Diseño de sistemas digitales mediante procesadores empotrados en un circuito reconfigurable

Enrique Soto Campos[1,2], Pablo Ríos Costas[2], Camilo Quintáns Graña[2], Enrique Mandado Pérez[1]
[1.] Instituto de Electrónica Aplicada
[2.] Departamento de Tecnología Electrónica
Universidad de Vigo, España
esotoc@uvigo.es, quintans@uvigo.es, enriquemandado@gmail.com

***Resumen*—Mediante los trabajos fin de grado y los proyectos fin de carrera de las Escuelas de Ingeniería se pueden desarrollar sistemas basados en procesadores empotrados en un circuito FPGA y estudiar sus características. En este trabajo se describe el desarrollo de este tipo de sistemas mediante una placa Altera DE2 que tiene un circuito FPGA Cyclone II. Como ejemplo se ha utilizado dicha placa para implementar un sistema generador de funciones y frecuencímetro digital controlado remotamente mediante la red Ethernet.**

***Palabras clave*—procesadores empotrados, FPGA, VHDL.**

I. Introducción

Este trabajo se enmarca dentro de los proyectos fin de carrera de la Escuelas de Ingeniería Industrial y de Telecomunicación de la Universidad de Vigo. El sistema logrado permite la gestión remota sobre Ethernet de tipo cliente-servidor UDP (*User Datagram Protocol*), y facilita la interactuación con una plataforma de instrumentos implementados sobre una FPGA (*Field Programable Gate Array*) [1].

Las FPGAs son dispositivos semiconductores cuya funcionalidad no está predeterminada de antemano, sino que es el diseñador del sistema el que, mediante un lenguaje de descripción hardware (VHDL, Verilog, etc.), programa la funcionalidad deseada en el dispositivo. Además, dicha funcionalidad o configuración se puede modificar, es decir reprogramar, lo que le otorga una gran ventaja frente a, por ejemplo, los sistemas de tipo ASIC. Esta capacidad de reprogramación es muy útil en el prototipado de sistemas, puesto que permite tener el sistema implementado físicamente y completamente funcional (no solo simulado) con un coste mucho más bajo que su equivalente ASIC. Tanto las FPGAs como los ASICs se utilizan en aplicaciones similares, sin embargo las FPGAs son más lentas, tienen un mayor consumo, y están más limitadas en cuanto a la complejidad del sistema a diseñar. Por ello, generalmente se suele diseñar el prototipo mediante una FPGA, y posteriormente, una vez verificado el sistema, se fabrica el producto final en tecnología de tipo ASIC.

La Figura 1 muestra una visión general y simplificada del sistema. El cliente es un ordenador de aplicación general (PC), que a través de una interfaz gráfica permite interactuar con los instrumentos que se encuentran en el servidor. Se puede tanto configurar los instrumentos, como obtener datos (medidas) de los mismos. Gracias a la utilización de Python y Qt, la aplicación es multiplataforma, lo que hace posible su ejecución en cualquier PC, independientemente del sistema operativo utilizado.

Por otro lado, el servidor es la placa de desarrollo basada en FPGA. En dicha placa se implementan los instrumentos: un frecuencímetro y un generador de funciones. Ambos instrumentos están conectados a un procesador embebido, el Nios II de Altera, que facilita su control interno. El Nios II es un procesador configurable (se puede adaptar a las necesidades del sistema) de arquitectura de 32 bits que está diseñado de forma específica para la familia de FPGAs de Altera. Como sistema operativo de dicho procesador se ha utiliza el MicroC OSII.

Los estudiantes del título de Ingeniero Técnico Industrial de la especialidad de Electrónica y Automática tenían (porque el plan se ha extinguido) 6 horas de docencia de lenguajes de descripción hardware y 45 de microcontroladores. Su experiencia en el trabajo con microcontroladores resultaron ser importantes en el diseño del microprocesador empotrado y sus periféricos. Los estudiantes del título de Ingeniería de Telecomunicación tenían, como es lógico, muchas más horas, incluyendo 60 horas de diseño hardware con VHDL, por lo que les resultaba más fácil diseñar los periféricos del microprocesador en VHDL. Además, sus conocimientos de protocolos de comunicaciones fueron los que hicieron posible el desarrollo de la aplicación cliente-servidor de este trabajo.

