EasyChair Preprint

№ 590

Design methodologies for FPGA using Matlab and Simulink.

Santiago Tomás Pérez Suárez

October 26, 2018

Metodologías de diseño para FPGA usando Matlab y Simulink

Santiago T. Pérez Suárez
Departamento de Señales y Comunicaciones
Universidad de Las Palmas de Gran Canaria (ULPGC)
Las Palmas de Gran Canaria, España
santiago.perez@ulpgc.es

***Abstract*— El primer método de diseño de sistemas digitales fue la edición en forma esquemática, desde entonces los métodos de diseño han avanzado enormemente, sobre todo en los últimos años. En esta ponencia se hará una breve descripción de los métodos de diseño, hasta llegar a los métodos avanzados que se apoyan en Matlab y Simulink. Estos últimos son rápidos y flexibles, permiten explorar distintas arquitecturas, y comprobar diferente formatos de datos y operadores en los diferentes puntos del sistema. Cada suministrador de FPGA (*Field Programmable Gate Array*) dispone de su propia herramienta para diseñar y verificar los diseños desde Simulink y Matlab; normalmente en aritmética de punto fijo, aunque últimamente se incluyen librerías en punto flotante. También es posible diseñar usando herramientas propias de Matlab y Simulink, desde modelos en aritmética de punto fijo o de punto flotante. En cualquier caso, desde el modelo en punto fijo se puede generar de forma automática el sistema en un lenguaje de descripción de hardware, que puede implementarse sobre una FPGA. Las técnicas propias de Simulink y Matlab están basadas en la edición de sistemas y en técnicas de verificación, estas últimas son la "cosimulación" y "*FPGA in the loop*". Estos métodos se están implantando en la industria y la investigación por ser rápidos y flexibles; su implantación no parece tener un seguimiento especial en el mundo académico, lo que puede dejar descolgadas estas técnicas del entorno universitario.**

Palabras clave— metodología de diseño, FPGA, Matlab, Simulink, VHDL, Verilog, punto fijo, punto flotante

I. INTRODUCCIÓN

Los dispositivos digitales programables por el diseñador (FPGA, *Field Programmable Gate Array*) [1] permiten su reconfiguración mediante una computadora, sin tener que ser enviados a una fábrica; esto lo hace óptimo para el prototipado en entornos educativos.

El dilema para el diseñador es optar por el método de diseño adecuado, son posibles múltiples métodos de diseño [2]. El tiempo de verificación excede casi siempre al tiempo de diseño; esto es así si la complejidad de los datos es elevada. La mayor parte de las implementaciones se han desarrollado en aritmética de punto fijo, no obstante en los últimos años se han desarrollado las técnicas en aritmética de punto flotante [3].

II. LOS MÉTODOS DE DISEÑO

El primer método que se aprende para el diseño de circuitos digitales es la edición de esquemáticos. En este caso existe el formato de intercambio EDIF (*Electronic Design Interchange Format*) [4], pero la posibilidad de transferir este tipo de diseños entre diferentes herramientas es muy limitada. Por otro lado, este método no es operativo cuando los circuitos aumentan en complejidad.

Para resolver los inconvenientes anteriores aparecieron los lenguajes de descripción hardware (HDL, *Hardware Description Language*), con ellos se describen los sistemas mediante ficheros de texto; esto facilita su edición y modificación. Los HDL estandarizados más usados son VHDL (*Very High Speed Integrates Circuit Hardware Description. Language*) [5] y Verilog [6].

Uno de los fabricantes más importantes de FPGA es Altera [7], cuya herramienta de implementación es *Quartus* [8], y la herramienta de diseño sobre Simulink [9] de Matlab [10] es *DSP Builder* [11]. El otro suministrador de FPGA igualmente relevante es Xilinx [12], que tiene a *Vivado* [13] como su herramienta de implementación, y a *System Generator* [14] a la utilidad para diseñar desde Simulink y Matlab. Tanto DSP Builder como System Generator aprovechan las ventajas intrínsecas de la plataforma sobre la que se instala. Existen otros entornos que operan sobre Simulink, por ejemplo Synplify [15] de la empresa Synopsys, con ella los diseños pueden llevarse a herramientas de implementación de diversos fabricantes.

Se deduce que existen multitud de formas de diseñar sobre FPGA; Oldfield y Dorf en 1995, ya lo manifestaron: "Otra fuente de confusión es la plétora de herramientas disponibles hoy en día" [1].

