DISEÑO E IMPLEMENTACIÓN ÓPTIMA DE PERIFÉRICOS DE DSP CON SYSTEM GENERATOR PARA MICROBLAZE

J. Viejo, E. Ostua, M.J. Bellido, J. Juan, A. Millan, P. Ruiz-de-Clavijo y D. Guerrero

Instituto de Microelectrónica de Sevilla-Centro Nacional de Microelectrónica
Av. Reina Mercedes, s/n (Edificio CICA)-41012 Sevilla (España)
Tel.: +34 955056666 - Fax: +34 955056686
http://www.imse.cnm.es

Departamento de Tecnología Electrónica-Universidad de Sevilla
Av. Reina Mercedes, s/n (E.T.S. Ingeniería Informática)-41012 Sevilla (España)
Tel.: +34 954556161 - Fax: +34 954552764
http://www.dte.us.es

{ julian ; ostua ; bellido ; jjchico ; amillan ; paulino ; guerre } @dte.us.es

ABSTRACT

Con este trabajo pretendemos analizar como se lleva a cabo el diseño de periféricos de DSP utilizando uno de los nuevos entornos de diseño de alto nivel: System Generator for DSP.

Así, en este documento el objetivo es mostrar por un lado las características del entorno de diseño y las herramientas utilizadas y por otro la metodología de diseño e implementación de periféricos de DSP para MicroBlaze usando System Generator. Asimismo, se presenta un ejemplo de diseño que permitirá demostrar la eficacia de la metodología propuesta.

1. INTRODUCCIÓN

La evolución tecnológica ha dado lugar a un aumento tan significativo de la densidad de integración que está propiciando que cada vez más partes de un sistema completo estén incluidas dentro del núcleo principal del mismo constituido por un único chip. Si bien, tradicionalmente se ha venido llamando a este chip Circuito Integrado (IC), debido a las múltiples partes del sistema que incluye, es más adecuado el término Sistema Integrado (IS). Este aumento de la capacidad de los sistemas integrados hace necesario establecer metodologías de diseño que permitan abordar la complejidad del diseño en unos plazos de tiempo y con unas garantías de fiabilidad razonables. Para lograrlo hay que establecer metodologías para las diferentes partes del sistema integrado que permitan realizar diseños al más alto nivel posible, pero con garantías de que la implementación final funciona de acuerdo a las especificaciones del sistema. Esto sólo es posible con herramientas que automaticen los pasos desde el diseño a alto nivel hasta la implementación final.

Uno de los esquemas de mayor interés en los sistemas integrados es aquél que incluye un microprocesador como elemento principal de control de las operaciones y, sobre él, un conjunto de periféricos específicos que realicen las operaciones a más bajo nivel que el sistema requiera. Son de particular interés los periféricos que realizan funciones de procesado digital de señal. Este tipo de funciones están recibiendo una atención especial por parte de las empresas desarrolladoras de CAD para el diseño de sistemas integrados. Así, por ejemplo, en la conferencia DATE 2005 celebrada en Marzo de ese año, se desarrolló un panel de discusión sobre las metodologías de diseño de DSP comparando la metodología tradicional basada en codificación directa en HDL con el empleo de nuevas herramientas de diseño a más alto nivel. En este panel participaron empresas de software como Xilinx, Altera y Synopsis, entre otras [1]. La conclusión, casi unánime, fue la de que actualmente es mucho más eficiente el diseño con los nuevos entornos automáticos que el método basado en codificación directa. El único argumento a favor de la codificación directa está en que se pueden obtener diseños de mejor rendimiento en cuanto a recursos y velocidad de operación. Sin embargo, en algunos trabajos se han hecho comparaciones de ambas metodologías de diseño y los resultados muestran que las diferencias en cuanto al rendimiento final del sistema son muy pequeñas[2].

En este trabajo vamos a analizar el diseño de periféricos de DSP empleando uno de estos nuevos entornos de diseño de alto nivel, en concreto, el plugin System Generator for DSP[3] para MATLAB/Simulink desarrollado por Xilinx para implementación de DSP sobre FPGA's.

El trabajo está organizado como sigue: el siguiente apartado está dedicado a presentar la metodología

concreta de diseño de DSP con System Generator. Así, en primer lugar se presenta el entorno y herramientas que se emplean y, posteriormente el proceso de diseño que se sigue. Para ver la eficiencia de la metodología propuesta, en el apartado 3 se presenta un ejemplo concreto de diseño realizado en el marco de un proyecto de desarrollo e innovación empresarial. Por último, en las conclusiones se destacan los principales resultados de este trabajo

2. METODOLOGÍA SEGUIDA EN EL DISEÑO DE PERIFÉRICOS DE DSP

Como hemos mencionado anteriormente vamos a establecer la metodología de diseño de periféricos de DSP que hemos seguido en el desarrollo del proyecto OPENRTU. Hemos dividido este apartado en dos subapartados: en el primero vamos a analizar el entorno y las herramientas de diseño y en el segundo vamos a explicitar el proceso seguido en el diseño de periféricos de DSP.

2.1. Entorno de desarrollo y herramientas

En la Figura 1 se muestra como quedan estructuradas y enlazadas las diferentes herramientas empleadas en la metodología que se sigue para el diseño de funciones de DSP.

System Generator para DSP es una plataforma software que usa las herramientas de The MathWorks MATLAB/Simulink para representar una visión abstracta de alto nivel del sistema de DSP, y que automáticamente genera el código VHDL de la función de DSP desarrollada usando los más optimizados LogiCOREs de Xilinx, asegurándonos así que se ha producido la implementación más eficiente del diseño.

De esta forma, System Generator permite modelar directamente mediante un entorno de alto nivel muy flexible, robusto y fácil de utilizar sistemas de DSP innovadores y de alto rendimiento para una plataforma hardware específica. Un diseño desarrollado con esta herramienta puede componerse de una gran variedad de "elementos": bloques específicos de System Generator, código de un lenguaje de descripción de hardware tradicional (VHDL, Verilog o EDIF) y funciones derivadas del lenguaje programación MATLAB. Así, todos estos elementos pueden ser usados simultáneamente, simulados en conjunto y sintetizados para obtener una función de DSP sobre FPGA. El aspecto más interesante de trabajar en MATLAB/Simulink es poder emplear la poderosa herramienta de simulación de sistemas Simulink para realizar la verificación del diseño. En este punto, es necesario destacar que gracias a System Generator existe una interfaz de comunicación entre Simulink y el simulador lógico ModelSim[4] para realizar la simulación de las partes del sistema codificadas en HDL.