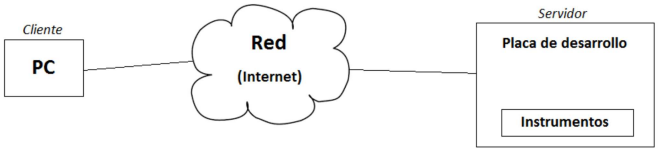

Fig. 1. Diagrama de bloques simplificado del sistema.

II. Descripción del Sistema Físico

A. *La placa de desarrollo ALTERA DE2*

La placa ALTERA DE2 [2] con la FPGA CYCLONE II, es el principal elemento hardware utilizado en este trabajo. La placa DE2 está formada por diversos componentes entre los que están la FPGA Cyclone II 2C35 y 8 Mbytes SDRAM para

las aplicaciones software. Además de diversos recursos de entrada/salida (Fig. 2).

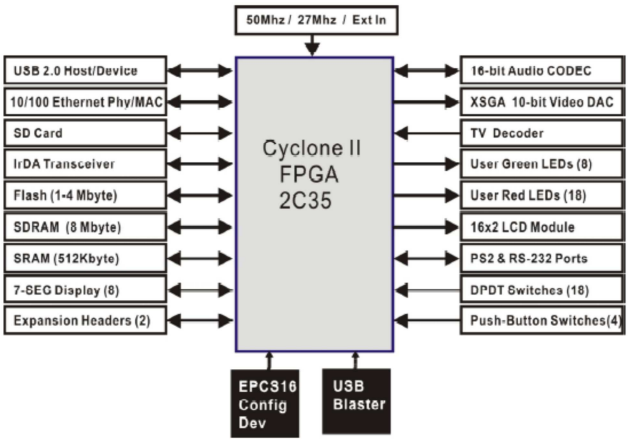

Fig. 2. Placa de desarrollo Altera DE2.

B. *El procesador empotrado NIOSII*

El NIOS II [3] es un microprocesador de 32 bits de aplicación general de arquitectura Harvard con 32 registros de 32 bits, 32 fuentes de interrupción, juego de instrucciones RISC de 32 bits, controlador de interrupciones, operaciones de multiplicación y división de 32 bits, instrucciones especiales para operaciones de 64 y 128 bits y capacidad de direccionamiento de 32 bits.

Se proporcionan tres versiones distintas según se busque maximizar el rendimiento del procesador o minimizar la cantidad de recursos lógicos utilizados: Nios II *economy*, *standard* y *fast*. En orden de mención progresan en complejidad y recursos de la FPGA utilizados.

C. *Plataforma de desarrollo Altera QuartusII*

La plataforma software de Altera para la creación de sistemas electrónicos en FPGA se llama *QuartusII*. Este programa utiliza, tanto lenguajes de descripción hardware como esquemas, que facilitan el diseño. La incorporación del microprocesador empotrado NIOS II se hace mediante una herramienta incluida dentro del *Quartus* llamada *Qsys* [4][5].

El *Qsys* (Fig. 3) muestra en su interfaz tres espacios diferenciados: *Component Library*, *Messages* y la página principal con varias pestañas, por defecto en *System contents*. Para configurar el microprocesador se debe primero buscar el componente en *Component Library* e instanciarlo. El componente instanciado se configura en una ventana desplegable.

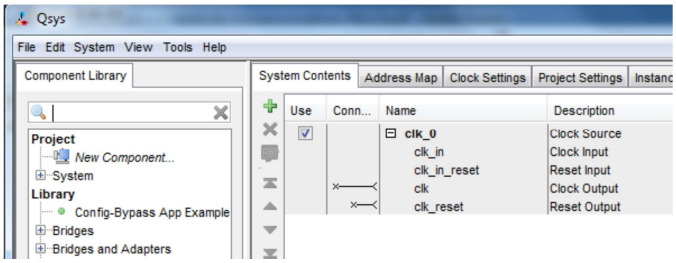

Fig. 3. Ejemplo de configuración en Qsys.

