DESARROLLO DE MÓDULOS-IP DE CONTROLADORES DIFUSOS PARA EL DISEÑO DE SISTEMAS EMPOTRADOS SOBRE FPGAS[1]

M. BROX[1], A. GERSNOVIEZ, S. SÁNCHEZ-SOLANO, A. CABRERA, I. BATURONE
[1] *Instituto de Microelectrónica de Sevilla, Centro Nacional de Microelectrónica, CSIC*
Avda. Reina Mercedes s/n, 41012-Sevilla, España. <maria@imse.cnm.es>

En esta comunicación se describe el diseño de controladores basados en lógica difusa como módulos de Propiedad Intelectual (IP) compatibles con los sistemas de procesado disponibles en las familias de FPGAs de Xilinx. El trabajo fue propuesto como caso práctico de un curso de postgrado de diseño de sistemas empotrados sobre FPGAs. Su realización permitió reforzar los conocimientos de los alumnos en disciplinas relacionadas con: diseño de sistemas digitales, arquitectura de computadores, codiseño hardware-software y aplicaciones de control.

1. Introducción

La lógica difusa representa una herramienta válida para emular los mecanismos de razonamiento utilizados por el cerebro humano [1]. Uno de los campos de mayor desarrollo teórico y de aplicaciones de la lógica difusa es el de control de procesos [2]. En los últimos años se ha producido un incremento considerable del número de aplicaciones de control que emplean técnicas de inferencia basadas en lógica difusa [3] debido a que estas técnicas permiten el desarrollo de sistemas de control complejos a partir de la descripción lingüística del conocimiento de un operador experto, sin necesidad de emplear modelos matemáticos y con buenas características de robustez frente a cambios de las condiciones de operación.

La base de conocimiento de un sistema de control basado en lógica difusa está formado por un conjunto de reglas en cuyos antecedentes y consecuentes se emplean términos lingüísticos representados mediante conjuntos difusos (variables lingüísticas). El núcleo fundamental del sistema es el módulo de inferencia difuso (FIM, *Fuzzy Inference Module*) el cual se encarga de fuzzificar las entradas, evaluar las diferentes reglas del sistema y defuzzificar las salidas, existiendo una gran diversidad de operadores para la realización de estas tareas [4].

A lo largo de los últimos años se han propuesto diversas alternativas para implementar los controladores difusos agrupándose en realizaciones software, realizaciones hardware y realizaciones híbridas. Las realizaciones software han sido las soluciones más ampliamente adoptadas y se caracterizan por su elevado grado de flexibilidad, aunque su tiempo de respuesta viene limitado por la inherente ejecución secuencial de los programas. Por otro lado, las realizaciones totalmente hardware del controlador difuso representan una alternativa totalmente opuesta a la anterior y son requeridas para aquellas aplicaciones en las que la velocidad de inferencia sea un factor determinante [5]. Por último, se encuentran las realizaciones híbridas basadas en técnicas de codiseño hardware/software (HW/SW) que permiten obtener un compromiso adecuado entre las ventajas e inconvenientes de las alternativas anteriores [6]. La distribución entre la implementación hardware y la ejecución software de las tareas que debe realizar el sistema de control difuso permite obtener una solución que combine las ventajas de flexibilidad y velocidad. Una posible estrategia de particionado de tareas consiste en la implementación sobre hardware dedicado del módulo de inferencia difusa y la realización software de las restantes tareas de configuración y procesado convencional.

Por otra parte, el vertiginoso desarrollo de la industria microelectrónica de los últimos años ha conllevado, entre otras, dos importantes consecuencias. Una de ellas es la importante transformación

[1] Proyectos TEC2005-04359/MIC (Ministerio de Educación y Ciencia) y TIC2006-635 (Junta de Andalucía).

sufrida por los dispositivos lógicos programables. Dicha transformación ha afectado, no solo al incremento del número de recursos de propósito general incluidos en los dispositivos, sino también a la introducción de elementos específicos (memorias, multiplicadores, generadores de señales de reloj, etc.), que permiten implementar sistemas muy complejos en una FPGA. Esto ha sido reforzado por la existencia de numerosos bloques de sistema disponibles como módulos-IP 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. Algunos ejemplos significativos de este tipo de módulos son los procesadores MicroBlaze de Xilinx y Nios de Altera.

La otra consecuencia significativa del avance de la microelectrónica es el planteamiento de nuevas estrategias de diseño debido a que continuamente se incrementa la diferencia entre los recursos proporcionados por las tecnologías y la productividad alcanzada por los diseñadores [7]. Para salvar este inconveniente se han propuesto nuevas estrategias de diseño basadas en la reutilización de bloques básicos previamente diseñados, denominados "Módulos de Propiedad Intelectual" (IP)[8] [9]. Estos módulos son fácilmente adaptables a diferentes funcionalidades, así como a diferentes tecnologías de fabricación. Su uso en combinación con herramientas de CAD que facilitan la realización de los sistemas supone importantes ventajas económicas, ya que reduce la duración del ciclo de desarrollo de nuevos productos y acelera su introducción en el mercado.

En esta comunicación se describe una estrategia hardware/software de realización de controladores difusos empotrados basada en la utilización de módulos-IP. Para ello, en la sección 2 se expone la estrategia de realización y los componentes físicos de la plataforma de desarrollo. La descripción del sistema MicroBlaze como sistema de procesado es realizada en la sección 3. La sección 4 muestra cómo se lleva a cabo la implementación hardware de los módulos de inferencia difusos con ayuda de las herramientas de síntesis de *Xfuzzy*. El ciclo de diseño del controlador difuso como módulo IP actuando como periférico estándar OPB del procesador MicroBlaze es descrito en la sección 5. La sección 6 describe la aplicación dentro del campo de la robótica móvil en la que esta investigación se encuadra. Por último, la sección 7 recoge las conclusiones de la comunicación.