Hoy en día hay multitud de herramientas de diseño, pero aparecen convergencias hacia métodos rápidos y flexibles de diseño. Debe destacarse los métodos a nivel de sistemas electrónicos (ESL, *Electronic System Level*), que se imparten en un master [16] del Instituto Tecnológico de Massachusetts (MIT, *Massachusetts Institute of Technology*), donde se engloban las técnicas anteriormente descritas.

III. Prototipado desde Matlab y Simulink

Los métodos avanzados de diseño para FPGA se apoyan en Simulink y Matlab. Estos aprovechan la funcionalidad y prestaciones de este paquete de cálculo matemático. Matlab está ideado como un laboratorio de matrices, su nombre *Matrix Laboratory* deriva del inglés; es un paquete de programas matemáticos que conforma un entorno de desarrollo integrado. Matlab tiene su propio lenguaje de programación, un lenguaje imterpretado, conocido por M. En cuestiones de representación gráfica Matlab tiene altas capacidades. Las funciones de Matlab se agrupan en *Toolboxes*. Matlab incluye Simulink, que es un paquete gráfico de diseño y simulación. Las librerías de Simulink se agrupan en *Blocksets*, y permiten el diseño en forma de diagrama de bloques. Debe resaltarse que en los bloques de Simulink pueden integrarse algoritmos y códigos, en lenguaje M, u otros lenguajes de alto nivel. Simulink y Matlab son ampliamente usados en entornos académicos, de investigación, e industriales. Debe incidirse en la capacidad de la visualización de las señales en las simulaciones y la facilidad para su posterior tratamiento en el espacio de variables de Matlab, una vez se haya finalizado la simulación en Simulink.

IV. Software de diseño de los suministradores de FPGA

Como ya se indicó los fabricantes de FPGA disponen de herramientas de diseño que operan sobre Simulink. Una vez instaladas aparecen los *Blocksets* específicos en Simulink , que son propios del fabricante. Los *Blocksets* agrupan bloques configurables de aritmética de punto fijo; en años recientes, además, incluyen bloques para el diseño en punto flotante. En la Fig. 1 se muestran, de los dos principales suministradores, las herramientas sobre Simulink y de implementación.

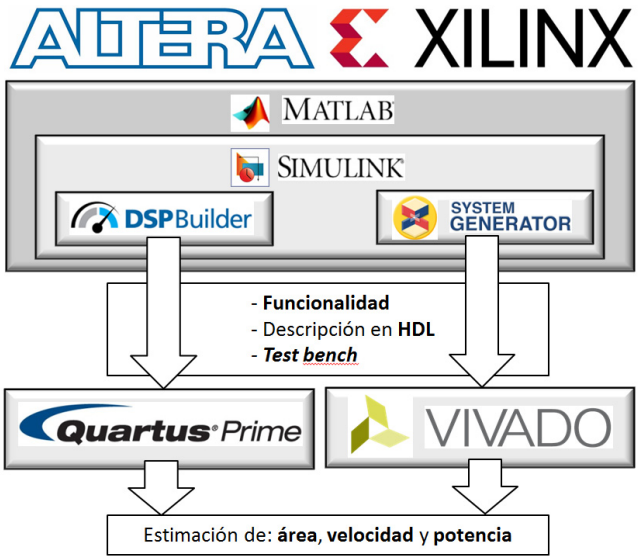

Fig. 1. Escenarios de diseño de FPGA sobre Simulink de Altera y Xilinx.

Como muestra la Fig. 1 al realizar la simulación en Simulink se puede comprobar la funcionalidad del diseño. Desde Simulink, al compilar el sistema de bloques, se obtiene la descripción en un HDL para las primitivas de ese fabricante; por lo tanto, no son códigos portables entre diferentes fabricantes. Asimismo, se obtienen en los *tests benches* necesarios para futuras simulaciones; esto es, las señales de entrada y salida en el HDL elegido.

Todo el código HDL generado es un proyecto manejable por la herramienta de implementación del fabricante, donde se realizan las fases habituales hasta finalizar con la configuración de la FPGA. Es el la herramienta de implementación donde se obtienen los recursos hardware necesarios, la máxima velocidad y el consumo de potencia del dispositivo elegido. La FPGA en cuestión puede elegirse en el entorno Simulink o actualizarse en la herramienta de implementación.

