Una herramienta web para diseñar sistemas digitales basados en procesadores implementados sobre FPGA

S. de Pablo*, L. C. Herrero*, A. B. Rey** y J. A. Cebrián*.

* Departamento de Tecnología Electrónica, Universidad de Valladolid, España,
** Dpto. Electrónica, Tecn. de Computadoras y Proyectos, Univ. Politécnica de Cartagena, Cartagena, Murcia, España.

Abstract — **Este artículo presenta una herramienta web que permite diseñar rápida y fácilmente un elevado número de circuitos digitales empleando procesadores incrustados. Desde el punto de vista del usuario, esta aplicación es un conjunto de páginas web enlazadas que pueden ser utilizadas sin necesidad de instalación. Gracias a los diversos menús y botones que van apareciendo, puede ir seleccionando los distintos módulos y conexiones que desea, personalizando a continuación los parámetros disponibles para cada módulo y para cada conexión. Cuando se ha terminado el diseño, la aplicación genera automáticamente todos los ficheros necesarios para la implementación del circuito en una FPGA: una descripción HDL de alto nivel que describe la implementación de los módulos elegidos y sus interconexiones, varios ficheros precompilados que describen todos los módulos necesarios, ficheros de restricciones, asignación de pines, especificaciones del proyecto, etc. Se puede pasar de la idea al circuito físico en unos pocos minutos.**

I. Introducción

Los primeros microprocesadores aparecieron hace 30-35 años (Intel 4004, Intel 8080, Zilog Z80), los primeros dispositivos complejos tipo CPLD y FPGA se crearon en 1984 (Altera [3], Xilinx [12]), y los primeros microprocesadores implementados sobre FPGA han estado disponibles desde hace más de diez años [6]. Actualmente empieza a ser habitual el uso de herramientas de generación automática de circuitos basados en procesadores, como la aplicación *SOPC Builder* de Altera [3] y el entorno *EDK* de Xilinx [12]. Este artículo presenta, precisamente, una herramienta web de este tipo, creada para acelerar el desarrollo de circuitos de control en proyectos de investigación y proyectos fin de carrera [1][4]. Su finalidad es claramente educativa.

Hoy en día es impensable trabajar con un procesador, tanto encapsulado de forma independiente como integrado en un chip (SoC), sin utilizar un entorno de desarrollo adecuado. Es práctica habitual el uso de un compilador de C incluso para los microcontroladores más pequeños, pues el programador puede conseguir así mejores prestaciones y tanto el resultado como el propio proceso de desarrollo es más fácil de supervisar. Sin embargo, cuando tenemos que diseñar el hardware alrededor de un procesador, en muchos casos seguimos haciéndolo "a mano".

El problema que se encuentra, o debemos decir encontraba, el ingeniero de hardware es que el interfaz de entrada y salida de cada procesador es en muchos casos distinto, incluye muchas señales que no siempre se corresponden con las deseadas, y algunas restricciones temporales que imponen los distintos fabricantes de componentes no son siempre fáciles de combinar. Por tanto, el proceso de conexión de un procesador con su entorno no siempre es fácil, y suele requerir laboriosos procesos de simulación temporal y verificación física.

El panorama, sin embargo, está cambiando. Cuando se trabaja dentro de un chip, tanto en FPGA como en ASIC, no es necesario emplear un interfaz asíncrono entre los diversos módulos, sino que se pueden realizar transferencias síncronas sin penalizar el coste, alcanzando a la vez una mayor robustez y sencillez. Se consigue también una mayor regularidad, algo no sólo deseable cuando se diseña "a mano", sino prácticamente imprescindible cuando se pretende generar el código que describe el circuito de forma automática.

En este ámbito se mueve la aplicación web descrita en este artículo. Su nombre es *"Hardware Highway"*[1], "HwHw" de forma abreviada, y lo que pretende es diseñar en minutos entre el 90% y el 100% del circuito, incluso cuando se requieren varios procesadores.

II. Dominios Síncronos

Un dominio síncrono se define como aquella región de un circuito digital que emplea una única señal de reloj y que responde únicamente a uno de sus flancos. No siempre es posible resolver una aplicación con un único dominio síncrono, ni siquiera cuando sólo se requiere una única frecuencia de reloj[2]. Sin embargo, a veces es posible aplicar algunas restricciones, y entonces se evitan muchos problemas. En particular:

1) Si el circuito incluye un único dominio síncrono.
2) Si el circuito supera las pruebas correspondientes de funcionamiento en baja frecuencia, lo que se puede comprobar a través de un simulador funcional o incluso con un simulador de HDL.

