CONTROLADORES DIFUSOS ADAPTATIVOS COMO MÓDULOS DE PROPIEDAD INTELECTUAL PARA FPGAS[1]

Santiago Sánchez-Solano[1], Alejandro J. Cabrera[2], María Brox[1], Ariel J. González[2]

[1] Instituto de Microelectrónica de Sevilla (CNM-CSIC), Sevilla, España.
[2] Dpto. Automática y Computación (ISPJAE), La Habana, Cuba.

Santiago.Sanchez@imse.cnm.es

ABSTRACT

La continua demanda por parte del mercado microelectrónico de aplicaciones novedosas, con elevados niveles de complejidad y tiempos de desarrollo cortos ha motivado el impulso de las técnicas de diseño basadas en el concepto de "reusabilidad" y el desarrollo de elementos de sistemas como módulos de propiedad intelectual o módulos IP. En esta comunicación se describe la implementación de controladores difusos como módulos IP para FPGAs. Los controladores operan como periféricos conectables al bus OPB para los procesadores disponibles en las FPGAs de Xilinx. El empleo de las memorias internas de las FPGAs para almacenar las bases de conocimiento permite definir o ajustar la funcionalidad en tiempo de operación.

1. INTRODUCCIÓN

La capacidad de los sistemas difusos para describir la operación de un sistema complejo mediante reglas simples expresadas en lenguaje natural ha motivado su aplicación a numerosos problemas de automatización y control industrial, dando lugar a lo que ha venido en denominarse "control difuso" [1]-[2].

Las técnicas de implementación de controladores difusos han evolucionado de manera considerable en los últimos 20 años. Las primeras aplicaciones industriales de los controladores difusos se llevaron a cabo en la primera mitad de la década de los 80 mediante software sobre los procesadores disponibles en la época. Sin embargo, pronto se puso de manifiesto que esta solución era incapaz de satisfacer los requisitos de velocidad necesarios para resolver problemas de control en tiempo real y se propusieron distintas aproximaciones para superar esta limitación.

La primera de ellas consiste en el uso de entornos software específicos que facilitan el desarrollo de aplicaciones, adaptan los algoritmos de inferencia y generan código optimizado para diferentes familias de microcontroladores. Entre los productos y compañías más significativos cabe citar a Togai InfraLogic, que desarrolló y comercializó a principios de los 90 productos como MicroFPL, Till Shell, TILGen y FuzzyCLIPS. Aptronix Inc. introdujo en 1992 el sistema FIDE para generar código ensamblador para distintos microcontroladores de Motorola, Intel, Siemens y Omron. La empresa alemana Inform ha desarrollado una extensa gama de productos basados en el sistema FuzzyTECH para proporcionar tanto soluciones de propósito general, que generan código C estándar, como soluciones específicas, que generan código ensamblador para diversos microcontroladores, DSPs, coprocesadores difusos, procesadores de propósito general con soporte difuso y PLCs. Más recientemente, la compañía Rigel Corporation, dedicada a la fabricación de sistemas de control empotrados, distribuye el software rFLASH.

Sin embargo, y a pesar de optimizar el código, la ejecución secuencial de los programas impone serias limitaciones a la velocidad de operación de las soluciones basadas en microprocesadores convencionales. Las causas principales de estas limitaciones radican en el paralelismo inherente a los algoritmos de inferencia difusos y la utilización de operaciones multioperando como el mínimo y el máximo. Surge entonces la necesidad de emplear estructuras hardware dedicadas que aceleraran parcial o totalmente la ejecución de los algoritmos difusos. Tras los trabajos iniciales de Togai y Watanabe [3] y Yamakawa [4], se reportaron un elevado número de propuestas de circuitos difusos utilizando técnicas de diseño analógicas, digitales y mixtas [5]. Muchas de las ideas planteadas

[1] Este trabajo ha sido parcialmente financiado por el proyecto TEC2005-04359/MIC.

fueron recogidas por la industria microelectrónica para desarrollar diferentes tipos de "coprocesadores difusos".

El empleo de coprocesadores difusos no proporciona una solución global al sistema de procesado pero permite una cierta flexibilidad y configurabilidad, aunque suele limitar el número y tipo de funciones de pertenencia, los mecanismos de inferencia y los operadores empleados. La mayor parte de los "chips difusos" comercializados en la década de los 90 se encuadran en esta categoría. Entre aquellos que alcanzaron mayor popularidad podemos mencionar los chips FC110 y VY86C570 de Togai InfraLogic, el T/FC150 de Toshiba, las familias SAE 81C99 y 81C991 [6] de Siemens, los coprocesadores FP1000, FP3000, FP5000 [7] y el módulo de procesador difuso para PLCs FZ001 de Omron, y la arquitectura W.A.R.P. de ST Microelectronics [8].

El incremento en velocidad y funcionalidad de los microcontroladores estándares por una parte y, por otra, la inclusión de la lógica difusa entre las tecnologías maduras motivaron, sin embargo, la desaparición del mercado de este tipo de productos a finales de los 90, ya que el desarrollo de la microelectrónica hizo que el coprocesador difuso aislado no tuviera demasiado sentido, aunque sí se mantenía la necesidad de contar con elementos que aceleraran la ejecución de algoritmos de inferencia en el contexto de las plataformas de procesado convencionales.

