la red que intervienen y su correcto manejo y conformación.

Por último, se probó la transmisión de paquetes RTP y RTCP (construidos por las clases del diseño) desde el FPGA a la PC, capturando la transmisión con el mencionado analizador de paquetes, como se muestra en la Figura 5.

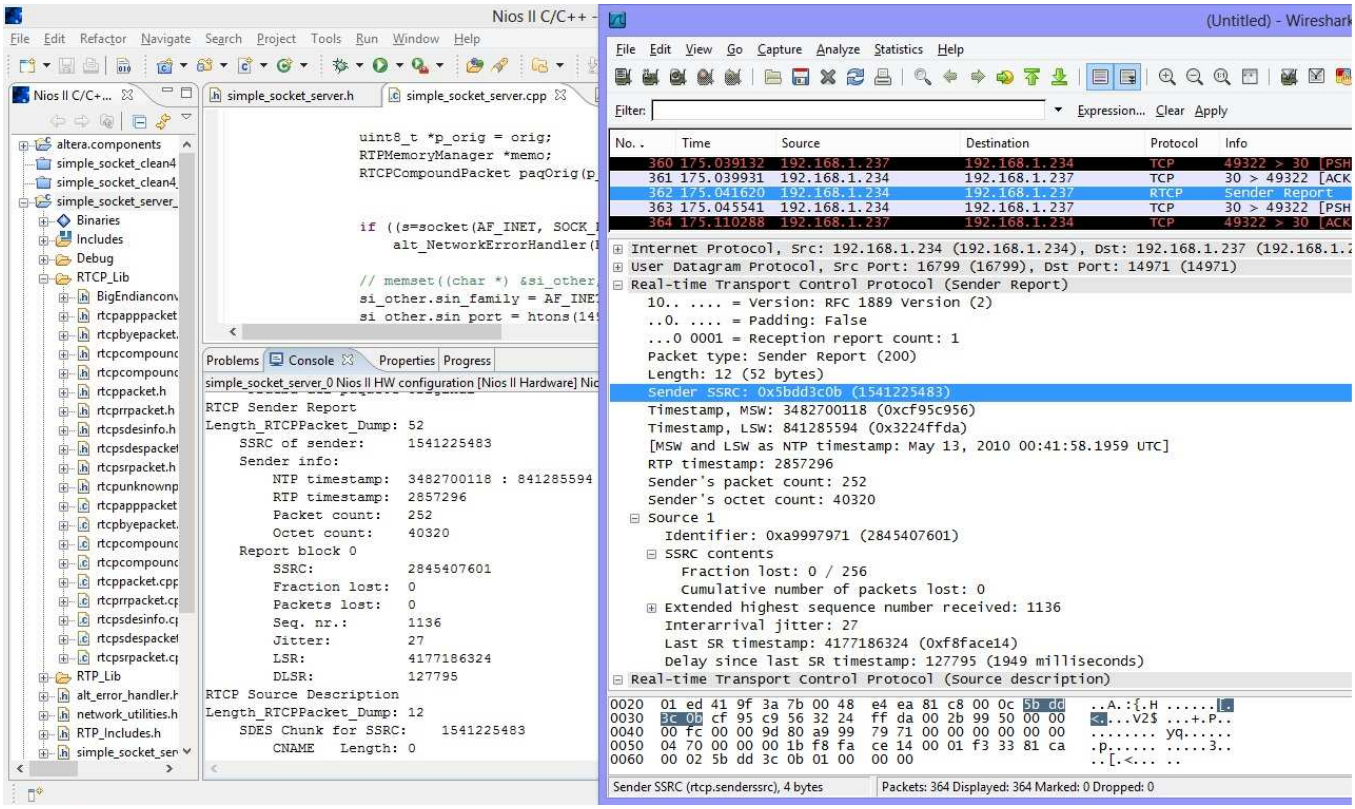
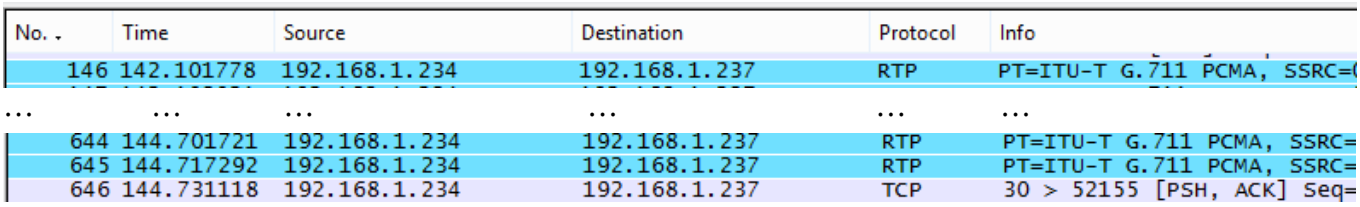


Figura 5. Envío de un paquete RTCP compuesto del FPGA a la PC.