Además, la aparición de cores de procesadores empotrados ha abierto nuevas puertas al diseño de aplicaciones de procesado digital de señal. Al mismo tiempo, System Generator para DSP posibilita que los diseños puedan ser extendidos para crear periféricos para estos procesadores. En concreto, en la metodología que se va a proponer el microprocesador utilizado es MicroBlaze[5]. A continuación, vamos a presentar algunas características funcionales y arquitecturales del mismo:

MicroBlaze es un microprocesador RISC de 32 bits desarrollado íntegramente por Xilinx Inc con una licencia comercial que permite la adaptación del mismo para aplicaciones SoC empleando unas herramientas de desarrollo propias: Xilinx EDK[6], *Embedded Development Kit*. Se distribuye como un *netlist* parametrizable, por lo que su código no está disponible para poder ser modificado. Está diseñado específicamente para la implementación sobre las familias de FPGA de Xilinx y, por tanto, resulta estar muy optimizado para estos dispositivos.

MicroBlaze dispone de una unidad entera (pipeline de 3 etapas) con 32 registros de propósito general de 32 bits, de cachés con arquitectura *Harvard*, es decir, cachés separadas para instrucciones y datos, además de un módulo de *debug* (MDM), controlador para memorias FLASH, SRAM, SDRAM y DDR SDRAM, interfaces *ethernet* y PCI, UART, una unidad de FPU de precisión sencilla comercial y la posibilidad de usar memorias RAM *on-chip* de acceso rápido. Incluye además soporte para un coprocesador propio.

Los buses internos usados por MicroBlaze para conectarse a los periféricos son: LMB (*Local Memory Bus*), OPBv2 (*On-chip Peripheral Bus*) de IBM y FSL (*Fast Simplex Link*).

También, vamos a destacar que este microprocesador tiene su propio juego de instrucciones, con dos formatos de instrucciones y dos modos de direccionamiento: inmediato y con desplazamiento, proporcionando instrucciones de multiplicación y división, con 3 y 34 ciclos de latencia respectivamente, pero sólo disponibles si la FPGA sobre la que se implementa incluye multiplicadores integrados en *hardware* (Virtex-II, Spartan-II o superior).

Una vez presentadas las características, podemos justificar la elección de este microprocesador basándonos en las dos razones siguientes:

1. System Generator dispone de un tipo de compilación, denominada *OPB Export Tool*, que genera directamente los archivos necesarios para importar el modelo como un periférico *CoreConnect On-Chip Peripheral Bus*. Así, los periféricos de DSP generados por System Generator para MicroBlaze pueden conectarse directamente y trabajar en unión con dicho microprocesador.
2. Xilinx ofrece un Kit denominado *Embedded Development Kit* que proporciona a los diseñadores a través de su plataforma de estudio para sistemas empotrados (XPS) un conjunto de herramientas para atender parte de las necesidades generadas en las fases