Esta última aproximación para aumentar la velocidad de procesado difuso sobre plataformas de propósito general empezó a dar sus primeros pasos en paralelo con el desarrollo de coprocesadores difusos y tiene como objetivo incrementar la funcionalidad de un procesador estándar para incluir instrucciones que aceleren los mecanismos de inferencia. Las primeras propuestas para añadir soporte difuso en arquitecturas CISC y RISC se realizaron a principios de los 90 [9]-[10], y se materializaron a nivel industrial con dos claros exponentes que han llegado hasta nuestros días. En 1996 Motorola introdujo la familia de microcontroladores de 16 bits 68HC12, que incluye una serie de instrucciones específicas para el procesado de algoritmos difusos que proporcionan un incremento de velocidad de un orden de magnitud en relación a su predecesor [11]. Uno de los desarrollos más interesantes lo constituye la familia ST FIVE, de ST Microelectronics, que ofrece una arquitectura de microcontrolador tradicional combinada con una arquitectura dedicada para algoritmos difusos [12].

Como se ha puesto de manifiesto en esta breve revisión histórica, la industria microelectrónica ha proporcionado en los últimos 20 años diferentes tipos de soluciones para la ejecución de controladores difusos sobre plataformas estándares, lo que ha permitido introducir estas técnicas de inferencia como una herramienta habitual en muchas aplicaciones industriales y de electrónica de consumo. No obstante, la necesidad de abordar problemas más complejos o con mayores restricciones en cuanto a velocidad tamaño o consumo ha motivado el desarrollo paralelo de circuitos integrados de aplicaciones específicas o Fuzzy-ASICs. En los últimos años se han reportado en la literatura científica numerosas realizaciones de ASICs para controladores difusos que emplean diferentes técnicas de diseño [5], [13], [14] y se han desarrollado metodologías y herramientas de CAD que facilitan y aceleran su realización [15]-[17].

Por otra parte, el constante avance de las tecnologías de fabricación de circuitos integrados impone hoy día nuevos retos al mercado microelectrónico cuya connotación más significativa es la demanda de aplicaciones novedosas, con elevados niveles de complejidad y tiempos de desarrollo cortos. Los intentos por satisfacer simultáneamente ambos requisitos han llevado a los diseñadores de sistemas electrónicos a proponer en los últimos años una serie de nuevas técnicas y estrategias de diseño entre las que cabe destacar: la concepción de los sistemas desde la perspectiva de System_on_Chip (SoC), la combinación de elementos de procesado de propósito general con otros de carácter específico, el uso de técnicas híbridas basadas en codiseño hardware/software, el empleo de módulos de Propiedad Intelectual (IP) y la inclusión en el ciclo de desarrollo de etapas de prototipado rápido basadas en dispositivos lógicos programables como las FPGAs.

En esta comunicación se describe el desarrollo de módulos IP que aceleran la ejecución de mecanismos de inferencia basados en técnicas neuro-fuzzy en sistemas que emplean los módulos de procesado disponibles en las FPGAs de Xilinx. Dichos módulos pueden utilizarse como periféricos convencionales para los procesadores MicroBlaze o PowerPC y aprovechan la capacidad de memoria con que cuentan las actuales familias de FPGAs.

2. SISTEMAS DE PROCESADO PARA FPGAS

Los dispositivos lógicos programables han sufrido una transformación importante en los últimos años. Dicha transformación ha afectado, no solo al incremento del número de recursos de propósito general incluidos en los dispositivos, sino también a la disponibilidad de elementos específicos como bloques de memorias, multiplicadores, generadores de señales de reloj, lógica de anticipación de acarreo, etc., lo que permite implementar sistemas de elevada complejidad sobre una FPGA. Esta tendencia se ha visto reforzada por la existencia de numerosos bloques de sistema disponibles como módulos de propiedad intelectual o *soft-cores* (procesadores, periféricos I/O, controladores de memoria, etc) que facilitan el desarrollo de sistemas de procesado empotrados adaptados a un determinado dominio de aplicación. Como ejemplo podemos citar los módulos MicroBlaze de Xilinx o Nios de Altera, además de otras soluciones no dependientes de una determinada familia de FPGAs, como el procesador LEON [18] y numerosos *cores* de microcontroladores

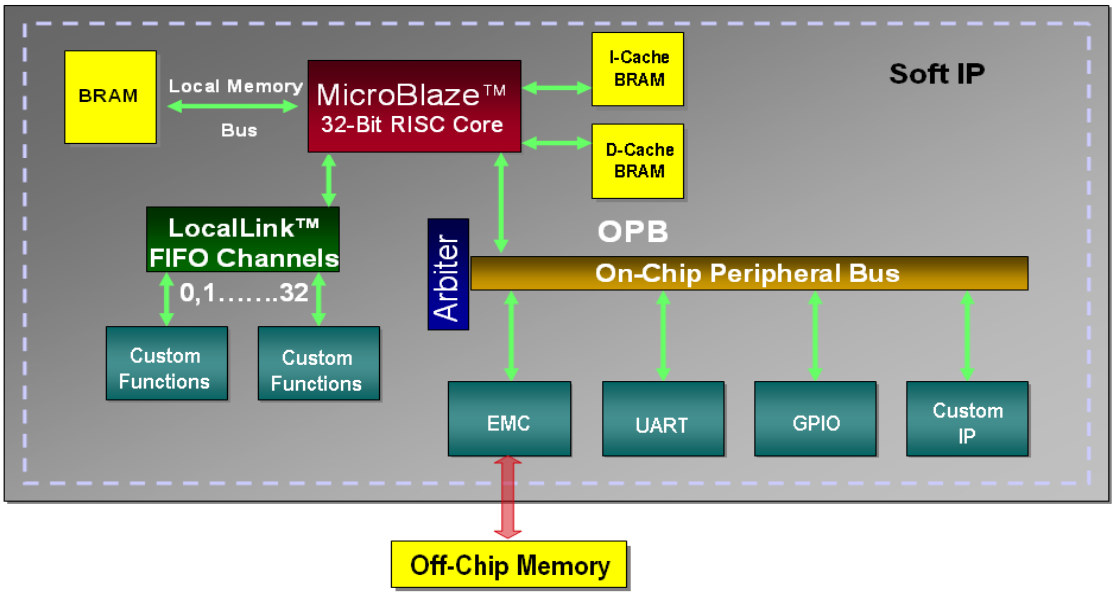

Fig 1. Diagrama de bloques de un sistema de procesado basado en MicroBlaze

comerciales. Adicionalmente, algunos fabricantes de FPGAs incorporan en sus dispositivos *hard-cores* de procesadores. Es el caso del procesador PowerPC incluido en las familias Virtex-II Pro y Virtex-4 de Xilinx y el ARM922T de la serie Excalibur de Altera.

