Prototipado de un MMR Simple en una FPGA

Manel Canseco, José M. Claver, Germán León, Iván Vilata

***Resumen*— Los recientes avances en la tecnología en redes de alta velocidad han creado oportunidades para el desarrollo de aplicaciones multimedia caracterizadas por múltiples requerimientos de calidad de servicio (QoS). Uno de los elementos críticos en este tipo de redes es el encaminador, responsable de asegurar la QoS de éstas al aumentar su tamaño y el número de conexiones.**

Por otra parte, la lógica reconfigurable, basada en el uso de FPGA, ha sido potenciada por los avances tecnológicos en VLSI, la disponibilidad de recursos hardware y su programación con lenguajes de especificación de hardware de alto nivel. Esta tecnología permite el rápido diseño de un circuito hardware complejo, como es un encaminador, en una FPGA de última generación.

En este artículo, presentamos el estado actual del diseño de un encaminador multimedia simplificado en una FPGA mediante el uso del lenguaje Handel-C.

Palabras clave—**Redes de ordenadores, Calidad de Servicio, Sistemas en un Chip, FPGA, Handel-C.**

I. INTRODUCCIÓN

UNA de las tecnologías en alza en el campo de las redes de ordenadores es, sin duda, el desarrollo de mecanismos que soporten tráfico multimedia con calidad de servicio. El abaratamiento de los costes en las conexiones de alta velocidad junto al aumento de las aplicaciones multimedia a través de la red, está provocando unas necesidades de soporte todavía no satisfechas con los esquemas actuales.

El tráfico multimedia conlleva la transferencia de gran cantidad de información, requiere un alto ancho de banda y precisa un número elevado de conexiones. Sin embargo sus características más singulares, y a la vez más problemáticas, son una importante sensibilidad al *Jitter* y la necesidad de una latencia controlada. Este tráfico debe compartir el uso de la red con otro tipo de información que posee características diferentes. La capacidad de una red para garantizar los requerimientos de los distintos tráficos que circulan a través de ella se denomina "calidad de sevicio" (QoS) [9].

Son muchos los factores que influyen en el soporte de este tipo de servicios (diseño de protocolos, tamaño y topología de la red, ancho de banda, etc.) pero uno de los más determinantes es el diseño de los encaminadores, encargados de conducir la información. Como se puede observar en la Fig. 1, la relación entre el lento crecimiento de velocidad de procesamiento de los encaminadores y el cada vez más acelerado incremento en el ancho de banda de los enlaces provoca que los primeros se estén convirtiendo en el gran cuello de botella de la comunicación. Este desfase en la velocidad de los encaminadores es debido a su arquitectura.

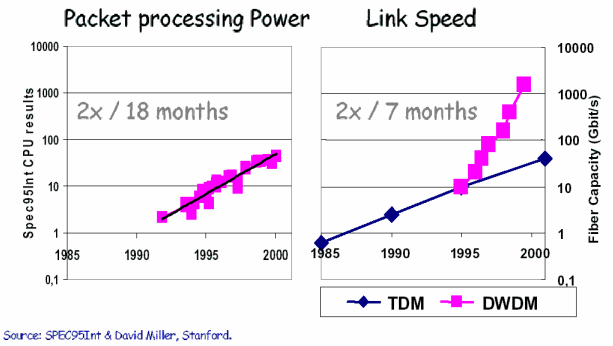

Fig. 1 Evolución de la relación entre la velocidad de procesado de los encaminadores y el ancho de banda de los enlace, bajo las técnicas de canalización TDM (multiplexado en el tiempo) y DWDM (multiplexado en la frecuencia en medio óptico)

De las tres clases de encaminadores que existen, este trabajo se centra en los corporativos. Para éste tipo de encaminadores se han propuesto varias arquitecturas: basadas en bus (con uno o varios procesadores), basadas en un conmutador con varios procesadores, etc. Entre estas propuestas destacan las basadas en conmutador con procesadores totalmente distribuidos. En esta arquitectura (ver Fig. 2) se distinguen tres elementos: los adaptadores de entrada (*Input Adapter*, *IA*) que realizan el desencapsulado y conversión entre el protocolo de la red y el interno del conmutador, el elemento de conmutación (*Switch Fabric*, *SF*) que distribuye la información procedente de los adaptadores de entrada hacia los adaptadores de salida, y los adaptadores de salida (*Output Adapter*, *OA*) que realizarán la función inversa a los adaptadores de entrada. El elemento de conmutación puede plantearse de diversas formas [2], aunque generalmente se ha concebido como un bus o como una memoria compartida. Existen otras alternativas como una red *crossbar* o una red *MIN*, que tienen un coste superior pero proporcionan una alta escalabilidad.

