Banco de test hardware para video streaming en FPGA

David Hernández[(1)], Manuel Rodríguez[(1)], Fernando Pérez[(2)], Eduardo Magdaleno[(1)], Alejandro Ayala[(1)]

davidhdeze@gmail.com, mrvalido@ull.es, fdoperez@ull.es, emagcas@ull.es, aayala@ull.es
(1)Departamento de Física Fundamental y Experimental, Electrónica y Sistemas, Universidad de La Laguna
(2)Departamento de Estadística, Investigación Operativa y Computación, Universidad de La Laguna

Resumen— **El objetivo de este trabajo es diseñar e implementar un banco de prueba en FPGA para algoritmos de vídeo streaming. La arquitectura del diseño propuesto está basada en una estructura típica de procesado de video streaming. Esto es, un receptor de contenidos implementado con un interfaz Ethernet, un sistema de buffers basado en AXI4 y alojados en memoria externa SDRAM-DDR3 y un consumidor de contenidos implementado con un interfaz DVI que servirá para reproducir el vídeo por una pantalla. Esta estructura se completa con un microprocesador empotrado usado como elemento de configuración y control.**

Un sistema de estas características servirá como soporte hardware para el desarrollo de futuras aplicaciones o algoritmos de procesado de vídeo en FPGA.

Palabras clave—**Video Streaming, FPGA, AXI.**

I. INTRODUCCIÓN

EL mercado de electrónica de consumo con capacidad de reproducción de contenidos multimedia crece constantemente. En el ámbito de diseño e implementación de estos dispositivos, es indiscutible que el procesado de video supone uno de los aspectos más relevantes por su importancia en el consumo de recursos, rendimiento, consumo energético, etc. Por este motivo el desarrollo de nuevos procesadores de vídeo es un campo que está en continuo cambio.

En la actualidad, en el mercado se pueden encontrar distintos tipos de procesadores de vídeo entre los que destacan las GPU y ARMs por su amplio uso [1]. Sin embargo, estos dispositivos son microprocesadores incapaces de variar su arquitectura lo que los hace poco versátiles a la hora de crear prototipos no estándares. Por ejemplo, la reproducción de vídeo 3D no ha sido aún estandarizada y existe una lucha de mercados por conseguir la mejor solución.

El objetivo de este trabajo es implementar un banco de test hardware para probar algoritmos de video streaming, que sirva de soporte para desarrollar prototipos de futuras aplicaciones, tales como adquisición y codificación de video 3D o multivista, generación de video 3D bajo demanda, representación de contenidos 3D, etc.

La arquitectura propuesta en este trabajo para el banco de test está basada en un procesador típico de video streaming [2, 3 y 4] como el mostrado en la Fig. 1.

En este diagrama podemos distinguir tres partes bien diferenciadas. Por un lado el receptor, encargado de dar acceso al canal de transmisión para recibir el stream de entrada. Por otro lado, el driver de salida que hace las tareas de acondicionar la señal para que pueda ser entendida por el medio de reproducción. Y en tercer lugar, entre el receptor y el driver de salida se encuentra lo que denominaremos medio de procesamiento que consiste en uno o varios procesos que usan un buffer de memoria compartida. La función que cumple este tercer bloque es permitir aplicar al stream de entrada distintos procesos antes de ser visualizado en el medio de reproducción. Además, la existencia del buffer de memoria en el medio de procesado tiene la finalidad de sincronizar todos los procesos y permitir un flujo ininterrumpido de vídeo a la salida.

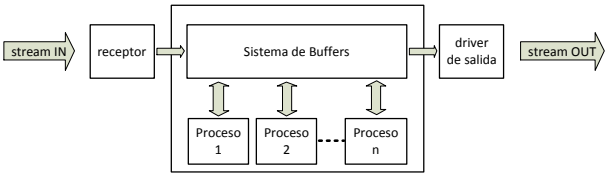

Fig. 1. Diagrama de bloques de un procesador típico de video streaming.

