CODISEÑO HARDWARE/SOFTWARE DE CONTROLADORES DIFUSOS MEDIANTE MÓDULOS DE PROPIEDAD INTELECTUAL

A. Cabrera[1], S. Sánchez-Solano[2], P. Brox[2], F. J. Moreno-Velo[2], A. Barriga[2], I. Baturone[2]

[1] Dpto. Automática y Computación, Facultad de Ingeniería Eléctrica, ISPJAE, Ciudad de la Habana, Cuba. <alex@electrica.cujae.edu.cu>
[2] Instituto de Microelectrónica de Sevilla, Centro Nacional de Microelectrónica, Avda. Reina Mercedes s/n, 41012-Sevilla, España. <santiago@imse.cnm.es>

RESUMEN

El uso de técnicas de diseño basadas en módulos de propiedad intelectual (IP) constituye una alternativa válida para salvar la creciente distancia entre los recursos proporcionados por las actuales tecnologías de fabricación de circuitos integrados y la productividad alcanzada por los diseñadores de sistemas. Esta comunicación describe el desarrollo de un sistema de control basado en lógica difusa mediante una técnica de codiseño hardware/software que combina un procesador de propósito general disponible como módulo-IP y hardware específico para la síntesis del módulo de inferencia. La implementación física se ha llevado a cabo mediante una plataforma de desarrollo basada en FPGAs, lo que permite la realización de todo el sistema como un SoPC (*System on Programmable Chip*).

1. INTRODUCCIÓN

Los continuos avances en las tecnologías de fabricación de circuitos integrados hacen hoy día posible la inclusión de todos los componentes de un sistema microelectrónico en un único chip de silicio, dando lugar a lo que se denomina comúnmente como *System on Chip* (SoC). Sin embargo, como consecuencia de la elevada complejidad de los sistemas actuales, la diferencia entre los recursos proporcionados por las tecnologías y la productividad alcanzada por los diseñadores se hace cada vez mayor. Para reducir este "gap de productividad" se han propuesto nuevas técnicas de diseño que explotan la reutilización de bloques básicos previamente diseñados, denominados módulos de propiedad intelectual o módulos-IP [1] [2]. El diseño basado en módulos-IP, en combinación con el uso de potentes herramientas de CAD que facilitan la realización de los sistemas, permite asimismo reducir la duración del ciclo de desarrollo de nuevos productos y acelerar su introducción en el mercado, lo que representa ventajas importantes desde el punto de vista económico.

Por otra parte, la disponibilidad de dispositivos lógicos programables como las FPGAs que incorporan una elevada cantidad de recursos genéricos (en forma de bloques lógicos configurables) y específicos (generadores de reloj, multiplicadores, memorias, etc.) permite la implementación del sistema total sobre dichos dispositivos, tanto para facilitar las etapas de prototipado previas a la fabricación de los circuitos integrados, como para obtener una implementación final del sistema como un *System on Programmable Chip* (SoPC) [3].

En esta comunicación se describe la implementación sobre FPGAs de un sistema de control basado en lógica difusa para aplicaciones de navegación de robots móviles. El desarrollo del sistema se ha llevado a cabo mediante una técnica de codiseño

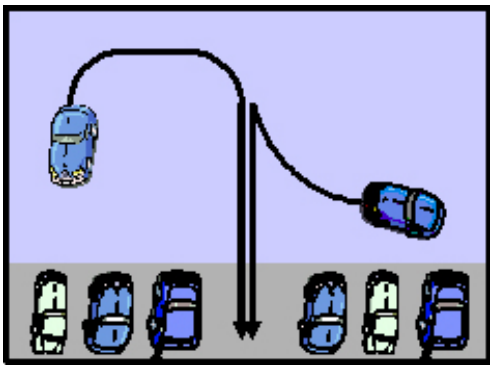
Trabajo parcialmente financiado por los proyectos TIC2001-1726 (CICYT) y DGAECE-58/02 (Junta de Andalucía).

hardware/software fuertemente apoyada en el uso de módulos-IP. El controlador incluye un sistema de procesado de propósito general basado en el procesador MicroBlaze de Xilinx y hardware específico para la realización del módulo de inferencia difuso que implementa la heurística de control. Las herramientas de diseño de FPGAs, el entorno de desarrollo de MicroBlaze y la placa de prototipado Digilab2E (de Digilent Inc) utilizada para dar soporte físico al controlador fueron obtenidos a través del *Programa Universitario de Xilinx* [4]. Las tareas de descripción, modelado y síntesis del módulo de inferencia se realizaron con ayuda del entorno de desarrollo de sistemas difusos *Xfuzzy* de libre distribución [5].

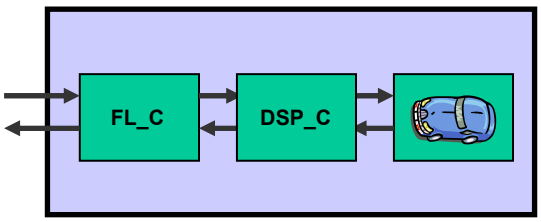
La Sección 2 del artículo introduce el problema planteado como ejemplo de diseño y describe las principales características del vehículo utilizado en la verificación experimental, así como la estructura general del sistema de control. La técnica de codiseño utilizada, la división hardware/software de las tareas del controlador y los diferentes componentes del sistema son descritos en la Sección 3. En la Sección 4 se comenta cómo ha sido necesario combinar las distintas herramientas de desarrollo para obtener un flujo de diseño unificado. Los detalles de implementación del módulo de inferencia difuso y la estrategia de control se muestran en la sección 5. La Sección 6 ilustra algunos de los resultados experimentales obtenidos en este trabajo. Finalmente, la Sección 7 recoge las principales conclusiones del mismo.