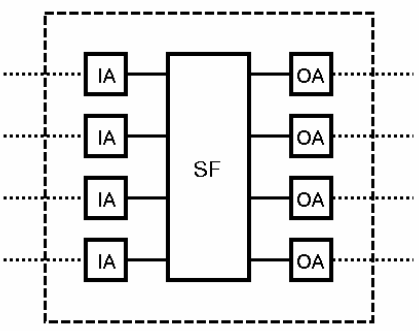

Fig. 2. Estructura básica de un encaminador basado en conmutador

Departamento de Ingeniería y Ciencia de los Computadores, Universitat Jaume I, Castellón (España). Correo electrónico: {canseco, claver, leon, vilata}@icc.uji.es. Este trabajo está subvencionado por el proyecto CICYT No TIC2003-08154-C06-06

Por otra parte, sabemos por la ley de Moore que el nivel de integración se duplica cada 18 meses a la vez que el coste de la lógica disminuye, por lo que no es descabellado pensar en la construcción de un encaminador en un único chip (SoC) o, al menos, integrar el elemento central del encaminador (SF).

En la actualidad existe un incremento en el uso de dispositivos reconfigurables para su utilización como coprocesadores o para el prototipado de sistemas hardware. Esta tecnología, basada en el uso de FPGA, ha sido potenciada por los avances tecnológicos en VLSI y la disponibilidad de recursos hardware.

[31]. La arquitectura de estas plataformas [20], y su programación [21], son objeto de muchos estudios que intentan encontrar configuraciones más eficientes.

El objetivo de nuestro trabajo es realizar una implementación de un encaminador basado en un modelo simplificado del *Multimedia Router* (*MMR*) [3]. Se trata de un encaminador con soporte de calidad de servicio (QoS) y una lógica suficientemente simple para caber dentro de una FPGA. Su finalidad es utilizarlo como herramienta de prototipado y experimentar nuevas alternativas de diseño, políticas y optimizaciones en el tiempo más reducido posible, obteniendo detalladas simulaciones próximas al funcionamiento real. Para ello, se utiliza en la programación de la FPGA un lenguaje de especificación hardware de alto nivel, el Handel-C [30].

El resto del trabajo se organizará como sigue. En la siguiente sección se realizará un estudio de las distintas opciones de diseño de encaminadores con QoS. En la tercera sección se presentará la arquitectura del encaminador MMR [3]. En la sección cuarta se plantean las modificaciones realizadas sobre el encaminador MMR para configurar el encaminador MMR Simple (SMMR) en una FPGA y las herramientas y lenguajes de diseño utilizados. En la sección 5 se comenta el estado actual del diseño del SMMR y sus características. Finalmente, en la última sección se expondrán las conclusiones más relevantes de este trabajo y se plantean los siguientes pasos a abordar.

II. Trabajo Previo

Esta sección se centrará en el estudio de las alternativas de diseño de un encaminador corporativo para la interconexión de redes de la misma o distinta arquitectura de red. Los objetivos a perseguir en esta clase de encaminadores son:

- Maximizar la **productividad** (*Throughput*) a una **alta velocidad** (similar o igual a la del enlace) para evitar que el encaminador sea el cuello de botella.
- Obtener un sistema **estable** (independiente del nivel de carga) y **equitativo** (en base al tipo de tráfico y sus necesidades) para garantizar las especificaciones de QoS.
- Realizar una **implementación simple** y eficiente que permita reducir costes, tamaño y consumo.

Existen distintas formas de distribuir las tareas entre los elementos que conforman el encaminador (Fig. 2). Un posible enfoque consiste en delegar al elemento de conmutación (SF), todas las tareas de encaminamiento, gestión de *buffers* y planificación, dejando únicamente a los adaptadores de entrada y salida (IA/OA) las tareas básicas de traducción. De este modo, el elemento de conmutación puede ser entendido como otro encaminador que opera con un protocolo (el propio interno) más adecuado para asegurar las características de QoS. Esta será la suposición que asumiremos.

Este encaminador tiene que realizar tres tareas fundamentales [17],

1. La resolución del encaminado y la admisión de éste.
2. La conmutación, que a su vez se divide en,
   2.1. La planificación y arbitraje del conmutador.
   2.2. La propia conmutación y transmisión de la información.
3. La gestión de los elementos de almacenamiento (*buffers*).

