SÍNTESIS HARDWARE DE MÓDULOS DE INFERENCIA DIFUSOS MEDIANTE HERRAMIENTAS DE DISEÑO DE DSP

M. BROX[1], S. SÁNCHEZ-SOLANO[2], P. BROX[2], I. BATURONE[2], A. BARRIGA[2], A. GERSNOVIEZ[1]
[1]*Departamento de Arquitectura de Computadores, Electrónica y Tecnología Electrónica. Escuela Politécnica Superior. Universidad de Córdoba. España.*
[2]*Instituto de Microelectrónica de Sevilla. Centro Nacional de Microelectrónica. CSIC. España.*

En esta comunicación se describe una nueva estrategia de desarrollo de sistemas de control basados en lógica difusa mediante un flujo de diseño que combina las herramientas de modelado y simulación del entorno Matlab y las herramientas de síntesis e implementación de FPGAs de Xilinx. Apoyada en el uso de una librería de módulos específicos para sistemas difusos, esta estrategia acelera las etapas de descripción, síntesis y verificación funcional de los sistemas bajo desarrollo.

1. Introducción

Los constantes avances en el desarrollo de dispositivos lógicos programables como las FPGAs han propiciado la utilización de este tipo de elementos para implementar sistemas de procesado digital de señal (DSP) de elevadas prestaciones. La estructura lógica de las FPGAs actuales incluye no solo un elevado número de bloques lógicos configurables (tablas look-up, registros, multiplexores, etc.), sino también circuitos dedicados como sumadores, multiplicadores y bloques de memoria. Estas características hacen que el empleo de FPGAs presente numerosas ventajas para el desarrollo de aplicaciones específicas de procesado digital de señal frente al uso de los DSPs tradicionales, cuya funcionalidad puede programarse mediante software, pero que disponen de una estructura hardware poco flexible: tamaño de datos predeterminado, capacidad de memoria fija y número limitado de bloques aritméticos.

En los últimos años se han desarrollado una serie de herramientas, como el entorno System Generator (SysGen) de Xilinx, que facilitan el diseño de algoritmos DSP sobre FPGAs. SysGen es una herramienta software para la descripción y síntesis de sistemas de procesado de señal basada en Simulink (la herramienta interactiva para el modelado, la simulación y el análisis de sistemas dinámicos integrada en Matlab). El entorno permite manejar un nivel de abstracción elevado y proporciona un entorno gráfico que facilita la descripción del algoritmo que se va a implementar. Una vez verificada la funcionalidad del diseño, SysGen lo traslada de forma automática en una implementación hardware optimizada en términos de área y velocidad.

Por otra parte, la disponibilidad de placas de prototipado que facilitan la implementación física de los diseños, permite recorrer de forma rápida todo el ciclo de desarrollo y verificación de los sistemas. Estas plataformas suelen incluir dispositivos lógicos programables, junto a una serie de periféricos que permiten ampliar la funcionalidad de las mismas. Los principales fabricantes de placas y dispositivos programables ofrecen, a través de sus programas universitarios [1]-[3], paquetes de desarrollo que incluyen la placa, las herramientas software necesarias para la programación de nuevos diseños y una serie de demostradores y notas de aplicación de gran valor didáctico. La adquisición de este material a través de dichos programas universitarios permite una reducción significativa del costo y facilita el acceso a una abundante documentación sobre aplicaciones, así como a foros de consultas y discusión.

[2] Este trabajo ha sido soportado por los proyectos TEC2005-04359/MIC del Ministerio de Educación y Ciencia y TIC2006-635 y TEP2006-375 de la Junta de Andalucía.

En este trabajo se describe una novedosa técnica de diseño de sistemas de inferencia difusos y su aplicación a la síntesis sobre una FPGA de un controlador difuso para el aparcamiento de un vehículo autónomo. Dicha técnica está basada en el uso de la herramienta SysGen dentro del entorno Matlab, lo que permite abordar el flujo de diseño completo sobre un único entorno de desarrollo que facilita, tanto la descripción y síntesis hardware del sistema de control, como su verificación funcional mediante técnicas de cosimulación hardware/software en las que un modelo del vehículo descrito en Matlab interacciona con la implementación del controlador residente en la placa de desarrollo de FPGAs. La estructura de la comunicación es la siguiente: en el apartado 2 se introducen brevemente las principales características de SysGen, el flujo de diseño asociado a dicha herramienta y su uso para la síntesis de sistemas difusos sobre FPGAs. La librería de módulos *XfuzzyLib*, que constituye la base de la nueva estrategia de síntesis, se describe en el apartado 3. El apartado 4 ilustra la aplicación de esta técnica de diseño al desarrollo de un sistema de control de navegación de un vehículo autónomo. Por último, las principales conclusiones del trabajo se resumen en el apartado 4.

2. Síntesis de sistemas difusos con SysGen

SysGen facilita el desarrollo de algoritmos de DSP sobre las FPGAs de Xilinx [4]. Integrada en el entorno Matlab, esta herramienta incluye una librería de módulos de Simulink denominada "Xilinx Blockset", así como el software necesario para trasladar el modelo matemático descrito en Simulink en una implementación hardware eficiente. El ciclo de diseño típico con esta herramienta comienza con la descripción esquemática del sistema mediante un modelo Simulink. La verificación se lleva a cabo mediante simulación y utilizando las distintas facilidades de generación y visualización de señales proporcionadas por Matlab. Una vez verificada la funcionalidad del diseño, SysGen lo traslada de forma automática a su especificación en un lenguaje de descripción de hardware (HDL) que puede usarse como entrada a las herramientas de síntesis e implementación de FPGAs. Para llevar a cabo este proceso, SysGen chequea los parámetros indicados en cada uno de los módulos del modelo Simulink y determina las entidades, arquitecturas, puertos, señales y atributos de la descripción HDL. El código obtenido puede ser sintetizado con la herramienta XST integrada en ISE o exportado a otras herramientas de síntesis.