En la Figura 5 se observa el envío de un paquete RTCP compuesto, mostrándose la decodificación de cada uno de los campos, con total coincidencia, en la consola del Nios II (ventana izquierda) y en el *Wireshark* (ventana derecha). En este último, puede verificarse que RTCP está encapsulado en UDP, y que el campo *Sender SSRC* (marcado en azul), coincide con el campo *SSRC of Sender* de la consola del Nios, y de esta manera pueden comprobarse el resto de los campos. La correcta identificación de los paquetes recibidos por esta herramienta de captura de tráfico prueba que el diseño cumple con los estándares de estos protocolos.

Adicionalmente, se simuló una sesión de VoIP, mediante la construcción y envío de 500 paquetes (el 5% de ellos eran RTCP y el resto RTP). En la Figura 6 se observa el primer mensaje RTP enviado (el número 146, en el segundo 142.101778 de esa captura) y el último (el número 645, en el segundo 144.717292), siendo la demora total: 2.62 segundos. Se enviaron 23 paquetes RTCP de 106 bytes y 477 paquetes RTP de 80 bytes cada uno. Estos datos nos permiten calcular la velocidad total de procesamiento y transmisión que es de 124 kbps, lo que garantiza el trabajo en tiempo real del sistema desarrollado. La verificación de que la totalidad de los paquetes enviados fue recibida sin reporte de errores, es un indicador de la calidad en la comunicación.



No.	Time	Source	Destination	Protocol	Info
146	142.101778	192.168.1.234	192.168.1.237	RTP	PT=ITU-T G.711 PCMA, SSRC=
...	...	...	...	...	...
644	144.701721	192.168.1.234	192.168.1.237	RTP	PT=ITU-T G.711 PCMA, SSRC=
645	144.717292	192.168.1.234	192.168.1.237	RTP	PT=ITU-T G.711 PCMA, SSRC=
646	144.731118	192.168.1.234	192.168.1.237	TCP	30 > 52155 [PSH, ACK] Seq=

Figura 6. Generación y envío de 500 paquetes RTP y RTCP.

ANÁLISIS ECONÓMICO

El costo inicial para la investigación y desarrollo del proyecto se corresponde con el precio de la tarjeta "Nios II Development Kit", que es de 647.15 USD. Sin embargo, el diseño conforma sólo una parte de la misma, por tanto su análisis económico está determinado por los principales componentes que se utilizan, ya que algunos serán integrados en un ASIC (*Application Specific Integrated Circuit*) a la hora de conformar el *Gateway* comercial:

- FPGA integrado (EP2C35F672C5N): 114.65 USD.
- Memoria de configuración (EPCS64): 37.37 USD.
- Tarjeta de 4 abonados desarrollada en el ICID: 60.00 USD (valor equivalente aproximado).

Por lo que el costo total del proyecto asciende a 212.02 USD, que es comparativamente inferior al costo total de un diseño como este, por ejemplo un *Gateway* de VoIP de 4 puertos de CISCO (SPA8800) cuyo valor actual es de 366.19 USD.

Además, el empleo de la tarjeta de desarrollo tributa a la reducción del ciclo de prueba pues no existirán demoras en el proceso de montaje de una maqueta del diseño realizado.

Por otra parte, el *know how* de este diseño constituye una fuente de incalculable valor agregado, que es muy superior al costo del hardware.

CONCLUSIONES

En este trabajo se realizó una implementación de los protocolos RTP y RTCP siguiendo estrictamente el estándar más actual (RFC 3550). Para esto se utilizó el 37% de los recursos de un FPGA EP2C35F672C5N de la familia *Cyclone* II de Altera, quedando suficientes recursos para desarrollar otros módulos del proyecto ramal.

Se evidenciaron algunas de las múltiples ventajas que este tipo de herramientas ofrecen a los diseñadores en la actualidad: la posibilidad de reutilizar el código, la versatilidad de los diseños, el aumento de las prestaciones de los dispositivos lógicos programables y sus precios más asequibles.

El uso de un esquema reconfigurable permite adaptar el diseño o incluirlo como parte de otro sistema más general, en este caso, para conformar un *Gateway* de acceso de VoIP.

El desarrollo e implementación de las librerías RTP y RTCP propiciaron la portabilidad del sistema, al hacerlo compatible con los sistemas operativos Windows y MicroC/OS-II.

Además, se comprobó el funcionamiento del sistema mediante la construcción y envío de paquetes desde el FPGA hacia la PC, demostrándose su operación exitosa con la correcta decodificación de la información transferida. Los tiempos de ejecución del procesamiento y transmisión de paquetes corroboraron la comunicación en tiempo real.

El desarrollo de este tipo de sistemas demuestra que es posible obtener productos que ofrecen soberanía tecnológica, siendo más económicos que otros equipos comerciales.