El significado de las pestañas de la página principal es:

System Contents: Facilita la edición de cada componente y el interconexionado entre ellos. Las columnas que aparecen en esta pestaña tienen el siguiente cometido: *Use* indica si el componente se utiliza o no en el sistema, *Connections* permite realizar las conexiones entre los componentes, *Name* indica el nombre del componente y el de sus interfaces de entrada y de salida, *Description* indica el tipo de componente y el tipo de entrada y salida, *Export* permite exportar una señal de forma que sea visible desde el exterior del sistema que se está generando en *Qsys*, *Clock* indica el reloj o relojes utilizados en cada componente, *Base* y *End* indican el comienzo y final de la dirección de memoria en la que reside el mapeado de memoria de dicho componente, e IRQ facilita la visualización y edición del nivel de prioridad de la interrupción del componente asociado.

Address Map: muestra el mapa de memoria actual.

Clock Settings: visualizan los relojes que están configurados (tanto el reloj básico como las derivaciones generadas por los circuitos PLL).

Project Settings: permite seleccionar la FPGA con la que se está trabajando.

Instance Parameters: facilita la definición de los parámetros para configurar posibles reutilizaciones de componentes.

System Inspector: visualiza el conjunto de componentes e interconexiones que componen el sistema completo diseñado con la herramienta.

HDL Example: muestra la declaración en HDL (Verilog o VHDL) del sistema.

Generation: genera un componente del sistema para integrarlo posteriormente en un esquema de Quartus II.

Los componentes que forman el sistema completo son: reloj, PLL, CPU, controlador SDRAM, JTAG UART, temporizadores, entradas/salidas para controlar la unidad de comparación y captura, entrada/salidas digitales, controlador de LCD, identificador del sistema y controlador Ethernet DM9000A.

D. *Unidad de captura y comparación*

Una unidad de captura y comparación es un periférico habitual en microcontroladores. Sus aplicaciones más frecuentes son las salidas de alta velocidad no dependientes de la ejecución de instrucciones, generación de señales moduladas en anchura de impulso y las entradas de alta velocidad utilizadas para medir períodos o desfases entre señales. Para la ejecución de todas estas tareas está compuesta de uno o más temporizadores y de dispositivos de comparación con registros que permiten la captura (entrada de alta velocidad) o la ejecución de una salida (salida de alta velocidad).

En esta aplicación se implementó en VHDL una unidad de captura y comparación con las siguientes características:

- Temporizador interno de 32 bits, que se utiliza para realizar una medición del tiempo transcurrido entre dos flancos de la señal a capturar. El valor de este temporizador se guarda en dos registros cuando se recibe una orden de captura.
- Memoria RAM de 36 posiciones con un ancho de 32 bits que permite tener 36 valores de comparación para generar cualquier tipo de onda PWM.
- Registros de 32 bits, para guardar el valor del temporizador ante una entrada de alta velocidad (captura).
- Detector de flancos para sincronizar la entrada de alta velocidad.

E. Finalización del proyecto

La placa DE2 está integrada por diversos componentes que interactúan con la FPGA y son necesarios para su correcto funcionamiento, como por ejemplo los interruptores y los diodos luminiscentes (LED). Todos estos componentes tienen asignado un terminal (*pin*) en la FPGA gracias al cual se puede controlar su funcionamiento. La asignación de terminales se puede consultar en el manual de la placa ALTERA DE2. Para controlar estos componentes desde el Nios II es necesario conectar los puertos del microprocesador a elementos de entrada y salida y después asignar a estas entradas y salidas un terminal de la FPGA.

Para saber el número de entradas, salidas o terminales bidireccionales que se han de conectar se puede ver en el bloque "niosIImicro" con "*Properties*> *Ports*" un listado de los diferentes tipos de entradas así como de su nombre y su longitud como se muestra en la Fig. 4.

Una vez que se comprueba que las conexiones están realizadas correctamente, se realiza el proceso de *Analysis & Synthesis* (*Processing*> *Start*> *Start Analysis & Synthesis*) y, a continuación, se realiza la asignación de terminales de la FPGA a las entradas y salidas del sistema. Esta tarea se realiza a través del *Pin Planner* como se muestra en la Figura 5.