Para la implementación de la arquitectura del banco de test se ha usado una FPGA como soporte hardware. En este contexto, esta elección está justificada por la flexibilidad que tienen estos dispositivos programables para adoptar resoluciones de pantalla, ritmos de fps y anchos de bit que no sean estándares. Al mismo tiempo, las FPGA están dotadas con bloques de entrada/salida programables que permiten implementar interfaces de alta velocidad. Por otro lado, son adecuadas para integrar todo el sistema en una misma pastilla dándole robustez al diseño y reduciendo los costes de implementación [5]. A todo esto se une el creciente esfuerzo de los fabricantes de FPGA en introducir sus dispositivos en este campo, aportando librerías de cores destinadas al tratamiento de vídeo. En el caso de Xilinx, se destaca el desarrollo de la librería de cores Video & Image processing, así como la apuesta por tarjetas de desarrollo orientadas a aplicaciones de vídeo como la Spartan 3A Video Started Kit y la Spartan-6 LX150T Eval Board. Concretamente, la FPGA usada en este trabajo es la Virtex 6 XC6VLX240T integrada en la tarjeta de desarrollo ML605 [6].

A continuación se hará una descripción general del sistema implementado haciendo una descripción más detallada de cada uno de los módulos que lo componen.

Por último mostraremos los resultados y conclusiones obtenidas tras la elaboración de este trabajo.

II. DESCRIPCIÓN DEL SISTEMA

Como hemos comentado anteriormente, la arquitectura de un procesador de streaming de vídeo está organizada en tres bloques funcionales básicos: el receptor o productor de contenidos, el medio de procesamiento y el driver de salida. Además de estos módulos, típicamente existe un módulo para configuración y control.

En la Fig. 2 se muestra la arquitectura del banco de test de vídeo streaming implementada. En primer lugar, el receptor del sistema se ha implementado con un Interfaz Ethernet Gigabit bajo el protocolo UDP/IP. En segundo lugar, el driver de salida, se ha implementado con un Interfaz DVI. Por último, como elemento central de nuestra arquitectura se ha implementado un sistema de buffers alojados en memoria SDRAM-DDR3. En cuanto al sistema de control, la arquitectura propuesta integra el microprocesador empotrado de Xilinx, MicroBlaze™ [7].

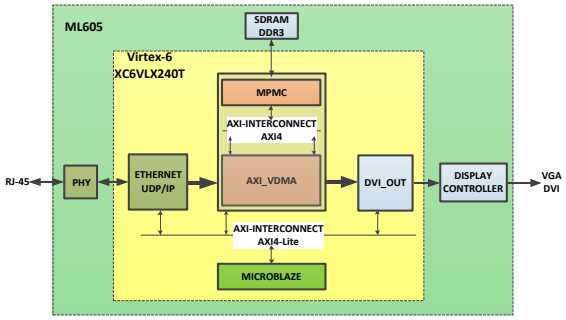

Fig. 2. Arquitectura del banco de pruebas de video streaming.

Destacamos que la comunicación de esta arquitectura está basada en el bus AXI4 (Advanced eXtensible Interface) [8]. Este bus tiene tres tipos de implementación en función del tipo de datos y la velocidad de transmisión requerida. Estas son: AXI4, orientado a mapeos de memoria de alto rendimiento, por ejemplo, movimiento de grandes cantidades de datos entre sistemas de memoria; AXI4-Lite, orientado a mapeos de memoria de baja velocidad, como tareas de configuración o monitorización de registros de control y registros de estado; y AXI4-Stream, orientado a mover datos de streaming a alta velocidad en conexiones tipo punto a punto.

Para la gestión de este bus, en nuestro diseño se hecho uso de los cores AXI INTERCONNECT de Xilinx. Estos cores son controladores de bus que permiten conectar uno o varios maestros Memory Mapped (AXI4 o AXI4-Lite) a uno o varios esclavos del mismo tipo. Como se puede ver en la Fig. 2, nuestra arquitectura implementa dos de estos módulos: uno de ellos en modo AXI4, situado en el elemento central, gestionando las operaciones de acceso a la memoria SDRAM; y el otro en modo AXI4-Lite encargado de arbitrar el bus de configuración y control, donde el MicroBlaze actúa como maestro.

En cuanto a la comunicación entre los módulos de procesado hardware, que es por donde fluye el vídeo desde el receptor hasta el driver de salida, se ha hecho mediante el bus AXI4- Stream.

A continuación describiremos los elementos principales del banco de pruebas.