MicroBlaze es un procesador RISC de 32 bits optimizado para implementación sobre FPGAs de Xilinx. Responde a una arquitectura Harvard, con buses separados para instrucciones y datos y posibilidad de utilizar cachés independientes para ambos buses. Su repertorio de instrucciones incluye instrucciones de 32 bits con tres operandos y dos modos de direccionamiento. En las familias Spartan 3, Virtex II y Virtex 4 las operaciones de multiplicación pueden realizarse por hardware mediante los multiplicadores disponibles en las FPGAs. El núcleo del procesador accede a las memorias de bloque de la FPGA a través del bus LMB (*Local Memory Bus*). El acceso a periféricos y memoria externa se realiza mediante el bus OPB (*On-chip Peripheral Bus*). Por último, dispone de diferentes canales FIFO para la conexión de funciones de usuario [19] (Fig. 1).

El bus OPB está basado en el estándar *CoreConect* de IBM. Se trata de un bus síncrono, con 32 bits de direcciones y 32 bits de datos, con un mecanismo de arbitración centralizado. El entorno de desarrollo EDK (*Embedded Development Kit*) de Xilinx incluye múltiples periféricos, compatibles con este estándar, que pueden ser parametrizados y disponen de drivers de software que facilitan su utilización. El entorno proporciona, asimismo, procedimientos que simplifican el desarrollo por parte del usuario de nuevos periféricos conectables al bus OPB.

3. MÓDULOS DE INFERENCIA DIFUSA

El entorno de diseño *Xfuzzy* facilita las diferentes etapas de desarrollo de un sistema difuso. Las herramientas de descripción y simulación permiten probar diferentes alternativas del sistema y comparar su comportamiento. Las herramientas de ajuste y simplificación ayudan a optimizar las bases de conocimiento que definen la operación del sistema. Por último, las herramientas de síntesis proporcionan implementaciones hardware o software del sistema de inferencia [20].

Para la implementación hardware de los módulos de inferencia, *Xfuzzy* utiliza una arquitectura específica basada en el procesado de reglas activas, la limitación del grado de solapamiento de las funciones de pertenencia de las entradas y la utilización de métodos de defuzzificación simplificados (Fig. 2), aspectos que contribuyen a incrementar la eficiencia de la realización digital [21].

A lo largo del proceso de síntesis el diseñador puede elegir entre diferentes opciones arquitecturales y de implementación con objeto de ajustar el sistema a las características del problema que tenga planteado. A nivel arquitectural es posible seleccionar la técnica (aritmética o basada en memoria) empleada en los circuitos generadores de funciones de pertenencia (MFCs), el operador usado como conectivo de antecedentes (mínimo o producto) y el método de defuzzificación utilizado. En cuanto a la implementación sobre la FPGA, además del dispositivo concreto a utilizar, puede optarse por la implementación de los distintos componentes de las bases de conocimiento como lógica combinacional o mediante almacenamiento en

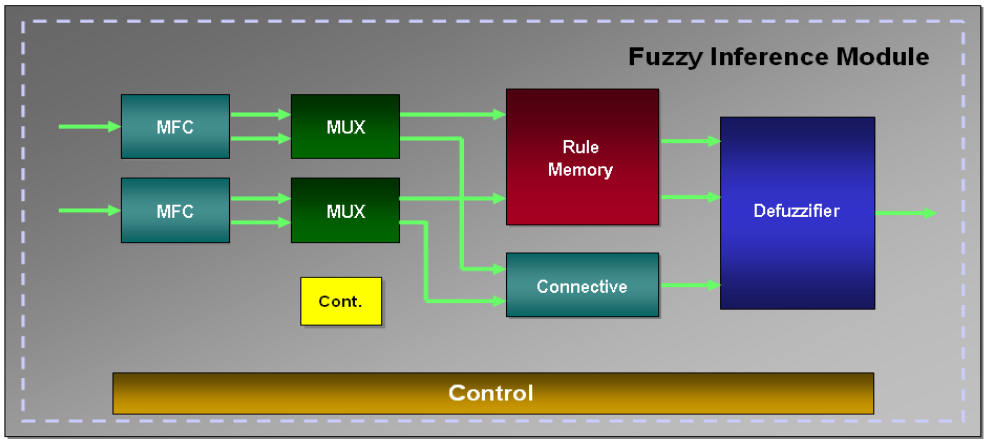

Fig 2. Diagrama de bloques de un módulo de inferencia difusa implementado mediante Xfuzzy

memorias ROM o RAM (en este último caso es posible elegir también entre la utilización de RAM de tipo bloque o asociada a los CLBs de la FPGA) [22].