[1] Se podría traducir como "Autopista para el Hardware".
[2] Por ejemplo, un *hub* de Ethernet puede trabajar a una única frecuencia, pero se ha de sincronizar con todos sus terminales, que emplean otros circuitos de reloj generalmente desfasados.

3) Y si el circuito supera las restricciones impuestas por la frecuencia de trabajo, algo que se ve inmediatamente aplicando un "Análisis Temporal Estático" que evalúa los retrasos de todas y cada una de las rutas combinacionales del circuito.

Si se dan estas tres condiciones tenemos la seguridad de que el circuito real va a funcionar correctamente, pues se tiene completa garantía de que el tiempo disponible para realizar cualquier transferencia de datos entre dos registros nunca será menor del establecido por la frecuencia de trabajo. Es evidente que estas restricciones confieren una elevada robustez al sistema.

Por supuesto, incluso cumpliendo estas condiciones, siempre tendremos que dejar cierta holgura en la frecuencia de trabajo para evitar efectos de metaestabilidad [2] en los registros, y luego quedará por comprobar la funcionalidad y los retrasos de todas las conexiones del circuito con el exterior, pues están fuera del entorno analizado.

Aplicando estas ideas a nuestro caso, podemos esperar que, si construimos un circuito sobre un dominio síncrono empleando módulos previamente verificados, y si imponemos restricciones temporales durante la síntesis e implementación del circuito para que pueda trabajar a la frecuencia deseada, entonces el circuito resultante *debería* funcionar correctamente.

III. Estructura De La Aplicación

Además de introducir restricciones en los circuitos para que sean suficientemente sencillos y fiables, pues sin eso no hubiera tenido sentido desarrollar esta aplicación, también hemos necesitado un lenguaje de programación apropiado. Hemos elegido PHP [10] porque, además de servir para el desarrollo de páginas web activas, permite manejar variables indexadas con estructura irregular. Esto ha sido importante para mantener la propia información de los diseños y para desarrollar las bases de datos que permiten saber qué módulos están disponibles y qué conexiones son posibles.

Para el diseñador esta herramienta es un conjunto de páginas web enlazadas, y el proceso de diseño consiste en ir seleccionando sucesivamente diversas opciones y pulsando botones para pasar cada vez a la siguiente página, que se genera de forma personalizada desde un servidor web. El volumen de información transferido cada vez no es muy elevado, por lo que incluso con una conexión con poco ancho de banda se obtiene un comportamiento aceptable. En el servidor, en cambio, sí se requiere cierta potencia de cálculo, pero un Pentium 4 es más que suficiente[3].

[3] Aunque de momento no se han hecho pruebas de producción con cientos de usuarios conectados simultáneamente.

La primera página de la aplicación se corresponde con la página de acceso al grupo de trabajo Epyme ("Electrónica de Potencia y Microelectrónica"), la cual realiza un proceso de identificación del usuario solicitando un nombre de usuario y una clave.

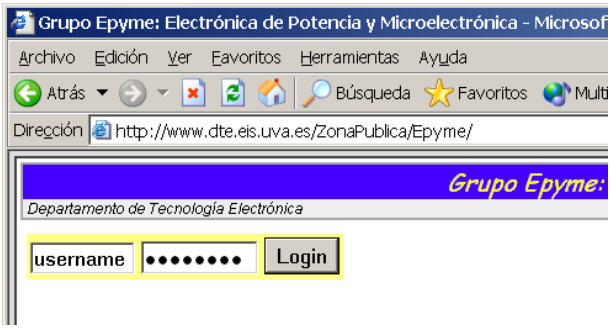

Fig. 1. Primera página web del grupo de trabajo Epyme y de la aplicación *Hardware Highway*.

Una vez se ha identificado y autentificado al usuario, se crea para él una "sesión", de modo que el servidor retiene cierta información de él mientras va pasando de una página web a otra. Cuando se cierra el navegador del cliente se pierde la información de la sesión.

Dentro de la página del grupo de trabajo se dispone de un enlace a la aplicación, llegando entonces a la página principal denominada *'Main Page'* (véase la figura 2), que reconoce la identidad del usuario. Esta página emplea el inglés igual que las restantes. Desde ella se pueden realizar tres operaciones distintas:

1) Se puede elegir un diseño anterior, para modificarlo o regenerar resultados.

Fig. 2. Página principal de la aplicación donde se administran los diseños.