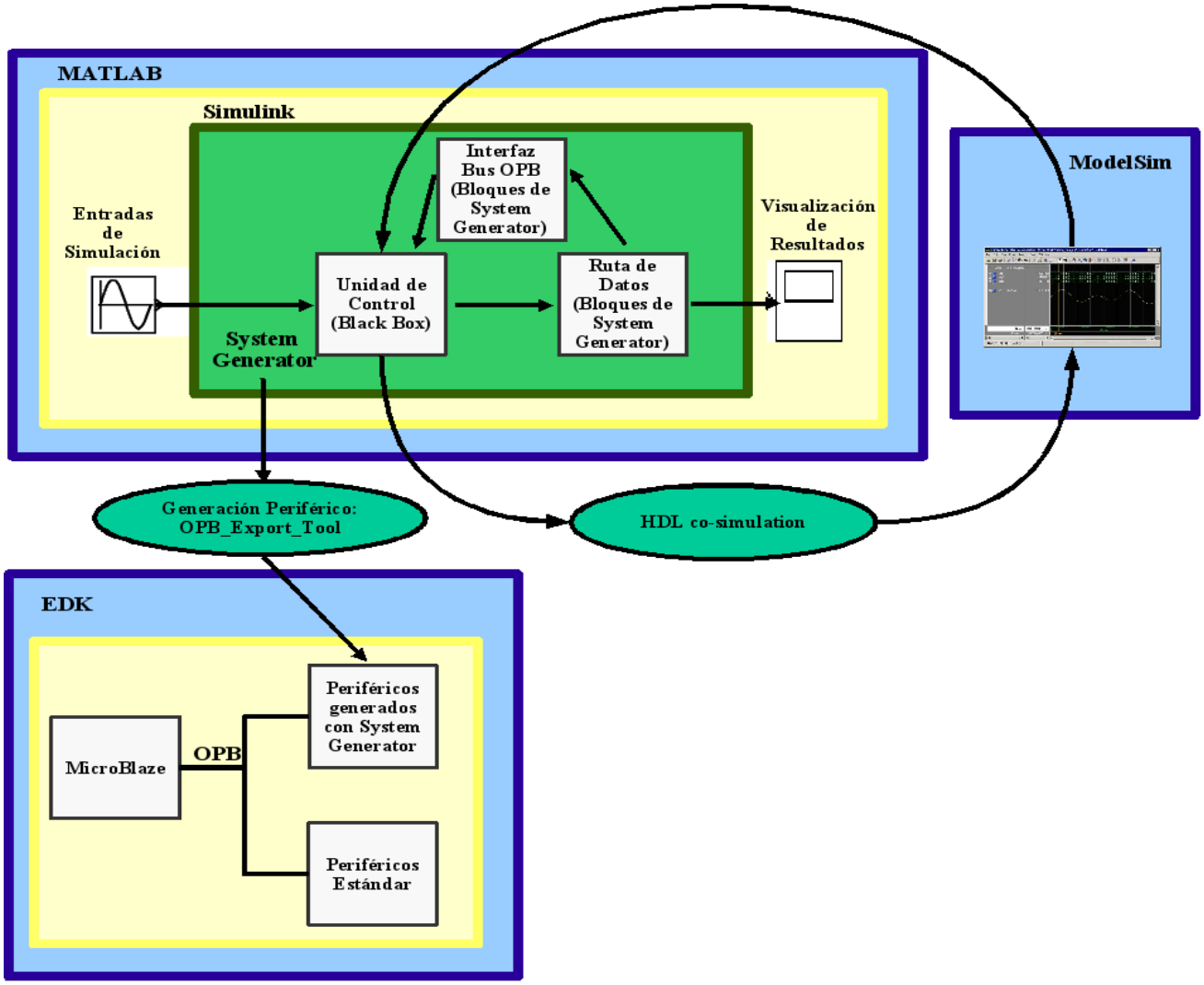

Figura 1: Entorno y herramientas utilizadas

de creación del diseño y una amplia selección de periféricos estándar para el desarrollo de sistemas empotrados basados en MicroBlaze. Por tanto, resulta que gracias a este entorno de trabajo y a sus herramientas tenemos un mecanismo sencillo para ir abordando las distintas fases de la metodología propuesta, lo que por otro lado se traduce en una disminución del tiempo de desarrollo del diseño.

2.2. Metodología de Diseño

A continuación, vamos a presentar la metodología seguida para diseñar funciones de DSP[3,7]. Como observamos en la Figura 2, en esta metodología, pueden distinguirse dos flujos de diseño claramente diferenciados. Dentro del primero, detallaremos como se lleva a cabo, usando las herramientas MATLAB/Simulink y System Generator, el modelado del periférico de DSP, su extensión y la generación del periférico para el OPB. Dentro del segundo, describiremos el flujo de diseño de sistemas empotrados basados en microprocesador (también conocidos como *System On Chip*), empleando la plataforma de estudio de Xilinx para sistemas empotrados *Xilinx Platform Studio (XPS),* incluida dentro del *Embedded Development Kit (EDK)*.

2.2.1. Flujo de diseño con MATLAB/Simulink y System Generator

En la Figura 2 se muestra el flujo de diseño seguido a la hora de diseñar un periférico de DSP para MicroBlaze usando las herramientas MATLAB/Simulink y System Generator. A continuación, pasamos a realizar una descripción del mismo.

En primer lugar, el diseñador creará un modelo del sistema de DSP, que en el caso de esta metodología va a estar compuesto por una ruta de datos, diseñada utilizando los bloques del Xilinx Blockset y del Xilinx Reference

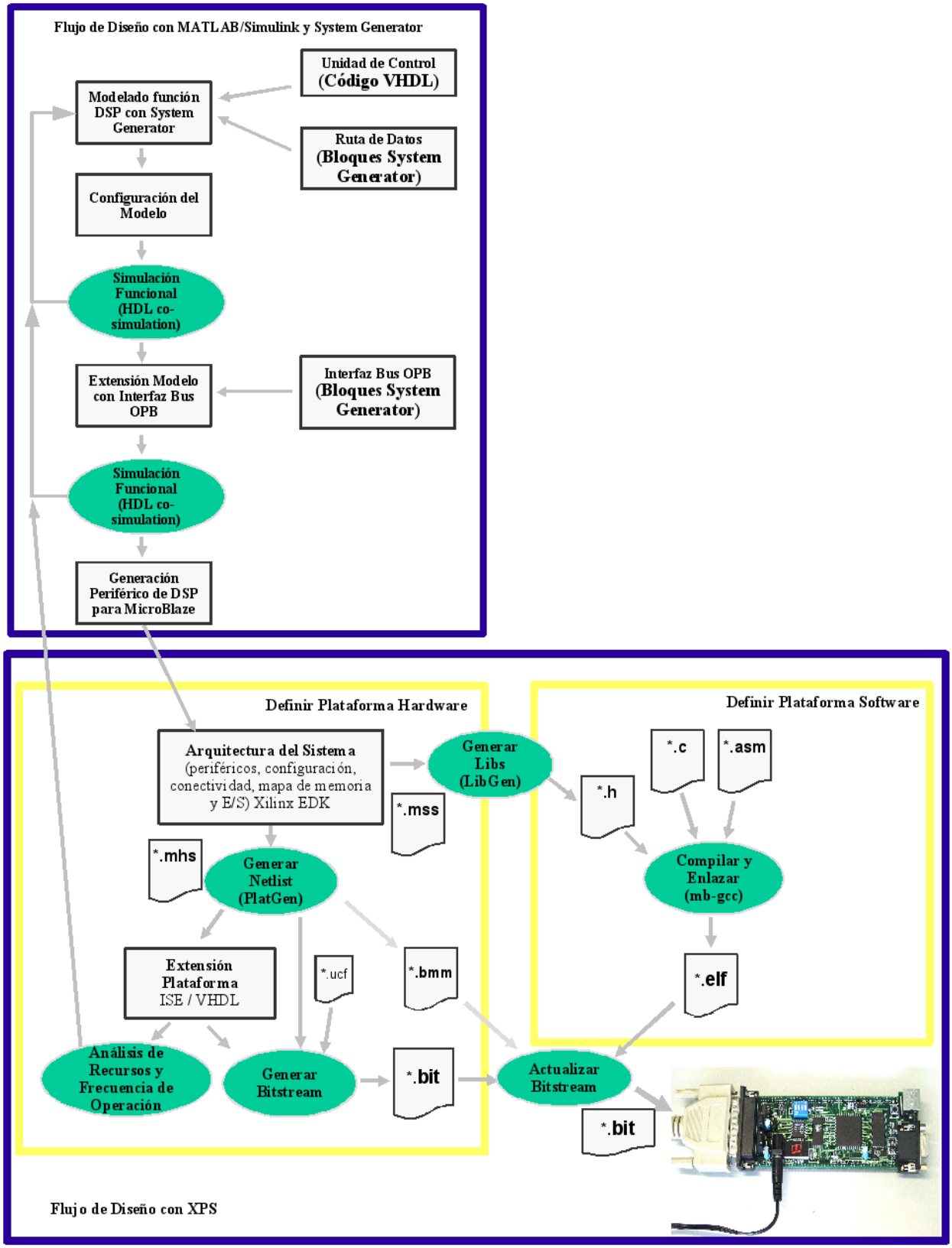

Figura 2: Metodología de diseño e implementación seguida

Blockset y por una unidad de control, descrita mediante código VHDL, que se encargará de controlar el flujo de datos de la ruta modelada mediante la generación de salidas de control. Aquí es preciso aclarar que System Generator nos permite importar diseños o módulos escritos en un lenguaje de descripción de hardware como puede ser VHDL y Verilog gracias a la System Generator Black Box Interface, que aparece dentro del Xilinx Blockset y se comporta exactamente igual a otros bloques de System Generator.

