Metodologías de diseño para dispositivos digitales programables

Santiago T. Pérez[1], Carlos M. Travieso[1], Jesús B. Alonso[1], José L. Vásquez[2]

[1]Departamento de Señales y Comunicaciones, Universidad de Las Palmas de Gran Canaria, Las Palmas de Gran Canaria, España
{sperez, ctravieso, jalonso}@dsc.ulpgc.es
[2]Instituto de Investigaciones en Ingeniería, Universidad de Costa Rica, San José, Costa Rica
jose.vasquez@ucr.ac.cr

Resumen. En esta aportación se describen las metodologías de diseño existentes para los circuitos digitales programables. Muchos de estos circuitos son incluso reprogramables por el diseñador con la ayuda de un ordenador personal. Con estos dispositivos se consiguen buenas prestaciones físicas. El aumento de la complejidad de los sistemas diseñados ha ido acompañado por multiplicidad de herramientas de diseño. Estos programas de diseño han crecido en complejidad y prestaciones. La elección del entorno adecuado no es una tarea fácil. Finalmente se hace una propuesta de métodos de diseño, a impartir en el Grado en Ingeniería en Tecnologías de la Telecomunicación y en el Máster Universitario en Ingeniería de Telecomunicación de la Universidad de Las Palmas de Gran Canaria.

Palabras clave: metodología, diseño, circuito, electrónica digital, programación, FPGA, HDL, VHDL, Verilog, compilación, simulación, Simulink, Matlab, grado, master

1 Introducción

Son varias las tecnologías disponibles para el diseño de sistemas digitales, cada una con sus ventajas e inconvenientes. Una de las posibilidades es usar circuitos integrados de aplicación específica (ASIC, *Application Specific Integrated Circuit*). Los diseños ASIC deben ser enviados a la fábrica para la creación del dispositivo; como ventajas tienen escasa área ocupada, bajo consumo de potencia y alta velocidad [1]. Como inconvenientes presentan alto precio, dificultosa depuración y verificación, gran tiempo de acceso al mercado, no permite la reprogramación y gran coste de ingeniería no recurrente. Esto lo hace normalmente desaconsejable en el desarrollo de prototipos.

Por otro lado, se pueden usar procesadores digitales de señales (DSP, *Digital Signal Processor*) [2,3], que son más baratos que los elementos ASIC. Los DSP alcanzan mayores frecuencias de reloj que los ASIC. Sin embargo, la tasa de datos que pueden procesar se ve limitada porque el paralelismo de los datos es limitado; el tamaño y formato de los datos es restringido; y el *pipelined* es fijo. Todo esto viene

ISBN: 978-84-617-1860-3

impuesto por la arquitectura fija de los DSP. También puede usarse una unidad de procesamiento gráfico (GPU, *Graphics Processing Unit*), que es un coprocesador que usa aritmética en coma flotante especializado en el procesamiento de imágenes [4,5]. Las GPU tienen mayor capacidad de procesamiento que las CPU, esto se debe a que cuenta con multitud de unidades aritméticas y lógicas.

Finalmente, se pueden usar matrices de puertas digitales programables por el diseñador (FPGA, *Field Programmable Gate Array*). Las FPGA constan de bloques de circuitos lógicos, líneas de conexión, matrices de interruptores y pines de entrada-salida. Los bloques de circuitos lógicos reciben diferentes nombres según el fabricante, en ellos se concentra la capacidad de computación de la FPGA. La arquitectura interna de los bloques lógicos depende del fabricante y del dispositivo. Las líneas de conexión son las encargadas de conectar los diferentes bloques y los pines; son largos conductores que atraviesan el integrado. Puede haber líneas específicas dedicadas, por ejemplo para señales de reloj o reinicio del sistema, que se extiende por toda la FPGA. La mayor parte de los pines son de entrada-salida de señal, pero también los hay exclusivos para alimentación, relojes, reinicio y programación de la FPGA.