2. CONTROL DE NAVEGACIÓN DE UN VEHÍCULO AUTÓNOMO

La aplicación elegida para validar la técnica de codiseño propuesta en este trabajo se centra en un problema típico de control de navegación en robótica. En particular, el control de la trayectoria que debe seguir un vehículo para aparcar en batería cuando parte de una situación cercana a la posición objetivo (Figura 1a) es una variante del problema clásico de aparcamiento marcha atrás que complica sensiblemente el sistema de control a utilizar, ya que



(a)



(b)

(c)

Figura 1. Sistema de control de aparcamiento de un vehículo autónomo: a) Descripción del problema. b) Estructura jerárquica de control. c) Vehículo autónomo Romeo4R.

en este caso se deben incorporar estrategias para planificar el sentido de marcha [6]. La verificación experimental se ha llevado a cabo sobre el vehículo Romeo4R (Figura 1c), diseñado y construido en la Escuela Superior de Ingenieros de la Universidad de Sevilla, equipado con dos motores (de tracción y de dirección) y múltiples sensores que permiten determinar el estado del vehículo [7].

El motor de tracción, localizado en la parte posterior del vehículo, establece la velocidad de éste. El sentido de marcha (hacia delante o hacia atrás) se determina según la polaridad del voltaje aplicado a dicho motor. Dado que no existe un sistema de control electrónico para el frenado de Romeo4R, esta maniobra debe llevarse a cabo reduciendo a cero la velocidad del motor de tracción. Por la misma razón, el cambio de sentido debe realizarse disminuyendo gradualmente la velocidad del vehículo antes de aplicar un voltaje de polaridad opuesta. El motor de dirección establece el ángulo de giro de las ruedas delanteras, es decir, la curvatura que debe desarrollar el vehículo.

De los diferentes sensores disponibles en Romeo4R, los utilizados en esta aplicación han sido dos encoders asociados a cada uno de los motores y un giróscopo ubicado en el techo del vehículo. El encoder ubicado en el eje trasero permite determinar la velocidad real (v) del vehículo mientras que el encoder localizado en la parte delantera mide el giro realizado por las ruedas, a partir del cual se determina la curvatura (γ) del vehículo. Conocidas la velocidad y la curvatura se puede determinar la posición del vehículo, es decir, sus coordenadas (x,y). La orientación (ϕ) del mismo viene dada directamente por la salida del giróscopo.

Romeo4R dispone de un procesador digital de señal (DSP) TMS-320LF de Texas Instruments que realiza el control a bajo nivel de los motores, así como la adquisición de la información procedente de los sensores y su procesado para determinar la velocidad, curvatura, orientación y posición del vehículo. Por este motivo hemos optado por utilizar una estructura jerárquica de control (Figura 1b) donde el controlador de bajo nivel (DSP_C) se encarga de las tareas de adquisición y procesado de la información procedente de los sensores y del control directo de los motores, liberando al controlador difuso de alto nivel (FL_C) para que se dedique exclusivamente a la ejecución del algoritmo de control de navegación del vehículo. La comunicación entre ambos niveles se realiza mediante un protocolo de comunicación serie incorporado en el DSP y que debe también ser incorporado en el controlador de alto nivel.

3. CODISEÑO HARDWARE/SOFTWARE DEL CONTROLADOR DIFUSO

La capacidad de la lógica difusa para describir el comportamiento de sistemas complejos mediante reglas simples basadas en la experiencia de un operador y descritas en forma lingüística ha motivado su aplicación a la realización de numerosos problemas de control de sistemas, como el planteado en este trabajo, en los que puede resultar difícil disponer de un modelo analítico preciso [8] [9].

De forma paralela, se han desarrollado diferentes alternativas para la implementación de sistemas de control difusos que contemplan, desde la realización por software del algoritmo de control mediante la ejecución de un programa en un computador, hasta su implementación microelectrónica mediante circuitos integrados para aplicaciones específicas [10] [11]. Las soluciones software se caracterizan por un elevado grado de flexibilidad, si bien su tiempo de respuesta viene limitado por la inherente ejecución secuencial de los programas. Por el contrario, las realizaciones hardware proporcionan una elevada velocidad de operación, pero en ellas es necesario reducir al mínimo el área del circuito integrado resultante, de aquí que sea preciso recurrir en la mayoría de las ocasiones a estrategias de realización que limitan en gran medida la flexibilidad de los sistemas.

