IMPLEMENTACIÓN DE UNA INTERFAZ MPEG-2 MEDIANTE LÓGICA PROGRAMABLE FPGA

Miguel González López

Departamento de Electrónica y Sistemas
Universidad de A Coruña
miguelgl@des.fi.udc.es

Luis Castedo Ribas

Departamento de Electrónica y Sistemas
Universidad de A Coruña
luis@des.fi.udc.es

ABSTRACT

A novel MPEG-2 interface is proposed for the reception of digital video through telephone subscriber loops using the ADSL technology. The resulting solution uses a PCI data bus and is implemented in FPGA programmable logic.

1. INTRODUCCIÓN

El objetivo del presente trabajo es desarrollar una interfaz que permita a un decodificador de vídeo digital MPEG-2 (*Moving Pictures Experts Group*) recibir el flujo de vídeo a través de un bucle de abonado telefónico utilizando la tecnología ADSL (*Asymmetric Digital Subscriber Line).* La característica más destacada de la solución que se ha escogido es la utilización del bus de datos PCI (*Peripheral Component Interconnection*) para realizar la transferencia del flujo de vídeo y la tecnología FPGA (*Field Programmable Gate Array*) para implementar el dispositivo propuesto (en adelante dispositivo MPEGRoute). De este modo, el prototipo resultante puede implementarse en una arquitectura PC con las ventajas que ello conlleva.

A lo largo del trabajo supondremos que el enlace con el decodificador se ajusta a la especificación TS-Parallel. Se asume, además, que la interfaz con el bucle de abonado se compone de dos capas, siendo la más interna un módem ADSL y la más externa un integrado que permita la utilización de ATM como tecnología de enlace y red. La coordinación de los distintos elementos conectados al bus PCI es realizada por un microprocesador.

En la figura 1 se muestra la estructura del prototipo desarrollado. El chip SAR (*Segmentation & Reassembly)*, encargado de soportar las funciones ATM, se encarga de enviar las celdas ATM que contienen las tramas de vídeo a la memoria principal del sistema mediante el inicio de una transferencia a través del bus PCI. Siguiendo la terminología de la especificación PCI [1], el dispositivo que inicia la transferencia se denomina *master* y el dispositivo que la recibe *target*, por lo que el SAR actúa como *master* de esta transferencia y el puente PCI-microprocesador como *target*. Este último es quien gobierna el acceso desde el bus PCI a la memoria principal del PC y, recíprocamente, el acceso por parte del microprocesador del sistema a las zonas de memoria alojadas en los periféricos del bus. Una vez finalizada la transferencia, el SAR genera una interrupción al procesador para que su *device driver* extraiga la carga útil de las celdas, consistente en vídeo en formato MPEG-2 Transport Stream [2]. El flujo de vídeo se transfiere entonces al dispositivo MPEGRoute, que debe atender la recepción de datos desde el bus PCI y enviarlos al decodificador a través de una interfaz TS-Parallel.

Se describen tres soluciones concretas al problema en orden creciente de complejidad [7]. En las implementaciones se han utilizado dos funciones proporcionadas por Altera Corp., *pci_t32* y *pci_b* [5][6], que garantizan que el dispositivo cumple las exigentes especificaciones temporales del bus PCI.