La Figura 1 muestra el diagrama de bloques de una arquitectura eficiente para la implementación digital de sistemas difusos basada en el procesado de reglas activas [5]. Los circuitos generadores de funciones de pertenencia (MFC), encargados de la etapa de fuzzificación, suministran parejas de datos etiqueta-grado de pertenencia (L_i, μ_i) para cada valor de las entradas del sistema. Las distintas combinaciones de las etiquetas determinarán las reglas que se activen. En la etapa de inferencia se procesan secuencialmente cada una de las reglas activas, utilizándose para ello un circuito selector de reglas activas formado por un conjunto de multiplexores controlados por un contador. En cada ciclo de reloj los grados de pertenencia de cada una de las entradas son combinados a través del conectivo de antecedentes para calcular el grado de activación de la regla, mientras que las respectivas etiquetas direccionan la posición de memoria que contiene los parámetros que representan su correspondiente consecuente. En la etapa de defuzzificación, un único bloque es el encargado de calcular el valor de salida combinando los consecuentes de las reglas con sus diferentes grados de activación según el método simplificado que se utilice. Finalmente, un bloque de control se encarga de regular la temporización del proceso de inferencia.

Esta arquitectura se caracteriza por ser altamente configurable debido a la posibilidad de implementar cada uno de los bloques de la Figura 1 a partir de diferentes realizaciones de circuito. Los MFCs pueden ser implementados siguiendo estrategias de almacenamiento en memoria o cálculo aritmético. El grado de activación de las reglas puede calcularse combinando los grados de pertenencia de los antecedentes mediante los operadores mínimo o producto. Por último, el diseñador puede seleccionar el tipo de defuzzificador que mejor se adapte a la heurística del sistema de inferencia bajo desarrollo.

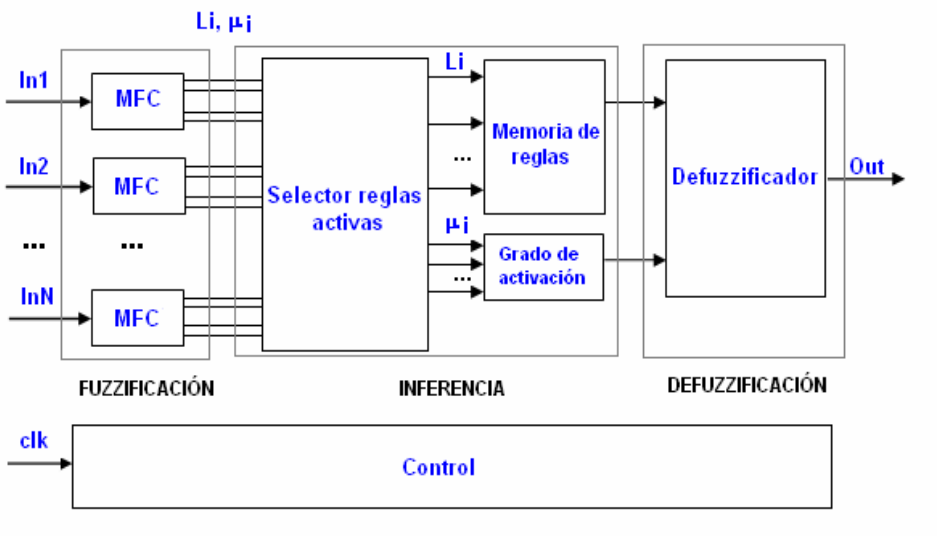

Figura 1. Arquitectura de un sistema de inferencia difuso basada en reglas activas

Haciendo uso de la librería "Xilinx Blockset" y utilizando las facilidades proporcionadas por Simulink, se ha generado una librería de módulos especializados, *XfuzzyLib*, compuesta por bloques que describen la estructura general de la arquitectura. A partir de los bloques de la librería, la construcción de un sistema difuso requiere únicamente la elección, interconexión y parametrización de los bloques necesarios. Una vez que se ha definido el sistema difuso de acuerdo a la arquitectura propuesta, se sigue el flujo de diseño descrito anteriormente para su implementación en una FPGA.

3. Librería de módulos SysGen para sistemas difusos

Para cubrir las etapas de fuzzificación, inferencia y defuzzificación de un sistema difuso se ha diseñado un conjunto de bloques encargados de la generación de funciones de pertenencia, el procesado secuencial de las reglas activas, el suministro de las señales de control, el cálculo del grado de activación de las reglas, el almacenamiento de los consecuentes de la base de reglas y la realización de tareas de acumulación y división para diversos métodos de defuzzificación (ver Figura 2).