V. Utilidades propias de Matlab y Simulink

Por otro lado, con Matlab se pueden desarrollar diseños basados en edición de modelos y técnicas de verificación; en la Fig. 2 se muestra el flujo de diseño [17] . Es posible, como indica la Fig. 3, generar el código HDL desde un sistema de Simulink, desde código de Matlab o de forma híbrida; en este último caso, el código M se incluye dentro de un bloque de Simulink. Para generar el código HDL se usa el *HDL Workflow Advisor* [18] y el *HDL Coder* [19].

Si se parte de código en punto flotante, este se puede convertir a punto fijo de forma eficaz en el HDL elegido (VHDL o Verilog); para esto se usa el *Fixed-Point Designer* [20]. La utilidad *Fixed-Point Designer* puede operar sobre código de Matlab, modelos de Simulink o diagramas de flujo generados con *Stateflow* [21]. Esta herramienta propone de forma automática el número de bits y el método de redondeo, que puede especificarse manualmente. En las simulaciones se observaría el efecto del rango y la precisión. Es decir, en esta fase se fija el formato de las señales en los diferentes puntos del sistema; si precisan bit de signo, el número de bits para la parte entera y los bits necesarios para la parte fraccionaria.

Una vez generado el código en punto fijo, o si ya se partiera de él, *HDL Workflow Advisor* permite elegir la utilidad que sintetiza el circuito, de las instaladas en la computadora. Igualmente, en esta fase se elige el fabricante y modelo de FPGA. En esta etapa es posible optar entre herramientas de síntesis de Altera o Xilinx. Se permiten otras opciones, como generar un bloque para Simulink, y ver el informe de generación de código. El código HDL generado es legible, está comentado convenientemente y es fácilmente integrable. Se mantiene la trazabilidad del diseño; esto es, se puede analizar la correspondencia entre el código de Matlab y el código HDL.

De la misma forma, es posible generar código HDL desde un diagrama de bloques de Simulink, estos bloques pueden incluir a su vez código Matlab o diagramas de estados diseñado con *Stateflow*.

Desde Simulink también se invoca su propio *HDL Workflow Advisor*, para generar HDL (VHDL o Verilog); este tiene una interface de usuario parecida a la usada en Matlab. Igualmente, se elige primero la herramienta de síntesis y la FPGA. El *HDL Workflow Advisor* de Simulink permite la comprobación de los parámetros del diseño: los bucles algebraicos, la compatibilidad entre los bloques y la compatibilidad de los diferentes intervalos de muestreo. También genera un informe del código HDL generado. En el modelo de Simulink todos los bloques deben ser soportados

por el generador de código, y las opciones de simulación deben ser apropiadas para permitir la generación del código HDL.

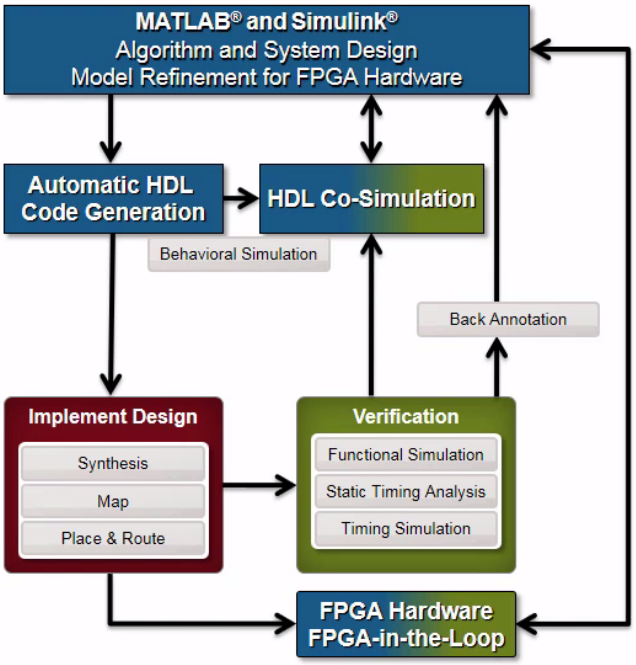

Fig. 2. Flujo de diseño para FPGA con Matlab y Simulink basado en edición de modelos y técnicas de verificación.