La gran ventaja de las FPGA frente a los ASIC es que son programables por el diseñador, sin necesidad de ser enviadas a una fábrica. Las FPGA presentan como ventajas: que no existe ingeniería no recurrente, mínimo tiempo de desarrollo, facilidad de depuración y verificación, menor tiempo de acceso al mercado, alto paralelismo de datos, tamaño y formato de datos flexible, y *pipelined* flexible [6,7]. Aunque la frecuencia de reloj en las FPGA no es tan elevada como en los DSP, con las características anteriores se consigue procesar un mayor flujo de datos. En general el precio de las FPGA es menor que los elementos ASIC, pero mayor que los microprocesadores. El consumo de energía en las FPGA es mayor que en los ASIC, pero menor que en los microprocesadores. Por todo lo anterior las FPGA son apropiadas para el desarrollo de prototipos cuando el flujo de datos a procesar es elevado. Por otro lado muchas empresas ofrecen placas de circuito impreso que incluyen la FPGA y dispositivos externos. Estas placas incluyen: convertidores analógicos-digitales y digitales-analógicos, memorias, dispositivos para entrada-salida de audio y video, puertos de comunicaciones, etc. El uso de estas placas evita la tediosa tarea de su diseño y fabricación; y lo más importante, permiten la creación de sistemas completos. Por los motivos anteriores para el desarrollo de prototipos es conveniente el uso de FPGA. Además, bajo ciertas condiciones, es posible migrar los diseños realizados para FPGA a dispositivos ASIC.

2 Metodologías de diseño para FPGA

Existen diferentes métodos de descripción de un circuito en una FPGA, en todos ellos se diseña de forma independiente de la tecnología. Es decir, el diseñador no tiene que conocer la arquitectura interna de la FPGA, ni las características de los circuitos semiconductores. En este apartado se describen los diferentes métodos disponibles para este tipo de dispositivos.

ISBN: 978-84-617-1860-3

2.1 Edición de esquemáticos

El método más intuitivo y usado desde hace decenios es la edición de esquemáticos. De esta forma el diseñador dibuja y describe los sistemas digitales usando puertas lógicas, registros, biestables, etc.; básicamente colocándolos y conectándolos. Esto se llama "captura de esquemáticos" y en el proceso se describe el sistema completo, pudiendo usarse niveles de jerarquía o sistemas ya creados. Obviamente, para evitar el dibujo manual de los sistemas, pronto se hizo necesario entornos de diseño gráfico sobre computador. Este método puede usarse para sistemas pequeños o bien conocidos por el diseñador, donde se pueden describir rápidamente y son fácilmente modificables. Para esta técnica se deben elaborar, con o sin ayuda de herramientas auxiliares, las tablas de Karnaugh de los sistemas combinacionales y secuenciales [8]. Cuando los sistemas aumentan en tamaño la elaboración de estas tablas y circuitos puede llegar a ser inviable. En este caso esta técnica deja de ser aconsejable. Existe un formato estándar de intercambio de diseño electrónico (EDIF, *Electronic Design Interchange Format*) independiente de los fabricantes [9]. El formato EDIF puede usarse para intercambio de esquemáticos, pero los sistemas están descritos en modo de texto. Tanto Xilinx como Altera, principales suministradores de FPGA, permiten introducir diseños en formato EDIF. Altera genera el formato EDIF desde otras formas de descripción en su propio entorno. En cambio, Xilinx solo genera el formato propio NGC (*Native Generic Database*); esto es así para mejora el rendimiento del flujo de diseño y optimizar el uso de sus circuitos. El formato estándar EDIF es de uso limitado en el intercambio de esquemáticos, y Xilinx obliga a diseñar los esquemáticos con su propio editor. Esto hace que un circuito esquemático editado sobre una herramienta de diseño de un fabricante de FPGA no sea exportable a la herramienta de otro fabricante; a menos que se genere el formato EDIF, que no siempre es posible. En resumen, si se cambiase de fabricante de FPGA se debe rediseñar el sistema en la nueva herramienta. Finalmente, pueden usarse herramientas de diseño de empresas, que no son suministradores de FPGA. Estas desarrollan entornos que soportan la edición de esquemáticos para diversos fabricantes; pero estas herramientas no son gratuitas [10,11].