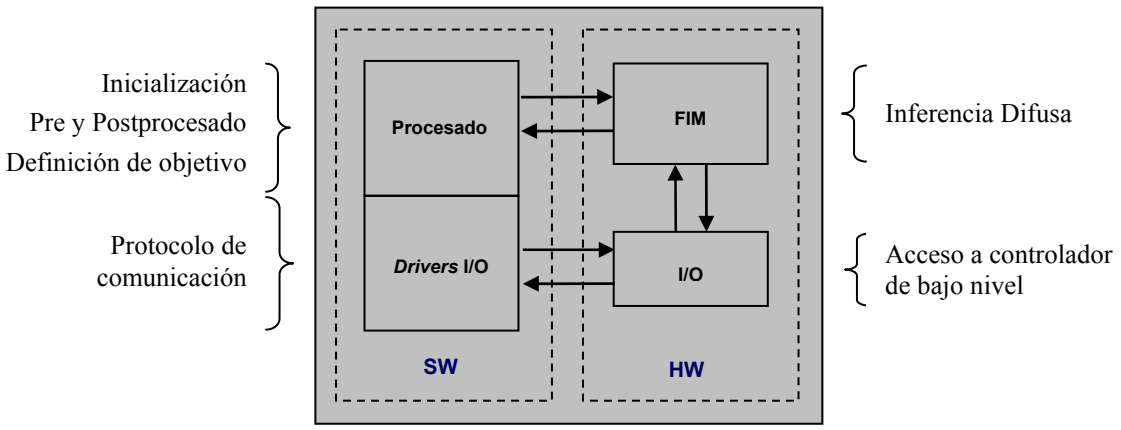


Figura 2. Codiseño hardware/software de un sistema de control basado en lógica difusa.

A medio camino entre las soluciones software y hardware, las técnicas de codiseño hardware/ software (HW/SW) tratan de obtener un compromiso adecuado entre las ventajas e inconvenientes de ambas alternativas [12]. La distribución entre el hardware y el software de las tareas que debe realizar un sistema de control difuso permite obtener una solución que sea a la vez rápida y flexible, donde las tareas de procesado convencional se realizarán por software mientras que aquellas que requieran recursos específicos serán implementadas mediante hardware dedicado [13].

La Figura 2 ilustra una estrategia de codiseño HW/SW para controladores difusos consistente en la partición "a priori" de las tareas del sistema. De acuerdo con dicha estrategia, un procesador de propósito general será el encargado de ejecutar tareas tales como la inicialización de los parámetros del sistema (base de tiempo, definición de objetivos, etc.), la adquisición y el preprocesado de las señales de entrada (linealización, filtrado, ajuste de la información, etc.), la comunicación con el módulo de inferencia (FIM) y con el controlador de bajo nivel, y la monitorización del comportamiento del controlador. Por otra parte, las tareas relacionadas con aplicación de técnicas neuro-difusas (cuya implementación en un procesador convencional resultaría poco eficiente), junto con los circuitos de interfaz del sistema, serán implementados mediante un soporte hardware específico para garantizar una alta velocidad de respuesta.

Adicionalmente, el grado de desarrollo alcanzado en las realizaciones de dispositivos programables y la disponibilidad de entornos de diseño que facilitan el rápido desarrollo de los distintos componentes del sistema hacen factible la idea de implementar todo el controlador difuso en una FPGA como un sistema empotrado sobre un circuito programable.

El elemento utilizado para soportar las tareas de software es el procesador MicroBlaze de Xilinx [14]. Dicho procesador está disponible como módulo IP conjuntamente con una gran diversidad de periféricos, controladores y estructuras de buses que permiten configurar el sistema según las necesidades de una determinada aplicación. MicroBlaze es un procesador RISC de 32 bits, con arquitectura Harvard que puede ser implementado sobre FPGAs de Xilinx del tipo SpartanII y superiores. Tanto el bus de instrucciones como el de datos presentan dos interfaces: una interfaz de bus de memoria local (LMB), optimizada para su utilización en FPGAs que contengan estructuras independientes de bloques de memoria RAM multipuerto; y una interfaz compatible con el bus estándar OPB (*On-chip Peripheral Bus*) de IBM que admite múltiples periféricos compatibles con este estándar y que pueden ser parametrizados y disponen de drivers de software que facilitan su utilización. La combinación de estas cuatro interfaces permite seis configuraciones de buses diferentes para el desarrollo de sistemas basados en MicroBlaze, que abarcan desde una estructura muy simple compuesta

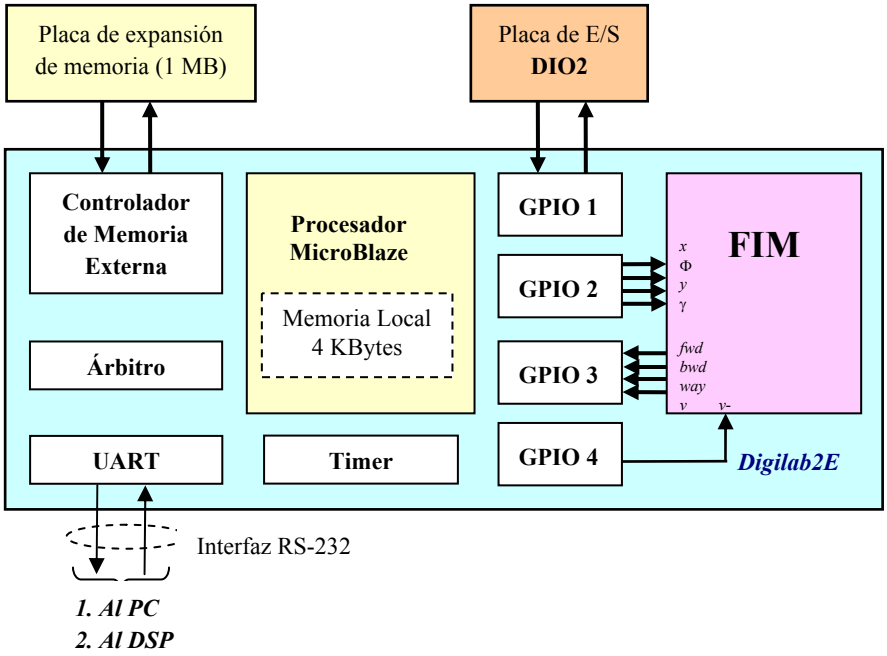


Figura 3. Diagrama de bloques del sistema de control basado en lógica difusa.