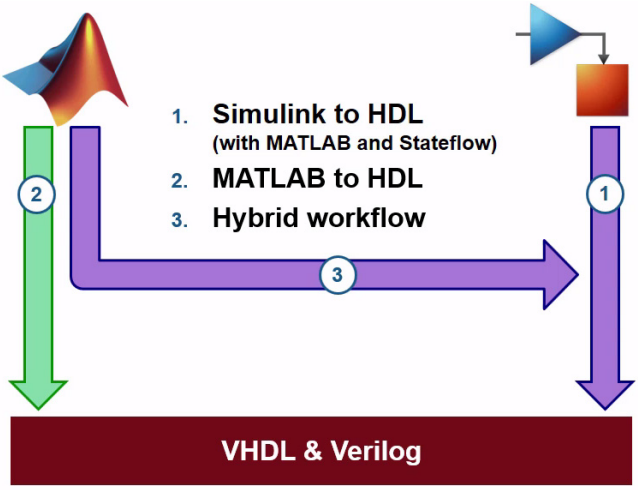

Fig. 3. Generación de código HDL desde Simulink, desde Matlab o de forma híbrida.

A. *Cosimulación*

Antes de la implementación en la FPGA, es habitual simular el código HDL, lo que es una verificación funcional. Analizar las señales en simuladores HDL puede llegar a ser inviable. Por otro lado, la generación de señales para estos simuladores (*test benches*) puede ser de enormemente complicada. Para resolver estos problemas se usa el *HDL Verifier* [22] que permite realizar la cosimulación del sistema [23], como se muestra en las Fig. 4 y 5, en este caso se usó ModelSim [24] como simulador HDL. La cosimulación consiste en enviar las señales de entrada desde Simulink al simulador HDL, iniciar este simulador, y devolver los resultados de a Simulink. Las señales que provienen del simulador HDL pueden ser observadas usando las altas capacidades gráficas de Simulink; por otro lado, pueden ser analizadas con programas auxiliares en Matlab, al estar almacenadas en su espacio de variables. Resumiendo, esta técnica permite la comparación de las simulaciones del código HDL con el modelo de referencia de Simulink. Para la cosimulación Matlab puede acceder a varios simuladores de HDL, debe elegirse uno de estos previamente instalado en la computadora. Las dos simulaciones se ejecutan en paralelo, siendo iniciadas desde el propio Simulink.

La cosimulación tiene como ventaja que evita la generación de señales de prueba (*test benches*), cuya generación manual puede llegar a ser imposible de realizar, si la complejidad de los datos es alta. Con esta técnica se realiza de forma automática las dos simulaciones y se comparan las señales de salida de los dos escenarios. Por otro lado; pueden capturarse y analizarse las señales de obtenidas en el simulador HDL en el entorno de Matlab y Simulink. Finalmente, es posible comparar la funcionalidad del código HDL generado con el modelo Simulink que se usó de referencia.

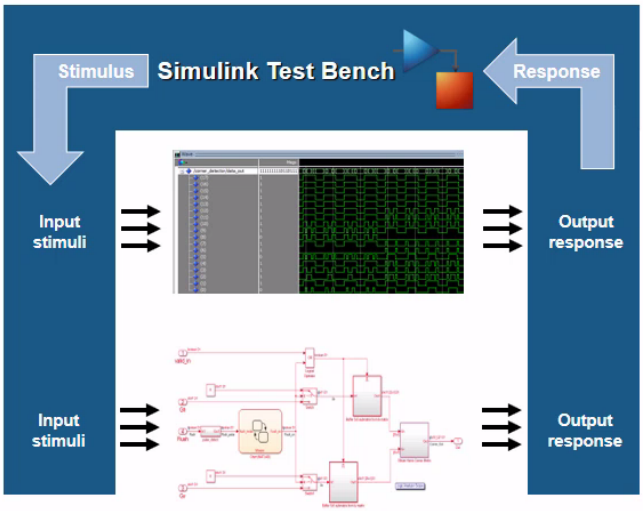

Fig. 4. Flujo de la cosimulación del código HDL y del modelo de Simulink.

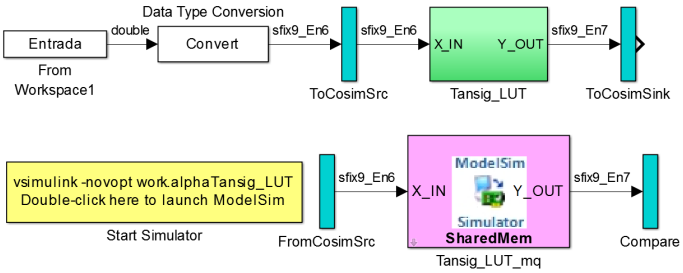

Fig. 5. Sistema Simulink generado para la cosimulación del modelo de referencia y del código HDL.