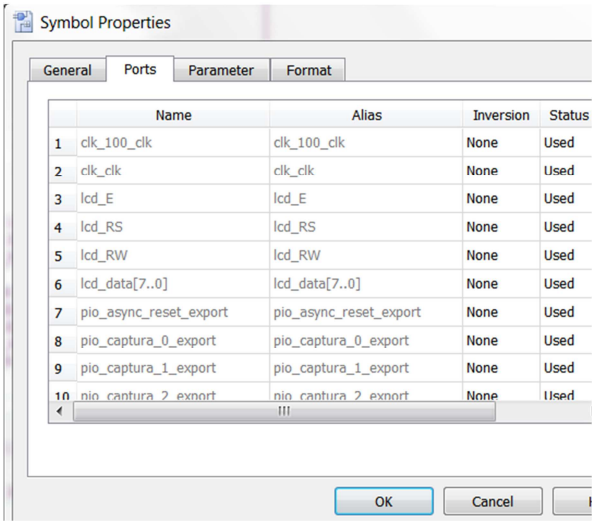

Fig. 4. Entradas y salidas del NiosII.

Los terminales de la placa que no se utilizan se establecen como entradas de tercer estado. Para ello en Quartus II hay que seleccionar *Assignments -> Device -> Device and Pin options -> unused pins*, y configurarlo como *As input tri-stated with weak pull-up*.

Finalmente se procede a la compilación del proyecto. Si no hay ningún error se muestran todos los procesos en verde (Fig. 6).

Por último, se debe configurar la FPGA con el sistema creado. Para tal fin, se deben seguir los siguientes pasos: conectar la placa a la alimentación, y conectar el USB Blaster a uno de los puertos USB del PC, poner el conmutador de programación en la posición RUN, seleccionar *Tools -> Programmer*, abrir el programador de la configuración hardware, seleccionar el fichero deseado (con extensión *.sof) y pulsar el botón *start*, tal y como se muestra en la Figura 7.

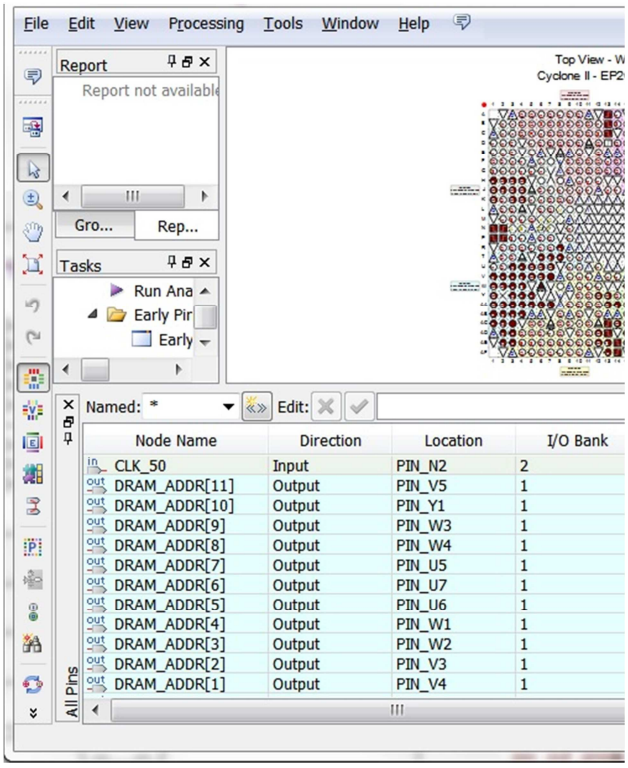

Fig. 5. Pin planner.

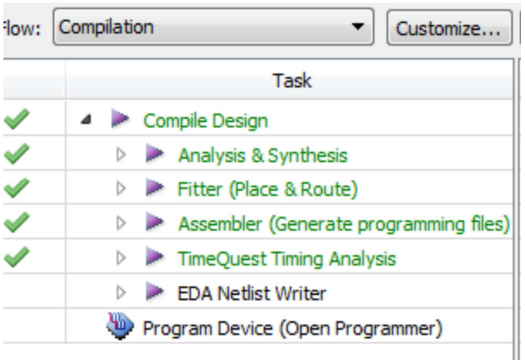

Fig. 6. Compilación completa del proyecto.