4. MÓDULO DIFUSO COMO PERIFÉRICO OPB

El entorno de desarrollo de Xilinx proporciona una serie de plantillas que facilitan la conexión de periféricos de usuario a los buses OPB (MicroBlaze y PowerPC) y PLB (PowerPC). Estas plantillas consisten en código VHDL que incluye los dos componentes que se muestran en la Figura 3: IPIF (*Intellectual-property interface*), encargado de realizar las funciones de interfaz con el bus OPB o PLB; y User_logic, que contiene la lógica desarrollada por el usuario. Estos dos componentes se comunican a través del IPIC (*Intellectual-property interconnect*), una interfaz independiente del bus del periférico y más sencilla de manejar que las correspondientes a los buses OPB o PLB. Existen diferentes tipos de plantillas dependiendo del modo de operación (master/slave) del periférico y de los servicios proporcionados por el bloque IPIF.

La interfaz gráfica de usuario XPS (*Xilinx Platform Studio*) incluye herramientas que facilitan la generación de estas plantillas, así como de los ficheros ".mdp" (*micro-processor peripheral definition*) y ".pao" (*peripheral analyze. order*) necesarios para que el módulo IP pueda ser utilizado dentro de XPS como cualquier otro periférico del sistema.

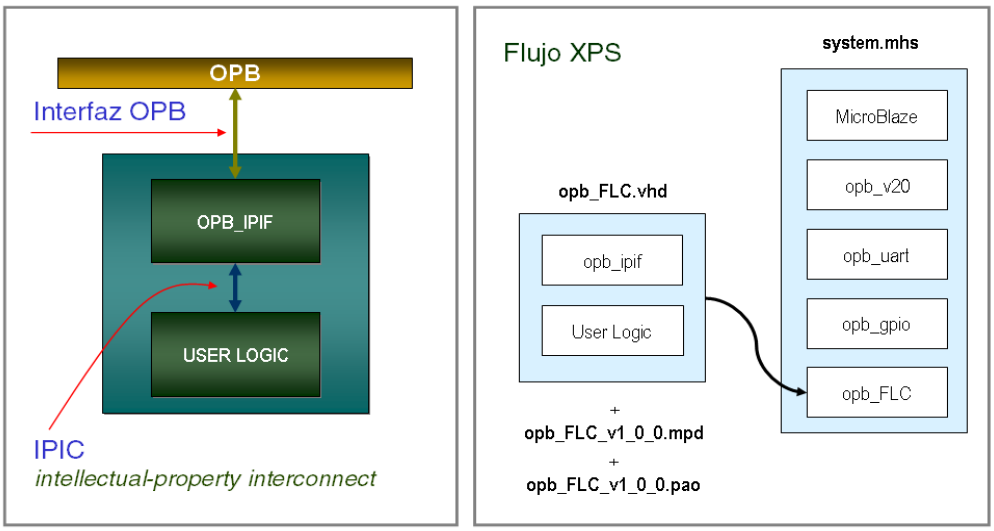

Fig 3. Conexión al bus OPB de periféricos de usuario

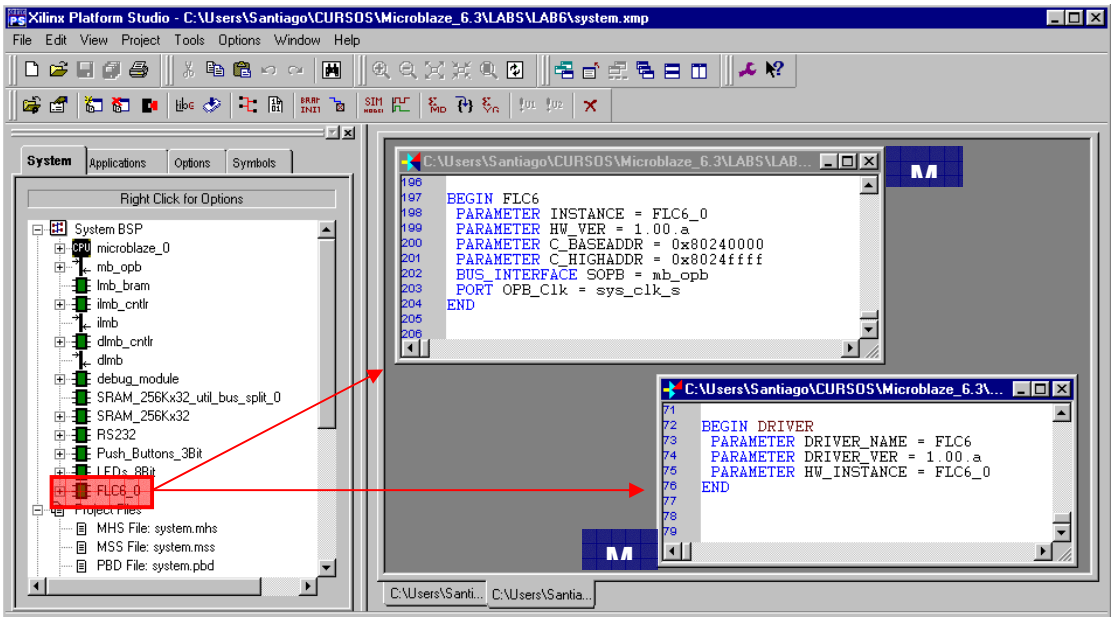

Fig 4. Uso del módulo IP del controlador difuso como un periférico de MicroBlaze

Combinando las facilidades proporcionadas por las herramientas de *Xfuzzy* y EDK, el ciclo de diseño de un controlador difuso como módulo IP requiere básicamente recorrer los siguientes tres pasos:

1.- Crear plantillas. Mediante el asistente para creación de periféricos de XPS se indica el nombre y versión del periférico, se selecciona el tipo de bus al que irá conectado y se configuran los diferentes servicios implementados por el bloque IPIF (operación como maestro o esclavo, soporte de interrupciones, número de registros, acceso directo a memoria, etc.).