REFERENCIAS

1. **LINGFEN, S., JAMMEH, E.:** *Guide to Voice and Video over IP*. Springer-Verlag London (2013).
2. **RADVISION, AN AVAYA COMPANY:** *Advanced RTP/RTCP Toolkit for transferring real-time media over UDP/IP*. (2012)
3. **LIESENBORGS, J.:** *JVOIPLIB.* Disponible en http://research.edm.uhasselt.be (2009)
4. **ALTERA CORPORATION.:** *Increase Flexibility in Layer 2 Switches by Integrating Ethernet ASSP Functions Into FPGAs*. [En línea]. Disponible en http://www.altera.com (2006).
5. **ALTERA CORPORATION.:** *Using FPGA-Based Channel Bonding for HDTV Over DSL*. [En línea]. Disponible en http://www.altera.com (2008).
6. **ALTERA CORPORATION.:** *Implementing Next-Generation Passive Optical Network Designs with FPGAs.* [En línea]. Disponible en http://www.altera.com (2012).
7. **SCHLEGEL, C.:** *Implementation of an Industrial Real-time Ethernet interface, the universal approach using FPGAs*. IXXAT Automation GmbH (2009).
8. **ALTERA CORPORATION.:** *Altera Video Over IP reference design*. [En línea]. Disponible en http://www.altera.com (2008).
9. **VIZZUSI, S. XILINX CORPORATION.:** *Streaming an MP3 File Using RTP*. XAPP915 (v1.0) [En línea]. Disponible en http://www.xilinx.com (Abril, 2006)
10. **YOUDONG, C., CHUNXIANG, X.:** *FPGA Implementation of Real-Time Ethernet for Motion Control.* Hindawi Publishing Corporation, Article ID 682085. (2013)
11. **MOGHADDAMI, N., POURSHAKOUR, S.:** *FPGA implementation of Real-time Ethernet communication using RMII Interface*. Malardalen University, Vasteras, Sweden (2011).
12. **BROADBAND MONTHLY NEWSLETTER:** *Macnica Americas delivers 10G RTP video over IP FPGA.* Vol. 10 (Noviembre, 2011).
13. **SCHULZRINNE H. NETWORK WORKING GROUP.:** *RTP: A Transport Protocol for Real-Time Applications.* IETF RFC 3550 [En línea]. Disponible en http://tools.ietf.org/pdf (Julio, 2003)
14. **ALTERA CORPORATION.:** *Nios Development Board. Cyclone II Edition Reference Manual*. [En línea]. Disponible en http://www.altera.com (Mayo, 2007).
15. **DEITEL, H. M. , DEITEL, P. J.:** *Como Programar en C/C++*, 2da edicion, Editorial Prentice Hall. ISBN: 0-13-226119-7, Julio de 1998.
16. **ALTERA CORPORATION.:** *Using MicroC/OS-II RTOS with the Nios II Processor Tutorial.* [En línea]. Disponible en http://www.altera.com (2007)
17. **LABROSSE, J. J.:** *MicroC/OS-II. The Real Time Kernel*, 2da edición, CMP Books, San Francisco, California, 2002.
18. **ALTERA CORPORATION.:** *Nios II Development Kit. Getting Started User Guide.* [En línea]. Disponible en: http://www.altera.com (2007)
19. **ALTERA CORPORATION.:** *Embedded Peripherals IP. User Guide.* [En línea]. Disponible en http://www.altera.com (2010).
20. **MARIN, V., YAÑEZ, R.:** "Interfaz Digital de Abonados" en *Revista Ingeniería Eléctrica Automática y Comunicaciones.* [En línea]. Vol. XXVII No. 1. Disponible en: www.cujae.edu.cu/ediciones/RElectronica.asp (2007)
21. **TANENBAUM, A. S.:** *Computer Networks*. Epígrafe 6.4.3. 4ta Edición. Prentice Hall. USA (2004).

AUTORES

Nelia Rosa León González, es Ingeniera en Telecomunicaciones y Electrónica, graduada en la CUJAE en el curso 2008-09. Trabaja en la Subdirección de Soporte Técnico a Tecnologías Huawei, en la Empresa Soluciones Integrales de Telecomunicaciones (SOLINTEL S.A.). Su email es nelia.rosa@solintel.cu

Mario Garcia Montoya, es Ingeniero en Automática graduado en la CUJAE en el curso 2005-06. Trabaja como especialista en la Empresa FONDON, Redes y Fluidos S.L. Su email es mariocujae@gmail.com

Victor Marin Contreras, es graduado en la especialidad de Telecomunicaciones en la CUJAE durante el curso 1969-70 y ostenta actualmente el grado científico de Doctor en Ciencias Técnicas. Labora como Profesor Titular en el Centro de Investigaciones en Microelectrónica (CIME) desde su fundación. Su email es vmarin@electrica.cujae.edu.cu

Rene Yañez de la Rivera, se graduó en la especialidad de Telecomunicaciones en la CUJAE durante el curso 1971-72 y ostenta actualmente el grado científico de Doctor en Ciencias Técnicas. Labora como Profesor Titular en el Dpto de Telecomunicaciones de la Facultad de Ing. Eléctrica. Su email es reneyanezdelarivera@yahoo.com.mx