2.2 Los lenguajes de descripción hardware

En la década de los años ochenta, aparecen los lenguajes de descripción hardware (HDL, *Hardware Description Language*). Estos lenguajes permiten describir sistemas digitales usando texto, facilitando su diseño y modificación. Por medio de una compilación se genera la arquitectura del circuito. De esta forma se solventan las dificultades del diseño usando esquemáticos. En los lenguajes de programación ordinarios las sentencias se ejecutan de forma secuencial, y se mantiene la dependencia entre ellas aunque haya una ejecución paralela. En contraposición, en los HDL las sentencias son concurrentes; es decir, se ejecutan de forma simultánea, tal y como se comportan un grupo de puertas lógicas en un circuito digital. Para ejecutar un grupo de sentencias de forma secuencial se han de incluir dentro de lo que se conoce con el nombre de proceso. Un proceso es la descripción de un sistema secuencial, que

ISBN: 978-84-617-1860-3

pasa por una serie de estados. Los procesos son concurrentes entre ellos; es decir, se ejecutan de forma simultánea.

Existen dos HDL estándar y ampliamente usados. Uno es VHDL [12], que es un doble acrónimo (*Very High Speed Integrated Circuit Hardware Description Language*, VHSIC-HDL). La traducción correcta de este acrónimo sería "lenguaje muy rápido de descripción hardware para circuitos integrados". El otro HDL estándar es Verilog [13].

El VHDL fue inicialmente desarrollado por el Departamento de Defensa de los Estados Unidos. Actualmente VHDL es un lenguaje estándar definido por el Instituto de Ingenieros Eléctricos y Electrónicos (IEEE, *Institute of Electrical and Electronics Engineers*) y ha sufrido diferentes revisiones [14]. Por otro lado, Verilog se empezó a desarrollar en la compañía *Automated Integrated Design Systems* y también es un estándar definido por IEEE a lo largo de una serie de revisiones [15]. Verilog es más compacto que VHDL, y más fácil de aprender y usar. Esta facilidad se debe a su parecido con el lenguaje C. Los dos lenguajes se usan para el diseño en FPGA. Verilog es preferido para el diseño en sistemas ASIC por acceder a niveles más bajos en hardware, pero VHDL permite el uso de tipos de datos más complejos.

Aparte de los dos HDL estándares cabe destacar AHDL (*Altera Hardware Description Language*), que es específico de Altera para sus dispositivos programables [16]. La sintaxis de este lenguaje es parecida a la del lenguaje Ada. Una ventaja de este lenguaje es que aprovecha mejor las características de los dispositivos de Altera, pero como desventaja no es portable a otros fabricantes.

2.3 Lenguajes de alto nivel

Con la aparición de VHDL y Verilog aparece un curioso fenómeno, que se describe a continuación. Los programadores siguen usando los lenguajes de programación de alto nivel (C, Java, etc.) para ejecutar programas secuenciales. Estos programas se ejecutan sobre computadoras, normalmente en punto flotante. Algunos de estos programadores, mayormente formados en el ámbito de la computación y no en el diseño electrónico, se preguntan por qué no pueden diseñar sistemas digitales sobre dispositivos programables. Este deseo se basa en el hecho de que la descripción del circuito se hace en forma de texto, igual que en los lenguajes de alto nivel. A partir de aquí empiezan a desarrollarse entornos de descripción de circuitos basados en esos lenguajes de alto nivel, que no fueron concebidos para la descripción hardware. Estos entornos aparecen a partir de los años noventa y se basan sobre todo en Java y C. Muchos de estos sistemas son abiertos y disponibles de forma libre, su origen está en foros y universidades. Otros entornos son propietarios de empresas. Como inconveniente presentan el hecho de que el diseño solo es portable a pocos fabricantes; y dentro de cada fabricante, a unas pocas familias de dispositivos. Además, los entornos gratuitos pueden quedar restringidos; bien por no incluir los dispositivos más recientes de un fabricante o por no dar suficiente soporte a los diseñadores.

En el lenguaje Java se basa JHDL (*Java Hardware Description Language*), este entorno es gratuito y solo soporta dispositivos de Xilinx, en el año 2006 se realizó su última actualización [17]. El entorno C-to-Verilog traduce código C a Verilog, es

 ISBN: 978-84-617-1860-3

gratuito, en el año 2009 se hizo su última actualización [18]. El entorno SystemC se basa en C++, que además es un estándar en código abierto de IEEE desde el año 2005 [19]. SystemC está soportado por más de veinte desarrolladores de herramientas de diseño.

También pueden describirse sistemas con editores de forma de onda, que si bien para circuitos combinacionales pueden ser útiles, para circuitos secuenciales pueden ser de uso complicado. Algunas herramientas incluyen editor de diagrama de flujo o de estados para sistemas secuenciales. Con estos entornos se edita el gráfico de los estados y las transiciones; son muy útiles para el diseño de este tipo de sistemas. Las diferentes formas de descripción de un circuito se pueden combinar; es decir, un sistema se puede componer de diferentes subsistemas donde cada uno de ellos se describe con algún método de los anteriormente descritos.