A partir de ellas, se pueden extraer dos tipos de parámetros del encaminador, cuantitativos y cualitativos, que se utilizarán en el estudio de su arquitectura [14]. Los parámetros cualitativos (técnicas de conmutación, organización del conmutador, algoritmos de encaminamiento, estrategia de control de admisión y reserva del ancho de banda, algoritmos de planificación y organización de los *buffers*) son los que definen la funcionalidad y características de cada uno de los elementos del encaminador. Los parámetros cuantitativos (tamaño de los mensajes, ancho de banda de los enlaces, grado del encaminador, número de canales virtuales (VC), frecuencia de funcionamiento, nº de *buffers*, etc.) son las variables a ajustar en base a la elección de los primeros. A continuación, se procederá a analizar cada uno de ellos.

A. Técnicas de Conmutación

Las técnicas de conmutación definen cómo se asignan los recursos, conectando los canales de entrada y salida [3][1]. Además, contienen implícito un mecanismo de control de flujo que permite la regulación del tráfico en la red. Existen dos clases de conmutación, la basada en conmutación de circuitos y la basada en conmutación de paquetes, utilizada frecuentemente en las redes actuales. Se han propuesto varias alternativas para solventar los problemas de baja utilización de los canales y alta latencia que adolece este modelo. La división de los paquetes de información en unidades encaminables más simples o *flits* (*flow units*) ha dado lugar a técnicas como el *Virtual Cut-Through* (VCT, [19]) y *Wormhole* [12]. La unión este concepto a la propuesta de Dally [10] del multiplexado virtual de los canales (canales virtuales) ha derivado en la posibilidad de desarrollar otras técnicas muy apropiadas para su uso en redes con QoS, como la Conmutación de Circuitos Segmentada (PCS, [13]).

B. Estrategias de Control de Admisión y Reserva de Ancho de Banda

Se pueden distinguir tres tipos básicos de tráfico: de mejor esfuerzo (*best effort* o ABR), de control y con QoS. Dentro del tráfico con QoS podemos distinguir, según su comportamiento [7], entre el de Caudal Constante (CBR), donde el tráfico se genera periódicamente a un ritmo constante, y el de Caudal Variable (VBR), generado en forma de ráfagas de distribución variable. Para parametrizar el ancho de banda requerido por los distintos tipos de tráfico, se puede dividir el espa-

cio/tiempo en particiones (rondas o *frames*) que contendrán un número determinado de *flits*.

Dado que los enlaces del encaminador serán compartidos por tráfico de distinto tipo, es necesario utilizar una política de asignación del ancho de banda y un mecanismo de control de admisión de la información entrante que garantice la adecuada QoS. El objetivo de estas estrategias es maximizar la utilización del canal, garantizando QoS y evitando la inanición del tráfico de mejor esfuerzo. Para ello se debe determinar la política de asignación a utilizar y la política de admisión a aplicar. La política de admisión varía en función del tipo de tráfico a tratar. En el caso de tráfico CBR, éste dependerá de un único parámetro, el ancho de banda a utilizar en cada ronda (*frame*). En el caso del tráfico VBR el ancho de banda puede variar en cada ronda, por lo que se aplica algún modelo estadístico para determinar los anchos de banda medio y pico, y relacionarlos con los ya reservados. El ancho de banda restante se otorgará al tráfico de mejor esfuerzo. La política de asignación en cada ronda se deriva de la de admisión. Así, en primer lugar se priorizan las conexiones CBR y VBR que no hayan consumido su ancho de banda medio, tras ellas las conexiones VBR hasta consumir su ancho de banda pico, y finalmente se asigna el tráfico de mejor esfuerzo y de control.

C. Algoritmos de Encaminamiento

La resolución del encaminado en el interior del encaminador suele ser una tarea más o menos sencilla a nivel de enlace (en comparación con otros protocolos de red como IP o ATM), siendo la resolución del camino la tarea más complicada a tratar. Existen muchas propuestas de algoritmos de encaminamiento, que pueden ser clasificadas por diversos criterios [1]: Centralizado o distribuido, determinista o adaptativo, etc. La utilización de un algoritmo centralizado conlleva mantener un conocimiento global de la información, lo que provoca un aumento en la lógica del encaminador. Dicha lógica podría extraerse del encaminador creando una unidad común de encaminado global, la cual podría simplificarse más si el algoritmo fuera determinista. Sin embargo, esta solución crea un cuello de botella lo que la hace muy poco escalable. Por ello, se suele elegir un algoritmo distribuido, que posee mayor escalabilidad, y con el objetivo de obtener simplicidad en la lógica a utilizar, sea además determinista. No obstante, si se consideran los objetivos de eficiencia y productividad, no cabe duda de la necesidad de un algoritmo adaptativo. Pero un algoritmo completa o parcialmente adaptativo puede presentar diversos problemas. Entre ellos el más importante es el de interbloqueo (sobretodo en redes irregulares). Para evitar este problema se han propuesto diversas soluciones [29], entre las que destaca *Backtracking*.