2.- Integrar lógica de usuario. La descripción VHDL del controlador difuso proporcionada por *Xfuzzy* constituye el núcleo fundamental del bloque "User_logic" que incluirá, además, el código necesario para acceder al controlador a través de los registros del módulo IP.

3.- Importar periférico. El asistente para importar periféricos en XPS combina las descripciones correspondientes al controlador difuso, las plantillas VHDL y los elementos de librerías, y genera los ficheros necesarios para que el nuevo módulo pueda ser utilizado como periférico.

Los módulos IP descritos en los ejemplos incluidos en la siguiente sección de este artículo emplean 4 registros para comunicar con los controladores difusos. Tres de estos registros están conectados a las entradas y salida del controlador. El cuarto registro se utiliza para programar la base de reglas en los casos en que se usa memoria RAM para implementar este componente del controlador difuso.

La Figura 4 muestra la utilización en XPS del módulo FLC6. En la parte izquierda de la figura se observa cómo el módulo del controlador difuso tiene un tratamiento similar al de los restantes módulos que conforman el sistema de procesado. En las dos ventanas de texto que aparecen en la parte derecha de la figura se ilustra la inclusión del módulo FLC6 en los ficheros de definición del sistema: ".mhs" (*Microprocessor Hardware Specification*) y ".mss" (*Microprocessor Software Specification*). El módulo se conecta como un periférico esclavo del bus OPB cuyos registros serán vistos por el procesador a partir de la dirección 0x80240000.

5. APLICACIONES DE ROBÓTICA MÓVIL

La técnica de realización descrita en las secciones anteriores está siendo aplicada en el desarrollo de sistemas de control para aplicaciones de guiado de robots móviles autónomos. En concreto, nos centraremos en un caso de aparcamiento evitando obstáculos fijos mediante una estrategia de navegación reactiva basada en lógica difusa que combina conocimiento heurístico con el análisis del problema desde el punto de vista geométrico. El vehículo utilizado dispone de una serie de sensores (encoders de tracción y dirección, giróscopo y láser de barrido) que permiten calcular su posición y orientación con respecto a una referencia fija, así como los ángulos comprendidos entre los extremos del obstáculo y la perpendicular al eje del vehículo.

El sistema global será implementado sobre una FPGA de acuerdo con la estrategia de implementación híbrida

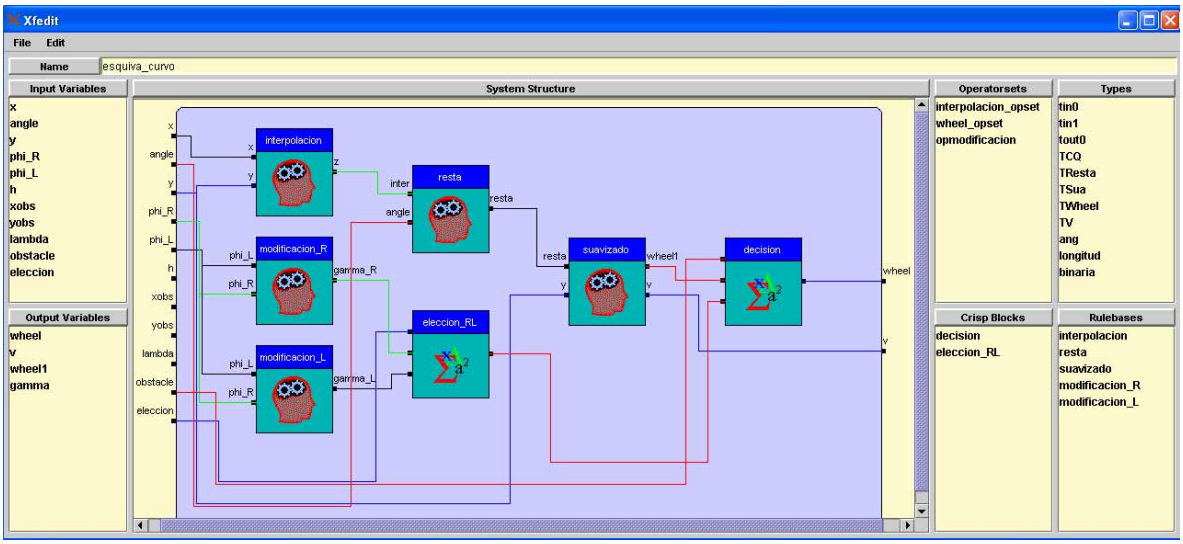

Fig 5. Descripción en *Xfuzzy* de un sistema de control difuso para la navegación de robots móviles autónomos

Hardware-Software descrita en [23]. Un sistema de procesado de propósito general basado en MicroBlaze se encargará de recibir el estado del vehículo, adaptar las señales de entrada y salida del controlador difuso y enviar los comandos de control a los actuadores de los motores de dirección y tracción. Los mecanismos de inferencia que llevan a cabo la heurística de control serán implementados mediante hardware dedicado de acuerdo con la arquitectura previamente comentada. La Figura 5 muestra la estructura del controlador difuso desarrollado con ayuda de las herramientas del entorno *Xfuzzy*. Puede observarse que se trata de un sistema jerárquico que combina módulos difusos, empleando diferentes bases de reglas y conjuntos de operadores, con módulos "crisp" que realizan funciones matemáticas convencionales.

La implementación de las bases de reglas de los módulos difusos del controlador mediante los bloques de memoria RAM disponibles en la FPGA permite dotar al sistema de características de programabilidad que son especialmente interesantes en la etapa de prototipado del sistema (para poder modificar el comportamiento de los módulos sin necesidad de re-implementar el diseño) y en la etapa de explotación (para facilitar su adaptación a diferentes condiciones de operación).

