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/RTCP, sobre FPGAs de altera

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

1 Empresa Mixta GKT S.A., Cuba, 2 Centro de Investigaciones en Microelectrónica CUJAE, Cuba,

3 Dpto de Telecomunicaciones CUJAE, Cuba

RESUMEN / *ABSTRACT*

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 lenguaje C++ y se recurre a las herramientas: NiosII IDE, Quartus, SOPC Builder, entre otras, 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. Finalmente, se prueba la comunicación de dos PC convencionales, en las que se ejecuta la aplicación sobre Windows XP.

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; those protocols are needed for Voice over IP technology start up. The design is based on C++ language and it employs the Altera tools such as: Nios II IDE, Quartus and SOPC Builder. Nios II embedded processor is used in an FPGA, with the real time operative system MicroC/OS-II. 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. Finally, the communication between two conventional PCs running the application over Windows XP is tested.

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

Implementation of VoIP Communication Protocols over Altera's FPGAs

INTRODUCCIÓN

El presente trabajo forma parte del proyecto ramal "Plataforma de Conmutación de Paquetes" que se lleva a cabo en el departamento de Investigación y Desarrollo (I+D) de la empresa Gran Kaiman Teleco (GKT s.a.) apoyado por el Ministerio de la Informática y las Comunicaciones (MIC). Dicho proyecto tiene como objetivo fundamental la implementación de un *Gateway* o 04. Implementación de los protocolos de comunicacion para VoIP RTPRTCP, sobre FPGAs de altera.doc(NGN).

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)*. Por lo que se necesita 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. [1]
Se persigue, además, la fabricación de un producto que sea capaz de funcionar con independencia de la PC, comercializable, que garantice soberanía tecnológica. Es por ello que resulta muy factible para implementar el diseño la utilización de dispositivos lógicos programables. En este caso, se recurre a la tarjeta "*Nios II Development Kit*", del fabricante Altera.
Los objetivos de este trabajo son: implementar los protocolos RTP/RTCP, utilizando el FPGA *Cyclone II* que proporciona el Kit de desarrollo de Altera, y estandarizar el diseño, de manera que pueda ser compatible con las plataformas más utilizadas por la comunidad científica.

GENERALIDADES DEL DISEÑO

La realización de un *Gateway*, tiene como propósito la integración de todas las funcionalidades que se exigen en un solo equipo. Por lo tanto, para la implementación del diseño se eligió la tarjeta *"Nios II Development Kit"*, la cual cuenta con el FPGA EP2C35F672C5N, que contiene 33216 elementos lógicos y 483840 bits de memoria *"on-chip"*, entre otros recursos; que tentativamente puede abarcar el producto final. [2]

Se utilizó para elaborar la aplicación 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 [3]. 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). Esto posibilitó que durante su realización se utilizaran conceptos avanzados de programación, tales como: el de objeto, clase, encapsulamiento, polimorfismo y herencia. 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 [9]. 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).

Además, es válido destacar que el fabricante Altera proporciona herramientas de diseño muy amigables, como son: el *Quartus* (para los diseños en lenguaje de descripción de hardware y captura esquemática) y el ya mencionado ambiente de desarrollo integrado (IDE) de Nios II.

Este último, ofrece una sólida plataforma de desarrollo que 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. [4]

Es posible también, utilizando estas herramientas, 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 y 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 gestión de memoria según se necesite. Con todas estas condiciones e incluyendo la pila TCP/IP proporcionada dentro del 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*. [5]

Se utilizó la herramienta *SOPC Builder*, que permite conformar el procesador a la medida, especificando la cantidad de memoria, núcleos, y periféricos requeridos. Se utilizaron interfaces paralelas de entrada/salida (PIO), para realizar pruebas durante la etapa de diseño. Por ejemplo, los LEDs, lámparas 7 segmentos y botones [6, 7]. Se implementaron bloques para acceder a la interfaz de abonados que será conectada mediante hardware, de la cual se adquiere la voz del abonado local y hacia la que se envía la del abonado remoto. [8]

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 (Fig. 1).

Algunos de estos procesos 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*.[5]

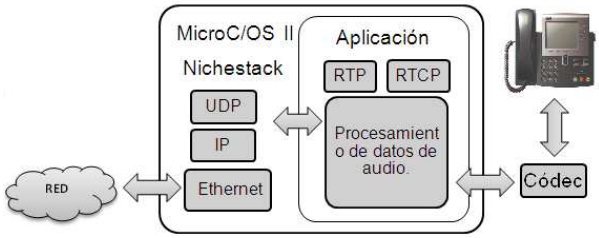

Figura 1. Utilización de las librerías de programa para el procesamiento de los datos desde la aplicación.