por una interfaz ILMB para el acceso a código en BRAM y una interfaz DOPB para el acceso a periféricos internos, hasta una estructura compuesta por las cuatro interfaces para el acceso a BRAM, periféricos internos y memoria externa de instrucciones y datos.

La Figura 3 muestra el diagrama de bloques del controlador difuso. Además del núcleo básico de MicroBlaze, configurado con buses para acceso a memoria externa de instrucciones y datos, el sistema de procesado incluye otros componentes. Para permitir acceso a memoria externa se han incorporado un controlador de memoria y el árbitro correspondiente. La interfaz con el módulo de inferencia difuso (FIM) se realiza a través de cuatro puertos de entrada/salida de 32 bits (GPIO 1-4). El bloque temporizador está formado por dos contadores descendentes de 32 bits que se utilizan para la generación de señales periódicas que controlan la operación del sistema. Por último, el UART se utiliza inicialmente para cargar el programa del controlador en la memoria externa y, posteriormente, como interfaz de comunicación.

La implementación hardware del módulo de inferencia difuso se ha realizado con ayuda de las herramientas de síntesis del entorno *Xfuzzy*, de acuerdo con la arquitectura y la metodología de diseño de circuitos difusos para aplicaciones específicas previamente reportada por los autores [15]. Las técnicas utilizadas para reducir el tamaño e incrementar la velocidad de los módulos de inferencia difusos son el procesado de reglas activas, la limitación del grado de solapamiento de las funciones de pertenencia y la utilización de métodos de defuzzificación simplificados. Una característica importante de dicha arquitectura es la existencia de diferentes opciones de implementación para los distintos bloques que la forman. El diseñador puede seleccionar distintas estrategias para la generación de funciones de pertenencia, diferentes operadores difusos para los conectivos de antecedentes de las reglas y distintos métodos de defuzzificación simplificados.

Siguiendo la estrategia de codiseño descrita previamente, el procesador MicroBlaze realiza las tareas de interfaz con el controlador de bajo nivel, secuencialización de las diferentes etapas del sistema e intercambio de información con el módulo de inferencia difusa. Este último implementa la estrategia de navegación del vehículo mediante un conjunto de reglas de actuación similares a las que emplearía cualquier conductor.

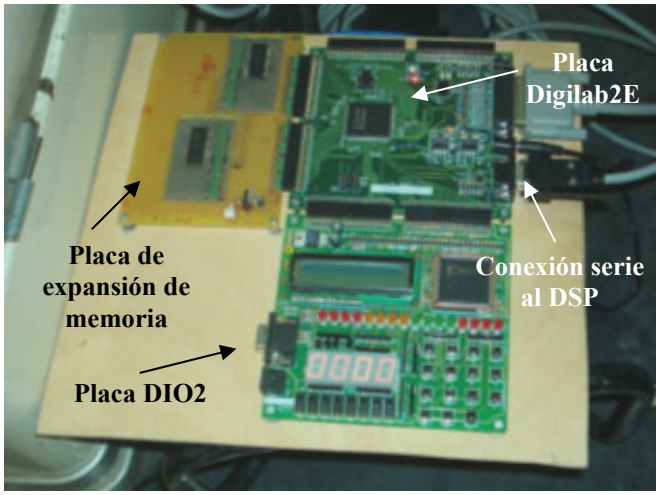


Figura 4. Prototipo experimental del sistema de control basado en lógica difusa.

Para dar soporte físico al controlador difuso se ha empleado una placa de desarrollo de FPGAs Digilab2E (de Digilent Inc). Esta placa dispone de una FPGA SpartanIIE de 200.000 puertas y 56 Kbits de memoria de tipo bloque [16]. También se dispone de una placa auxiliar de entrada/salida (DIO2, de Digilent Inc.) que puede utilizarse para introducir la condición de arranque del controlador y visualizar los resultados, así como de una placa de expansión de memoria externa (con dos SRAM AS7C34098 de 256K x 16 bits cada una) para aumentar el espacio de memoria disponible. La Figura 4 muestra una fotografía del conjunto formado por estas tres placas.

4. HERRAMIENTAS Y FLUJO DE DISEÑO

Una de las tareas que ha sido preciso realizar para llevar a la práctica el sistema de control propuesto ha consistido en combinar las herramientas de desarrollo de los componentes software y hardware del controlador con objeto de disponer de un flujo de diseño unificado y coherente.