A. Receptor: Interfaz Ethernet

Para implementar el receptor o fuente de datos para nuestro banco de test, disponemos típicamente de varios interfaces como pueden ser VGA/DVI, PCI-E, USB y Ethernet. De entre todos ellos, el que mayor velocidad de transmisión soporta es el PCI-E alcanzando velocidades del orden de varios GB. Sin embargo su utilización no es sencilla, pues requiere integrar la tarjeta de desarrollo en el PC lo que implica el diseño de drivers específicos. En cuanto a Ethernet y USB, el manejo de ambas tecnologías implica una complejidad similar. Sin embargo, nos decantamos por la utilización de Ethernet ya que aporta mayor velocidad de transmisión y tiene la ventaja de que permite incorporar nuestro sistema a una red de ordenadores.

Sobre la interfaz Ethernet, el protocolo de internet IP/UDP resulta la combinación más óptima y extendida para el transporte de vídeo entre un PC, que funcione como servidor, y una FPGA remota. Además, al establecer una capa de red con capacidad de enrutado, dotamos al sistema con la posibilidad de permitir la entrada de datos desde cualquier servidor de vídeo en red a través de direccionamiento IP a nuestro banco de pruebas.

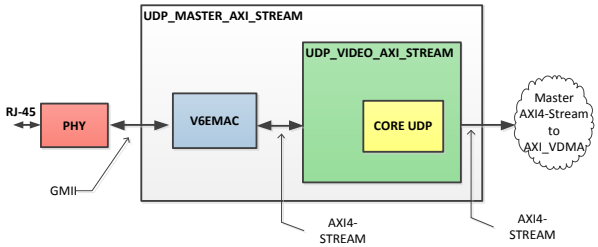

Fig. 3. Interfaz Ethernet

La implementación del Interfaz Ethernet consta de varios bloques funcionales que implementan las diferentes capas del modelo de referencia OSI. En la Fig. 3 se muestra su diagrama de bloques. De izquierda a derecha encontramos primero un chip situado físicamente en la ML605 que implementa la capa física. La capa de Enlace de Datos se ha implementado con el core IP Virtex-6 FPGA Embedded Tri-Mode Ethernet MAC v2.1 (V6EMAC) [9] de la librería Communication & Networking de Xilinx. Éste permite tres modos de operación seleccionables: 10 Mbps, 100 Mbps y 1000 Mbps. El V6EMAC está diseñado específicamente para las FPGA de la familia Virtex-6 ya que éstas incorporan unidades hardware empotradas que tienen esta funcionalidad. Concretamente, la XC6VLX240T usada en este trabajo cuenta con cuatro unidades empotradas. Para otras FPGA, Xilinx ofrece múltiples soft cores que cumplen con este cometido. Por último, para la implementación de las capas de transporte y de red, se ha optado por una implementación puramente hardware, descartando una implementación software con librerías como LWIP, pues éstas requieren de recursos para implementar funcionalidades que no son necesarias

en nuestro diseño. En el mercado existen diferentes cores IP hardware que implementan los protocolos UDP/IP. El core elegido [10] destaca por su eficiencia en el uso de recursos, su alto rendimiento y porque se distribuye con licencia Open-Source. Este core fue diseñado para hacer conexiones UDP/IP punto a punto desde un PC a una FPGA, concretamente para las FPGAs Spartan-3 y Virtex-5 de Xilinx. El hecho de que haya sido diseñado específicamente para estas FPGA en concreto, nos ha obligado a hacer algunas modificaciones en el código VHDL que describe el core para poder adaptarlo a la Virtex-6 usada en este trabajo. Además, se ha adaptado el interfaz para hacerlo compatible con el bus AXI4-Stream.

B. Medio de procesamiento: Frame Buffer DDR3

Para la implementación de un procesador de vídeo streaming es necesario un conjunto de buffers que mantengan parte de la información almacenada en memoria y que así sirvan de medio de sincronización entre los elementos de procesado o para hacer operaciones entre fotogramas distintos. Por ejemplo, es usual en aplicaciones de detección de movimiento, comparar dos fotogramas consecutivos en el tiempo.