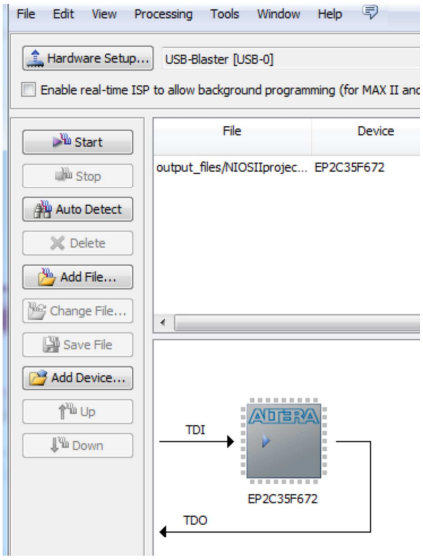

Fig. 7. Configuración de la FPGA.

III. Programación

A. *Servidor*

La aplicación que se ejecuta en la placa de desarrollo Altera DE2 funciona como servidor en una configuración cliente/servidor. La programación se realiza en lenguaje C de la plataforma de programación *Eclipse*. Este entorno está integrado en *Quartus* II con un *plugin* para el Nios II. Para arrancarlo se debe seleccionar *Tools -> Nios II Software Build for Eclipse* [6].

La ventana principal de *Eclipse* está compuesta por una ventana denominada *Project Explorer*, en la que se indican los archivos que componen el proyecto, una ventana llamada *Outline* que muestra un índice para acceder de forma rápida a las distintas partes del programa y otra ventana en la parte inferior con distintas pestañas que indica los errores, advertencias, y otra información relevante del proyecto. Por último, la parte central es el editor de textos, que facilita la edición el código.

Para crear un nuevo proyecto se selecciona *File -> New -> Nios II Application and BSP for Template*. En el apartado SOPC Information File Name se debe incorporar el fichero **.sopcinfo*, que incorpora la información necesaria sobre el hardware diseñado en *Qsys*. En el apartado *Project Name* se le da el nombre deseado al proyecto, este caso 'frec_gen'. En *Project Template* se pueden escoger distintos formatos preparados para ayudar en el desarrollo, como por ejemplo uno que incluya el sistema operativo Micro/OS-II. Esto incluye todas las bibliotecas del sistema operativo para poder introducir y gestionar los hilos de ejecución. Una vez hecho esto en la ventana *Project Explorer* muestra dos carpetas: una denominada frec_gen y otra frec_gen_bsp (Fig. 8). La primera contiene el código que se desarrolla para la aplicación. La otra contiene toda la información sobre el hardware y sus bibliotecas asociadas.

El diagrama de la Figura 9 muestra una imagen conceptual del diseño software.

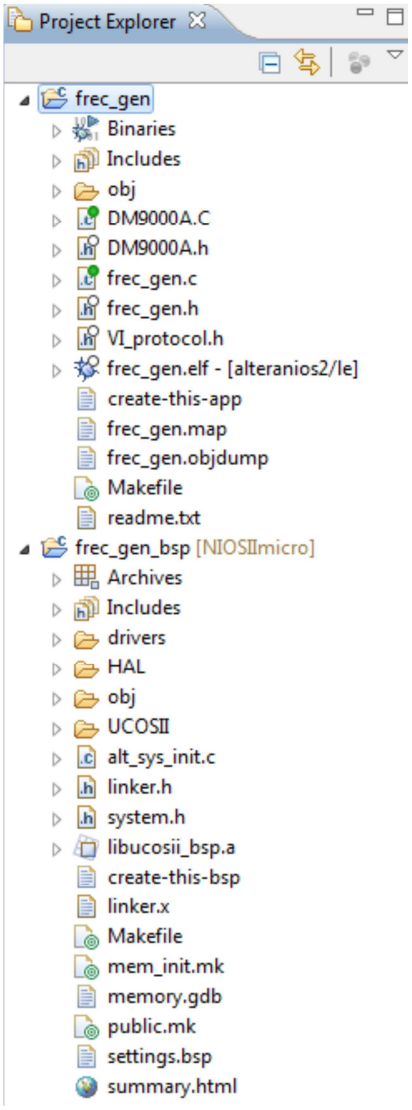

Fig. 8. Ventana explorador de proyecto en Eclipse.