D. Organización del Conmutador

La arquitectura del conmutador es un tema crítico a la hora del abordar el diseño de un conmutador. Se parte del objetivo de obtener un encaminador de alta velocidad con una alta productividad, en rivalidad con la velocidad del enlace, que obliga a asegurar un *speed-up* que ronde la unidad. También se ha de considerar si se utilizaran canales virtuales.

Se han propuesto varias arquitecturas [2], de las que destacan las basadas en bus y las que utilizan una red conmutada o conmutador. Se suelen descartar las basadas en bus dada la contención que produce un multiplexado en el tiempo de la transmisión. Por lo tanto, aunque se incremente considerablemente la lógica, usualmente se escoge una arquitectura basada en conmutador. Existen varios tipos de redes conmutadas [27], entre las que podemos distinguir en primera instancia las bloqueantes (*Banyan, Delta*, etc.) de las no bloqueantes (*crossbar, Benês, Batcher Banyan*, etc.). La elegidas para formar suelen ser las no bloqueantes, destacando entre ellas la arquitectura en crossbar, por su relación prestación/coste cuando el número de elementos a conmutar es reducido.

Las arquitecturas en crossbar con canales virtuales pueden organizarse de tres maneras: completamente demultiplexada, parcialmente demultiplexada y totalmente multiplexada. Debido al coste y complejidad que conllevan, las dos primeras opciones suelen descartarse, adoptándose la última forma. Sin embargo, si se plantea un diseño con pocos canales virtuales, el segundo esquema será una opción a tener en cuenta.

E. Organización de la memoria

La organización de los almacenamientos intermedios es un factor muy importante, pues condiciona la elección del planificador del conmutador. Las organizaciones más destacadas [17] son: el encolado a la entrada (*input queueing*), *input smoothing*, encolado a la salida (*output queueing*) y almacenamiento totalmente compartido (*completely shared buffering*). Ninguna de las propuestas anteriores es ideal, puesto que todas presentan problemas. En almacenamiento totalmente compartido y encolado a la salida, el conmutador debe funcionar más rápido que el puerto de salida. En *input smoothing*, el demultiplexado hace encarecer la lógica del conmutador e incrementar el retardo. Finalmente, el encolado a la entrada presenta el problema de bloqueo de cabecera (HOL) que limita su productividad al 58'6% [18]. Aplicando mejoras a algunas de ellas se han conseguido resultados mejores, a destacar las aplicadas sobre encolado a la entrada, como *windowing*, expansión del conmutador (SE), incremento de la velocidad del conmutador, agrupado de canales de salida (OCG) y encolado virtual de salida (VOQ, [17]) .

Otro factor determinante es la tecnología utilizada en los elementos de almacenamiento y su estrategia de gestión, que se convierte en el gran cuello de botella en el funcionamiento del conmutador. Si se utiliza DRAM se obtiene gran densidad de almacenamiento, aunque se sacrifica velocidad y se pierde el acceso concurrente aleatorio (solucionable, en parte, con el uso de DRAM entrelazada). Al utilizar SRAM se gana en velocidad y aleatoriedad de acceso, pero se pierde en capacidad de almacenado debido a su elevado coste. Se han propuesto diversos modelos basados en el jerarquizado, organización, optimización e incluso paralelización, como en el *Tiny Tera* [24], para equilibrar ambos parámetros.

F. Algoritmos de Planificación.

La planificación de la conexión de los canales de entrada con los de salida, determinará la productividad y el comportamiento de QoS del encaminador. Los algoritmos de planificación propuestos recientemente [17][7] garantizan resultados de productividad aceptables [26]. Muchos de ellos han sido mejorados, como el eSLIP [25], hasta alcanzar tasas cercanas al 100% de productividad teniendo en cuenta la estabilidad (OCF) [23]. No obstante, no garantizan en modo alguno el cumplimiento de los requisitos de QoS. Luego, se precisa de algoritmos que garanticen la QoS y que consigan el máximo de productividad. Además, si se utiliza VOQ existen múltiples *flits* candidatos asociados a una misma entrada, lo que conlleva a tener en cuenta no sólo los conflictos de salida, sino también los derivados de las diversas colas de entrada. La estrategia habitual es realizar una planificación previa para priorizar la elección del candidato en caso de empate para la misma entrada, denominada planificación de enlace. Posteriormente se aplica la planificación a nivel de conmutador que finalmente realiza los emparejamientos.