Los buffers para aplicaciones de vídeo requieren el uso de memorias con gran capacidad y con altas velocidades de lectura/escritura. Esto hace que se descarte el uso de memoria interna de las FPGA, ya sea en bloques o distribuida, por la poca capacidad que pueden llegar a implementar. De entre las memorias externas disponibles, las que más se ajustan a los requerimientos del vídeo streaming son las DDRx SDRAM, pues tienen capacidades que pueden llegar a los 16GiB con un capacidades de transferencia del orden de miles de MB/s (hasta 17.000 MB/s las de mejores prestaciones). Estas memorias tienen un único puerto, pero con el gran ancho de banda que éste tiene se puede multiplexar su uso simulando memorias multipuerto. Además, el coste de estas memorias es relativamente bajo (menos de 5 €/GB para DDR3) y casi la totalidad de las tarjetas de desarrollo de FPGA cuentan con algún módulo DDR2 o DDR3.

Un hándicap a superar para poder hacer uso de las memorias DDRx es el diseño e implementación del controlador de memoria. Se trata de un dispositivo de compleja arquitectura que implementa arduos protocolos de comunicaciones a nivel físico y que tiene que ceñirse a estrictas restricciones temporales. Sin embargo, Xilinx aporta la herramienta Memory Interface Generator (MIG) [11] de la librería Memory & Storage Elements, que nos permite, seleccionando algunos parámetros de diseño, implementar dicho controlador de memoria abstrayéndonos casi por completo de su funcionamiento interno. Haciendo uso del MIG tenemos la posibilidad de generar controladores de memoria para DDR2 o DDR3 que implementen un interfaz AXI4, haciendo que estas memorias sean accesibles desde nuestro sistema de buses.

Como ya se ha mencionado, las memorias DDR sólo tienen un puerto de lectura/escritura pero éste puede ser multiplexado para implementar varios puertos. Esta función la cumple en nuestro diseño el controlador de bus AXI INTERCONNECT v1.04.a mostrado en la Fig. 4. A este conjunto de controlador de memoria (MIG) más el multiplexor de puertos se le denomina en terminología de Xilinx Multiport Memory Controller (MPMC).

En este caso nuestra aplicación requiere únicamente de dos puertos, dedicado al canal de lectura y de escritura respectivamente. Sin embargo, el AXI INTERCONNECT permite conectar hasta 16 puertos.

En cuanto a los interfaces, en la Fig.4 se ve lo hasta ahora mencionado. El AXI MIG implementa el interfaz físico (PHY) con la DDR3 y lo convierte en un interfaz AXI4 esclavo. Por otro lado, el AXI INTERCONNECT recibe las órdenes de lectura/escritura por los interfaces AXI4 esclavos, las organiza y las manda por el interfaz AXI4 master hacia el AXI MIG. De este modo conseguimos con el MPMC una memoria de doble puerto integrada en nuestro sistema por un bus AXI4.

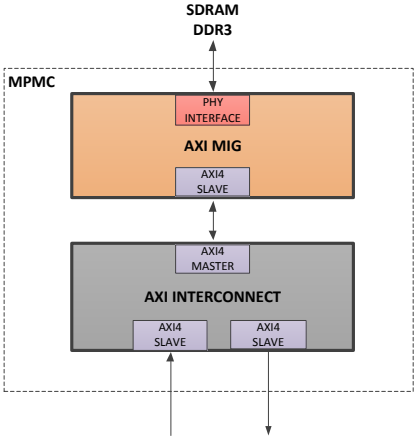

Fig. 4. Multiport Memory Controller.

Sin embargo, recordemos que los datos que proceden del Interfaz Ethernet viajan en forma de stream en un bus AXI4-Stream. Es por eso que incluimos en nuestro diseño el core AXI VDMA (Video Direct Memory Access) [12] de Xilinx. Este módulo cumplirá la función de convertir los datos en formato Stream a formato Memory Mapped (S2MM) y viceversa (MM2S) para poder ser escritos/leídos en memoria en modo ráfaga. En la Fig.5 se muestra la estructura completa del FRAME BUFER. En nuestro diseño se ha usado la versión *v4.00.a* del *AXI VDMA*, de la librería *Video & Image processing* de Xilinx.

Como se ha mencionado anteriormente, su cometido es mover grandes paquetes de datos entre interfaces *AXI4-Stream* y *AXI4 Memory Mapped*. Estas grandes cantidades de datos no son otras que el conjunto de bytes que componen un frame de vídeo.