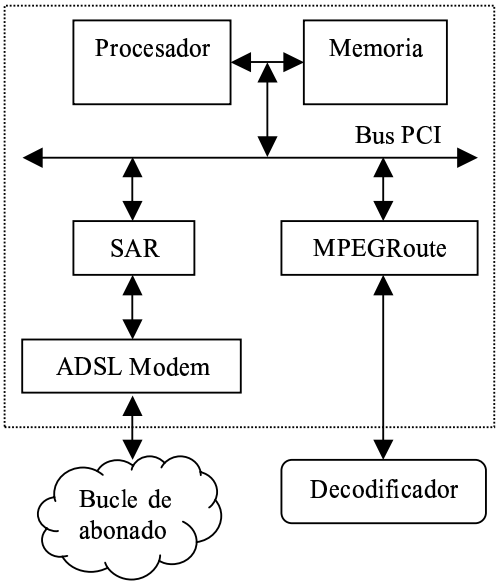

Figura 1. Estructura del sistema planteado

2. PRIMERA SOLUCIÓN

Consiste en un dispositivo PCI que actúa sólo en modo *target* y que dispone de un registro de 32 bits para la recepción de los datos. Cuando el procesador del sistema (*master)* efectúe una escritura en la zona del espacio de E/S asignada al registro antes mencionado a través del puente PCI-microprocesador, el dispositivo MPEGRoute transferirá la palabra recibida al decodificador a través de la interfaz TS-Parallel. Ésta cuenta con un bus de datos de ocho líneas, dos líneas de control (*dvalid*, que

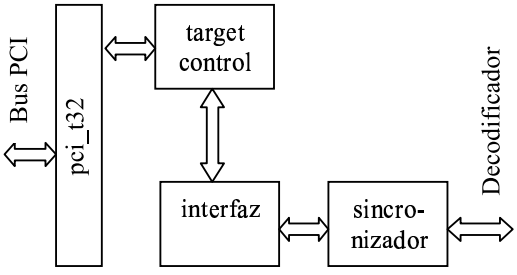

Figura 2. Primera y segunda soluciones

indica que el dato presentado en el bus de datos es significativo, y *psync*, que señaliza el comienzo de cada trama) y un reloj de sincronismo. Se deben adaptar las velocidades de llegada de los datos a través del bus PCI (4 bytes/ciclo con un reloj de 33 MHz) y la soportada por el TS-Parallel (1 byte/ciclo con un reloj cuya frecuencia debe ser como máximo 9 MHz). La solución propuesta estructura en los siguientes bloques funcionales (ver figura 2): *pci_t32*, función de acceso al bus PCI; *target_control*, que se ocupa del control de la parte *target* de la función *pci_t32*, regulando el tráfico de datos desde el bus PCI al módulo *interfaz*, que los enviará al exterior; *interfaz*, encargada de recibir los datos provenientes de la función *pci_t32* y regulados por *target_control*, y enviarlos al exterior a través de la interfaz TS-Parallel adaptando las velocidades del bus PCI y el TS-Parallel; y, por último, *sincronizador*, que genera las señales de sincronismo *psync* y *dvalid* a partir del flujo de datos proveniente de la función *interfaz*. La adaptación de velocidades se realiza obteniendo un nuevo reloj a partir del reloj del bus PCI cuya frecuencia es cuatro veces inferior, i.e., 8,33 MHz. Para evitar la desventaja de que la herramienta de simulación no puede calcular los retardos de los caminos entre nodos sincronizados por distintos relojes se ha optado por sincronizar todos los elementos con el reloj del bus PCI, y generar una señal que sirva de *clock enable* para los elementos que deben operar a la frecuencia del reloj de salida.

3. SEGUNDA SOLUCIÓN

Su novedad consiste en hacer un uso más eficiente del bus PCI dotando al dispositivo MPEGRoute de la capacidad de soportar una transferencia que comprenda no sólo una palabra sino una ráfaga de datos y aumentando así la tasa de transferencia media. Para ello es necesario sustituir en la solución anterior el registro de recepción por un *buffer* que pueda albergar una ráfaga del tamaño deseado y añadir la lógica de control necesaria para controlar el llenado y vaciado del mismo. La adaptación de velocidades se efectúa de modo análogo: una vez que el buffer se llena, se procede a vaciarlo a través de la interfaz TS-Parallel a una velocidad de reloj cuatro veces inferior a la del bus PCI.