MicroBlaze dispone de un entorno de desarrollo que facilita tanto la síntesis hardware del sistema como la compilación y depuración del software. La parte izquierda de la Figura 5 ilustra el flujo de diseño de un sistema basado en MicroBlaze. Los componentes del sistema se declaran en un fichero de especificaciones de hardware (.mhs), en donde se configura el procesador así como los diferentes periféricos y controladores utilizados (espacio de direcciones de E/S, buses de datos y direcciones, señales, etc.). Este fichero constituye la entrada a la herramienta *platgen* encargada de integrar todo el hardware del sistema MicroBlaze. Opcionalmente se generará una descripción VHDL que puede ser utilizada para crear un sistema jerárquicamente superior, como se requiere para la inclusión del módulo de inferencia. La implementación final del sistema completo se lleva a cabo utilizando las herramientas de desarrollo de FPGAs de Xilinx incluidas en el entorno ISE. De forma similar los drivers de los diferentes periféricos, el modo de operación del sistema y los enlaces a las rutinas de interrupción, se declaran en un fichero de especificaciones de software (.mss). Utilizando esta información, la herramienta *libgen* compila las funciones de los diferentes periféricos, obteniéndose las librerías y ficheros de cabecera que facilitarán la codificación de los programas en lenguaje C. Por último el compilador *mb-gcc* genera el código ejecutable correspondiente a dicho programa. El entorno se completa con la herramienta de depuración *xmd* que facilita la carga y verificación del programa.

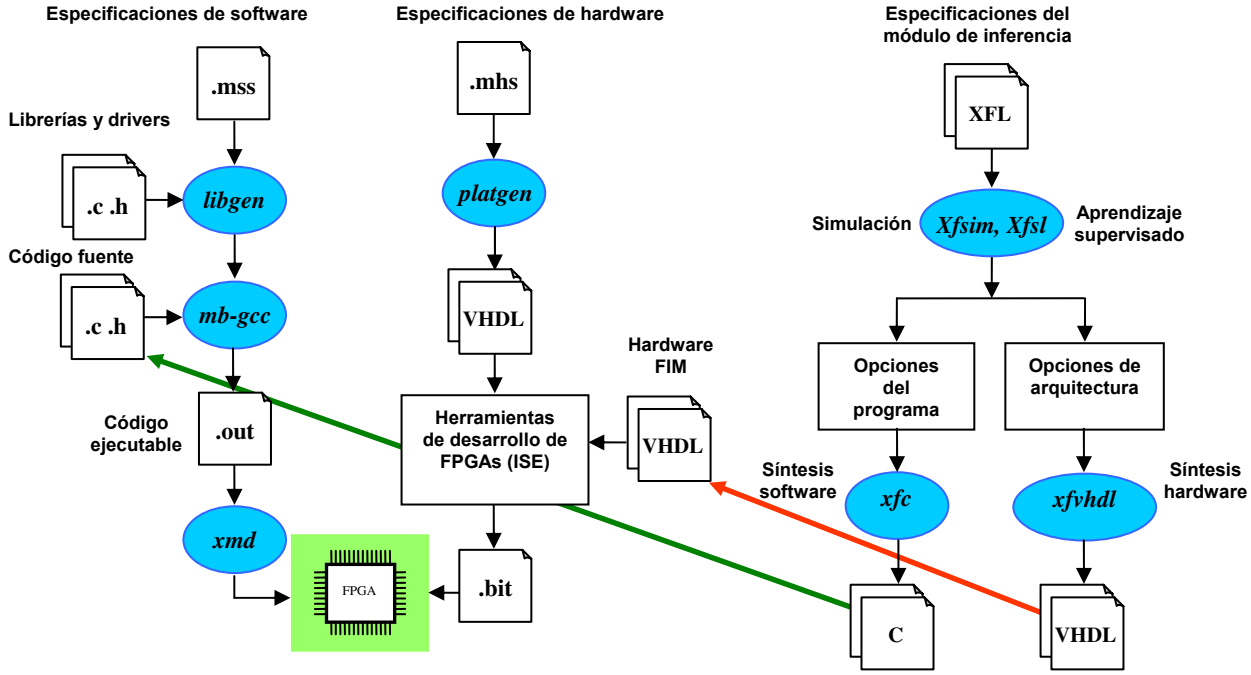


Figura 5. Combinación de los flujos de diseño de MicroBlaze y *Xfuzzy* para la implementación de sistemas de control difusos.

La parte derecha de la Figura 5 corresponde a las etapas de desarrollo de los módulos de inferencia difusos con ayuda del entorno *Xfuzzy*. Las distintas herramientas de dicho entorno permiten realizar la descripción del sistema difuso, simular su comportamiento, ajustar la base de conocimiento y realizar el proceso de síntesis del mismo. La herramienta de síntesis hardware del entorno (*xfvhdl*) genera una descripción VHDL del sistema difuso a partir de su especificación XFL. La herramienta permite dimensionar los bloques del sistema y seleccionar entre las distintas opciones de la arquitectura, generando código VHDL sintetizable compatible con las herramientas de síntesis de FPGAs. Al ejecutar el proceso de síntesis el usuario puede seleccionar diferentes opciones de implementación de la base de conocimiento [17].

Como *Xfuzzy* incorpora herramientas para la síntesis software del módulo de inferencia difuso, es posible también realizar una implementación totalmente software del controlador difuso utilizando para su ejecución el procesador empotrado en la FPGA. De esta forma, la misma plataforma realizada sobre la FPGA permite ser configurada para la realización de un controlador difuso software o de uno basado en codiseño hardware/software.