B. *FPGA in the loop*

El código HDL generado se implementa en la FPGA con la herramienta de Altera [8] o Xilinx [13]. Entonces es posible realizar otra etapa de verificación mediante *HDL Verifier*. Esta etapa se conoce con el nombre de *FPGA in the loop* (FPGA en el bucle) [25], y son pruebas directas sobre la FPGA

configurada, como muestra la Fig. 6. Con esta técnica se vuelve a simular el modelo de referencia de Simulink; de manera simultánea, se envían a la FPGA las señales de entrada, y desde la FPGA son devueltas sus señales de salida. Obviamente la placa que contiene la FPGA se conecta de forma apropiada al ordenador, como muestra la Fig. 7.

Dependiendo de la placa, existen hasta tres formas distintas de conectarlas: a través de un cable JTAG-USB, con un cable Ethernet o una conexión PCI Express.

Las señales de salida en la FPGA son devueltas a Simulink, como muestra la Fig. 8. Esto implica la generación de las señales de entrada a la FPGA y el análisis de sus señales de salida en Simulink. Con *FPGA in the loop* se está comparando la implementación en la FPGA con el modelo de referencia de Simulink. Desde el *HDL Workflow Advisor* se elige la placa para crear el sistema en Simulink, se elige una de las placas soportadas o se configura la placa que se quiera personalizar. Como ventaja cabe destacar que con esta técnica se está probando el sistema implementado sobre la FPGA, sin necesidad de osciloscopios o analizadores lógicos. Las señales de la FPGA, aún capturadas por un analizador lógico, pueden ser imposibles de analizar.

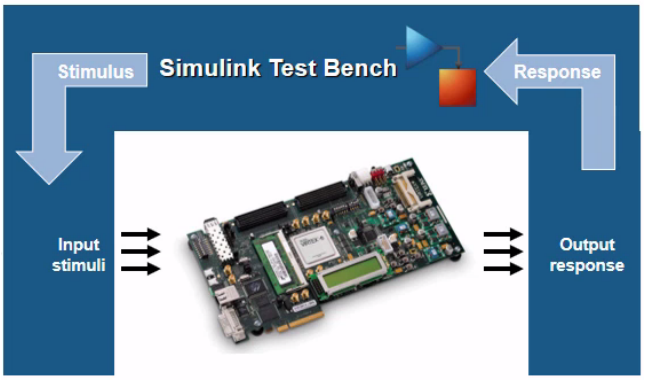

Fig. 6. Flujo de ejecución de *FPGA in the loop* desde Simulink.

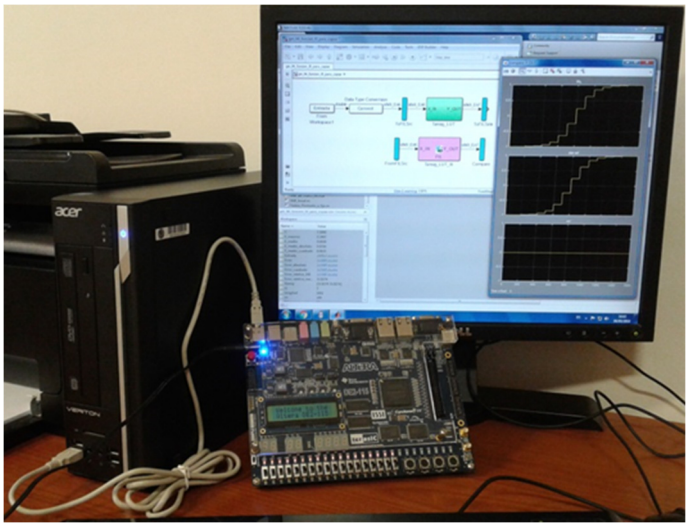

Fig. 7. Montaje para realizar *FPGA in the loop*.