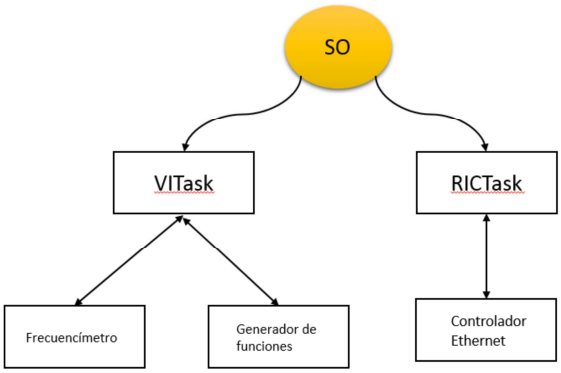

Fig. 9. Diseño del programa servidor.

Se diferencian dos tareas:

- VITask (*Virtual Instrument Task*): en función del instrumento configurado (frecuencímetro o generador

de funciones) se ejecuta la función asociada de cada instrumento.

- RICTask (*Refresh Instrument Configuration*): tarea prioritaria que actualiza la configuración del instrumento y habilita/deshabilita el instrumento actual. Controla la recepción de paquetes de la conexión Ethernet, que es a través de la interfaz por la cual se configuran los instrumentos de la placa.

Para realizar la medida de la frecuencia se utilizan dos variables, captura y captura_anterior. Se calcula la frecuencia a partir de los valores guardados en dichas variables. Cada vez que se realiza una captura, se produce una interrupción por el puerto PIO_CAPTURA_0_BASE. Esta interrupción se atiende mediante una subrutina en la que se deshabilita la interrupción, se borra el indicador, se procede a la lectura de los valores y se vuelve a habilitar la interrupción. Se realiza, además, un promediado de frecuencia de N nodos, siendo N configurable por el usuario, con el objetivo de reducir el error de frecuencia.

El generador de funciones se realiza mediante una salida de modulación de anchura de impulso con un filtro paso bajo añadido a la salida del terminal correspondiente de la placa DE2.

El código se ha hecho con el objetivo de que sea lo suficientemente modular para que se pueda ampliar a otro número de instrumentos de forma sencilla. Se divide en los archivos siguientes:

- vi_protocol.h: contiene las órdenes del VI Protocol y la estructura de los paquetes [7], [8].
- DM9000A.h, DM9000A.c: contienen el driver del controlador Ethernet Davicom DM9000A [9].
- frec_gen.h, frec_gen.c: contienen el programa principal.

El entorno Eclipse, además de soportar el lenguaje normalizado C, proporciona órdenes para acceder a los periféricos del NiosII. IOWR y IORD son las funciones que manejan la entrada y salida de datos de los puertos PIO del microprocesador:

- IOWR (dirección base, offset, valor): carga el valor en el registro cuya dirección base y offset se indica.
- IORD (dirección base, offset): lee el valor del registro cuya dirección base y offset se indica.

Las direcciones base de cada elemento se encuentran definidas en el archivo system.h, que se encuentra en la carpeta frec_gen.bsp.

El programa principal crea las dos tareas VITask y RICTask con la orden del sistema operativo *OSTaskCreateExt* y lanza el sistema operativo para que conmute entre las dos tareas y se ejecuten ambas con la función *OSStart*(). Para ello se sigue una estrategia de reparto de tiempo de procesador basada en prioridades.

B. *Cliente*

El programa del usuario facilita la reconfiguración remota de los instrumentos disponibles en la placa de desarrollo y está diseñada, al igual que la placa para ser fácilmente ampliable a otro número y tipo de instrumentos.

Se implementa en *Python* y *Qt* (Fig. 10) [10], lo que hace que sea una aplicación multiplataforma. Para desarrollar la aplicación se hace en el entorno *Python* (x, y), que incluye, entre otros, los siguientes programas:

- *Spider*: es un editor de programación. Es muy útil porque integra diversas funcionalidades (explicación de funciones, consola integrada, etc.) lo que simplifica el desarrollo de del sistema.
- *Qt Designer*: esta herramienta permite diseñar la interfaz gráfica deseada [11].

El aspecto final del interfaz gráfico del sistema en modo frecuencímetro se muestra en la Figura 11.