5. IMPLEMENTACIÓN DEL MÓDULO DE INFERENCIA

La utilización de un dispositivo lógico programable y la disponibilidad de un flujo de diseño que automatiza las tareas de síntesis nos han permitido explorar distintas soluciones para realizar el control del vehículo Romeo4R. La Figura 6a muestra una de estas alternativas, en la que el módulo de inferencia difuso presenta una estructura jerárquica compuesta por seis bases de conocimiento simples. Las bases de conocimiento *Position*, *Planning* y *Direction* corresponden a estructuras de toma de decisiones. *Position* decide 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 *Planning* para obtener la propuesta de sentido de circulación (plan). Sin embargo, dado que no se puede invertir bruscamente la

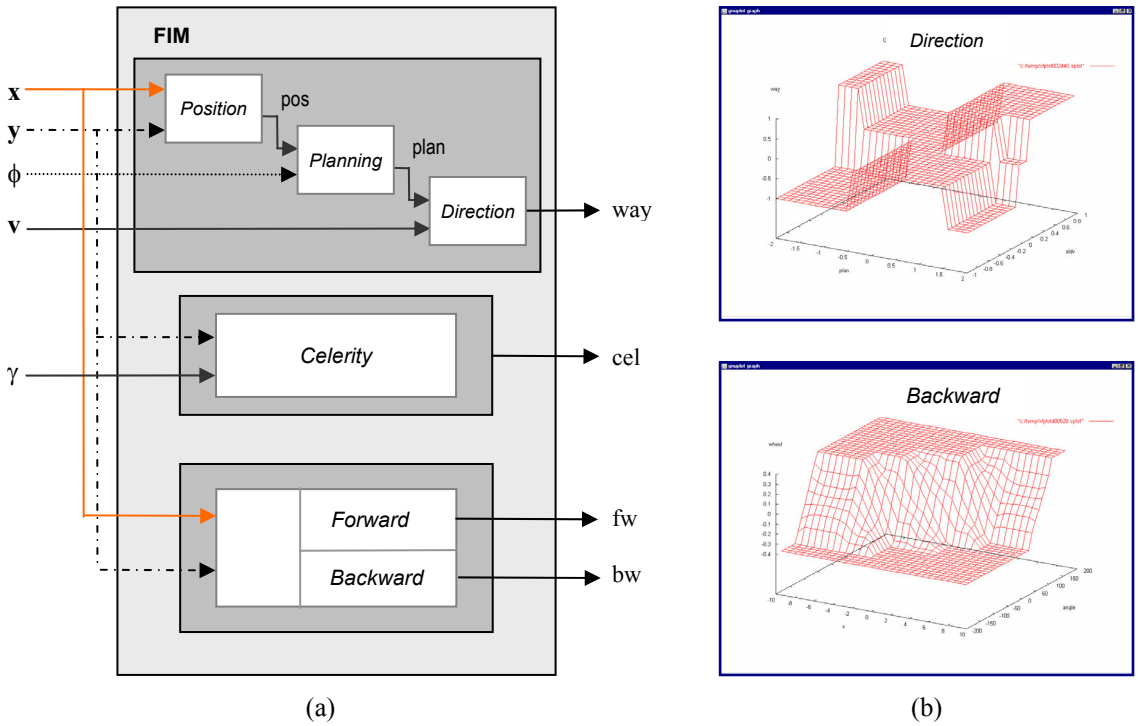


Figura 6. a) Estructura del módulo de inferencia difuso del sistema de control de navegación para el vehículo Romeo4R. b) Superficies de control de las bases de reglas *Direction* y *Backward.*

polaridad del voltaje aplicado al motor de tracción, esta salida se combina con el valor anterior de la velocidad (v-) en la base *Direction*, cuya salida way indica el sentido que definitivamente debe seguir el vehículo. En la implementación de estas bases de conocimiento se utilizan antecedentes basados en memoria y el método de defuzzyficación MaxLabel. En todos los casos se emplearon 5 bits para codificar las entradas y salidas y 3 bits para los grados de pertenencia.

Las restantes bases de conocimiento corresponden a sistemas con salidas interpoladas, utilizándose en todas ellas una resolución de 8 bits en entradas y salidas así como 6 bits para codificar los grados de pertenencia. En ellas se emplean técnicas aritméticas para generar los antecedentes y el método de defuzzificación FuzzyMean. La base de conocimiento *Celerity* determina el valor absoluto de la velocidad (cel), mientras que las bases *Backward* y *Forward* determinan la curvatura de Romeo4R en un sentido u otro (bw, fw). Dado que estas dos bases comparten las mismas entradas, es posible su unificación en un solo módulo. La Figura 6b muestra las superficies de control correspondientes a las bases *Direction* y *Backward.*

La salida final del controlador se obtiene mediante la combinación de la salida que decide el sentido de la circulación (way) con las correspondientes al valor absoluto de la velocidad (cel) y las de curvatura de Romeo4R en un sentido u otro (bw y fw) de acuerdo con las siguientes expresiones:

$$newv = way \cdot cel \tag{1}$$

$$new\gamma = \begin{cases} fw & if \quad v > 0 \\ bw & if \quad v < 0 \\ 0 & if \quad v = 0 \end{cases} \tag{2}$$