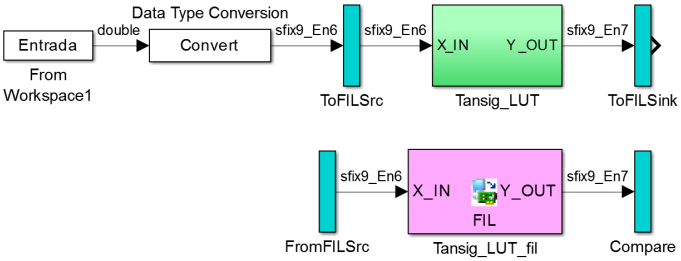

Fig. 8. Sistema Simulink para ejecutar *FPGA in the loop*.

No es posible hacer *FPGA in the loop* para cualquier familia de FPGA; en particular para Altera, las disponibles son las mostradas en la Tabla I [26], donde también se muestran algunas tarjetas disponibles. Debe destacarse que para realizar las pruebas se usó la placa de desarrollo y educación DE2-115 [27], que incluye el dispositivo EP4CE115F29C7N de la familia Cyclone IV E. Esta placa puede ser pedida a Altera como donación dentro de su programa universitario. Para realizar *FPGA in the loop* en esta placa se usó la conexión JTAG, por la que también se configuró la FPGA. El montaje coincide con la Fig. 7.

TABLA I. FAMILIAS DE DISPOSITIVOS DE ALTERA Y TARJETAS DISPONIBLES CON LOS QUE ES POSIBLE REALIZAR *FPGA IN THE LOOP*.

Device Family	Board	Ethernet	JTAG	PCI Express
Altera® Arria® II	Arria II GX FPGA Development Kit	X	X	
Altera Arria V	Arria V SoC Development Kit		X	
	Arria V Starter Kit	X	X	
Altera Arria 10	Arria 10 SoC Development Kit		X	
Altera Cyclone® III	Cyclone III FPGA Starter Kit		X	
	Cyclone III FPGA Development Kit	X	X	
	Altera Nios II Embedded Evaluation Kit, Cyclone III Edition	X	X	
Altera Cyclone IV	Cyclone IV GX FPGA Development Kit	X	X	
	DE2-115 Development and Education Board	X	X	
	BeMicro SDK	X	X	
Altera Cyclone V	Cyclone V GX FPGA Development Board	X	X	
	Cyclone V SoC Development Kit		X	
	Cyclone V GT Development Kit	X	X	X
	Atlas-SoC Kit/DE0–Nano SoC Kit		X	
	Arrow® SoC Kit development kit		X	
Altera MAX® 10	Arrow MAX 10 DECA	X	X	
Altera Stratix® IV	Stratix IV GX FPGA Development Board	X	X	
Altera Stratix V	Stratix V DSP Development Kit	X	X	X

La conexión del ordenador anfitrión a la placa se muestra en la Fig. 9. Para ejecutar la *FPGA in the loop* se genera un proyecto que incluye el sistema diseñado (DUT, *Device Under Test)* a la que se añade un bloque de lógica para comunicar el diseño con la computadora a través del cable de conexión. Esta conexión es la que permite llevar las señales de entradas de Matlab al circuito digital y devolver sus salidas a Matlab. Obviamente, el proyecto final en la FPGA debe coincidir con el bloque llamado DUT, que es el código HDl generado automáticamente sin la opción *FPGA in the loop*.

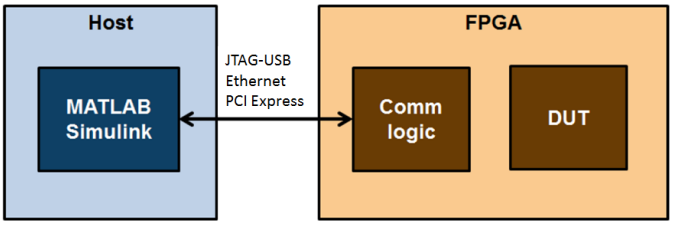

Fig. 9. Esquema de conexión para *FPGA in the loop*.

Este método de diseño integrado en Matlab permite explorar el espacio de soluciones. Para un sistema son posibles múltiples arquitecturas; además, para cada arquitectura es posible cambiar el formato de los datos y los operadores en los diferentes puntos del sistema. El diseño debe cumplir con una funcionalidad dada, que se puede medir en Matlab; cada posible solución tiene sus propias prestaciones físicas de área, velocidad y consumo de potencia; que se puede medir con la herramienta de implementación. Solo serían válidos los diseños que están dentro del paralelepípedo dibujado en la Fig. 10 si se establecen limitaciones para estos tres parámetros.