2.4 Los entornos de los fabricantes

Actualmente los suministradores de FPGA ponen a disposición de los usuarios entornos de diseño estándar, en ellos se pueden describir los circuitos de las formas descritas anteriormente. Estas herramientas no son en principio gratuitas, aunque se pueden pedir como donación a instituciones académicas. También tienen versiones de demostración, bien por limitación en el tiempo de uso o por el tamaño del diseño. Estas herramientas permiten compilar y simular el diseño, para finalmente generar el fichero de programación para la FPGA del fabricante. Estos entornos gestionan el conjunto de ficheros generados en el diseño, asigna los pines de entrada-salida, selecciona opciones para la compilación, etc. Finalmente dan información de los recursos hardware necesarios, de la velocidad máxima del circuito y de la potencia consumida.

Los suministradores de FPGA también incluyen librerías de componentes parametrizables: memorias, unidades aritméticas, etc. También disponen de módulos con propiedad intelectual (IP, *Intellectual Property*) disponibles después del pago del canon correspondiente; a veces se ofrece de forma gratuita, con limitaciones en el tiempo de uso. El código HDL generado en estos casos solo es válido para los dispositivos del propio fabricante. En el dispositivo FPGA pueden integrarse otros módulos empotrados que incluyen software, como por ejemplo microprocesadores, funciones de comunicaciones, etc. Así se constituye lo que se conoce por sistema en un chip (SoC, *System on Chip*).

Por otro lado los principales suministradores de FPGA ofrecen entornos de diseño sobre Simulink de Matlab [20]. Estas herramientas suponen una ampliación del entorno estándar, están disponibles en las mismas condiciones y solo sirven para los dispositivos del fabricante. Tras la instalación de estas herramientas aparecen uno o varios *Blocksets* en Simulink, que incluyen los bloques del fabricante. Simulink es un entorno de diseño gráfico que usa diagrama de bloques, permitiendo diseñar de forma rápida y flexible. Los bloques son configurables mediante ventanas de diálogo. Con estas utilidades se diseña el sistema usando aritmética en punto fijo, aunque a partir del año 2013 también se incluye unidades aritméticas en punto flotante. Una vez diseñado el sistema puede simularse aprovechando las ventajas de Simulink. La primera ventaja es el acceso directo al espacio de variables de Matlab, lo que facilita

ISBN: 978-84-617-1860-3

la generación de las señales de entrada y el análisis de las salidas. Por un lado, se pueden usar las diferentes fuentes de señal de Simulink, estas se encuentran en el *Blockset Sources*. Por otro lado, se pueden usar sus visualizadores y destinos de señal, que se encuentran en el *Blockset Sinks*. Las señales obtenidas se muestran como si fueran de punto flotante, lo que facilita el análisis de las formas de onda. Estas simulaciones son muy rápidas porque incluyen un nivel pobre de detalle de los circuitos de la FPGA, esto hace que la estimación de área ocupada sea aproximada. Por otro lado no aporta estimaciones de máxima velocidad o potencia consumida. Como ventaja es posible la comprobación de la total funcionalidad del sistema. Esto es posible por ser estas simulaciones rápidas y Matlab poder manejar gran cantidad de datos. Al disminuir el tiempo de diseño, y ser entornos más versátiles, se facilita al diseñador la comprobación de diferentes arquitecturas y configuraciones.

Una vez simulado el sistema se realiza una compilación que genera el proyecto para el entorno estándar del fabricante. El sistema queda descrito de forma estructural en VHDL o Verilog. Aquí los HDL son usados como una etapa intermedia. El código HDL generado no es portable a otros fabricantes, porque aprovecha las peculiaridades y arquitectura de la FPGA; dicho de otra forma, usa lo que se llaman primitivas, que son librerías de bloques específicas del fabricante. De todas formas, este tipo de herramientas pueden ser usadas para comprobar la arquitectura y prestaciones numéricas de un sistema digital; por ejemplo, el efecto del número de bits en diferentes partes de un circuito, sin necesidad de programar finalmente una FPGA.