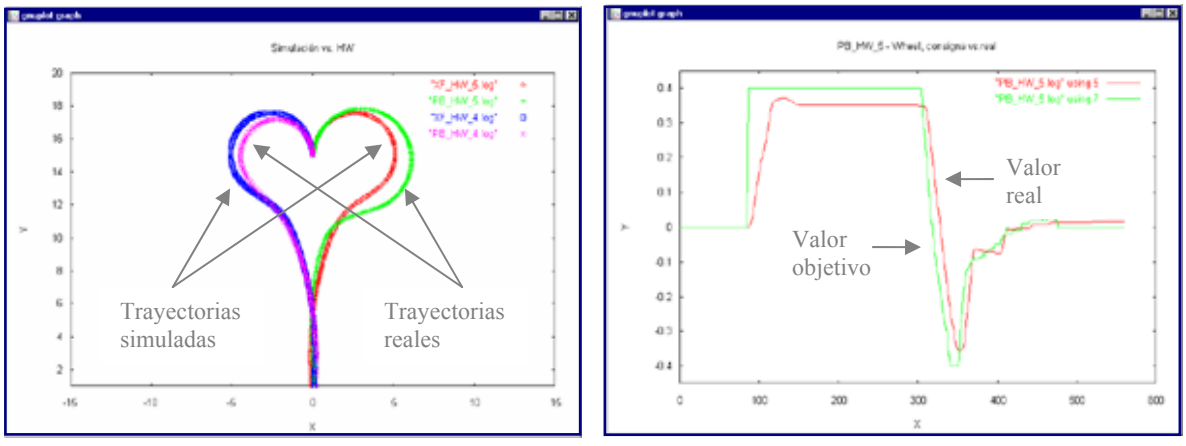
Siguiendo el flujo de diseño discutido en la Sección 4, las descripciones VHDL de las diferentes bases de conocimiento se combinan con la descripción del sistema MicroBlaze para proceder a la implementación conjunta de todos los componentes del sistema. Como resultado, se obtiene el fichero que contiene la configuración de la FPGA y puede ser descargado en la misma mediante la herramienta iMPACT. Los resultados de implementación de este controlador indican un consumo de recursos de 1.899 slices (80% de la Spartan-IIE200), con un periodo mínimo para la señal de reloj de 17,70 ns.

5. RESULTADOS EXPERIMENTALES

Con objeto de analizar el comportamiento del sistema de control desarrollado se realizaron distintas pruebas en las que se situó el vehículo en diferentes posiciones iniciales. En todas ellas se fue registrando la trayectoria del vehículo, almacenandose los datos correspondientes a las coordenadas y orientación, así como los valores de velocidad y curvatura establecidos por el módulo de inferencia y los valores reales medidos por Romeo4R. Estos ficheros fueron posteriormente procesados y analizados para evaluar los resultados. Entre otros aspectos, se compararon las trayectorias obtenidas experimentalmente con los resultados de la simulación realizada mediante las herramientas del entorno *Xfuzzy*.

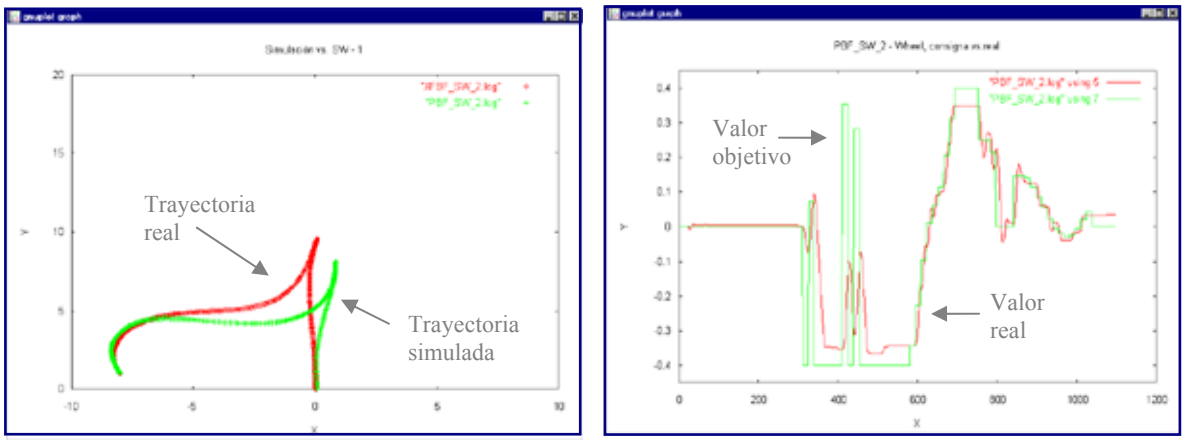
La Figura 7a muestra dos trayectorias reales de Romeo4R obtenidas mediante el sistema de control difuso y las trayectorias equivalentes obtenidas como resultado de la simulación. Ambos casos corresponden a situaciones iniciales en las que la maniobra de aparcamiento puede realizarse marcha atrás. Las diferencias entre las trayectorias simuladas y las reales se deben a las irregularidades existentes en el terreno, no contempladas en el modelo de la simulación. Nótese no obstante cómo el controlador difuso corrige la desviación y logra aparcar correctamente. En la Figura 7b se comparan las salidas del módulo de inferencia para el ángulo de giro de las ruedas con los valores reales que adopta el vehículo, comprobándose que Romeo4R es capaz de seguir la consigna aunque presenta una cierta inercia que impide variaciones bruscas de dirección.