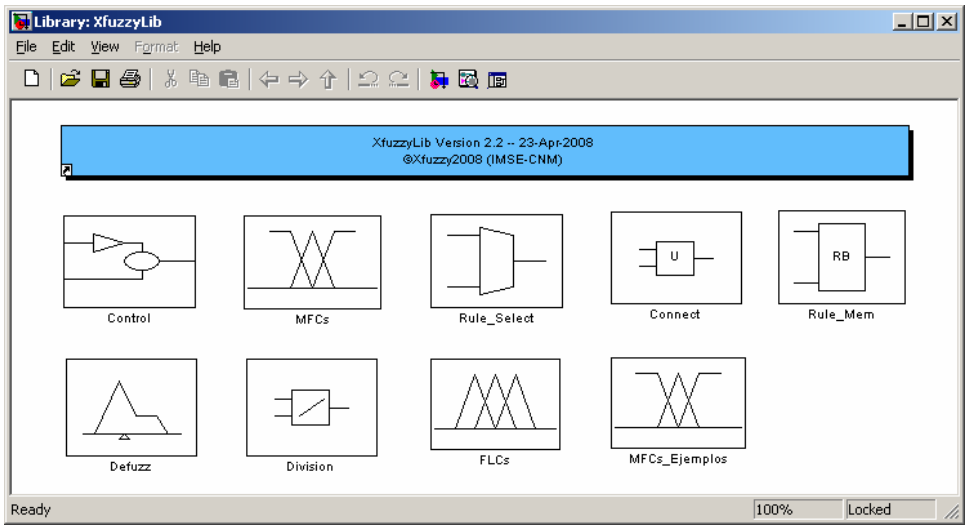

Figura 2. Conjunto de bloques de la librería de módulos *XfuzzyLib*

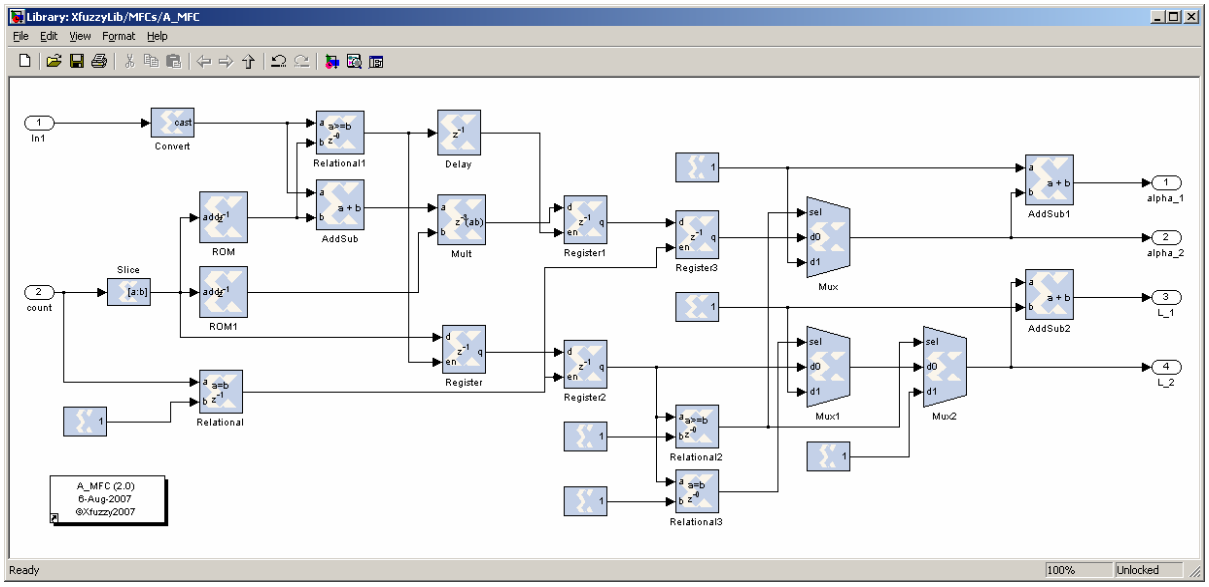

Figura 3. Modelo Simulink del módulo de generación de funciones de pertenencia (A_MFC)

A modo de ejemplo, la Figura 3 muestra el esquema de conexionado del módulo encargado de la generación de funciones de pertenencia mediante técnicas aritméticas. Este bloque es capaz de generar familias de funciones de pertenencia triangulares (salvo la primera y la última función que pueden ser de tipo "Z" y "S", respectivamente) con grado de solapamiento 2 y normalizadas (en cualquier punto la suma de ambos grados de pertenencia es igual a la unidad). Los distintos módulos de la librería "Xilinx Blockset" admiten una serie de parámetros que definen aspectos tales como su funcionalidad, tamaño o tipo de aritmética utilizada. Del mismo modo, una vez realizado el diagrama de bloques de un nuevo módulo de la librería *XfuzzyLib*, se encapsula como un "subsistema" y se añade una "máscara" para identificar los distintos parámetros que caracterizan al módulo. De esta forma, cuando dicho subsistema es usado en un nivel jerárquico superior, estos parámetros pueden asignarse mediante valores numéricos o mediante variables a las que se les asigna un valor numérico en la ventana de comandos de Matlab o a través de un fichero ".m". Como muestra la Figura 4, la funcionalidad del diseño puede verificarse en todo momento con ayuda del simulador Simulink y sus diferentes facilidades para generar señales de excitación y para capturar y representar gráficamente los datos de salida.