Una vez modelado el sistema, procederemos a la configuración de los bloques que forman el diseño, estableciendo el tipo de datos de las señales, el tratamiento del overflow y la saturación, la latencia de cada uno, el uso de recursos hardware disponibles en el dispositivo programable como pueden ser multiplicadores empotrados o memorias BRAM, etc.

Cuando se ha finalizado la configuración del sistema, el siguiente paso a realizar es la simulación funcional del mismo, de forma que podamos verificar si su comportamiento es el adecuado o no. Al disponernos a simular el modelo hay que tener en cuenta si hemos usado la opción anteriormente descrita de la Black Box Interface, ya que esto condicionará el tipo de simulación que se ha de efectuar, en cuanto que si no se emplea esta interface se simulará únicamente con Simulink, mientras que si la empleamos se llevará a cabo lo que se ha venido a denominar *HDL co-simulation*. En el primero de los casos, la simulación la realizará Simulink, que se encargará de la generación de los estímulos de entrada al diseño (mediante los Simulink sources), del cálculo de los resultados y de su visualización (mediante los Simulink sinks). En el segundo de los supuestos, que es precisamente el que se va a emplear en esta metodología, usando HDL co-simulation, a todo lo indicado anteriormente habría de añadirse la consideración de que las cajas negras (Black Boxes) pueden ahora participar directamente en la simulación. Esto es posible porque System Generator incluye una interfaz dinámica y flexible al simulador ModelSim de Mentor Graphics. System Generator simula estos componentes lanzando al comienzo de la simulación el entorno de ModelSim, y acto seguido, generando el código HDL adicional que necesite (propio de la simulación). En este instante, todo el código HDL es compilado por ModelSim. Finalmente, una vez concluida la compilación, System Generator comienza a programar eventos de simulación, controlando el intercambio de información entre Simulink y ModelSim.

Una vez que la simulación funcional del modelo se ajusta a las especificaciones, procederemos a la extensión del diseño añadiéndole la interfaz lógica con el bus OPB [8] (construida igualmente con bloques del Xilinx Blockset). A continuación, vamos a comentar algunos aspectos genéricos que deben contemplarse a la hora de diseñar esta interfaz:

1. Es necesario definir los puertos de entrada y salida de acuerdo al estándar OPBv2.0.
2. El diseñador debe concretar como se va a realizar la puesta en marcha del periférico.
3. Esta interfaz también debe encargarse de decodificar la dirección, decidir si se encuentra dentro del espacio de direcciones asignado al periférico y en caso afirmativo generar una señal ack de periférico seleccionado.
4. El periférico de DSP utiliza una serie de parámetros de configuración que MicroBlaze antes de la puesta en marcha del periférico debe registrar en las posiciones de memorias asignadas a cada uno. Asimismo, una vez realizado el procesado de las señales de entradas, los datos calculados por el periférico deben estar accesibles de forma que MicroBlaze los pueda leer.

Antes de la generación del periférico, debemos asegurarnos que el modelo extendido se sigue comportando correctamente, por lo que volveremos a simular el sistema pero en este caso incluyendo también la lógica de conexión con el bus OPB.

Finalmente, cuando hayamos comprobado mediante simulación que el periférico se comporta adecuadamente pasaremos a ejecutar el último paso dentro del flujo de diseño con MATLAB/Simulink y System Generator for DSP: la generación del periférico de DSP para MicroBlaze.

Para dicha generación, es necesario instalar un plugin adicional que incorpora a System Generator un nuevo tipo de compilación denominada OPB_Export_Tool[8]. Cuando tengamos instalada esta opción, para generar el periférico, en primer lugar, debemos abrir el cuadro de diálogo del bloque System Generator. Dentro de este cuadro, debemos seleccionar el nuevo tipo de compilación instalada así como un proyecto EDK en el que el periférico será importado. Además, podemos configurar otras opciones comunes como el dispositivo programable sobre el que se va a implementar el diseño, la herramienta de síntesis, el periodo de reloj de la FPGA o el lenguaje de descripción de hardware utilizado. Para comenzar el proceso de generación pulsamos dentro de dicho bloque el botón *Generate*.

Cuando System Generator termina de compilar el diseño tendremos disponible el nuevo periférico como si se tratara de un periférico estándar más de MicroBlaze.

2.2.2. Flujo de diseño con Xilinx Platform Studio

Una vez generado el periférico de DSP para MicroBlaze comienza la segunda parte de la metodología propuesta que abarca el diseño e implementación de una aplicación SoC basada en MicroBlaze donde tendremos como elemento destacado el periférico creado con System Generator. Por tanto, como podemos observar en la Figura 2, planteamos una metodología que se puede interpretar como un doble flujo de diseño en el que, a partir de las especificaciones de la aplicación SoC a desarrollar, debemos considerar tanto el flujo de diseño de la

plataforma hardware como el de la plataforma software[5,6,9].

2.2.2.1. Flujo de diseño de la Plataforma Hardware

En este apartado, vamos a describir todos los pasos que debemos realizar para obtener la plataforma hardware del diseño.