2) Se puede crear un nuevo diseño, dándole un nombre y eligiendo diversas características básicas: lenguaje HDL de salida (de momento sólo Verilog, pero está previsto incorporar VHDL), herramienta de síntesis que

será empleada (de momento sólo WebPack de Xilinx [12]) y tarjeta de FPGA de destino (de momento sólo dos, ambas de Burch Electronic Designs [5])[4].

3) Se pueden administrar los ficheros que contienen diseños y que están en el servidor web, y también se puede revisar, modificar o cancelar la información personal y la cuenta de usuario.

Lo habitual será empezar o continuar un diseño. En la página correspondiente, mostrada en la figura 3, se describe en la parte central el estado del diseño, con sus módulos y sus conexiones. En la parte superior se muestra un conjunto de controles que permiten añadir, editar y eliminar elementos del diseño, mientras la parte inferior nos indica que podemos verificar el circuito, generar resultados o modificar diversas propiedades del diseño.

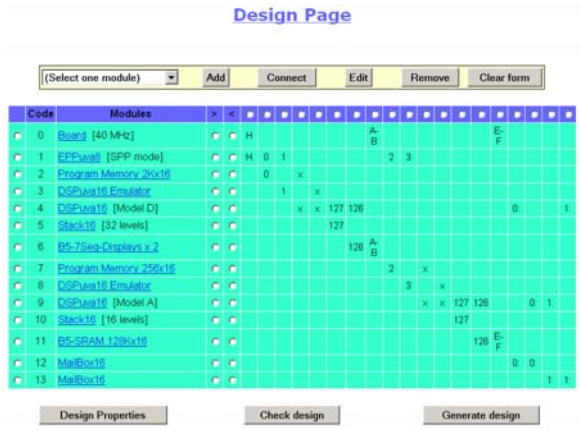

Fig 3. Ampliación de la página de diseño donde se modifica el circuito, pudiendo añadir y eliminar módulos, y especificar los parámetros posibles de sus interconexiones.

Si se analiza con detalle la figura o, mejor, si se utiliza esta herramienta para diseñar personalmente un circuito, se descubre que su uso no es excesivamente complicado. Para añadir un nuevo módulo basta con seleccionarlo en el menú desplegable superior y pulsar el botón *'Add'*. Para modificar sus parámetros es suficiente con seleccionarlo en la primera columna de la tabla y pulsar el botón *'Edit'*. Para conectar dos módulos, y aquí radica una parte importante de la potencia de esta herramienta, es suficiente con indicarlo a través de los selectores centrales (columnas identificadas con los símbolos ">" y "<") y pulsar el botón *'Connect'*: no es necesario decir *cómo* se debe hacer la conexión, pues si ésta es posible la herramienta ya lo sabe y lo hace por nosotros; sólo tenemos que especificar qué conexiones queremos. Las propiedades de cada conexión se pueden editar igual que los parámetros de los módulos, y siempre se pueden eliminar elementos que ya no deseemos utilizando el botón *'Remove'*.

Actualmente la herramienta incluye soporte para dos procesadores[5] con sus correspondientes emuladores, un interfaz a través del puerto paralelo que permite controlar los distintos procesadores desde un PC, varios modelos distintos de memoria de programa y de memoria de datos, algunas compartidas entre varios procesadores, varias pilas LIFO y FIFO, y varias definiciones para conectar periféricos específicos de dos tarjetas de desarrollo de BurchED [5]. Recientemente se han añadido módulos que permiten controlar módulos RF, convertidores A/D y generadores de PWM.

La principal ventaja de diseñar empleando esta herramienta es que el código HDL sólo se ha de escribir una vez. A partir de entonces la aplicación es capaz de generar el código necesario personalizándolo para cada situación. Lo mismo ocurre con la asignación de pines, que también se realiza de forma automática.

IV. Generación De Resultados

Cuando se ha terminado de introducir un circuito, lo que puede llevar unos pocos minutos, es el momento de generar los ficheros de diseño necesarios para su implementación en FPGA.

El primer paso consiste en comprobar que las especificaciones introducidas no contienen ninguna inconsistencia. Pulsando el botón *'Check design'* se realizan un conjunto de comprobaciones que verifican el circuito. En particular, se revisa que toda la información relativa al circuito se corresponde con las bases de datos que en ese momento tiene la aplicación, pues algún módulo se ha podido quedar obsoleto. Por el mismo motivo se revisan todas las conexiones y sus propiedades. Por último, se verifica que el diseño no tenga módulos desconectados y que al menos exista un componente conectado con el exterior, para que se generen entradas y salidas. Está en desarrollo el chequeo de posibles cortocircuitos, pero en cualquier caso esta eventualidad será verificada por las herramientas de síntesis.