En la Figura 6 se ilustra el esquema de conexionado de la memoria de doble puerto que mantiene la base de reglas de un modulo de inferencia. El puerto superior de la memoria se utiliza en la fase de configuración del sistema para programar su comportamiento. El bus de direcciones, el bus de datos de entrada y la señal de escritura van conectadas a uno de los registros accesibles a través de la interfaz OPB del módulo IP. Una conexión similar del bus de datos de salida permite monitorizar, en cualquier fase de operación, el contenido de la memoria desde el programa ejecutado por el procesador MicroBlaze. El puerto inferior de la memoria es utilizado en la fase de operación normal del sistema de inferencia. El bus de direcciones proviene, en este caso, de los circuitos generadores de funciones de pertenencia. La información que transmite corresponde a la codificación de los etiquetas lingüísticas que intervienen en cada regla. El bus de datos de salida se conecta al bloque defuzzificador para proporcionar información sobre el consecuente de la regla.

La utilidad de la técnica de realización propuesta se pone de manifiesto en las dos situaciones relacionadas con la aplicación de navegación de robots móviles autónomos que se describen a continuación.

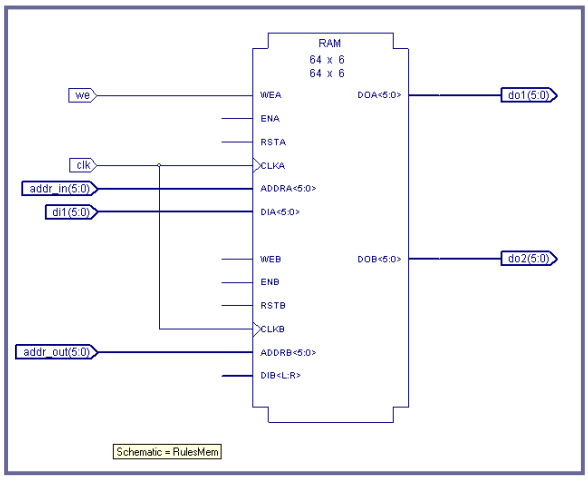

Fig 6. Utilización de la memoria de bloques de la FPGA en el sistema difuso adaptativo

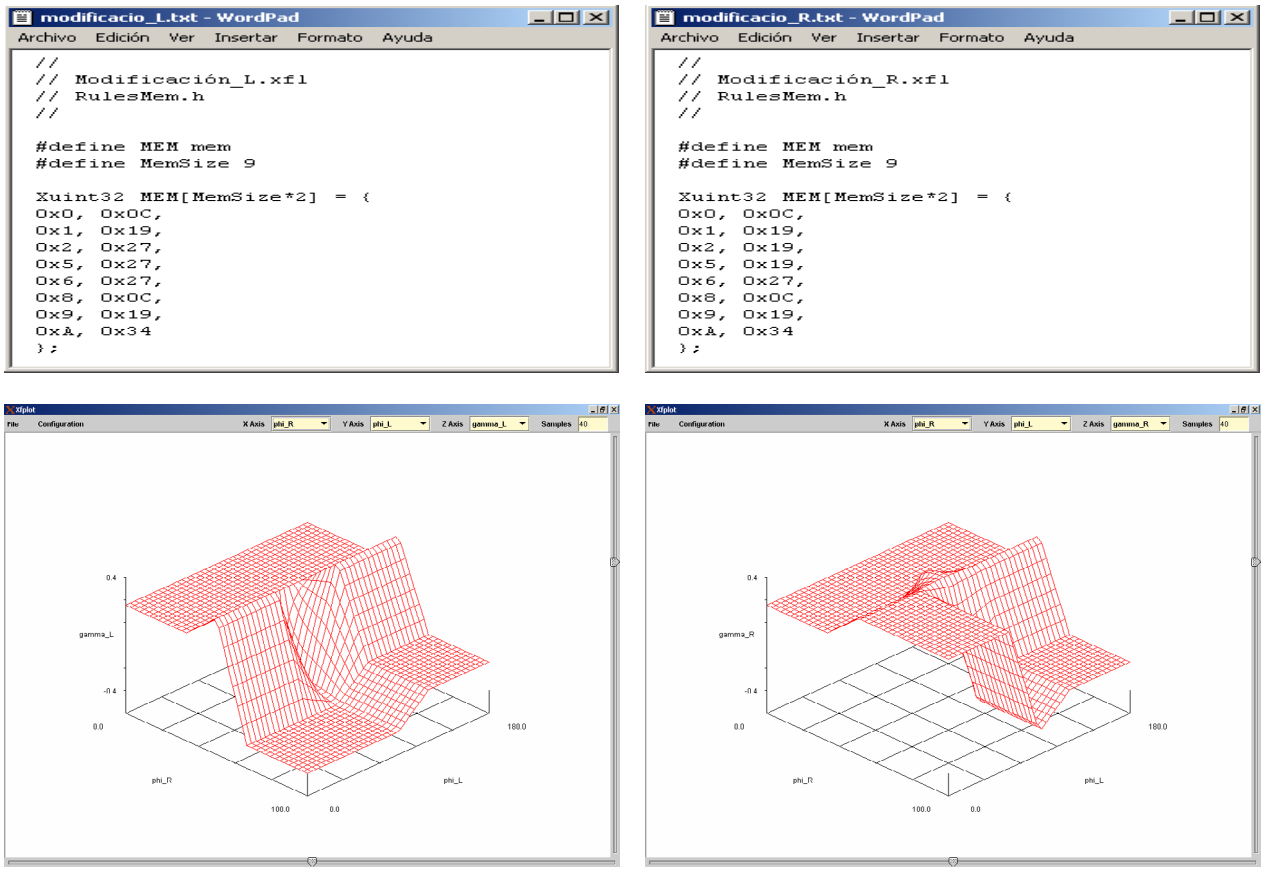

Fig 7. Configuración de diferentes bases de reglas sobre un mismo módulo IP con memoria RAM