Como se muestra en la Figura 2, en primer lugar el diseñador deberá definir la arquitectura del sistema mediante la creación de un archivo de especificaciones hardware (se trata de un archivo con extensión MHS: *Microprocessor Hardware Specification*) que incluirá información relativa a:

1. La configuración del microprocesador: frecuencia de reloj, tamaño de la memoria de datos e instrucciones local, habilitación del módulo de debug y de la cache, etc.
2. El periférico de DSP generado y del resto de periféricos estándar que componen el diseño. Para cada uno especificaremos el bus al que está conectado, el rango de direcciones asignado dentro del mapa de memoria, así como una serie de parámetros de configuración propios de cada periférico.
3. La declaración de los puertos de entrada/salida.

Una vez definida la arquitectura del sistema, utilizamos la herramienta PlatGen (Platform Generator) para crear la plataforma hardware a partir del fichero MHS. PlatGen construye el sistema de procesado empotrado como un conjunto de listas de conexionado de hardware, es decir, ficheros HDL y netlists de implementación, que serán el punto de partida para otra herramienta (Xflow) que se encargará de implementar el diseño sobre un dispositivo FPGA concreto. Para ello, Xflow ejecuta las herramientas de implementación del entorno ISE [10]:

1. Traslación (**ngdbuild**).
2. Mapeo tecnológico (**map**).
3. Place and Route (**par**).

Para realizar este proceso es necesario definir el fichero de restricciones con la localización de los pines I/O de la FPGA (archivo UCF).

En este punto, nos encontramos en una parte importante dentro de la metodología como es el análisis de los recursos utilizados y de la frecuencia de operación. Para realizar este análisis vamos a utilizar los *reports* que genera automáticamente la herramienta Xflow.

Así, en lo que se refiere a los recursos podemos decir que estos *reports* proporcionan, por un lado, información acerca del total de recursos utilizados por el sistema completo, y por otro, información específica del periférico generado y conectado a MicroBlaze: número de slices, flip-flops, BRAMs, LUTs, IOBs y multiplicadores empotrados utilizados. Asimismo, podemos encontrar información acerca de la frecuencia de operación: frecuencia máxima conseguida y descripción de los caminos críticos, especificándose en este segundo caso las señales que los forman y el retraso que añade cada una. De forma opcional, podemos utilizar otra herramienta del entorno ISE llamada Time Analyzer, siendo su uso recomendable debido a que incluye un *wizard* que nos proporciona una serie de recomendaciones de como podemos solucionar dichos caminos críticos.

A partir de los resultados obtenidos, debemos concluir si el diseño cumple las especificaciones, de forma que procederemos a la generación del fichero Bitstream de configuración de la FPGA (herramienta **bitgen**), finalizando así el flujo de diseño de la plataforma hardware, o no las cumple, teniendo en este caso que volver a System Generator, realizar las modificaciones del diseño que se consideren oportunas y generar de nuevo el periférico.

2.2.2.2. Flujo de diseño de la Plataforma Software

Para completar el diseño del sistema es necesario crear unos drivers que permitan controlar el funcionamiento del periférico generado: funciones que permitan la escritura de los registros de configuración y la lectura de los datos de salida. Una vez que dispongamos de estos drivers realizaremos la definición de un fichero de especificaciones software (archivo con extensión MSS: *Microprocessor Software Specification*), a través del cual el diseñador especifica:

1. El nombre del driver y la versión utilizada para cada componente instanciado en el diseño: MicroBlaze, periférico de DSP y resto de periféricos estándar.
2. El modo de depuración del procesador, asignando el componente encargado de la comunicación en la depuración.
3. Los dispositivos estándar de entrada/salida del sistema.
4. La frecuencia de trabajo del procesador y el manejador de interrupciones asociado a la señal de interrupción.
5. Otras características relacionadas con el software: selección de librerías y de sistema operativo, etc.

En la Figura 2 observamos como la herramienta LibGen (Library Generator) toma como entrada este fichero MSS y genera las librerías que utilizarán las aplicaciones software del sistema empotrado.

Gracias a las librerías generadas por LibGen, podemos abordar a partir de este momento el desarrollo de software de aplicación. Para ello, XPS incorpora un gestor de proyectos software que permite la edición del código fuente del programa (escrito en lenguaje C, C++ o en ensamblador), cuya finalidad básica es explotar funcionalmente los recursos de la plataforma hardware diseñada.

Siguiendo con el flujo de diseño, XPS proporciona un conjunto de herramientas de desarrollo GNU, que

permiten obtener el fichero ejecutable (archivo ELF), y que pasamos a describir a continuación:

mb-gcc. Esta herramienta realiza las siguientes tareas:
1. Preprocesado: Expande los ficheros fuentes incluyendo los ficheros de cabecera.
2. Compilación: Verifica la sintaxis y genera el código ensamblador.
3. Llamada al ensamblador **(mb-as)**: Convierte el código ensamblado a lenguaje máquina (código objeto).
4. Llamada al linker (**mb-ld**): Combina los módulos binarios: código objeto, librerías y ficheros de inicialización. Determina el mapa de memoria del ejecutable.

Con la obtención del fichero ejecutable hemos finalizado el flujo de diseño de la plataforma software, pero aún queda el paso final de esta metodología: la conjunción de ambos flujos, es decir, la implementación de la aplicación SoC en una plataforma hardware concreta.

2.2.2.3. Implementación de la Aplicación SoC en Hardware

Una vez completados los flujos de diseño de la plataforma *hardware* y *software* tenemos, por un lado, la arquitectura hardware, y por otro, la aplicación software que correrá sobre el microprocesador.

Para completar nuestra metodología de diseño faltaría:

1. Poner a trabajar a ambos componentes de forma simultánea y sobre un hardware adecuado, en este caso una placa de desarrollo provista de una FPGA.
2. Realizar la depuración del sistema final.

Para la primera tarea utilizamos la herramienta **data2bram** que genera un fichero bitstream (archivo BIT) compuesto por la configuración del dispositivo y el código de inicialización de los módulos BRAM (Memorias de Bloques) del mismo. Una vez generado, podemos iniciar la programación de la placa de desarrollo.

Para la programación se suele utilizar el *standard* JTAG, conectando la placa con el PC mediante un cable de programación a través del puerto paralelo o de uno de los puertos USB. Se usará la herramienta de *Xilinx iMPACT* para la descarga del fichero *bitstream* hacia la FPGA de la placa de desarrollo, consiguiendo en pocos segundos la programación de la misma y la puesta en marcha de nuestra aplicación SoC.