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, 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.[3]

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* (Fig. 2)[10]. 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.

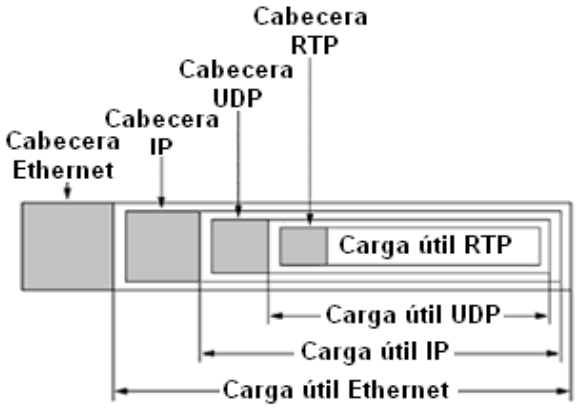

Figura 2. Estructura del paquete *Ethernet* partiendo de RTP.

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 (Fig.3).

En la Fig. 3 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="2">0..1</td><td>2</td><td>3</td><td>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 colspan="2">CC</td><td>M</td><td>PT</td><td>Numero de Secuencia</td></tr>
<tr><td colspan="8">Marca de Tiempo</td></tr>
<tr><td colspan="8">Identificador de fuente de sincronización</td></tr>
<tr><td colspan="8">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).
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). Se podrían analizar los campos de mayor complejidad de algunos tipos de encabezado RTCP:

- 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. [3]

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. 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 ser tenidos 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 influyen 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 (Fig.4).