La figura 4 muestra la página resultante de pulsar el botón *'Generate Design'*. La función principal es generar el código HDL (*"Verilog Top Level File"*) que implementa e interconecta todos los módulos requeridos por el diseñador en la página anterior. Si hay disponible información sobre la tarjeta de desarrollo o la FPGA empleada se genera además un fichero UCF (*"User Constraints File"*) que indica en qué pines de la FPGA se han de conectar las entradas y salidas del circuito y qué restricciones temporales se ha de imponer a la señal

[4] Estas características sólo son importantes si se pretende generar desde esta herramienta el 100% del diseño.

[5] Un procesador de propósito general de 8 bits pendiente de publicación en FPGAworld 2006 y un DSP en coma fija de 16/24 bits publicado en [7]-[9] y usado en [11].

de reloj. Finalmente, si existe información sobre la herramienta de síntesis que el diseñador va a usar, también se genera un fichero de proyecto (*.prj*), de modo que el usuario sólo tendrá que hacer un doble-click sobre él para que su herramienta de síntesis esté preparada para implementar el circuito.

Fig. 4. Ampliación de la página de generación de resultados de la aplicación *Hardware Highway*, desde donde se pueden descargar los ficheros generados.

Adicionalmente se suministran diversos enlaces hacia un conjunto de ficheros predefinidos que describen, en código HDL o en formato EDIF con sus correspondientes ficheros de enlace (*wrappers*), los diversos componentes que integran el diseño. Estos ficheros también se deben descargar para completar la información que necesitará la herramienta de síntesis del usuario. De momento las descargas típicas transfieren unos 100~200 KBytes, por lo que el proceso es rápido.

V. Páginas De Ayuda

Como estamos viendo, es la propia herramienta la que sabe cómo implantar y conectar los diversos módulos, pero es el usuario el que debe decidir qué módulos utiliza y ha de saber qué conexiones puede realizar. Para eso, además de la página genérica de ayuda que se activa de forma personalizada al pulsar en el enlace de la esquina inferior derecha, esta aplicación es capaz de crear una página de ayuda específica para cada módulo. Para activar esta ayuda es suficiente con pulsar sobre el propio nombre del módulo, que es un hiperenlace.

Al realizar esta tarea la aplicación acude directamente a sus bases de datos, que veremos enseguida, y evalúa para cada módulo cuáles son sus parámetros y sus conexiones posibles. También aporta información relativa a enlaces y páginas de documentación acerca de los módulos. El resultado es una página como la que se muestra en la figura 5.

Fig. 5. Página de ayuda autogenerada para un módulo.

VI. Bases De Datos

La complejidad de esta aplicación no reside tanto en la programación de sus páginas web como en la adecuada elección de la estructura de sus múltiples bases de datos. Precisamente por este motivo se ha elegido el lenguaje PHP, por la facilidad que aporta en la creación y posterior tratamiento de las estructuras de datos que almacenan cómo se implementan los módulos y cómo se interconectan.

Hasta el momento actual, en el que suele haber pocos usuarios conectados simultáneamente y el volumen de las bases de datos todavía es reducido, se ha estructurado la información de esta aplicación en los siguientes ficheros:

1) *Users.(!)*: Listado con información de los usuarios[6] en formato XML.
2) *General.php*: Diversos listados que contienen los lenguajes HDL disponibles, las herramientas de síntesis y las tarjetas de desarrollo reconocidas.
3) *Modules.php*: Información básica de los módulos existentes, su categoría básica y sus parámetros.
4) *ModulesVerilog.php*: Información específica de cómo implementar los diversos módulos en lenguaje Verilog. De momento no se ha incorporado ningún HDL más, pero está previsto.
5) *Connections.php*: Información básica de cómo se interconectan los módulos y qué propiedades pueden afectar a las conexiones.
6) *ConnectionsVerilog.php*: Información específica de cómo se ha de generar el código HDL de las conexiones. Cuando se incorporen otros lenguajes de descripción de circuitos bastará con crear otros ficheros similares a éste.
7) *Boards.php*: Información sobre los conectores de las tarjetas de desarrollo.
8) *Pinouts.php*: Información específica de cómo asociar señales con pines al crear los ficheros de restricciones empleados por las herramientas de implementación.