La comparación entre fabricantes, de herramientas similares de diseño, es una tarea compleja. Estas herramientas van cambiando rápidamente, y los fabricantes mantienen en sus páginas web las diferentes versiones, incluso con diferentes simuladores. A esto hay que añadir la dificultad de que cambian la estructura de las páginas web. Debe destacarse que la forma de adquirir la licencia y de habilitarla para las herramientas también sufre cambios. Los fabricantes tienen las herramientas para diferentes versiones de sistemas operativos, principalmente Windows y Linux; las utilidades disponibles pueden depender del sistema operativo elegido. Cuando el software de diseño se apoya en Simulink solo son válidas unas determinadas versiones de Matlab. Resumiendo, si alguien quiere instalar un paquete de estas herramientas tendrá que comprobar la compatibilidad con su sistema operativo, incluso si es de 32 o 64 bits. Por otro lado debe disponer de una versión de Matlab válida.

2.5 Xilinx versus Altera

La empresa Xilinx es la que acapara la mayor parte del mercado de FPGA a nivel mundial [21]. Actualmente Xilinx dispone de dos entornos de diseño estándar; por un lado ISE (*Integrated System Environment*), con versiones hasta octubre de 2013 [22]; y por otro Vivado, con versiones desde 2012 hasta la actualidad [23]. Junto a estos entornos se dispone de System Generator, que es la utilidad de diseño sobre Simulink [24]. El segundo suministrador en importancia de FPGA es Altera [25]. El entorno de diseño de Altera se llama Quartus II [26] y la utilidad para Simulink es DSP Builder [27]. Es posible obtener donación de placas, por parte de Altera, dentro de su programa universitario.

ISBN: 978-84-617-1860-3

El software de diseño AccelDSP fue ofertado entre los años 2006 y 2010 por Xilinx [28]. Este software era de pago, aunque Xilinx permitía usarlo durante un periodo de tiempo de forma gratuita como evaluación. Esta herramienta se basaba en System Generator; y por tanto en Matlab y Simulink. AccelDSP automatizaba la conversión del código en punto flotante a punto fijo, pero esto se hacía con fuertes restricciones en el código original. El código de Matlab se convertía a C++ en punto fijo o a una descripción de bloques en System Generator. Finalmente se generaba el sistema descrito en VHDL o Verilog, no portable a otros fabricantes. Curiosamente AccelDSP de Xilinx nunca tuvo su herramienta equivalente en el competidor Altera, y siendo AccelDSP una herramienta novedosa desapareció tan rápido como apareció. Los motivos pueden ser la falta de rentabilidad económica, quizás los diseñadores se conformaban con System Generator y la interactuación que puede tener con Matlab en punto fijo, y no consideraron necesario pagar el canon.

En cuanto a las herramientas de diseño estándar de Xilinx y Altera se puede hacer una comparación cualitativa. La herramienta de diseño de Xilinx es tradicionalmente más compleja y difícil de manejar, pero más versátil y potente; tiene tiempos de compilación mayores y dispone de utilidades que permiten una precisa estimación del consumo de potencia. Por otro lado, su equivalente de Altera es más fácil de manejar y de uso intuitivo, pero menos potente y flexible. Cualquier proyecto puede realizarse con cualquiera de los dos principales fabricantes, pero se puede afirmar que Altera es más apropiado para el uso docente y académico; mientras que Xilinx es preferido en tareas de desarrollo e investigación.

2.6 Otras empresas y entornos académicos

Existen compañías que crean software no gratuito que puede usarse para diseñar en FPGA de diferentes fabricantes. La empresa Synopsys ofrece la utilidad Synplify en diferentes modalidades [29]. Este entorno de diseño permite elegir entre los fabricantes de FPGA: Xilinx, Altera, Lattice, Atmel, Microsemi y Achronix. Para usar esta herramienta se necesita pagar un canon, aunque puede ser usada en forma de evaluación. Este sistema, aunque más caro, pero permite la portabilidad del diseño para los fabricantes habilitados. Como se puede compilar el diseño para cada uno de los fabricantes es posible comparar las prestaciones, sin necesidad de usar el entorno de cada fabricante por separado.