En la planificación a nivel de enlace se suele utilizar algún algoritmo basado en prioridades que tome como base los parámetros principales de la QoS: el *jitter* o el retardo [6][22], bien orientados al ancho de banda reservado, a la espera acumulada, o a una combinación de ambos. De la misma forma, en la planificación del conmutador se precisa de un algoritmo que garantice los requerimientos de QoS e intente maximizar, en la medida de lo posible la productividad en la salida [3][14].

III. EL ENCAMINADOR MMR

El MMR [3][14] es un modelo de encaminador con encolado de entrada (IQ) concebido para redes con tráfico multimedia en entornos LAN, que garantiza QoS para tráfico combinado CBR y VBR, y en presencia de mejor esfuerzo (ver Fig.3). Para el tráfico con QoS utiliza la técnica de conmutación híbrida denominada PCS [13], que adopta un control de flujo basado en créditos, y VCT para el tráfico de mejor esfuerzo.

El MMR puede gestionar gran número de conexiones con QoS simultáneas debido al elevado número de canales virtuales por puerto (entre 64 y 128). El encaminamiento se lleva acabo mediante un algoritmo adaptativo basado en *Backtraking*, denominado *Exhaustive Profitable Backtraking* (EBP) [15]. La organización de su conmutador es de tipo *crossbar* multiplexado y la planificación se realiza a dos niveles. A nivel de enlace utiliza un algoritmo basado en prioridades denominado IABP que relaciona el ancho de banda con el retardo [6][22]. Este ha sido posteriormente optimizado y simplificado, SIABP (*Simple*-IAB) [4]. El conmutador recibe de cada enlace un vector de N candidatos ordenado por prioridad, y en base a esta información se plantea un esquema en el que equilibra dos parámetros, el nivel de prioridad de los candidatos de cada enlace y el grado de conflicto acontecido en cada una de las salidas. A partir de este esquema se proponen dos algoritmos dependiendo de qué parámetro sea el más prioritario. Basándose en el nivel de los candidatos surge el algoritmo COA (*Candidate Order Arbiter*, [5]), y basándose en el grado de conflictos se propone el algoritmo CCA (*Candidate Conflict Arbiter*).

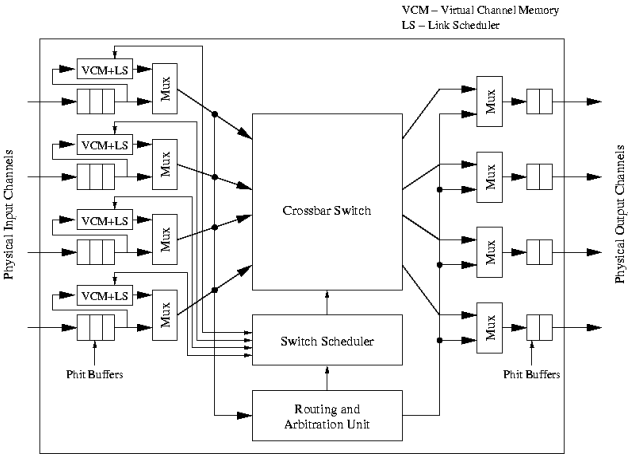

Fig. 3. Esquema del encaminador multimedia MMR.

IV. EL ENCAMINADOR SMMR

El encaminador MMR Simple o SMMR se concibe como una adaptación del modelo MMR con los siguientes objetivos:

- Reducir el coste material y de diseño. Utilizar la mínima lógica necesaria para conseguir las prestaciones deseadas, sin exceder el tiempo invertido en la implementación.
- Aumentar el grado de independencia del dispositivo. Enfocar al SMMR como un dispositivo autónomo.
- Incrementar la velocidad de trabajo. Detectar, analizar y optimizar las rutas críticas incrementando así la frecuencia máxima de trabajo.
- Ajustar la relación entre garantías de QoS y productividad, tratando de maximizar ambas.
- Orientado a la reconfiguración y prototipado. Experimentar nuevas alternativas de diseño, políticas y optimizaciones.

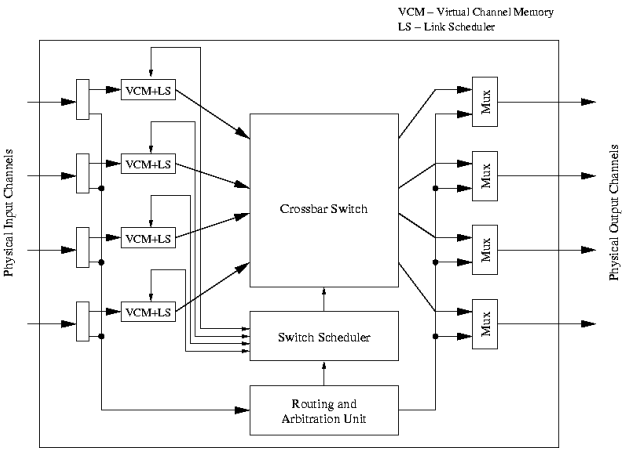

Fig. 4. Esquema del encaminador MMR simplificado (SMMR).