5.1. Bases de reglas programables

Las bases de reglas correspondientes a los bloques denominados "modificación_L" y "modificación_R" en Fig. 5 se encargan de modificar la trayectoria del vehículo para los casos en que se detecta la presencia de obstáculos. Ambos bloques utilizan las mismas variables de entrada (los ángulos phi_R y phi_L) y proporcionan como salida la corrección que debe realizarse en la curvatura del vehículo. La elección entre la salida de ambos bloques se realiza en función del estado previo del robot y tiene por objeto minimizar la trayectoria que éste debe recorrer para alcanzar su objetivo.

La Figura 7 ilustra cómo puede programarse la memoria del módulo IP para que éste responda al comportamiento definido por las dos bases de reglas. Ya que los datos de programación de la memoria deben estar disponibles en tiempo de compilación, se ha optado por definir dichos datos en el fichero de cabecera *RulesMem.h* referenciado desde el programa principal de la aplicación. El formato del fichero incluye parejas "dirección-dato", ya que el número de reglas será, en general, menor que el tamaño de la memoria que las almacena.

5.2. Modificación *on-line* de la base de reglas

La utilización de técnicas de codiseño Hardware-Software para la implementación del sistema de control facilita la inclusión de nuevas y más complejas funcionalidades. Por ejemplo, la potencia de cálculo proporcionada por el procesador MicroBlaze permite programar un algoritmo de aprendizaje basado en *backpropagation* para identificar y ajustar los parámetros de la base de reglas a partir de un fichero de entrenamiento.

La Figura 8 muestra algunas facetas del proceso de aprendizaje para la base de reglas del módulo difuso que calcula la curvatura del vehículo en función de su distancia y orientación con respecto al objetivo. Los datos de entrenamiento son calculados de acuerdo con la geometría del problema y almacenados con el formato "entrada-entrada-salida" en el fichero de cabecera F4_trn.h. La sección inicial de dicho fichero se muestra en la parte izquierda de la Figura 8, junto con la sección del programa principal que lanza la rutina de aprendizaje. En la zona derecha de la figura se muestran la superficie de control objetivo y la que finalmente ha sido aprendida por el sistema en el proceso de aprendizaje.

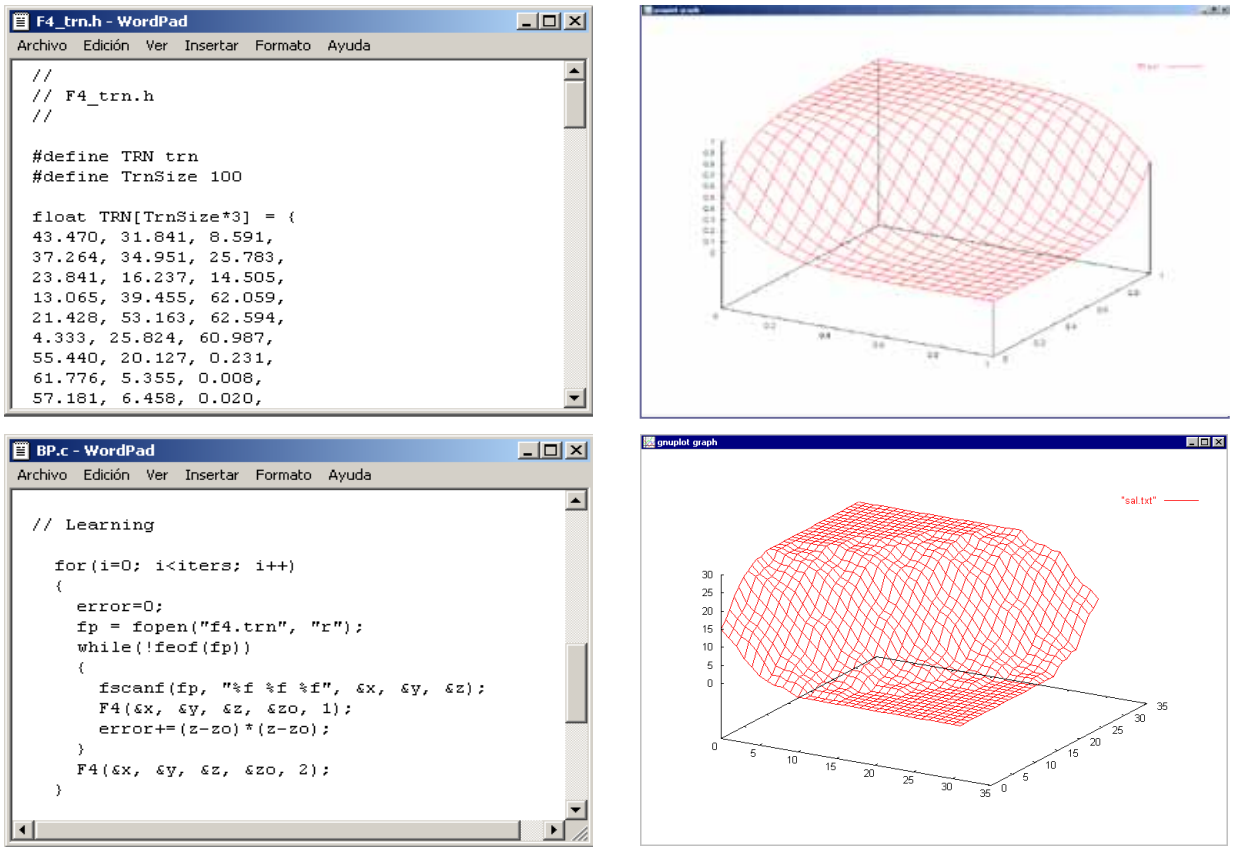

Fig 8. Ajuste de parámetros de la base de reglas mediante un algoritmo de aprendizaje *on-chip*