Para finalizar el proceso de diseño del sistema, podemos depurar y verificar el software desarrollado en hardware real.

La depuración y verificación del software en EDK recae sobre dos herramientas: GDB (GNU Debugger), que permite depurar y verificar el diseño sobre un simulador y XDM (Xilinx Microprocessor Debug), que permite servir datos de depuración desde un host remoto a GDB.

La depuración de la aplicación se realiza sobre la plataforma de desarrollo en tiempo de ejecución. El código de la aplicación es sustituido en el archivo bitstream de configuración e inicialización de los bloques BRAM por un programa de comunicación asociado al periférico de depuración. A través de GDB, XMD descarga el código de la aplicación y establece los puntos de ruptura así como las trazas necesarias para la verificación completa de su funcionamiento.

3. EJEMPLO DE DISEÑO

Como ejemplo de diseño, se ha modelado un periférico que realiza el procesamiento de medidas directas sobre señales analógicas adquiridas por medio de dispositivos de adecuación de señal externos, formados básicamente por transformadores de medida de sistemas trifásicos (tensiones y/o corrientes en líneas de potencia: fases R, S T y neutro). Por tanto, el periférico está diseñado para manejar 8 entradas analógicas, que se podrán organizar como entradas de tensión o de corriente. Además, las entradas de tensión y de corriente de una misma fase también se podrán asociar para poder calcular potencias.

3.1. Descripción del Diseño

En la Figura 3 se muestra un diagrama de los módulos que forman este periférico de medidas directas[11]. Como observamos en ella, en primer lugar hemos diseñado una interfaz entre el periférico y el Analog Front-End (AFE). Esta interfaz se compone de los submódulos de:

1. **Control de conversión analógico-digital**, que permite controlar la frecuencia de muestreo de las entradas analógicas. En concreto, este periférico se ha diseñado para que, según configuración, muestre a 64 ó 128 muestras por ciclo de la señal analógica de entrada, lo que se traduce en frecuencias de muestreo de: 3200 Hz (64 muestras/ciclo) y 6400 Hz (128 muestras/ciclo) para señales de entrada de 50 Hz y de 3840 Hz (64 muestras/ciclo) y 7680 Hz (128 muestras/ciclo) para señales de entrada de 60 Hz. Todas las entradas tendrán la misma tasa de muestreo.
2. **Control de ganancia**, que controla la ganancia de cada una, con idea de obtener la máxima precisión. De esta forma, se definen límites en el rango dinámico de las entradas que establecen la nueva ganancia a utilizar. El valor de la ganancia se calculará teniendo en cuenta la ganancia actual y los valores de amplitud de las señales a la salida del filtro FIR en cada ciclo de la señal eléctrica.

Para el diseño de este módulo se han implementado un código VHDL que describe el comportamiento anterior, utilizando para ello los datasheets de los dispositivos AD7656 [12] y AD5233 [13].

En segundo lugar, se encuentra la interfaz con MicroBlaze. Como se ha explicado en el apartado 2.2.1,

MicroBlaze y el periférico de DSP intercambian información en dos procesos, estando por una parte el envío de parámetros de configuración desde MicroBlaze al periférico, y por otro, el envío de datos calculados desde el periférico a MicroBlaze. En el caso del periférico de medidas directas, como parámetros de configuración disponemos de registros de control y de coeficientes de calibración. Los registros de control permiten establecer la frecuencia de muestreo, el número de ciclos de promedio y habilitar o deshabilitar la eliminación de la componente de continua en los cálculos así como el control de ganancia. Los coeficientes de calibración son utilizados por el módulo de calibración para calcular los valores de salida del periférico.

Esta parte del diseño se ha realizado con System Generator de acuerdo a como viene descrito en [8].

Finalmente, aparecen la unidad de control y la ruta de datos. En lo que se refiere a la unidad de control, como observamos en la Figura 3, ésta recibe por un lado los parámetros de configuración y por otro una señal de validación de la interfaz con el AFE que indica que se ha recibido una muestra y que debe empezar su procesamiento. Así, una vez validada la nueva muestra, la unidad de control comienza a generar un conjunto de señales de control que llegan a la ruta de datos realizándose el siguiente procesado de señal:

1. **Filtrado FIR**. Este submódulo efectúa la rutina de filtrado FIR de las 8 entradas. En concreto, el filtro implementado es un filtro FIR lineal paso de baja que elimina las componentes espectrales de alta frecuencia.
2. **Control de Offset**. Esta función calcula la componente de continua de cada una de las entradas a partir de las señales filtradas (pero no la elimina). Estos valores calculados se utilizarán en el submódulo de cálculo de parámetros instantáneos. Todos los parámetros calculados son valores promediados utilizando una media móvil. El periodo de promedio tanto para entradas de tensión como de corriente será de 1 ciclo de la señal eléctrica.
3. **Valores instantáneos**. Este submódulo calcula los valores eficaces de las 8 entradas y las potencias activas y reactivas, implicando estas dos últimas operaciones el uso de 2 entradas, una de tensión y otra de intensidad. Al igual que en el punto 2 los valores instantáneos se calculan utilizando una media móvil cuyo periodo de promedio es de 1 ciclo.
4. **Promediado**. El objetivo es realizar un promediado de los valores instantáneos calculados de acuerdo al parámetro de configuración NCP que informa del número de ciclos de promedio.
5. **Calibración**. Los parámetros instantáneos promediados deben ser calibrados. La calibración está formada por dos procesos: el primero es la corrección de distorsiones de la tarjeta y el segundo es la corrección de distorsiones externas. Para realizar el calibrado utilizaremos los coeficientes de calibración disponibles por configuración. Los resultados obtenidos por este submódulo estarán disponibles de forma que las operaciones de lectura de MicroBlaze sobre el periférico irán dirigidas a estos parámetros calculados.

A continuación, vamos a explicar brevemente como se ha afrontado el diseño de los distintos submódulos que forman la ruta de datos del periférico.