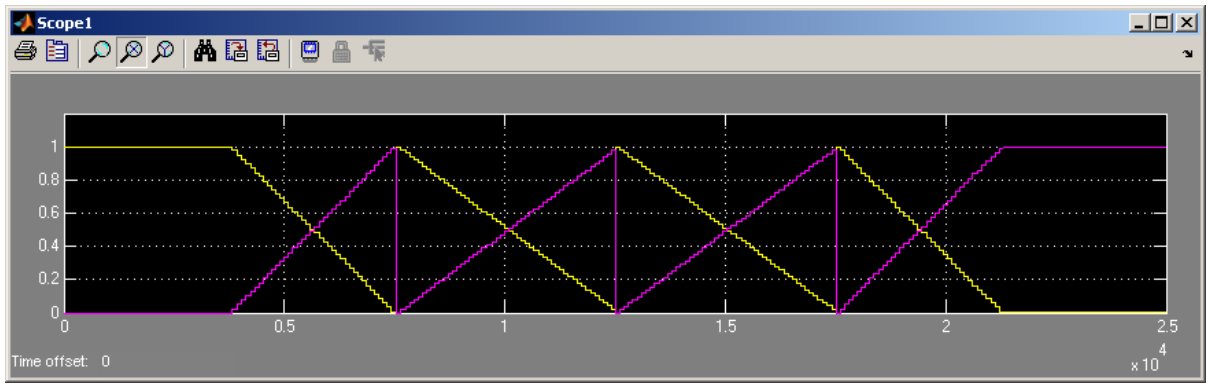

Figura 4. Verificación de la funcionalidad del módulo A_MFC mediante Simulink

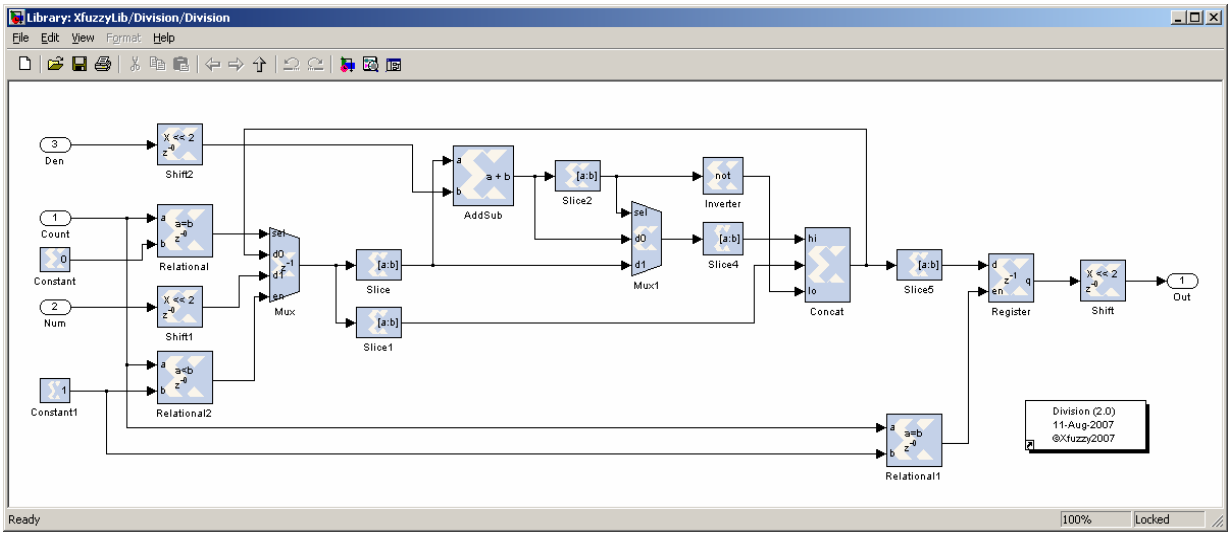

Figura 5. Esquema de conexión del módulo de la librería correspondiente a la etapa de división

Por razones de eficacia en la estructuración de la librería, la etapa de división necesaria en la mayoría de los métodos de defuzzificación se ha dejado fuera de los módulos que implementan dichos métodos y se ha incluido un módulo divisor que puede ser utilizado por cualquiera de ellos. El esquema mostrado en la Figura 5 corresponde a un algoritmo de "resto sin restauración", el cual se basa en la realización de desplazamientos y operaciones de sumas y restas para el cálculo de cada uno de los bits del cociente.

Además de los bloques que constituyen los elementos constructivos básicos de un sistema difuso, la librería *XfuzzyLib* incluye bloques con descripciones de arquitecturas tipo de módulos de inferencia difusos que se diferencian entre sí en el número de entradas, el conectivo que usan para calcular el grado de activación de las reglas y el método de defuzzificación empleado. La inclusión de estos bloques da la opción al usuario de elegir entre construir un sistema difuso determinado a partir de estas arquitecturas tipo o bien mediante la interconexión de los bloques de elementos básicos que sean necesarios. Asimismo, es posible combinar de forma jerárquica estos módulos de inferencia simples para definir sistemas difusos de mayor complejidad. El diagrama de bloques de un controlador difuso (FLC) de dos entradas que emplea el mínimo como conectivo y FuzzyMean como defuzzificador se muestra en la Figura 6.

Como ocurre con los bloques básicos de la librería *XfuzzyLib*, los bloques que describen arquitecturas de FLCs son totalmente parametrizables, por lo que el diseñador de sistemas difusos puede adaptar la funcionalidad del módulo a los requerimientos del problema planteado sin más que definir los parámetros adecuados. Básicamente hay dos tipos de parámetros: los que corresponden al dimensionado del sistema de inferencia, como son el número de bits de las entradas, salidas y grados de pertenencia, y los que definen la base de conocimiento del mismo, es decir, las funciones de pertenencia y la base de reglas. Para facilitar su uso por parte del diseñador, los distintos parámetros corresponden a variables y estructuras de datos a las que pueden asignarse valores a través de la propia interfaz gráfica de Simulink o utilizando un fichero ".m" de Matlab.

Análogamente a los bloques básicos de la librería, también en este caso es posible aprovechar toda la funcionalidad del entorno Matlab para verificar la funcionalidad del sistema de inferencia. En particular