6. CONCLUSIONES

El uso combinado de las herramientas del entorno *XFuzzy* y las herramientas de Xilinx para diseño de sistemas empotrados facilita el desarrollo de sistemas de inferencia difusos como módulos IP para FPGAs. Dichos módulos pueden actuar como periféricos de los procesadores MicroBlaze o PowerPC y aprovechan la capacidad de memoria con que cuentan las actuales familias de FPGAs. La viabilidad de la técnica de realización propuesta en el artículo se ilustra mediante su aplicación a un problema de navegación de robots móviles autónomos.

7. REFERENCES

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

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

[3] M. Togai, H. A. Watanabe, "A VLSI implementation of a fuzzy inference engine: toward an expert system on a chip", *Information Science*, Vol. 38, N. 2, pp. 147-163, 1985.

[4] T. Yamakawa, T. Miki, "The current mode fuzzy logic integrated circuits fabricated by the standard CMOS process", *IEEE Transactions on Computer*, Vol. 35, N. 2, pp. 161-167, 1986.

[5] I. Baturone, A. Barriga, S. Sánchez Solano, C. J. Jiménez Fernández, D. R. López, *Microelectronic design of fuzzy logic-based systems*, CRC Press, 2000.

[6] H. Eichfeld, T. Kunemund, M. Menke, "A 12b general-purpose fuzzy logic controller chip", *IEEE Transactions on Fuzzy Systems*. vol.4, N. 4, pp. 460-475, Nov. 1996.

[7] K. Shimizu, M. Osumi, F. Imae, "Digital Fuzzy Processor FP-5000", Proc. of 2nd Int. Conf. on Fuzzy Logic & Neural Networks, pp. 539-542, Iizuka, 1992.

[8] A. Pagni, "Digital approaches", in *Handbook of Fuzzy Computation*, Institute of Physics Publishing, 1998.

[9] A. P. Ungering, H. Bauer, K. Goser, "Architecture of a fuzzy-processor based on an 8-bit microprocessor", Proc. IEEE Int. Conf. on Fuzzy Systems, pp. 297-301, Orlando, 1994.

[10] V. Salapura, "A fuzzy RISC processor", *IEEE Transactions on Fuzzy Systems*, vol. 8, no. 6, pp. 781-90, Dec. 2000.

[11] R. Bannalyne, "Motorola's 68HC12 an evolution from 8-bit to 16-bit", *Embedded-System Engineering*, Vol. 4, N. 4, pp. 32-3, June-July 1996.

[12] L. Fortuna, M. Lo Presti, C. Vinci, A. Cucuccio, "Recent trends in fuzzy control of electrical drives: an industry point of view", Proc. Int. Simp. Circuits and Systems, vol. 3 pp. 459-461, 2003.

[13] U. Çilingiroglu, B. Pamir, Z. S. Günay, F. Dülger, "Sampled-analog implementation of application-specific fuzzy controllers", *IEEE Transactions on Fuzzy Systems*, Vol. 5, N. 3, pp. 431-442, 1997.

[14] N. Evmorfopoulos, J. Avaritsiotis, "An adaptive digital fuzzy architecture for application-specific integrated circuits", *Journal of Active and Passive Electronic Components*, vol. 25, n. 4, pp. 289-306, Apr. 2002.

[15] A. Barriga, R. Senhadji, C. J. Jimenez, I. Baturone, S. Sánchez-Solano, "A design methodology for application specific fuzzy integrated circuits", Proc. IEEE Int. Conf. on Electronics, Circuits and Systems, vol. 1, pp. 431-434, Sept. 1998.

[16] T. Hollstein, S. Halgamuge, M. Glesner, "Computer aided design of fuzzy systems based on generic VHDL specifications", *IEEE Transactions on Fuzzy Systems*, Vol. 4, N. 4, pp. 403-417, 1996.

[17] N. Manaresi, R. Rovatti, E. Franchi, R. Guerrieri, G. Baccarani, "A silicon compiler of analog fuzzy controllers: from behavioral specifications to layout", *IEEE Transactions on Fuzzy Systems*, Vol. 4, N. 4, pp. 418-428, 1996.

[18] Gaisler Research Company: http://www.gaisler.com/

[19] MicroBlaze Reference Guides, Xilinx, Inc.

[20] F. J. Moreno-Velo, I. Baturone, S. Sánchez-Solano, A. Barriga. "Rapid Design of Complex Fuzzy Systems with Xfuzzy", Proc. IEEE Int. Conf. on Fuzzy Systems , pp. 342-347, St. Louis, May. 2003.

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

[22] E. Lago, C. J. Jiménez, D. R. López, S. Sánchez-Solano, A. Barriga, "Xfvhdl: A Tool for the Synthesis of Fuzzy Logic Controllers", DATE'98, pp. 102-107, Paris, Feb. 1998.

[23] A. Cabrera, S. Sánchez-Solano, P. Brox, A. Barriga, R. Senhadji, "Hardware/software codesign of configurable fuzzy control systems", *Applied Soft Computing*, Vol. 4, nº 3, pp. 271-285, Elsevier, Aug. 2004.