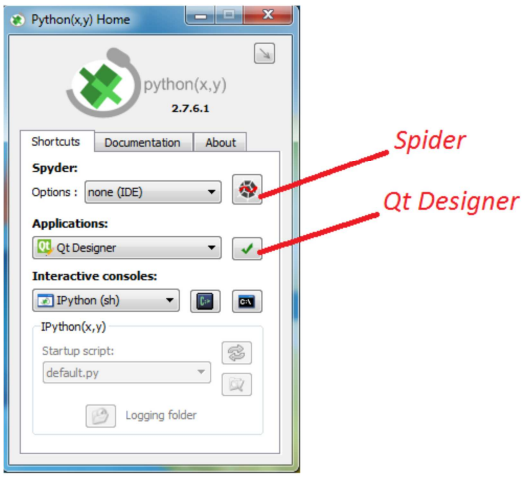

Fig. 10. Entorno Python.

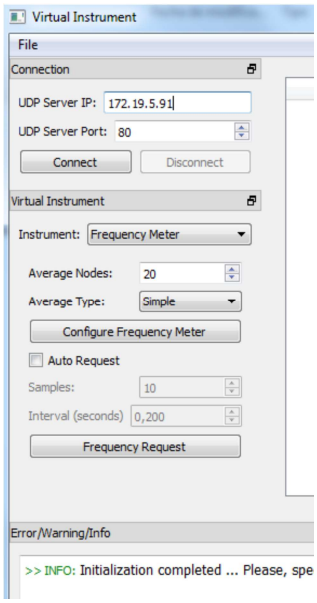

Fig. 11. Interfaz gráfico cliente en modo frecuencímetro.

IV. FUNCIONAMIENTO

Para el servidor se debe introducir la IP y MAC deseada en los archivos frec_gen.h (campos DEFAULT_IP_SOURCE, DEFAULT_SOURCE_MAC) y DM9000A.h (campo *ether_addr*). A continuación se configura (con *Quartus II Programmer*), y se programa (con *Eclipse*) la placa de desarrollo. Por último se conecta el cable Ethernet entre la placa y el conmutador (*switch*) o enrutador (*router*) de la red.

Para el cliente se indica la IP y puerto del servidor y se activa 'Connect'. A continuación se selecciona el instrumento deseado y su configuración.

Para calcular la precisión del frecuencímetro se compara la desviación típica con respecto a la de un osciloscopio Tektronix THS710 [12]. Para cada frecuencia del generador de funciones PM5192 [13] se obtienen los dos valores extremos que mide el osciloscopio (Osciloscopio (1) y Osciloscopio (2)) y calcula su desviación típica:

$$Desv.típica = \sqrt{[(fmax-f)^2+(fmin-f)^2]}$$

En la que fmax y fmin son los extremos del intervalo de frecuencias que mide el osciloscopio para una determinada frecuencia f de salida del generador de funciones Philips PM5192. La desviación típica obtenida en la FPGA para cada una de las frecuencias de entrada se calcula de una forma similar.

La Figura 12 muestra la desviación típica entre el osciloscopio y el sistema desarrollado.

El generador de señales permite definir cualquier señal periódica con 36 valores definidos por el usuario. Las figuras 13 y 14 muestran, respectivamente, la definición en el cliente de una señal arbitraria y el aspecto de dicha señal en el osciloscopio.

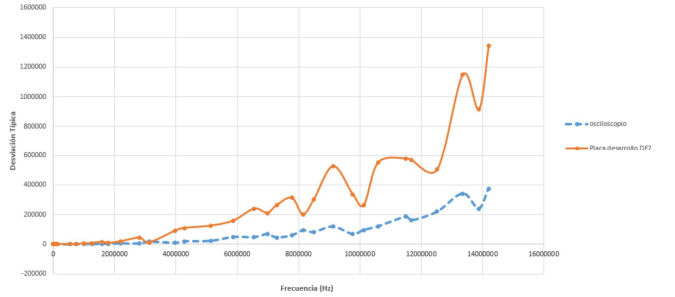

Fig. 12. Comparativa de la desviación típica entre el osciloscopio Tektronix THS710 y la placa de desarrollo Altera DE2 .

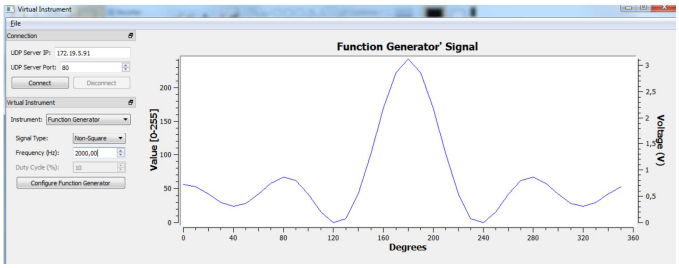

Fig. 13. Definición de una señal arbitraria en el cliente.