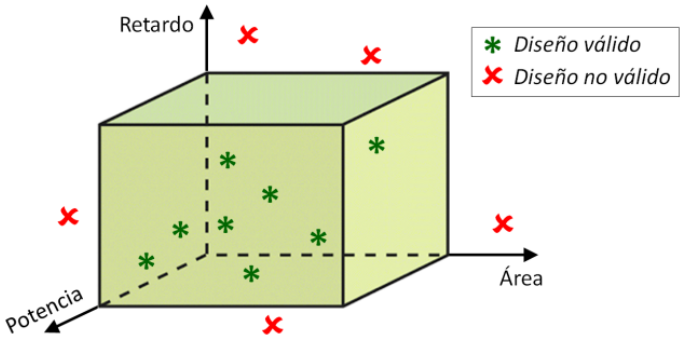

Fig. 10. Exploración de los diseños que cumplen con las restricciones de área, velocidad y potencia.

VI. Conclusiones

El método de diseño descrito, basado en modelos y en etapas de verificación, permite la exploración del espacio de soluciones. Si en alguna de las fases de verificación, cosimulación o *FPGA in the loop*, no se alcanza la funcionalidad o rendimiento deseado, se puede retomar el flujo de la Fig. 2.

Este flujo permite la rápida descripción del sistema en punto flotante, su traducción a punto fijo, y la generación automática del código HDL.

La rapidez de las simulaciones, y el análisis y visualización de las señales, permite verificar rápidamente la funcionalidad de los sistemas. Resumiendo, se acorta el tiempo de diseño.

Deben destacarse, para sistemas complejos, que la generación manual de código HDL puede ser altamente complicada; tanto la descripción del circuito como del *test bench*. Esta generación manual presenta gran tiempo de diseño, dificultad para modificaciones, dificultad en la detección de errores, complejidad para probar diferentes arquitecturas y formatos de datos. Por otro lado en los simuladores de código HDL puede ser imposible analizar datos complejos.

En las universidades españolas los planes de estudio inciden en los métodos clásicos de diseño basados en la descripción directa en un HDL; si bien deben impartirse, datan de hace una treintena de años. Los métodos sobre Matlab y Simulink están relegados a actividades de investigación.

Algunos diseñadores piensan que Matlab es exclusivo para procesado de señal en aritmética de punto flotante, cuando puede ayudar en la implementación de sistemas digitales en punto fijo; otros creen que Simulink es un mero editor de diagrama de bloques, pero en realidad puede incluirse código secuencial y máquinas de estado. Por lo tanto, debe hacerse una profunda reflexión acerca de los métodos de diseño en escenarios académicos; por descontado deben impartirse los métodos clásicos de generación manual de código HDL y edición de esquemáticos, pero han de incluirse en los planes de estudio, principalmente en master y doctorado, estos métodos de diseño avanzados. De no ser así, estos métodos que además evolucionan rápidamente, quedarán sin los especialistas deseados.

Agradecimientos

Se agradece a MathWorks el permiso para usar sus gráficos y figuras en la redacción de esta ponencia.

A Xilinx se le agradece el uso de sus gráficos, y la donación de software y licencias dentro de su programa universitario.

Finalmente, se agradece a Altera el uso de las figuras, la donación de software y licencias, y la donación de la placa de desarrollo y educación DE2-115 dentro de su programa universitario.

Referencias

[1] J. Oldfield y R. Dorf, Field-Programmable Gate Array, John Wiley and Sons, 1995.

[2] W. Meeus, K. Van, T. Goedemé, J. Meel y D. Stroobandt, «An overview of today's high-level synthesis tools,» *Design Automation of Embedded Systems,* vol. 16, nº 3, pp. 31-51, 2012.

[3] IEEE, «IEEE Standard for Floating-Point Arithmetic,» [En línea]. Available: http://ieeexplore.ieee.org/servlet/opac?punumber=4610933, permanent link. [Último acceso: june 2017].

[4] International Electrotechnical Commission, «Electronic Design Interchange Format (EDIF), Part 2, Version 4.0.0,» 2000.

[5] IEEE, «IEEE Standard VHDL Language Reference Manual,» [En línea]. Available: http://ieeexplore.ieee.org/servlet/opac?punumber=4772738, permanent link. [Último acceso: june 2017].