En primer lugar, para la caracterización del filtro se ha utilizado la herramienta FDATool, que permite definir los siguientes parámetros del filtro: tipo (paso de baja, de alta, de banda o de rechazo de banda), frecuencia de paso y de rechazo, rizado en la banda de paso y atenuación en la banda de rechazo. FDATool además dispone de una opción para establecer el orden del filtro, pero en este caso hemos dejado esta opción marcada como Minumun order de manera que sea la propia herramienta la que determine el orden mínimo. Una vez especificadas las características del FIR generamos los coeficientes pulsando la opción Design Filter. Para la implementación se ha utilizado el bloque de System Generator FIR que permite establecer los siguientes parámetros: coeficientes del filtro (los generados por FDATool), número de bits de los coeficientes (en este caso, 16 bits por coeficiente), número de canales (8 canales de entrada) y latencia. Otro aspecto importante es que cuando el número de canales es mayor de uno, como es el caso, las entradas al filtro pueden ser en serie o en paralelo. En este diseño, hemos optado porque la primera opción.

En segundo lugar, para el diseño de los submódulos control de offset, valores instantáneos y promediado, dado que la operación a realizar en todos es una media móvil, se han planteado rutas de datos parecidas, de manera que todas están formadas por: un sumador, lógica de multiplexación que permite seleccionar la entrada a procesar, acumuladores para registrar los parámetros intermedios calculados y finalmente un bloque que al finalizar los cálculos realice una operación de escalado (dividir por el periodo de promedio). Esta división, lejos de parecer una operación complicada de implementar en hardware, no comporta más que desplazamientos debido a que el periodo de promedio será siempre una potencia de dos.

Sin embargo, como es comprensible existen diferencias que vamos a describir a continuación:

1. Para el cálculo del control de offset y valores instantáneos el periodo de promedio se toma como un ciclo de la señal de entrada, lo que significa 64 muestras por ciclos para señales eléctricas de 50 Hz ó 128 muestras por ciclo para señales 60 Hz. Para el promediado el periodo de promedio será el establecido por el parámetro de configuración NCP (número de ciclos promedio de la señal de entrada).
2. En el submódulo de valores instantáneos es necesario añadir un multiplicador ya que para realizar los

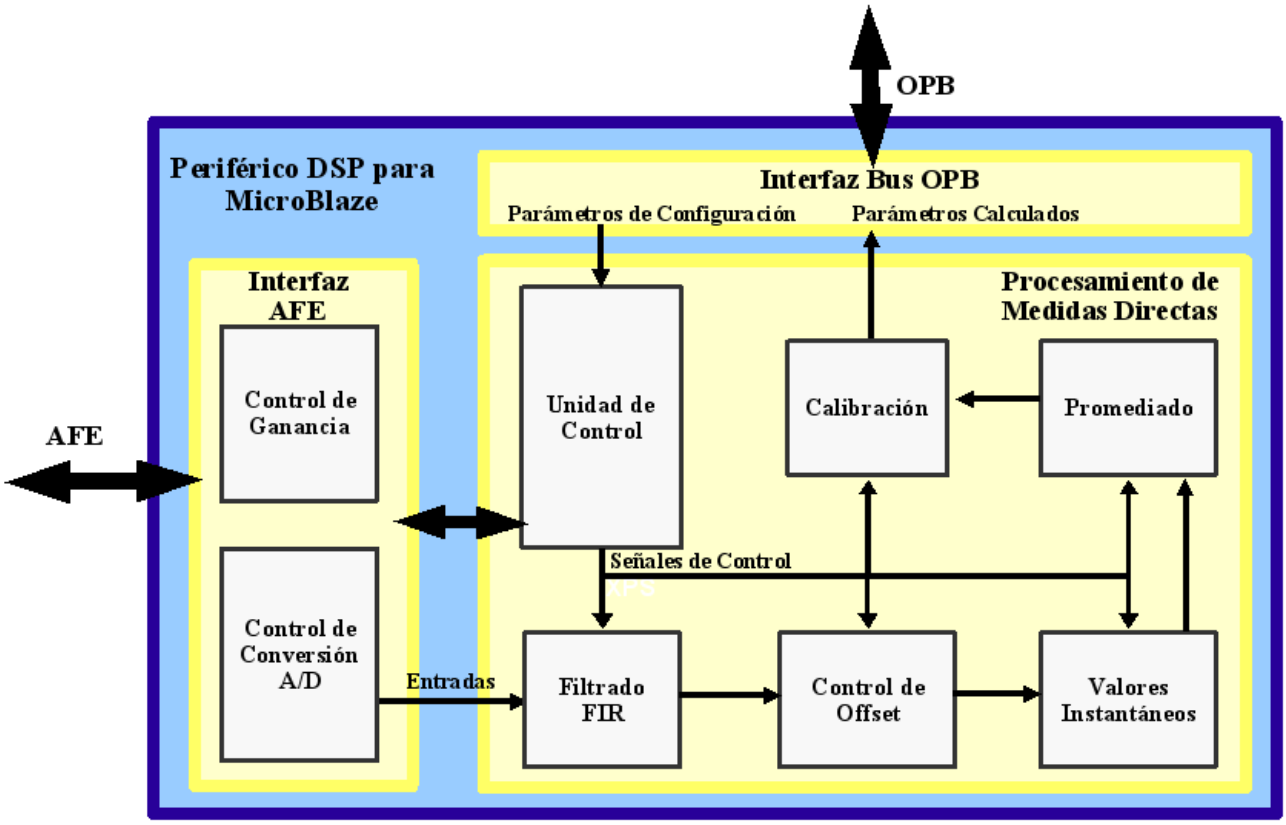

Figura 3: Esquema del diseño propuesto

promediados necesitamos las entradas al cuadrado. Además, por configuración es posible habilitar la eliminación de la componente de continua, de forma que necesitamos un restador que realice esta función cuando sea necesaria. Finalmente, el cálculo de los valores eficaces conlleva una operación de raíz cuadrada que se ha implementado mediante un compuesto de System Generator denominado CORDIC SQRT. Los compuestos de System Generator están diseñados con elementos del Xilinx Blockset, y tienen la propiedad de que pueden modificarse para adaptarlos a las especificaciones de cada diseño concreto. En el caso del periférico de medidas directas, hemos modificado este bloque de forma que en el caso de entrada negativa saque un cero como resultado.