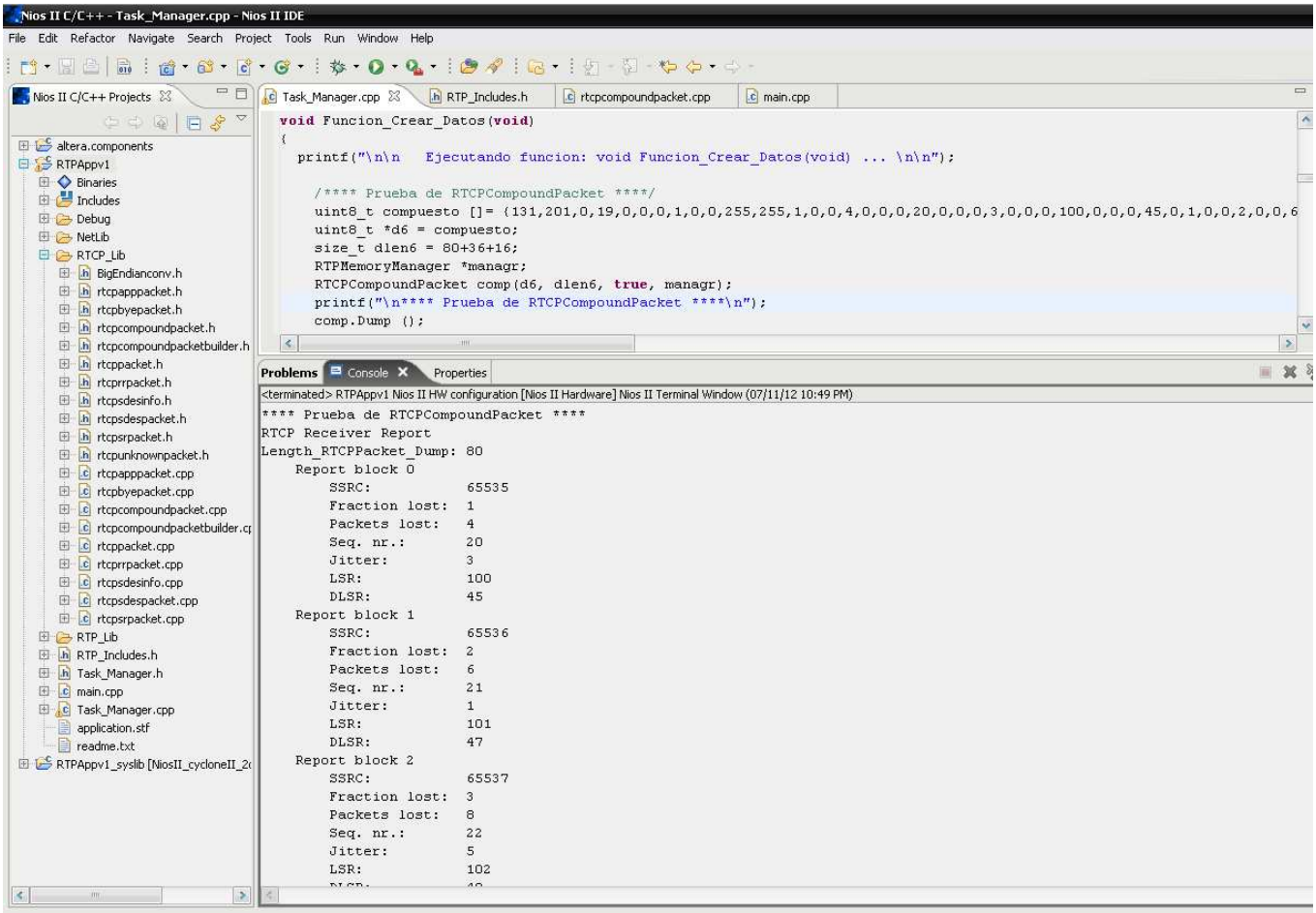

Figura 4. Validación de los distintos campos de un paquete RTCP.

De esta misma forma, se probaron disímiles paquetes con errores (elementos no permitidos por el estándar). Por ejemplo: con 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, 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), con una versión diferente de dos, y otros.

En el caso de RTCP, se tuvieron en cuenta casos como: un paquete compuesto cuyo primer tipo difiere de RR o SR, 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 SDES (*Source Description Message*) sin el indicador de fin de la lista, entre otros. [3]

Además se sometieron a prueba del diseño, paquetes reales, provenientes de capturas en la red, realizadas con la herramienta *WireShark*.

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 dispositivo *Cyclone* II y a la tarjeta en sí.

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. Estos estaban interconectados por un elemento activo de la red. Luego se compiló la aplicación en cada una de estas PC que utilizaban Windows XP como sistema operativo y se estableció una comunicación utilizando dicho protocolo.

Dicha 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.

CONCLUSIONES

Se obtuvo una implementación de los protocolos RTP y RTCP siguiendo estrictamente con el estándar más actual (RFC 3550). Se utilizó el 37% de los recursos de un FPGA EP2C35F672C5N de la familia *Cyclone* II de Altera.

El hecho de trabajar con 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.

Se logró portabilidad en el trabajo, al hacerlo compatible con varios sistemas operativos y plataformas de desarrollo.

La utilización de las herramientas de Altera, conjuntamente con el lenguaje de alto nivel C++, proporcionó agilidad y eficiencia en la etapa de diseño.

Se comprobó la veracidad del diseño utilizando herramientas *software* y comunicación en tiempo real.

REFERENCIAS

1. BLACK, U.: *Voice over IP*. Prentice Hall PTR, New Jersey USA, 1999.
2. ALTERA CORPORATION.: *Nios Development Board. Cyclone II Edition Reference Manual*. [En línea]. Disponible en http://www.altera.com (Mayo, 2007).
3. SCHULZRINNE H. NETWORK WORKING GROUP.: *RTP: A Transport Protocol for Real-Time Applications.* [En línea]. Disponible en http://tools.ietf.org/pdf (Julio, 2003)
4. ALTERA CORPORATION.: *Using MicroC/OS-II RTOS with the Nios II Processor Tutorial.* [En línea]. Disponible en http://www.altera.com (2007)
5. LABROSSE, J. J.: *MicroC/OS-II. The Real Time Kernel*, 2da edición, CMP Books, San Francisco, California, 2002.
6. ALTERA CORPORATION.: *Nios II Development Kit. Getting Started User Guide.* [En línea]. Disponible en: http://www.altera.com (2007)
7. ALTERA CORPORATION.: *Embedded Peripherals IP. User Guide.* [En línea]. Disponible en http://www.altera.com (2010).
8. 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)
9. DEITEL, H. M. , DEITEL, P. J.: *Como Programar en C/C++*, 2da edicion, Editorial Prentice Hall. ISBN: 0-13-226119-7, Julio de 1998.
10. TANENBAUM, A. S.: *Computer Networks*. Epígrafe 6.4.3. 4ta Edición. Prentice Hall. USA (2004).

AUTORES

Mario Garcia Montoya, es Ingeniero en Automática graduado en la CUJAE en el curso 2005-06 y trabaja como especialista en la Empresa Mixta GKT S. A. Su email es mayito@gkt.cu

Nelia Rosa León González, se graduó en la CUJAE en la especialidad de Telecomunicaciones en el curso 2008-09 y actualmente trabaja en la Subdirección de Soporte Técnico de la Empresa Mixta GKT S. A. Su email es nelia.rosa@gkt.cu

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, impartiendo asignaturas de la disciplina Electrónica y dirige un proyecto de colaboración con la Empresa Mixta GKT S. A. Su email es victor@gkt.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. Electrica, y desde el año 2003 ha sido asesor para las Telecomunicaciones en la Empresa Mixta GKT S. A. Su email es yanez@gkt.cu