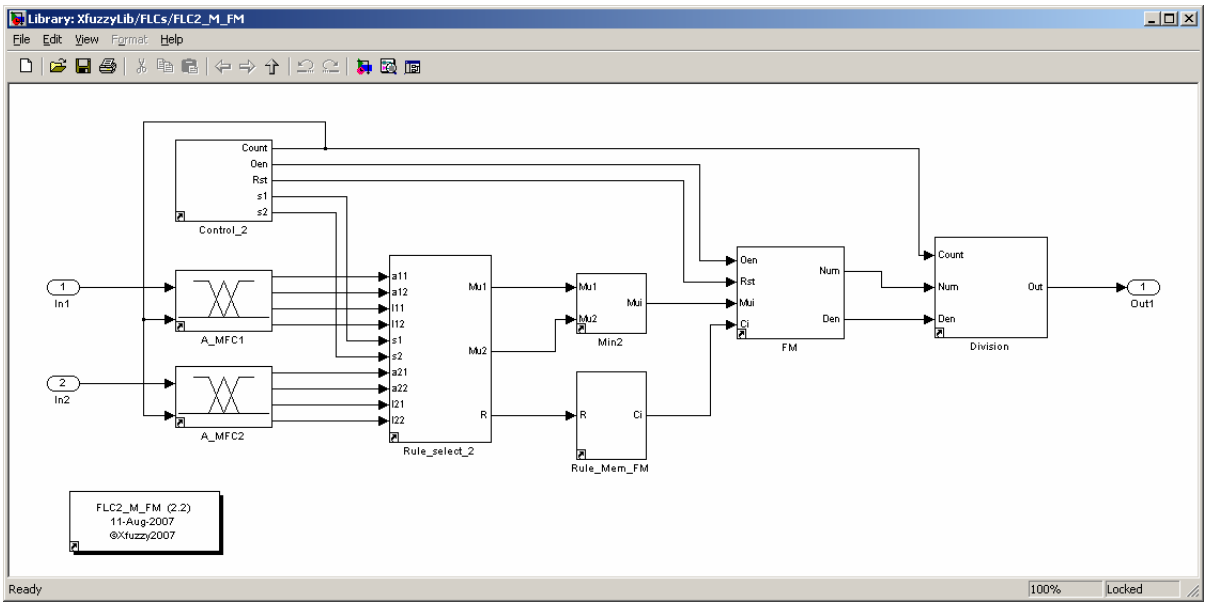

Figura 6. Esquema de un FLC de 2 entradas con conectivo mínimo y defuzzificación FuzzyMean

resulta de utilidad el uso de fuentes de señal para recorrer los universos de discurso de las variables de entrada, de elementos de adquisición de datos que permiten observar la evolución temporal de la salida del sistema, y de bloques de almacenamiento de datos en ficheros que facilitan la representación gráfica de la superficie entrada/salida del sistema (Figura 7).

Finalmente, la incorporación en el modelo del bloque "System Generator" facilita la síntesis hardware del diseño descrito en Simulink mediante la traslación del modelo del FLC a diferentes tipos de netlists, la generación del fichero de programación de la FPGA e incluso, como describiremos en el siguiente apartado, la verificación funcional del controlador en lazo cerrado a través de la cosimulación hardware/software de la implementación del sistema de control junto a un modelo de la planta.

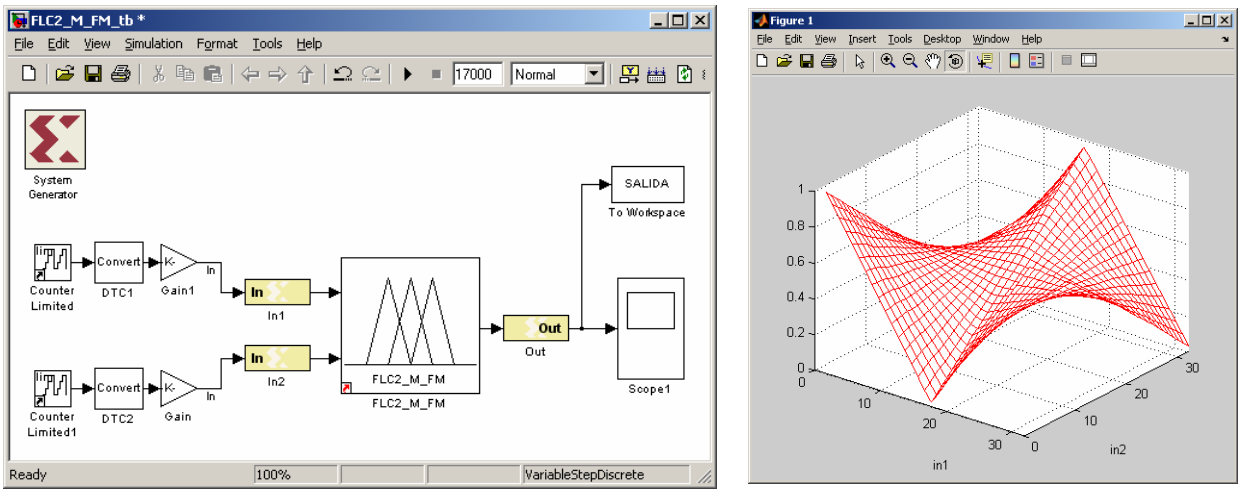

Figura 7. Modelado de un sistema de inferencia difuso y "superficie de control" obtenida

4. Aplicación a un sistema de control para robótica

Con objeto de validar la técnica de desarrollo propuesta, se ha empleado en el diseño un sistema de control típico en robótica: el problema de aparcamiento de un vehículo autónomo en un espacio limitado [6]-[7]. En el caso considerado, el vehículo puede estar ubicado en una posición cercana a la plaza de aparcamiento, por lo que la trayectoria a seguir debe combinar desplazamientos hacia delante y hacia atrás para alcanzar la posición objetivo. El vehículo utilizado es un robot eléctrico autónomo (ver Figura 8) dotado de una serie de sensores entre los que destacan, para nuestro objetivo de control, un giróscopo y encoders de tracción y dirección. El encoder de tracción, junto con el giróscopo, permite obtener los valores de la posición, orientación y velocidad del vehículo, aplicando odometría. A su vez, el encoder de dirección permite obtener los valores de la curvatura del vehículo. El control a bajo nivel de los motores se realiza mediante un procesador digital de señal (DSP) TMS-320LF de Texas Instruments, que además se ocupa de la adquisición de la información procedente de los sensores y su procesado para determinar la orientación y posición del vehículo de acuerdo al modelo cinemático normalmente empleado para robots móviles [8]. De esta forma se libera al controlador de alto nivel de las tareas de adquisición y procesado de la información procedente de los sensores así como del control directo de los motores, pudiendo dedicarse en exclusividad a la ejecución del algoritmo de control de navegación del vehículo.