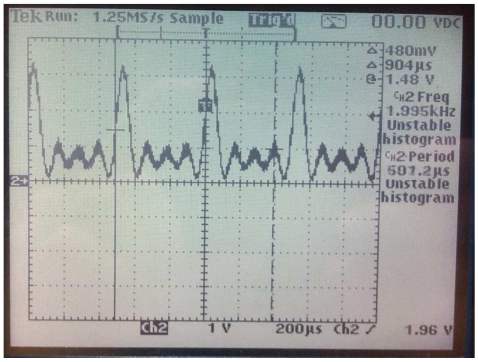

Fig. 14. Visualización en el osciloscopio de la señal arbitraria generada.

V. CONCLUSIONES

Se ha aprendido mucho y obtenido interesantes experiencias. Algunas de estas se comparten en este apartado.

La utilización de un microprocesador empotrado en una FPGA permite hacer mediante hardware las partes que necesiten mayor rapidez y fiabilidad. El software proporciona una gran flexibilidad para hacer cambios en el funcionamiento, como por ejemplo modificar los valores a introducir en la RAM y generar formas de onda distintas.

Otro aspecto que se quería analizar al comenzar a trabajar en estos temas se trataba de la importancia de los sistemas operativos. La conclusión es que el trabajo sin el sistema operativo Micro/OS-II para el control de los instrumentos (sin la conexión Ethernet activada) no refleja diferencias de rendimiento. Sin embargo, el SO facilita la programación, principalmente la gestión de interrupciones, y sobre todo la gestión de tiempo del procesador una vez que se incorpora la tarea que controla la conexión Ethernet.

Las versiones de las herramientas para el desarrollo se han ido actualizando de acuerdo con las actualizaciones de los fabricantes. A pesar de ello, especialmente *QuartusII* y *Eclipse*, demuestran tener problemas de madurez e inconsistencia en los procesos. Lo que obliga a tener muy claros los pasos que se deben dar para no encontrar errores cuya solución parece a priori imposible. Un ejemplo de un problema nada intuitivo con los mensajes de error que proporcionan las herramientas es que no se puede modificar de forma alguna, aunque sea pequeña, el diseño hardware y continuar en el mismo proyecto de *Eclipse*. Cualquier cambio en la parte hardware del proyecto (Altera *QuartusII* y *Qsys*) exige la creación de un proyecto nuevo en *Eclipse*.

Los resultados para los estudiantes de las dos titulaciones mencionadas siempre fueron muy satisfactorios. Consideraban que los trabajos eran muy interesantes para su desarrollo profesional. Adicionalmente los trabajos eran evaluados por los tribunales de fin de carrera de forma muy positiva.

AGRADECIMIENTOS

Los alumnos que han participado en estos trabajos sin los que no hubiese sido posible aprender tanto son: Alfredo Saa Barros, Marcos García Argibay, Rubén Losada Bastos, Victor González Peixoto, Pablo Ríos Costas y Miguel Domínguez Campos.

BIBLIOGRAFÍA

[1] E. Mandado, J.L. Martín, “Sistemas electrónicos digitales,” Marcombo, 2015.

[2] Altera, “ DE2 User Manual”, 2012.

[3] Altera, “Nios II Processor Reference Handbook”, 2014.

[4] Altera, “Embedded Design Handbook”, 2011.

[5] Altera, “Nios II Hardware Development Tutorial”, 2011.

[6] Altera, “Nios II Software Developer's Handbook”, 2014.

[7] R. Hollenbeck, R, “The IEEE 802.3 Standard (Ethernet): An Overview of the Technology”, 2001.

[8] C.E. Spurgeon, “Ethernet: The Definitive Guide”. Sebastopol, California: O'Reilly & Associates, Inc., 2000.

[9] DM9000A Data Sheet.

[10] H. Röst, “Python Programming”, Wikibooks, 2013.

[11] M. Summerfield, “Rapid GUI Programming with Python and Qt”, Prentice Hall 2007.

[12] Tektronix, “THS710 & THS720 User Manual”.

[13] Philips, “PM5192 Programmable Synthesizer Operating Manual”.