Finalmente, para llevar a cabo el diseño del módulo de calibrado necesitamos un multiplicador y un sumador, de forma que a partir de los valores instantáneos promediados y de los coeficientes de calibración se realizarán los dos procesos descritos anteriormente y los resultados serán almacenados en registros accesibles por MicroBlaze.

3.2. Optimización de los recursos

En este apartado se van a comentar las decisiones más relevantes que se han tomado durante el proceso de diseño del periférico y que han tenido como finalidad una utilización optimizada de los recursos.

La primera decisión tomada fue la de no llevar el procesado de cada una de las ocho entradas por separado sino plantear una única ruta de datos controlada mediante una unidad de control que procesara todas a la vez. Con esta decisión estamos ahorrando recursos en el sentido de que no tenemos elementos de procesado (multiplicadores, sumadores, módulo de raíz cuadrada, etc.) específicos para cada una de las entradas sino que estos son compartidos por todas. No obstante, esto conlleva tener que introducir en el diseño lógica de multiplexación, además de tener que diseñar una unidad de control que controle el procesado de todas las señales de acuerdo a un orden establecido.

La siguiente decisión importante fue la utilización de los recursos hardware disponibles en el dispositivo programable: multiplicadores empotrados, memorias BRAMs, ect.

El último punto en el que tuvimos que detenernos y tomar una decisión fue a la hora de configurar los bloques del periférico. Sin duda, es aquí donde más abierto está el tema de la optimización, ya que las herramientas utilizadas permiten cambiar rápidamente la configuración

del diseño (sin que varíe su comportamiento funcional) y comprobar los recursos utilizados, de forma que podemos probar con distintas configuraciones hasta que alguna se adapte al número de recursos que se desea emplear. Así, el trabajo realizado en este sentido se ha centrado en los siguientes aspectos: precisión de los datos (tanto de entrada como calculados) y tratamiento del overflow y la saturación.

3.3. Optimización de la frecuencia de operación

En lo que se refiere a la optimización de la frecuencia de operación podemos decir que hemos actuado de forma similar al caso de la optimización de los recursos, en el sentido de que es en el proceso de configuración donde podemos afrontar la optimización a la que se refiere este apartado. No obstante, en el caso que nos ocupa los parámetros configurados han sido Pipeline to Greatest Extent Possible y la latencia. Así, para conseguir la frecuencia de operación deseada hemos introducido etapas de pipeline en los multiplicadores y sumadores de las distintas partes del diseño y utilizado una latencia de 8 ciclos. La decisión de colocar este valor a 8 se ha debido principalmente a que estos cambios implican tener que rediseñar la unidad de control para que incluya ese comportamiento, de forma que con esa latencia y de acuerdo a como estaba diseñada la unidad de control en un primer momento (procesamiento de las 8 entradas con todos los bloques combinacionales) nos ha resultado casi inmediato recuperar el correcto funcionamiento del periférico.

Con estas medidas hemos logrado que el periférico funcione de acuerdo a las especificaciones, es decir, a una frecuencia de al menos 75 MHz.

4. CONCLUSIONES

En este trabajo, se ha analizado el diseño de periférico de DSP empleando la herramienta de Xilinx System Generator.

Para ello, hemos explicado las características del entorno y herramientas utilizadas y planteado una metodología de diseño e implementación de periféricos de DSP para el microprocesador MicroBlaze. Por último, hemos desarrollado un ejemplo de diseño concreto, que consiste en un periférico de Medidas Directas.

5. AGRADECIMIENTOS

Este trabajo ha sido parcialmente financiado por el proyecto MCYT META TEC 2004-00840/MIC y el proyecto PROFIT-MITC OPENRTU FIT-330101-2004-5 del Gobierno Español.

6. REFERENCIAS

[1] PANEL: “DSP in FPGA: Automated 'v' Manual Tool Flows”. DATE Focus on Business and Industry. Design, Automation and Test in Europe (DATE'05), Munich, March 2005.

[2] M. Font, J. Ordeix, M. Serra, P. Marti y J. Carrabina, “Diseño de sistemas SoC mediante MATLAB-Simulink. Aplicación en sistemas de comunicación OFDM”, XVII Simposium Nacional de la Unión Científica Internacional de Radio, Alcalá de Henares (España), Septiembre 2002.

[3] Xilinx System Generator v7.1 User Guide, Xilinx Inc., 2005: http://www.xilinx.com/products/software/sysgen/app_docs/user_guide.htm

[4] ModelSim Xilinx Edition II: http://www.xilinx.com/ise/mxe2/mxe2_5.8.zip

[5] MicroBlaze Processor Reference Guide, Xilinx Inc., 2005: http://www.xilinx.com/ise/embedded/mb_ref_guide.pdf

[6] Platform Studio User Guide, Xilinx Inc., 2004: http://www.xilinx.com/ise/embedded/edk6_3docs/ps_ug.pdf

[7] R.D. Turney, C. Dick, D.B. Parlour and J.Hwang, “Modeling and Implementation of DSP Solutions”, Xilinx Inc.: http://www.xilinx.com/products/logicore/dsp/matlab_final.pdf

[8] J. Ballagh, J. Hwang, P. James-Roxby, E. Keller, S. Seng and B. Taylor, “Building OPB Slave Peripherals using System Generator for DSP”, Xilinx Inc., 2004: http://www.xilinx.com/bvdocs/appnotes/xapp264.zip

[9] I. Bravo, R. Piñeiro y J.L. Lázaro, “Explotación de la plataforma EDK para el desarrollo de sistemas empotrados basados en MicroBlaze”, International Conference on Reconfigurable Computing and FPGA's (RECONFIG'04), Colima (México), Septiembre 2004.

[10] ISE 6 In-Depth Tutorial, Xilinx Inc. 2003: http://direct.xilinx.com/direct/ise6_tutorials/ise6tut.pdf

[11] OpenRTU: Descripción detallada de los Periféricos de Procesado Digital de Señales rev.3, TELVENT Energía y Medio Ambiente S.A., 2005.

[12]Datasheets Analog Devices AD7656, Analog Devices Inc., 2005: http://www.analog.com

[13]Datasheets Analog Devices AD5233, Analog Devices Inc., 2005: http://www.analog.com