4. TERCERA SOLUCIÓN

Incorpora la capacidad de ejercer como *master* del bus PCI, por lo que no es necesario el concurso del procesador principal para la transferencia (figura 3). Además no hace ninguna suposición acerca del reloj de salida, pudiendo provenir éste de un oscilador independiente. El procesador principal puede notificar al dispositivo en qué zona de memoria principal se encuentra la trama MPEG-2 a transferir a través de un registro incluido en este último. Una vez escrita la dirección de comienzo de la trama, el procesador sólo tiene que activar un bit disponible en el dispositivo para indicarle que comience la transferencia. Se compone de los siguientes bloques funcionales: pci_b, función similar a la pci_t32 pero con soporte *master*; *master_control*, que efectúa el proceso de leer la trama de memoria controlando las señales de la interfaz local de la función pci_b relacionadas con el acceso al bus en modo *master*; *FIFO & Interfaz*, que media entre la función *master_control* y la interfaz TS-Parallel de salida, recibiendo los datos de la primera para enviarlos al exterior a través de la segunda y de acuerdo a una señal de reloj externa. Actúa como buffer FIFO de adaptación de velocidades entre el bus PCI y la interfaz TS-Parallel; y, por último, *ini* y *direcc*, que almacenan el bit de inicio de transferencia y la dirección en memoria principal de la trama a transferir.

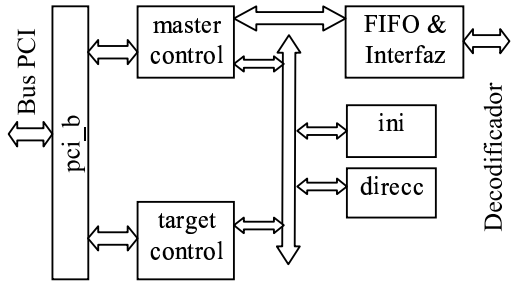

Figura 3. Tercera solución

5. RESULTADOS Y CONCLUSIONES

Los resultados de compilar los distintos diseños en diferentes arquitecturas FPGA [3][4] utilizando la herramienta Max+Plus II de Altera Corp. se muestran en la Tabla 1. Se comprueba que todos los diseños soportan la especificación PCI a 33 MHz, e incluso algunos la de 66 MHz. La complejidad creciente de las distintas soluciones se ve reflejada en el número de celdas lógicas (LCs) y bits de memoria requeridos. Se ha usado un buffer de cuatro palabras de 32 bits en la segunda solución y de dieciséis palabras en la tercera. La tasa de transferencia media se ha calculado teniendo en cuenta los ciclos de control adicionales necesarios para completar una transferencia. Por ello, cuanto mayor es la ráfaga de datos aceptada por el dispositivo (lo cual depende del tamaño del *buffer* de recepción) menos ciclos del total de la transferencia corresponden a ciclos de control, incrementándose la tasa de transferencia media.

Sol.	TT (MB/s)	Dispositivo	LCs	MB	FMáx (MHz)
1	33	EPF6010ATC144-1	659		42.55
		EPF10K30ETC144-1	652		84.74
2	75.43	EPF6016ATC100-1	921		37.17
		EPF10K30ETC144-1	714	128	67.56
3	111.16	EPF10K30AFC484-1	1545	512	36.63

TT = Tasa de transferencia media, LCs = *Logic Elements* usados, MB = *Memory Bits* usados, FMáx = Frecuencia Máxima

Tabla 1 Comparativa de las distintas soluciones

6. AGRADECIMIENTOS

Este trabajo ha sido financiado por FEDER (1FD97-0082) y Xunta de Galicia (PGIDT00PXI10504PR).

7. REFERENCIAS

[1] Shanley, T., Anderson, D., *PCI System Architecture*, MindShare, Inc., 1995.
[2] Benoit, H., *Digital Televisión. MPEG-1, MPEG-2 and principles of the DVB system*, Arnold, 1999.
[3] *FLEX 6000 Programmable Logic Family Data Sheet v4.02*, Altera Corp., 1999.
[4] *FLEX 10K Embedded Programmable Logic Family Data Sheet v4.01*, Altera Corp., 1999.
[5] *pci_t32 MegaCore Function User Guide*, Altera Corp., 1999
[6] *pci_b & pcit1 MegaCore Function User Guide*, Altera Corp., 1999.
[7] González López, M., *Diseño de dispositivos PCI con lógica programable FPGA*, Proyecto fin de carrera, UDC, 2000.