[6] IEEE, «IEEE Standard for Verilog Hardware Description Language,» [En línea]. Available: http://ieeexplore.ieee.org/servlet/opac?punumber=10779, permanent link. [Último acceso: june 2017].

[7] Altera, «Altera Corporation,» [En línea]. Available: http://www.altera.com/. [Último acceso: june 2014].

[8] Altera, «Intel Quartus Prime,» [En línea]. Available: https://www.altera.com/products/design-software/fpga-design/quartus-prime/overview.html. [Último acceso: June 2017].

[9] MathWorks, «Simulink,» [En línea]. Available: http://www.mathworks.com/products/simulink. [Último acceso: june 2014].

[10] MathWorks, «MATrix LABoratory de MathWorks (Matlab),» [En línea]. Available: http://www.mathworks.com. [Último acceso: june

2014].

[11] Altera, «DSP Builder,» [En línea]. Available: http://www.altera.com/products/software/products/dsp/dsp-builder.html. [Último acceso: june 2014].

[12] Xilinx, «Xilinx Corporation,» [En línea]. Available: http://www.xilinx.com. [Último acceso: june 2014].

[13] Xilinx, «Vivado Design Suite,» [En línea]. Available: https://www.xilinx.com/products/design-tools/vivado.html. [Último acceso: june 2017].

[14] Xilinx, «System Generator for DSP,» [En línea]. Available: http://www.x.com/products/design-tools/vivado/integration/sysgen.html. [Último acceso: june 2014].

[15] Synopsys, «Synplify,» [En línea]. Available: http://www.synopsys.com/Tools/Implementation/FPGAImplementation/FPGASynthesis/Pages/SynplifyFeatureComparisonChart.aspx. [Último acceso: October 2016].

[16] XCell, «MIT Prof Uses ESL Tools, FPGAs to Teach System Architecture,» [En línea]. Available: http://www.bluespec.com/downloads/XCELL_76-Xpert_opinion.pdf. [Último acceso: October 2016].

[17] MathWorks, «Generación y Verificación de HDL para FPGAs, ASICs y SoCs desde MATLAB y Simulink,» [En línea]. Available: http://es.mathworks.com/videos/hdl-code-generation-and-verification-for-fpgas-asics-and-socs-from-matlab-and-simulink-122850.html?form_seq=conf756&elqsid=1475083337040&potential_use=Education&country_code=ES. [Último acceso: Octubre 2016].

[18] MathWorks, «HDL Workflow Advisor,» [En línea]. Available: https://www.mathworks.com/examples/matlab-hdl-coder/mw/hdlcoder_product-mlhdlc_tutorial_hdlcodegen-basic-hdl-code-generation-with-the-workflow-advisor. [Último acceso: june 2017].

[19] MathWorks, «HDL Coder,» [En línea]. Available: http://www.mathworks.es/products/hdl-coder. [Último acceso: October 2016].

[20] MathWorks, «Fixed-Point Designer,» [En línea]. Available: http://www.mathworks.es/products/fixed-point-designer. [Último acceso: june 2017].

[21] MathWorks, «Stateflow,» [En línea]. Available: https://es.mathworks.com/products/stateflow/. [Último acceso: October 2016].

[22] MathWorks, «HDL Verifier,» [En línea]. Available: http://www.mathworks.es/products/hdl-verifier. [Último acceso: October 2016].

[23] MathWorks, «HDL Cosimulation,» [En línea]. Available: https://es.mathworks.com/help/hdlverifier/gs/hdl-cosimulation.html. [Último acceso: November 2017].

[24] Altera, «ModelSim-Altera Edition,» [En línea]. Available: https://www.altera.com/products/design-software/model---simulation/modelsim-altera-software.html. [Último acceso: june 2017].

[25] MathWorks, «FPGA in the loop,» [En línea]. Available: http://es.mathworks.com/help/hdlverifier/ug/fpga-in-the-loop-fil-simulation.html. [Último acceso: October 2016].

[26] MathWorks, «Intel FPGA Board Support from HDL Verifier,» [En línea]. Available: https://es.mathworks.com/help/supportpkg/alterafpgaboards/ug/altera-fpga-board-support-from-hdl-verifier.html. [Último acceso: March 2018].

[27] Altera, «DE2-115 User Manual,» 2012. [En línea]. Available: ftp://ftp.altera.com/up/pub/Altera_Material/13.0/Boards/DE2-115/DE2_115_User_Manual.pdf. [Último acceso: March 2018].