El sistema de control utilizado emplea una base de conocimiento jerárquica que incluye módulos de toma de decisiones (para discriminar si el vehículo debe circular en un sentido u otro) y módulos de control (para calcular la velocidad y el ángulo de giro de las ruedas). Las bases de reglas *Position*, *Planning* y *Direction* (ver Figura 8) corresponden a estructuras de toma de decisiones. El espacio de aparcamiento se divide en varias zonas, siendo *Position* la que nos indica en qué zona se encuentra el vehículo. De esta forma, se sabe si el vehículo se encuentra cerca o lejos de la posición objetivo en función del valor de las coordenadas (x, y). Su salida se combina con la orientación del vehículo (Φ) en la base de reglas *Planning* para obtener la planificación del aparcamiento (plan). Esta base de reglas toma la

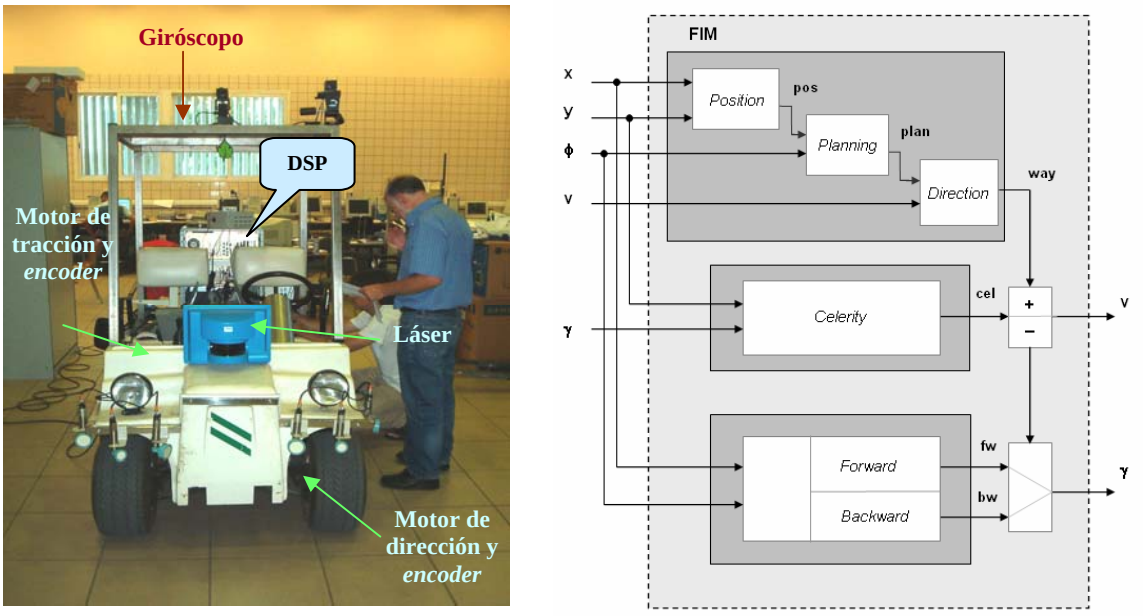

Figura 8. Vehículo autónomo Romeo-4R y sistema de control para el problema de aparcamiento

decisión del sentido de la marcha y debe garantizar que finalmente se llegue al lugar de aparcamiento sin que se produzcan colisiones con el muro y=0. Su salida plan se combina con el valor anterior de la velocidad (v) en la base de reglas *Direction* debido a que no se puede invertir bruscamente la polaridad del voltaje aplicado al motor de tracción. Su salida (way) indica el sentido que definitivamente debe seguir el vehículo. Por otro lado, las bases de reglas *Celerity*, *Backward* y *Forward* corresponden a estructuras de control. La base de reglas *Celerity* determina el valor absoluto de la velocidad (cel), mientras que las bases de reglas *Backward* y *Forward* determinan la curvatura del vehículo en un sentido u otro (bw, fw). La salida final del controlador (v) se obtiene mediante la combinación en un multiplicador de la salida que decide el sentido de la circulación (way) con las correspondientes al valor absoluto de la velocidad (cel). Por otro lado, la otra salida final (γ) se obtiene mediante un simple selector que elige entre las salidas de curvatura de Romeo 4R en un sentido u otro (bw y fw).

Para desarrollar esta estructura del controlador se ha hecho uso del entorno *Xfuzzy* [9], que permite describir sistemas difusos con estructura jerárquica. *Xfuzzy* incluye asimismo herramientas que facilitan la verificación funcional del controlador combinando la descripción del controlador con un modelo matemático de la planta escrito en un lenguaje de programación de alto nivel. Una vez obtenida una especificación funcionalmente correcta, puede comenzar la fase de síntesis HDL mediante la técnica de implementación basada en SysGen descrita anteriormente. La traslación de la especificación de un sistema difuso en el entorno *Xfuzzy* a la herramienta SysGen de Matlab requiere la identificación de los bloques de la librería de módulos *XfuzzyLib* que implementarán las distintas bases de reglas y la definición de los parámetros que caracterizan a dichos bloques: los relativos al dimensionado del sistema de inferencia y los que definen las bases de conocimiento. El resultado de este proceso para el sistema de control considerado se muestra en la Figura 9.