La compañía National Instruments ha desarrollado LabView (*Laboratory Virtual Instrumentation Engineering Workbench*), este es un entorno donde se diseña de forma visual y gráfica [30]. Para su uso hay que pagar una licencia, aunque se puede solicitar su evaluación. Se usa para diseñar sistemas de instrumentación, control, procesado de señal y comunicaciones; admite gran cantidad de equipamiento externo y puertos de entrada-salida. Esta herramienta sobrepasa el diseño para FPGA porque cubre una gran cantidad de equipamiento de las áreas anteriores. Solo soporta las FPGA de Xilinx. Los diseños se pueden introducir usando VHDL, Verilog y de forma esquemática. Este fabricante muestra como una ventaja el evitar el uso de lenguajes tipo HDL en modo de texto; esto no es necesariamente una ventaja, más bien puede ser un inconveniente. Entonces puede ser conveniente usar el software de Xilinx, que puede llegar a ser gratis, tanto su herramienta estándar de diseño, como System

ISBN: 978-84-617-1860-3

Generator sobre Simulink. El uso de LabView puede justificarse si el diseño de la FPGA se integra en un sistema mayor e interdisciplinar que aprovecha las ventajas globales de LabView.

Se han desarrollado muchas otras herramientas de ayuda para el diseño en FPGA, algunas gratuitas y otras de pago. Las primeras provienen de entornos académicos o grupos de investigación, las segundas han sido generadas por empresas. Entre las gratuitas cabe destacar *Floating-Point to Fixed-Point Toolbox* para Matlab [31], que permite el paso de código de punto flotante a punto fijo, y analiza las prestaciones del sistema frente a un error establecido para la representación de punto fijo. Este *Toolbox* es un ejemplo de herramienta pionera en este campo, aunque ya está en desuso y ha sido ampliamente superado. Su página web no se actualiza desde 2006.

2.7 Matlab para FPGA

El entorno Matlab se ha convertido en un estándar de hecho, tanto para la comunidad académica y científica, como para la industria. Si bien no es gratuita, en la práctica los diseñadores disponen de ella, entre otros motivos, por el uso de licencias institucionales. Primeramente cabe destacar *Filter Design HDL Coder* que tiene la capacidad de generar el hardware necesario para el diseño de filtros [32]. Permite obtener en punto fijo la descripción del filtro en VHDL o Verilog, este código es portable a todos los fabricantes de FPGA. También genera las señales necesarias de entrada para la simulación y verificación del filtro.

La utilidad *Fixed-Point Designer* de Matlab permite manejar tipos de datos y herramientas para el desarrollo de algoritmos en punto fijo, usando código de Matlab, modelos de Simulink o el editor de diagramas de flujo *Stateflow* [33]. Esta herramienta propone automáticamente el número de bits y el método de redondeo, que también se pueden especificar manualmente.

En Matlab cabe destacar la utilidad *HDL Coder* donde los sistemas se describen usando funciones de Matlab, modelos de Simulink o el editor de diagramas de flujo *Stateflow* [34]. El código generado puede ser en VHDL o Verilog, y es totalmente portable y sintetizable; además tiene un flujo de trabajo integrado con Altera y Xilinx, tanto para diseños en FPGA como ASIC. Esta herramienta automatiza y facilita el flujo del diseño, permite control de la arquitectura y de los retardos críticos; además, da una estimación del área ocupada. Permite también comparar la respuesta del código HDL generado con el modelo inicial de Simulink. *HDL Coder* es una herramienta relativamente nueva, la primera versión data de marzo de 2012.

Finalmente Matlab ofrece un verificador del diseño realizado en VHDL o Verilog [35]. Este sistema permite comunicar las señales de Matlab con la placa que contiene la FPGA de Altera o Xilinx, hacer funcionar la FPGA en tiempo real y compara las señales obtenidas con las del modelo de Matlab. En resumen, con *HDL Coder* y *HDL Verifier* se facilita el diseño de sistemas en FPGA o ASIC. Se puede generar un código portable a cualquier fabricante; o generar un código optimizado para dispositivos de Altera o Xilinx. En este último caso se puede programar los dispositivos desde Matlab y realizar la verificación del diseño.