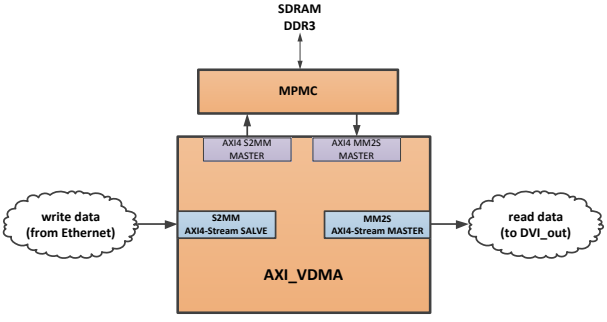

Fig. 5. Frame Buffer. AXI_VDMA con MPMC.

Su funcionamiento se basa básicamente en organizar el espacio de memoria que se le asigne en buffers bidimensionales del tamaño que ocupa un fotograma. De modo que estos buffers pueden ser leídos/escritos de forma secuencial a partir de una señal de disparo. Esto permite leer y escribir en la memoria SDRAM-DDR3 a través del sencillo interfaz AXI4-Stream.

El *AXI VDMA* puede implementar dos canales, uno dedicado a lectura (*MM2S*) y otro a escritura (*S2MM*) que funcionan de forma simultánea e independiente, permitiendo así simultanear las operaciones de lectura y escritura en memoria.

En este ejemplo se ha añadido sólo un módulo AXI_VDMA, pero es posible adecuar el número de módulos a las necesidades del algoritmo a implementar.

C. Driver de salida: Interfaz VGA/DVI

La arquitectura propuesta dispone de un driver de salida que implementa un interfaz de DVI (Digital Video Interface).

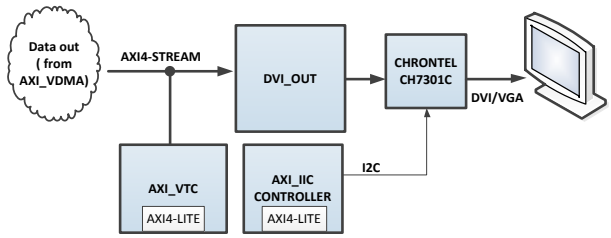

Fig. 6. Interfaz DVI.

Este interfaz de salida de vídeo comprende los elementos necesarios para reproducir una señal de video a través de un monitor. Para ello cuenta, por un lado, con el core AXI Video Timing Controller (AXI_VTC) [13] de Xilinx, encargado de generar la señales de sincronismo horizontal y vertical. Por otro lado, para adaptar la señal de vídeo para sacarla de la FPGA hacia el controlador de pantalla, el interfaz cuenta con el módulo DVI_OUT. Además, el controlador de pantalla (CHRONTEL CH7301C), requiere ser configurado por medio de bus IIC. Por tanto, nuestro diseño implementa un controlador de bus IIC mediante el core AXI IIC Bus Interface de Xilinx [14]. Este último, así como el AXI_VTC son controlados por el MicroBlaze a través del bus AXI4-Lite. En la Fig. 6 se muestran los bloques funcionales que forman el interfaz de DVI.

III. RESULTADOS Y CONCLUSIONES

En este apartado mostraremos los resultados obtenidos con la elaboración de este trabajo.

Se ha diseñado e implementado un banco de test para procesado hardware de video streaming que se compone de: Interfaz Ethernet Gigabit como receptor o productor de contenidos; medio de procesamiento implementado con un sistema de buffers basado en el bus AXI4 y alojado en memoria SDRAM-DDR3; y, como consumidor de contenidos, un driver de salida implementado con un interfaz DVI.

Toda la arquitectura, ha sido diseñada y ensamblada en el entorno de desarrollo EDK, usando el SDK para la programación del MicroBlaze.

El sistema ha sido sintetizado en la FPGA Virtex-6 XC6VLX240T de la tarjeta de desarrollo ML605. Los recursos hardware utilizados de la mencionada FPGA se muestran en la TABLA I.