El caso mostrado en la Figura 8 corresponde, sin embargo, a una situación en la que el vehículo debe dirigirse en primer lugar hacia delante, hasta llegar a una posición centrada con respecto a la posición final y, posteriormente, completa la maniobra marcha atrás. Las trayectorias real y simulada se muestran en la Figura 8a mientras que la Figura 8b ilustra la evolución del ángulo de giro de las ruedas.



(a) (b)

Figura 7. Resultados experimentales frente a simulación. a) Trayectorias; b) Control de curvatura.



(a) (b)

Figura 8. Resultados experimentales frente a simulación. a) Trayectorias; b) Control de curvatura.

6. CONCLUSIONES

El uso combinado de módulos de propiedad intelectual, herramientas de síntesis automáticas y dispositivos lógicos programables permite recorrer de forma rápida y fiable las etapas de descripción, síntesis e implementación de sistemas digitales de elevada complejidad.

Utilizando estas técnicas de diseño, en esta comunicación se ha descrito la realización de un sistema de control basado en lógica difusa para aplicaciones de guiado de un vehículo autónomo. El sistema incorpora un módulo de procesado basado en MicroBlaze y un modulo de inferencia difuso diseñado con ayuda de las herramientas de síntesis del entorno de desarrollo *Xfuzzy*.

Tanto el sistema de procesado basado en MicroBlaze como el módulo de inferencia pueden ser fácilmente configurados para obtener una estructura de controlador difuso acorde a las necesidades del problema planteado.

La implementación del sistema de control sobre una placa de desarrollo de FPGAs y su aplicación al control del vehículo Romeo4R permiten validar las técnicas utilizadas. Los resultados obtenidos ponen de manifiesto la similitud existente entre los resultados de simulación y los datos experimentales.

7. BIBLIOGRAFÍA

[1] Savage, W., Chilton, J., Camposano, R., "IP Reuse in the System on a Chip Era", *13th International Symposium on System Synthesis*, pp. 2-7, Madrid, Sep. 2000.

[2] Keating, M., "The reuse methodology manual: Toward a reuse discipline," *Design, Automation and Test in Europe (DATE),* pp. 141–145, Paris, Feb. 1998.

[3] Sancho-Pradel, D. L., Jones, S. R., Goodall, R. M., "System on Programmable Chip for Real-Time Control Implementations", *IEEE International Conference on Field-Programmable Technology*, pp. 276-283, Hong Kong, Dec. 2002.

[4] http://www.xilinx.com/univ/index.htm

[5] López, D.R., Jiménez, C.J., Baturone, I., Barriga, A. Sánchez-Solano, S., "Xfuzzy: A Design Environment for Fuzzy Systems", *IEEE International Conference on Fuzzy Systems*, pp. 1060-1065, Anchorage, May 1998.

[6] Baturone, I., Moreno-Velo, F. J., Sánchez-Solano, S., Martín de Agar, R., Ollero, A., "Automatic Design of Fuzzy Control Systems for Autonomous Mobile Robots", *28th Annual Conference of the IEEE Industrial Electronics Society*, pp. 2457-2462, Sevilla, Nov. 2002.

[7] Ollero, A., Arrue, B., et al, "Control and perception components for autonomous vehicles guidance", *Control Engineering Practice*, vol. 7, n. 10, p. 1291-1299, Oct. 1999

[8] Passino, K. M., Yurkovich, S., *Fuzzy Control*, Addison-Wesley, 1998.

[9] Yen, J., Langari, R., Zadeh, L. A., Eds., *Industrial Applications of Fuzzy Logic and Intelligent Systems*, IEEE Press, 1995.

[10] C. Von Altrock, "Adapting existing Hardware for Fuzzy Computation", in *Handbook of Fuzzy Computation*, Institute of Physics Publishing, 1998.

[11] Baturone, I., Barriga, A., Sánchez-Solano, S., Jiménez, C.J., López, D., *Microelectronic Design of Fuzzy Logic-Based Systems*, CRC Press, 2000.

[12] Reyneri, L., "Implementation issues of neuro-fuzzy hardware: going toward HW/SW codesign", *IEEE-Transactions-on-Neural-Networks*, vol. 14, n.1, pp. 176-194, Jan. 2003.

[13] Cabrera, A., Sánchez-Solano, S., Senhadji, R., Barriga, A., Jiménez, C.J. "Hardware/software codesign methodology for fuzzy controllers implementation", *IEEE International Conference on Fuzzy Systems*, pp. 464-469, Honolulu, May 2002.

[14] MicroBlaze Reference Guides, Xilinx, Inc.

[15] Sánchez-Solano, S., Barriga, A., Jiménez, C.J., Huertas, J.L, "Design and Applications of Digital Fuzzy Controllers", *IEEE International Conference on Fuzzy Systems*, pp. 869-874, Barcelona, Jul. 1997.

[16] Digilab 2E Reference Manual, Digilent Inc, 2002.

[17] Lago, E., Jiménez, C.J., López, D.R., Sánchez-Solano, S., Barriga, A., "Xfvhdl: A Tool for the Synthesis of Fuzzy Logic Controllers", *Design, Automation and Test in Europe (DATE),* pp. 102-107, Paris, Feb. 1998.