ISBN: 978-84-617-1860-3

3 Conclusiones y propuesta

Es previsible que los diferentes métodos coexistan en el futuro. La edición de esquemáticos se usará principalmente para diseñar pequeños bloques que se usarán en un sistema jerárquico. Los HDL estándar se usarán porque garantizan la portabilidad a todos los fabricantes. Además, las herramientas de diseño avanzadas generan el sistema en un HDL, que es usado posteriormente por las herramientas estándar. La descripción usando lenguajes de programación de alto nivel, sin llegar a desaparecer, se mantendrán, sobre todo los estandarizados dependientes de empresas. Las herramientas gratuitas, de entornos académicos o instituciones, quedan normalmente obsoletos por falta de actualización; quizás por escasez de fondos. Los suministradores de FPGA mantendrán activas sus herramientas de diseño para sus dispositivos. Estos suministradores desarrollarán herramientas sobre Matlab, que facilite el diseño y la simulación. Otras empresas de desarrollo de herramientas de diseño mantendrán los entornos necesarios que permitan exportan los diseños a varios fabricantes, pero serán más caras. La tendencia es que los nuevos métodos de diseño se integren con Matlab y Simulink, lo que facilita el diseño, la simulación del sistema y la comprobación de diferentes arquitecturas.

Cabe preguntarse cuál es el mejor método de diseño, esto dependerá del escenario del diseño y de una serie de parámetros: del coste económico, de la portabilidad a diferentes fabricantes, de la flexibilidad del diseño, etc. El método elegido será capaz de garantizar las restricciones de área, potencia consumida y velocidad.

Para finalizar se realiza una propuesta de los métodos que se consideran apropiados impartir en el Grado en Ingeniería en Tecnologías de la Telecomunicación [36] de la Universidad de Las Palmas de Gran Canaria. En este grado existen cuatro módulos: Sistemas Electrónicos, Sistemas de Telecomunicación, Telemática y Sonido e Imagen. Todos ellos tienen la asignatura *Electrónica Digital* en segundo curso, esta debería centrarse en el diseño de esquemáticos. Todos estos módulos tienen también *Sistemas Digitales y Microprocesadores* en el tercer curso, en ellas se considera conveniente aprender al menos uno de los HDL habituales: VHDL o Verilog. Solo Sistemas Electrónicos tiene las asignaturas de *Hardware Programable* y *Sistemas Electrónicos*, en ellas sería conveniente estudiar la herramienta estándar de diseño de Altera, dada su facilidad de uso.

En lo que respecta al Máster Universitario en Ingeniería de Telecomunicación [37] de la Universidad de Las Palmas de Gran Canaria, se dispone de tres asignaturas: *Sistemas Integrados*, *Ingeniería de Sistemas* y *Desarrollo Hardware-Software de Productos Electrónicos*. En ellas podría introducirse la herramienta estándar de Xilinx, el entorno de diseño de Xilinx sobre Simulink y finalmente las utilidades de Matlab específicas para el diseño en FPGA.

Reconocimientos. Los autores agradecen a Xilinx la donación de licencias gratuitas para el uso de su software. Igualmente agradecen a Altera la donación de licencias y diferentes placas con FPGA.