Principalmente se plantean las siguientes diferencias respecto del MMR:

- Pensado como componente hardware, sin interfaces de red (aumenta la independencia del dispositivo).
- Se reduce del número de canales virtuales: Uno de los factores fundamentales es el ajuste del número de canales virtuales por enlace. El número de canales virtuales está íntimamente relacionado con la lógica a emplear y la densidad de almacenamiento precisada. A mayor número de canales virtuales es necesaria mayor lógica para su tratamiento, que estará relacionada con el tiempo empleado en la planificación. Esto puede forzar a aumentar el tamaño del *flit*, lo que implica un aumento del tamaño de los *buffers*. Si además se ha incrementado el número de estos, la densidad de información a almacenar crece considerablemente. En base a varias propuestas, como la red Infiniband [28], se han fijado 16 canales virtuales por puerto (el primero para tráfico de mejor esfuerzo y los restantes para tráfico QoS).
- Se resta importancia a los paquetes de control (reducción del coste, aumento de la productividad). Dado que el porcentaje de dichos paquetes es muy reducido en relación con el tráfico total y que no son sensibles al retardo, no es necesario mantener una conexión QoS reservada para ellos. La solución planteada es la utilización de un bus que compartirá el intercambio de paquetes de control y créditos de forma multiplexada.
- Altamente parametrizable y escalable (orientado al prototipado). Existe gran facilidad para configurar los parámetros cuantitativos (nº de puertos, nº de canales virtuales, tamaño de *flit*, etc.) y cualitativos (cualquiera de los componentes) del encaminador.

Respecto a la implementación se destacan los cambios en:

- El cálculo de la prioridad en la planificación de enlace. El módulo SIABP ([3], Pág. 97) incrementa la prioridad desplazándola a la izquierda aumentándola en potencias de dos (pe. 0..0101 => 0..1010). Esto plantea un problema si los incrementos de prioridad exceden del tamaño del contador de prioridad, provocando una disminución de ésta (pe, 1010..0 => 0100..0). Puesto que se considera un número limitado de prioridades (alta, media y baja), se propone codificarlas en notación de temperatura. Así, al llegar a la misma situación, el contador de temperatura permanecerá estable con la máxima prioridad (1..1).
- El tratamiento de los paquetes de control. En el MMR ([3], Pág. 83) se propone enviar los paquetes de control durante el ciclo de reconfiguración. Sin embargo, esto es imposible dado que el tamaño máximo de información a enviar en ese periodo (un *phit*) es inferior a la longitud de los paquetes de control. Así pues, se propone enviarlos durante el ciclo de *flit*, tratándolos como tráfico ABR.

El prototipo del SMMR se ha desarrollado sobre una tarjeta RC1000 de Celoxica [8] que posee, además de una Virtex 2000E de Xilinx [32] (con dos millones de puertas equivalentes), 8 Mbytes de memoria SRAM dividida en cuatro bancos, y dos tarjetas PCI Mezzanine (PMC) para la entrada/salida con el exterior. La FPGA Virtex-2000E contiene 38.400 LUT y 640 Kbits de BRAM.

Para el diseño físico de un encaminador de la magnitud del SMMR, se ha utilizado un lenguaje de alto nivel, el Handel-C [30]. Se trata de un lenguaje de descripción del hardware derivado del C al que se le han incorporado conceptos provenientes del modelo CSP de Hoare, más concretamente del lenguaje Occam. En particular, posee sentencias para indicar el paralelismo intrínseco de los circuitos hardware y expresar el sincronismo y la transferencia de información entre las distintas partes del circuito a través de canales. Además, utiliza tipos de datos estándar con tamaños de bit definidos por el usuario, siendo muy eficiente en el uso de los recursos de la FPGA.

El entorno de desarrollo utilizado es el DK1 de Celoxica, que posee un entorno de edición, compilación, simulación y síntesis. El código generado se expresa en formato EDIF o VHDL, que es finalmente emplazado en la FPGA mediante las herramientas del fabricante.