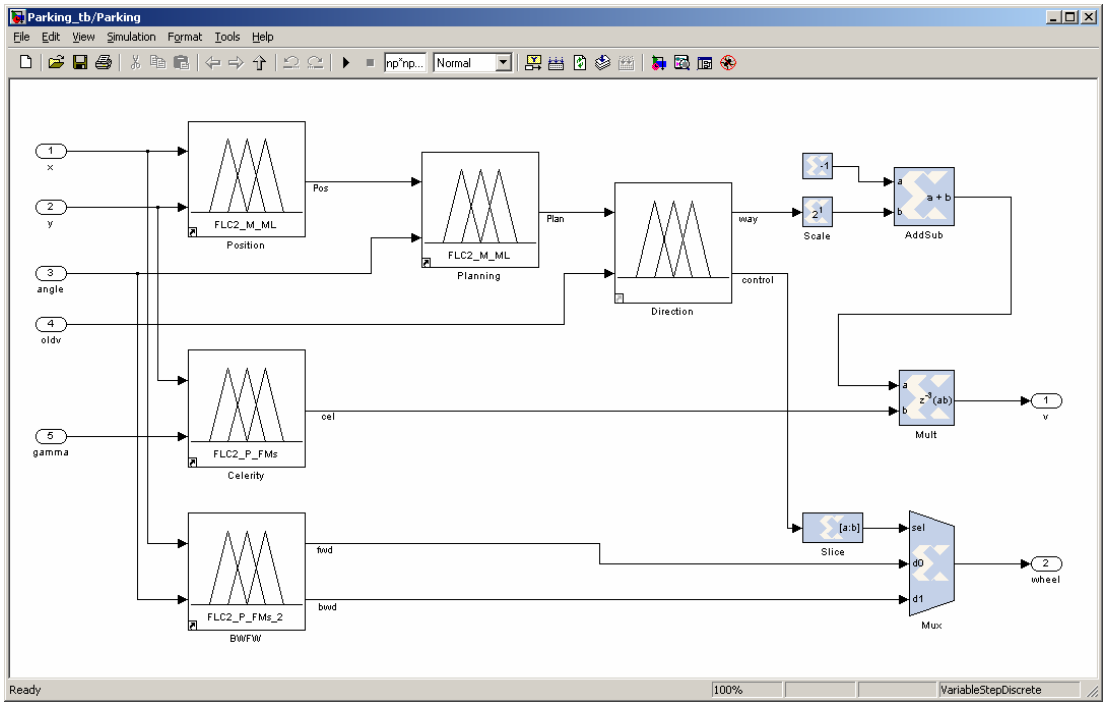

Figura 9. Descripción del controlador difuso con *XfuzzyLib*

Una vez completado el modelo del sistema, las herramientas del entorno Matlab permiten llevar a cabo la verificación funcional del controlador. En esta etapa de desarrollo resulta de especial interés la posibilidad de simular el sistema en lazo cerrado incluyendo un modelo Simulink del vehículo bajo control. Cuando los resultados de simulación son aceptables se procede a la síntesis del controlador difuso seleccionando uno de los tipos de compilación disponibles a través del bloque "System Generator". La Figura 10 ilustra una de las facilidades proporcionadas por la herramienta, que permite verificar la funcionalidad del sistema sintetizado mediante la integración en un lazo cerrado de cosimulación del modelo software del robot y la implementación hardware del controlador sobre la FPGA. Algunos de los resultados obtenidos en esta etapa se muestran en la Figura 11.

La técnica de diseño propuesta es válida para las distintas placas de desarrollo que incluyen FPGAs de Xilinx. La implementación del controlador mostrado en la Figura 9, utilizando 8 bits en las entradas y salidas de las distintas bases de reglas y empleando una placa XUP-V2Pro (que incluye una FPGA Virtex-2 Pro XC2VP30 con 13.696 slices, 136 multiplicadores de 18 bits y 2.448Kb de memoria RAM) ocupa 1.094 slices y emplea 16 multiplicadores (lo que representa, respectivamente, el 7% y el 11% de los recursos disponibles en la FPGA). La frecuencia de operación alcanzada supera los 100 Mhz.

5. Conclusiones

Se ha presenta una técnica de implementación de sistemas difusos sobre FPGAs basada en el uso de la herramienta de diseño de DSP de Xilinx. Al estar integrado en el entorno Matlab, el flujo de diseño propuesto presenta ventajas significativas en cuanto a flexibilidad y configurabilidad, y permite el uso de las distintas facilidades de cálculo y representación gráfica que proporciona dicho entorno. La estrategia propuesta es la base de una nueva herramienta de síntesis para la próxima versión del entorno *Xfuzzy*,

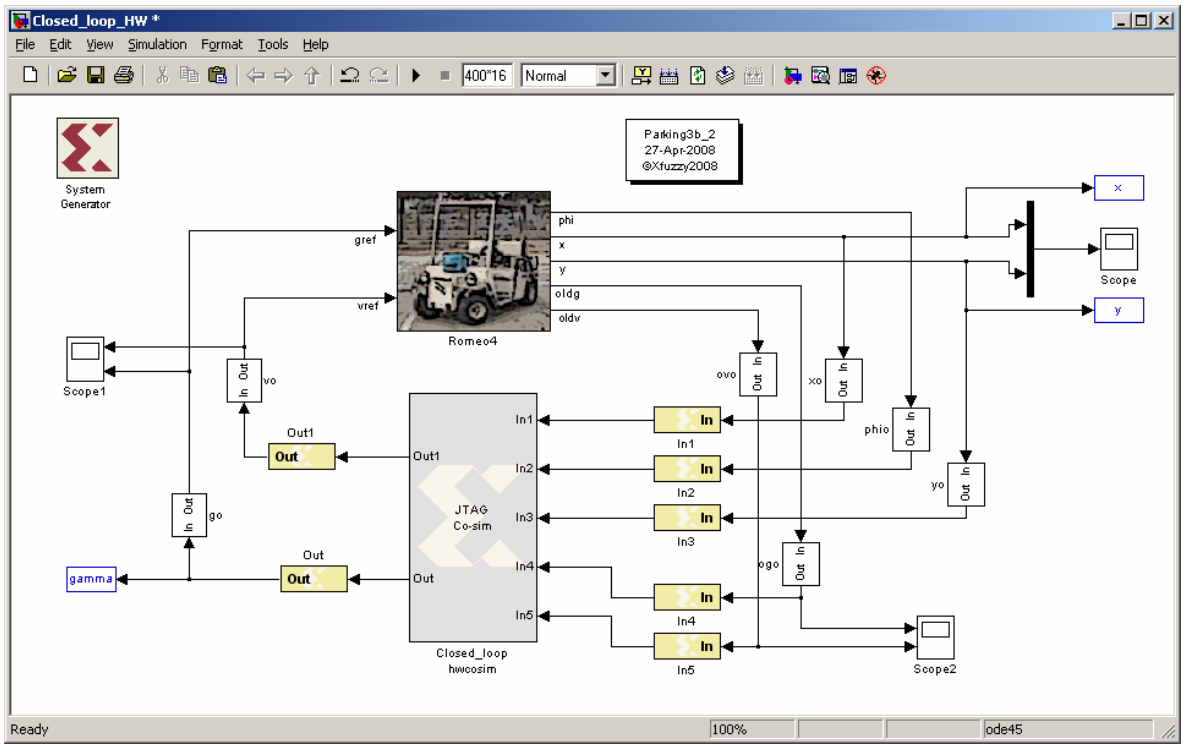

Figura 10. Simulación en lazo cerrado del sistema de control junto al modelo del vehículo