Referencias

1. Chandrasetty, V. A.; *VLSI Design a Practical Guide for FPGA and ASIC Implementations*, Springer, 2011
2. Oniga, S.; Tisan, A.; Lung, C.; Buchman, A.; Orha, I.; "Adaptive Hardware-Software Co-Design Platform for Fast Prototyping of Embedded Systems", *12th International Conference on Optimization of Electrical and Electronic Equipment*, pp. 1004-1009, 2010
3. Wang, X.; Ma, Z.; "Discussion on the methodology of neural network hardware design and implementation", *6th International Conference on Solid-State and Integrated-Circuit Technology* (acrónimo), pp. 113-116, 2001
4. Ho, T.; Lam, P.; Leung, C.; "Parallelization of cellular neural networks on GPU", *Pattern Recognition*, vol. 41, nº 8, pp. 2684-2692, 2008
5. Oh, K.; Jung K.; "GPU implementation of neural networks", *Pattern Recognition*, vol. 37, nº 6, pp. 1311-1314, 2004
6. Cofer, R. C.; Harding, B. F.; *Rapid System Prototyping with FPGAs*, Elsevier, 2006
7. Maxfield, C; *The Design Warrior's Guide to FPGAs*, Elsevier, 2004
8. Floyd, T. L.; *Fundamentos de Sistemas Digitales*, novena edición, Prentice Hall, 2006
9. *Electronic Design Interchange Format (EDIF), Part 2, Version 4.0.0*, International Electrotechnical Commission, 2000
10. Active-HDL de Aldec, *http://www.aldec.com/en/products/fpga_simulation/active-hdl*, última visita el 4 de junio de 2014
11. Synplify Pro de Synopsys, *http://www.synopsys.com/Tools/Implementation/FPGAImplementation/FPGASynthesis/Pages/SynplifyPro.aspx*, última visita el 4 de junio de 2014
12. Pedroni, V. A.; *Circuit Design with VHDL*, MIT Press, 2004
13. Palnitkar, S.; *Verilog HDL: A Guide to Digital Design and Synthesis*, 2ª edición, Prentice Hall, 2003
14. *IEEE Standard VHDL Language Reference Manual*, enlace permanente: http://ieeexplore.ieee.org/servlet/opac?punumber=4772738
15. *IEEE Standard for Verilog Hardware Description Language*, enlace permanente: http://ieeexplore.ieee.org/servlet/opac?punumber=10779
16. Altera Hardware Description Language, *http://www.alterawiki.com/uploads/c/c9/Altera_AHDL_Language_Reference.pdf*, última visita el 4 de junio de 2014
17. Java Hardware Description Language, *http://www.jhdl.org*, última visita el 6 de junio de 2014
18. C-to-Verilog por Nadav Rotem, *http://www.c-to-verilog.com*, última visita el 6 de junio de 2014
19. IEEE Standard for Standard SystemC Language Reference Manual, enlace permanente: *http://ieeexplore.ieee.org/servlet/opac?punumber=6134617*
20. Simulink de MathWorks, *http://www.mathworks.com/products/simulink*, última visita el 11 de junio de 2014
21. Xilinx Corporation, *http://www.xilinx.com*, última visita el 11 de junio de 2014
22. Integrated System Environment, *http://www.xilinx.com/products/design-tools/ise-design-suite/index.htm*, última visita el 11 de junio de 2014
23. Vivado Design Suite, http://www.xilinx.com/products/design-tools/vivado/index.htm, última visita el 11 de junio de 2014
24. System Generator for DSP, *http://www.xilinx.com/products/design-tools/vivado/integration/sysgen.html*, última visita el 11 de junio de 2014
25. Altera Corporation, *http://www.altera.com/*, última visita el 11 de junio de 2014
26. Quartus, *http://www.altera.com/products/software/quartus-ii/web-edition/qts-we-index.html*, última visita el 11 de junio de 2014
27. DSP Builder, *http://www.altera.com/products/software/products/dsp/dsp-builder.html*, última visita el 11 de junio de 2014
28. AccelDSP Synthesis Tool, *http://www.xilinx.com/tools/acceldsp.htm*, última visita el 11 de junio de 2014

 ISBN: 978-84-617-1860-3

29. Synplify, *http://www.synopsys.com/Tools/Implementation/FPGAImplementation/FPGASynthesis/Pages/SynplifyFeatureComparisonChart.aspx*, última visita el 11 de junio de 2014
30. Laboratory Virtual Instrumentation Engineering Workbench, *http://www.ni.com/labview/esa*, última visita el 11 de junio de 2014
31. Floating-Point to Fixed-Point Transformation Toolbox, *http://users.ece.utexas.edu/~bevans/projects/wordlength/converter/*, última visita el 13 de junio de 2014
32. Matlab Filter Design HDL Coder, *http://www.mathworks.es/products/filterhdl*, última visita el 13 de junio de 2014
33. Fixed-Point Designer, *http://www.mathworks.es/products/fixed-point-designer*, última visita el 13 de junio de 2014
34. Matlab HDL Coder, *http://www.mathworks.es/products/hdl-coder*, última visita el 13 de junio de 2014
35. Matlab HDL Verifier, *http://www.mathworks.es/products/hdl-verifier*, última visita el 13 de junio de 2014
36. Verifica del Grado en Ingeniería en Tecnologías de la Telecomunicación de la ULPGC, *www.eite.ulpgc.es/documentos/documentacion_interna/verificas/gitt-verifica%20eite_aneca2011.pdf*, última visita el 6 de noviembre de 2014
37. Verifica del Máster Universitario en Ingeniería de Telecomunicación de la ULPGC, *www.eite.ulpgc.es/documentos/documentacion_interna/verificas/muit-verifica%20eite_aneca2012.pdf*, última visita el 6 de noviembre de 2014

ISBN: 978-84-617-1860-3

ISBN: 978-84-617-1860-3