V. Resultados y estado actual

Actualmente el proyecto se encuentra en fase de implementación. En la tabla I se presentan los módulos desarrollados junto con el área ocupada y la latencia crítica de funcionamiento. Los resultados presentados corresponden a un encaminador con 4 puertos y 16 canales virtuales por puerto, con un tamaño de *flit* de 64 *phits*, siendo 16 bits el tamaño de un *phit*. El tamaño de memoria empleado en las colas es de 64 Kbits de BRAM.

TABLA I

Módulos Desarrollados del SMMR, mostrando su área y su latencia.

Módulo / Componente	LUT	Latencia
Módulo de Entrada.		
PktSep	45	3’71 ns.
Memoria de Canales Virtuales (16 Kbits de BRAM)		
VCM	351	3’99 ns.
Módulo de variación de la prioridad (SIABP [4] modificado)		
SIABP (1 unidad)	144	3’40 ns.
Módulo de Ordenación		
SORT	198	3’98 ns.
Planificador de Enlace		
LnkSched	3743	4’69 ns.
Planificador del Conmutador (tipo COA [34], requiere menos líneas y permite mayor segmentación).		
SwSched	1143	3’98 ns.
Conmutador (Crossbar por enmascaramiento)		
XBAR	254	2’56 ns.

En la tabla II se muestra una estimación del área ocupada para diversas configuraciones del encaminador SMMR. No se ha incluido el incremento de memoria BRAM, puesto que ésta crece linealmente en relación a los canales virtuales empleados.

TABLA II

ESTIMACIÓN DE ÁREA OCUPADA EN LUT PARA DIFERENTES CONFIGURACIONES DE SMMR

Config.	Total	Conmutador / Planificador	1 Puerto
4 Puertos 16 CV	16.835	1.071	3.941
4 Puertos 32 CV	38.216	1.071	9.104
8 Puertos 16 CV	36.027	3.352	3.941
8 Puertos 32 CV	77.946	3.352	9.104

En base a los datos de la Tabla II se concluye que:

- Para la configuración de 4 puertos con 16 canales virtuales se obtiene un 43'84 % de ocupación de la Virtex 2000E, lo que hace viable su implementación. Las restantes configuraciones no se pueden implementar en esta FPGA ya que a partir del 70 % de ocupación surgen problemas de enrutado.
- Existe muy bajo acoplamiento entre los puertos y el núcleo del conmutador.

Analizando las latencias críticas de funcionamiento, se estima que la frecuencia de trabajo del encaminador implementado es de 200 MHz. Considerando un *speed-up* de 1 respecto del enlace y un ancho de palabra de 16 bits por puerto, se obtiene una velocidad pico de 3'2 Gbits/s por enlace.

Actualmente se está acabando de implementar el circuito generador/monitor. En esta primera versión del generador sólo producirá tráfico CBR, aunque está previsto incorporar mejor esfuerzo y tráfico VBR. El modelo de tráfico generado es sintético, y a diferencia de otros modelos de tráfico preconfigurado como *Backto-Back* (BB) o *Smooth-Rate* (SR) [4][6], existe control de flujo en la comunicación.

VI. CONCLUSIONES

En este artículo se han analizado las distintas opciones de diseño de los encaminadores para entornos multimedia y se ha presentado el estado actual de una propuesta de diseño de un encaminador multimedia simple (SMMR) en una FPGA a partir del modelo del encaminador MMR. El diseño de este encaminador ha sido pensado como herramienta de prototipado de encaminadores "*on chip*" y supone una apuesta por la utilización de lenguajes de especificación hardware de alto nivel para la programación de las FPGA, abstrayendo y acelerando el proceso de diseño de sistemas complejos como el de un encaminador.

A partir de los resultados obtenidos, se puede comprobar que el modelo se puede emplazar dentro de una FPGA Virtex 2000E, logrando un área de ocupación razonable. En base a las estimaciones realizadas, se prevé conseguir unas altas prestaciones (3'2 Gb/s por puerto), a un coste espacial relativamente aceptable. Una vez terminado el circuito generador/monitor de tráfico podremos comprobar el comportamiento del encaminador SMMR.