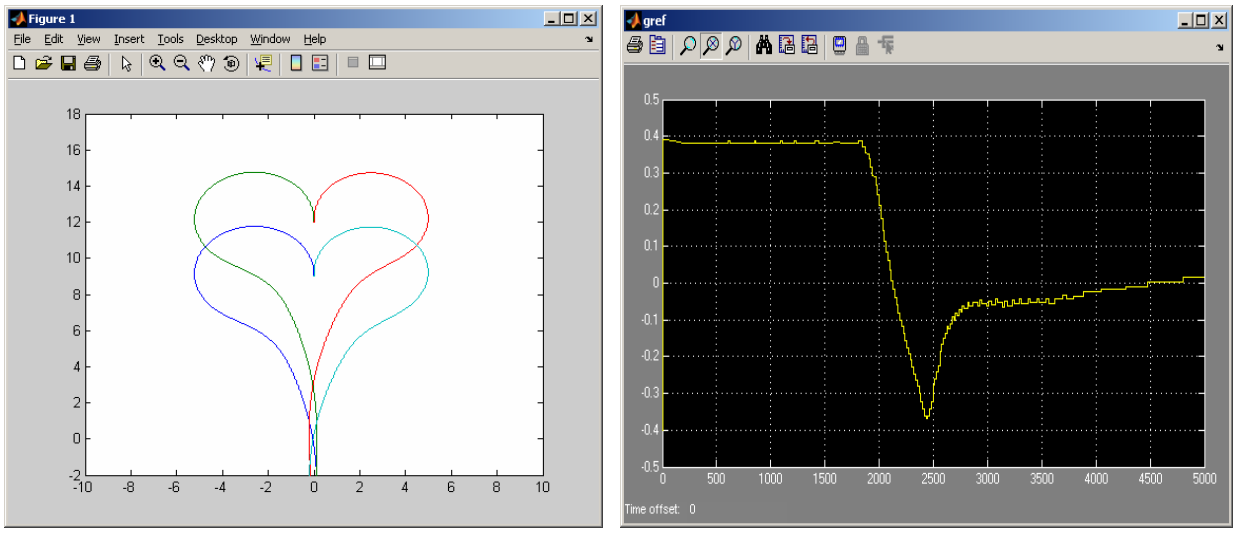

Figura 11. Resultados experimentales mostrando algunas trayectorias obtenidas y la evolución temporal de la curvatura del vehículo

aportando como principales novedades la inclusión de nuevos operadores y métodos de defuzzificación, una funcionalidad mejorada de muchos de los elementos básicos de la arquitectura y la implementación directa de sistemas difusos con estructuras jerárquicas. Debido a la creciente importancia de los dispositivos lógicos programables, la técnica de diseño propuesta representa una herramienta muy interesante para el desarrollo de aplicaciones prácticas en los cursos relacionados con el diseño digital que se imparten en diversas enseñanzas técnicas. Su empleo, junto con la disponibilidad de placas de desarrollo obtenidas a través de los programas universitarios de los principales fabricantes de dispositivos programables, permitiría que los alumnos pudieran recorrer de una forma rápida y sencilla las diferentes etapas de desarrollo de los sistemas, reforzando sus conocimientos en disciplinas relacionadas con diseño de sistemas digitales, síntesis hardware, lógica difusa y aplicaciones de control.

Referencias

[1] Xilinx University Program: http://www.xilinx.com/univ/index.htm

[2] Altera University Program: http://www.altera.com/education/univ/unv-index.html

[3] Celoxica University Program: http://www.celoxica.com/partner/university/default.asp

[4] Xilinx. "Xilinx System Generator v8.2 for Simulink". User Guide. Xilinx Blockset Reference Guide

[5] A. Cabrera, S. Sánchez-Solano, C. J. Jiménez, A. Barriga, I. Baturone, "Arquitectura eficiente para la implementación hardware de sistemas de inferencia difusos", Ingeniería Electrónica, Automática y Comunicaciones, Vol. XXIII, No. 1, pp. 59-66, 2003

[6] T.-H. S. Li, C. Shih-Jie, C. Yi-Xiang, "Implementation of human-like driving skills by autonomous fuzzy behavior control on an FPGA-based car-like mobile robot," IEEE Trans. Ind. Electron., vol. 50, no. 5, pp. 867–880, Oct. 2003

[7] S. Sánchez-Solano, A. Cabrera, I. Baturone, F. J. Moreno-Velo, M. Brox, "FPGA Implementation of Embedded Fuzzy Controllers for Robotic Applications". IEEE Trans. on Industrial Electronics, Vol. 54, No. 4, pp. 1937-1945, Aug. 2007

[8] J. Ferruz, V. Blanco, A. Ollero, and J. V. Acevedo. "*An embedded DSP-based controller for the Romeo-4R vehicle*". *Proc. IFAC Int. Symp. Intell. Compon. and Instrum. Control Appl.*, pp. 101–106, Jul. 2003

[9] Página web de Xfuzzy: http://www.imse.cnm.es/Xfuzzy/, Instituto de Microelectrónica de Sevilla