La utilización de recursos de la FPGA no es crítica y por tanto, nuestro sistema permite ampliaciones incluyendo nuevos AXI_VDMA así como los módulos de procesado hardware que se quieran poner bajo test. Destacamos que la utilización de DSP48E1s es sólo del 1%, dejando numerosos DSPs disponibles para la implementación de algoritmos de procesado que requieran operaciones aritméticas.

TABLA I

UTILIZACIÓN DE RECURSOS. FPGA XC6VLX240T.

Recurso	Utilización
Slice Registers	22,338 (7%)
Slice LUTs	17,681 (11%)
Occupied Slices	8,683 (23%)
Bonded IOBs	185 (30%)
RAMB18E1	6 (1%)
Number of DSP48E1s	3 (1%)

Para verificar el sistema se ha hecho el montaje que se muestra en el esquema de la Fig 7.

Fig. 7. Montaje de verificación del sistema.

Por un lado, se ha conectado la tarjeta ML605 a un PC (servidor de vídeo) mediante un cable *RJ-45* para formar así el enlace Ethernet. Y por otro lado, se ha conectado a un monitor VGA haciendo uso de un conversor *DVI-VGA.*

En cuanto al servidor de vídeo, éste ha sido diseñado con un programa Java que implementa un socket UDP/IP y se ejecuta en un PC con procesador Intel Core 2 6400@2.13GHz.

El éxito de esta experiencia acredita el correcto funcionamiento de cada uno de los componentes diseñados e implementados en nuestro diseño, así como del sistema completo. Esto es, recibimos el flujo de vídeo a través de Ethernet a una velocidad media en torno a 40 Mbps mientras se reproduce por el monitor de forma ininterrumpida. Asimismo, el sistema de buffers garantiza el refresco de pantalla a una frecuencia de 60 Hz sin que se produzca un efecto de parpadeo o rellenado de pantalla.

En nuestro caso sólo hemos probado el sistema enviando un video desde el PC a nuestra arquitectura y comprobando su reproducción por pantalla, sin aplicar ninguna transformación sobre los datos.

Para probar algoritmos sólo es necesario insertar el bloque que se quiera poner bajo test, previo diseño con las interfaces adecuadas (AXI-4-Stream).

REFERENCIAS

[1] M. Owaida, N.Bellas, C. D. Antonopoulos, K. Daloukas, Ch. Antoniadis, K. Krommydasy and G. Tsoumblekas, Implementation and Performance Comparison of the Motion Compensation Kernel of the AVS Video Decoder on FPGA, GPU and Multicore Processors, IEEE International Symposium on Field-Programmable Custom Computing Machines, 2011

[2] Graeme Stewart, David Renshaw ,Martyn Riley, a low-cost, fpga based, video streaming server, Southern Conference on Programmable Logic, 2007. SPL '07. 2007 .

[3] Christophe Desmouliers, Erdal Oruklu and Jafar Saniie ,FPGA-based design of a high-performance and modular video processing platform, IEEE International Conference on Electro/Information Technology, 2009. eit '09.

[4] Jai Gopa Pandey, An Embedded Architecture for Implementation of a Video Acquisition Module of a Smart Camera International Conference on System, Devices, Circuits and Systems (ICDCS), 2012

[5] Philip H.W. Leong, Recent Trends in FPGA Architectures and Applications, 4th IEEE International Symposium on Electronic Design, Test & Applications, 0-7695-3110-5/08.

[6] ML605 Hardware User Guide. (UG534.pdf)

[7] MicroBlaze Processor Reference Guide. (UG081.pdf)

[8] AXI Reference Guide. (UG761.pdf)

[9] Virtex-6 FPGA Embedded Tri-Mode Ethernet MAC. User Guide. (UG368.pdf)

[10] Nikolaos Alachiotis, Simon A. Berger, Alexandros Stamatakis "EFFICIENT PC-FPGA COMMUNICATION OVER GIGABIT ETHERNET", IEEE International Conference on Computer and Information Technology (CIT), 2010

[11] Virtex-6 FPGA Memory Interface Solution. User Guide. (UG406.pdf)

[12] LogiCORE IP AXI Video Direct Memory Access v4.00.a. Product Guide. (PG020.pdf)

[13] LogiCORE IPVideo Timing Controller v3.0. (DS857.pdf)

[14] LogiCORE IP AXI IIC Bus Interface. (axi_iic_ds756.pdf)