VII. REFERENCIAS

[1] F. Ashraf, M. Abd-El-Barr, K. Al-Tawil. "Introduction to Routing in Multicomputer Networks". Computer Architecture News, 26 (5)
[2] J. Aweya. "IP Router Architectures: An Overview", Journal of Systems Architecture 46 (2000) pp.483-511, 1999.
[3] M.B. Caminero. "Diseño de un Encaminador Orientado a Tráfico Multimedia en Entornos LAN". Tesis doctoral, Albacete, Junio 2002.
[4] M.B. Caminero, C. Carrión, F.J. Quiles, J. Duato, S. Yamalanchili. "A Cost-Effective Hardware Link Scheduling Algorithm for the Multimedia Router (MMR)". Lecture Notes in Computer Science: Networking – ICN'01, 2001.
[5] M.B. Caminero, C. Carrión, F.J. Quiles, J. Duato, S. Yamalanchili. "Investigating switch scheduling algorithms to support QoS in the Multimedia Router (MMR)". Proceedings - Workshop on Communication Architecture for Clusters (CAC'02). IEEE Computer Society, April 2002.
[6] M.B. Caminero, C. Carrión, F.J. Quiles, J. Duato, S. Yamalanchili. "Tunning Buffer Size in Multimedia Router (MMR)". Workshop on Communication Architecture for Clusters CAC'01.
[7] M.B. Caminero, F.J. Quiles. "Técnicas de Planificación de Conmutadores Orientadas a Garantizar QoS a Tráfico Multimedia". Technical Report UCLM, 2000.
[8] Celoxica. http://www.celoxica.com/methodology/c2fpga.asp
[9] Cisco. "Quality of Service (QoS) Networking". http://www.cisco.com/univercd/cc/td/doc/cisintwk/ito_doc/qos.htm
[10] W.J. Dally. "Virtual-channel flow control". IEEE Transactions on Parallel an Distributed Computing.
[11] W.J. Dally et al. "The J-machine: A fine grain concurrent computer". Proceedings - Hot Interconnects II.
[12] W.J. Dally, C.L. Seitz. "The Torus routing chip". Journal of Distributed Compunting.
[13] B.V. Dao, J. Duato, S. Yamalanchili. "Configurable flow control mechanisms for fault-tolerant routing". Proceedings - 22nd International Symposium on Computer Architecture (ISCA), Jun 1995.
[14] J. Duato, S. Yalamanchili, M.B. Caminero, D. Love, F.J. Quiles. "MMR: A High-Performance Multimedia Router-Architecture and Design Trade-Offs". Fifth International Symposium on High Performance Computer Architecture (HPCA-5).
[15] P.T. Gaughan, S. Yalamanchili. "Adaptative routing protocols for direct multiprocessor networks". IEEE Computer , May 1993.
[16] P.T. Gaughan, S. Yalamanchili. "Pipelined Circuit-Switching: A fault-tolerant variant of wormhole routing" Proceedings - International Conference in Network Protocols.
[17] P. Gupta. "Scheduling in Input Queued Switches: A Survey". Unpublished Manuscripts
[18] M.J. Karol et al. "Input versus output queueing on a space-division packet switch". IEEE Transactions on Communications.
[19] P. Kermani, L. Kleinrock. "Virtual-Cut Through: A new computer communication switching technique". Computer Networks.
[20] C. Kozyrakis, D. Patterson. "A new direction for computer architecture research". IEEE Computer, 33 (11), pp. 24-32, 1998.
[21] E. Lee. "What's ahead for embedded software". IEEE Computer, 33 (9), pp. 18-26, 2000.
[22] D. Love, S. Yamalanchili, J. Duato, M.B. Caminero, F.J. Quiles. "Switch Scheduling in Multimedia Router (MMR)". International Parallel and Distributed Processing Symposium 2000.
[23] N. Mckeown, A. Mekkittikul, V. Anantharaman, J. Walrand. "Achieving 100% Througput in an Input Queued Switch". IEEE Transactions on Communications, Vol.47, No.8, August 1999.
[24] N. McKeown, M. Izzard, A. Mekkittikul, W. Ellersick, M. Horowitz. "The Tiny Tera: A Packet Switch Core". IEEE Micro, pp 26 - 33. Jan/Feb 1997.
[25] N. Mckeown, P. Gupta. "Design and Implementation of a Fast Crossbar Scheduler". IEEE Micro Magazine, Jan/Feb 1999.
[26] N. Mckeown, T.E. Anderson. "A Quantitative Comparison for Scheduling Algorithms". Computer Networks and ISDN Systems, Vol 30, No 24, pp 2309-2326, December 1998.
[27] Y. Oie, T. Suda, M. Murata, H. Miyahara. "Switching Techniques in High-Speed Networks and Their Performance". INFOCOM, pp. 1242-1251, 1990.
[28] T. Shanley "Infiniband Network Architecture". Addison-Wesley, 2003.
[29] F. Silla. "Routing and flow control in networks of workstations". Tesis Doctoral, UPV, 1998.
[30] C. Sullivan, S. Chapell. "Handel-C for co-processing and codesign of field programmable systems on chip". JCRA'02, pp.65-70.
[31] J. Villasenor, W. Mangione-Smith. "Configurable Computing". Scientific American, pp. 66-71, Jun 1997.
[32] Xilinx Company http://www.xilinx.com/