[6] En la configuración local del servidor web se ha especificado que los ficheros con extensión *"(!)"* no puedan ser descargados ni a través de *http* ni a través de *ftp*.

9) *HelpData.php*: Información empleada para generar las páginas de ayuda personalizadas para cada módulo.
10) *Design.(!)*: Información de cada diseño, en formato XML debido a su estructura irregular.

Aunque, como hemos dicho, todavía el número de usuarios no es elevado, sí se ha programado toda la aplicación pensando en reducir en lo posible la carga del servidor. Por este motivo, no todas las bases de datos se cargan en todas las páginas, sino solamente las imprescindibles. En particular, la página de diseño, que es la más utilizada por los usuarios, sólo carga aquellas bases de datos que aportan información básica. Precisamente se han separado las partes con información específica no sólo para permitir una fácil extensión de la aplicación a otros lenguajes y tarjetas, sino también para acelerar la respuesta del servidor al reducir su carga computacional. En cualquier caso, de momento el funcionamiento es fluido desde el punto de vista del cliente y ninguna respuesta del servidor, que es un Pentium 4 a 2.4 GHz, se demora más de uno o dos segundos, un poco más en conexiones de 56 Kbps.

V. CONCLUSIONES

En este artículo se ha mostrado una aplicación web que permite diseñar rápidamente un gran número de circuitos digitales basados en varios procesadores, de propósito general o de cálculo, integrados en FPGA. Con esta herramienta es posible generar circuitos de control en unos veinte o treinta minutos, obteniendo al final un circuito sintetizado y transferido a una FPGA sobre un sistema de desarrollo conectado a nuestro ordenador, en nuestra mesa de trabajo. Con esto ya tenemos el hardware de nuestro equipo y sólo nos queda programar los procesadores que hayamos incorporado al diseño.

Si actualmente resulta impensable emplear un procesador sin un entorno adecuado para programar su *software* de forma cómoda, dentro de poco también será extraño usar un procesador si no podemos decidir su entorno *hardware* con la misma facilidad.

REFERENCES

[1] R. Aceves, *Desarrollo de un enlace inalámbrico para telefonía fija empleando una FPGA*. Proyecto Fin de Carrera, Universidad de Valladolid, España, 2006.

[2] P. Alfke, "Metastable Recovery in Virtex-II Pro FPGAs", XAPP 094, Febrero de 2005, también en *http://direct.xilinx.com/bvdocs/appnotes/xapp094.pdf*.

[3] Altera Corporation: *http://www.altera.com*.
SOPC Builder: *http://www.altera.com/products/software /products/sopc/sop-index.html*.

[4] J. del Barrio, *Desarrollo sobre FPGA de un emulador de un sistema de microgeneración eléctrica*, Proyecto Fin de Carrera, Universidad de Valladolid, España, 2004.

[5] Burch Electronic Designs (BurchED), Macquarie Centre, North Ryde, Australia. *http://www.burched.biz*.

[6] Jan Gray, FPGA CPU list (lista de procesadores para FPGA), *http://www.fpgacpu.org/links.html*.

[7] S. de Pablo, J.A. Cebrián, L.C. Herrero y A.B. Rey, "Un DSP de coma fija integrado en FPGA aplicado en Electrónica de Potencia". *Seminario Anual de Automática, Electrónica Industrial e Instrumentación (SAAEI'2005)*, pp. 678-683, Septiembre de 2005.

[8] S. de Pablo, J.A. Cebrián, L.C. Herrero y A.B. Rey. "A soft fixed-point Digital Signal Processor applied in Power Electronics", FPGAworld Conference 2005, Estocolmo (Suecia), pp. 19-24, Septiembre de 2005.

[9] S. de Pablo y otros, "Project OpenDSP: A 16-bit fixed-point DSP processor for FPGA", Departamento de Tecnología Electrónica, Univ. de Valladolid, España, *http://www.dte.eis.uva.es/OpenProjects/OpenDSP*.

[10] Lenguaje PHP: *http://www.php.net*.

[11] A.J. Salim, M. Othman y M.A. Mohd Ali, "OpenDSP in Xilinx FPGA", *2005 Asian Conference on Sensors and the International Conference on New Techniques in Pharmaceutical and Biomedical Research Proceedings*, pp. 185-189, Septiembre de 2005.

[12] Xilinx: *http://www.xilinx.com*.
EDK: *http://www.xilinx.com/ise/embedded_design_prod /platform_studio.htm*.