2. Plataforma de desarrollo para sistemas de control difusos

La Fig. 1 muestra una estrategia de codiseño HW/SW para controladores difusos consistente en un particionado de las tareas del sistema. De acuerdo con dicha estrategia, todas aquellas tareas que no requieren el uso de recursos específicos serán ejecutadas por un procesador de propósito general. Entre tales tareas se encuentran 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 la monitorización del comportamiento del controlador. Por otra parte, las tareas relacionadas con la aplicación de técnicas neuro-difusas, junto con los circuitos de interfaz del sistema, serán implementados mediante un soporte hardware específico debido a que su realización en un procesador convencional resultaría poco eficiente.

Para dar soporte físico a los controladores difusos se ha empleado una placa de desarrollo de FPGAs Spartan 3 (de Digilent Inc). Esta placa dispone de un millón de puertas, 24 multiplicadores de 18 bits y 432 Kbits de memoria RAM de tipo bloque. También contiene recursos de entrada/salida (switches, botones, LEDs, y un display de siete-segementos), puertos serie y JTAG, y una SRAM asíncrona de 1MB.

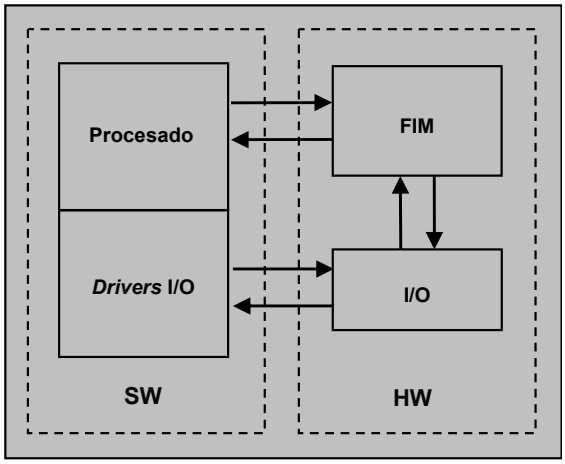


Figura 1. Codiseño hardware/software de un sistema de control basado en lógica difusa

3. MicroBlaze como sistema de procesado

Para el diseño del sistema de procesado se ha utilizado el procesador MicroBlaze [10]. Dicho procesador está disponible como módulo-IP junto a una gran variedad de periféricos, controladores y estructuras de buses que permiten configurar el sistema según las necesidades de una aplicación concreta (Fig. 2). MicroBlaze es un procesador RISC de 32 bits basado en arquitectura Harvard, con buses separados para instrucciones y datos y posibilidad de usar memorias cachés independientes para ambos buses. Esta arquitectura está optimizada para su implementación sobre FPGAs de Xilinx. Su conjunto 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 ser realizadas por hardware a través de los multiplicadores disponibles en las FPGAs. MicroBlaze usa un bus específico (LMB) para acceder a las memorias de bloque disponibles en la FPGA, mientras que emplea el bus estándar OPB de IBM para conectar memoria externa y periféricos. Finalmente, posee diferentes canales FIFO para la conexión de funciones de usuario.

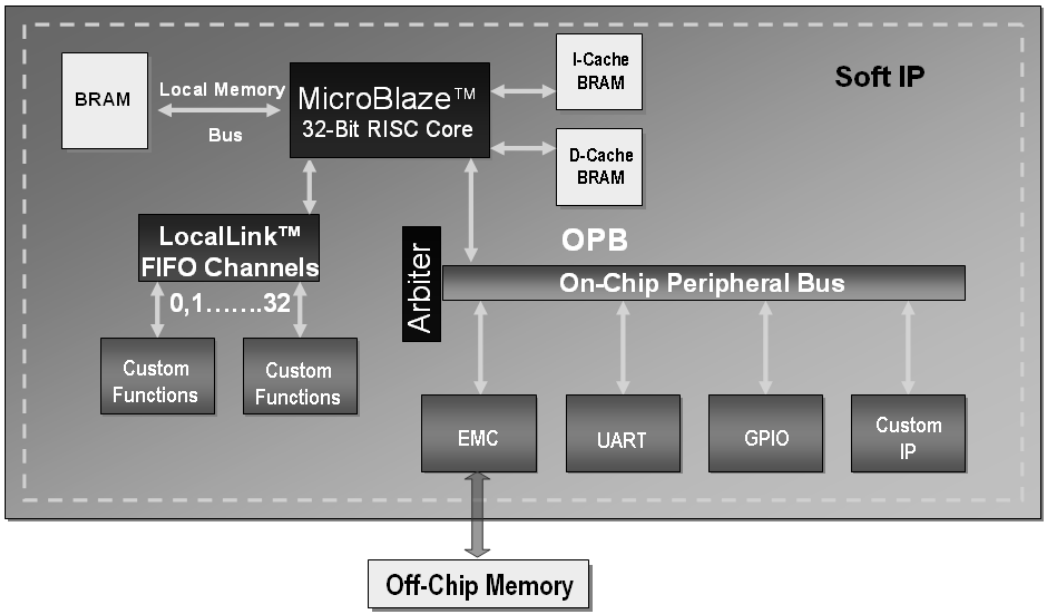


Figura 2. Diagrama de bloques de un sistema de procesado basado en MicroBlaze

4. Implementación hardware de módulos de inferencia difusos

El entorno de diseño *Xfuzzy* facilita las diferentes etapas de desarrollo de un sistema de inferencia basado en lógica difusa. Para ello incluye herramientas de descripción, simulación y simplificación que permiten definir, verificar, y optimizar el sistema difuso. El entorno también incluye herramientas de síntesis que proporcionan implementaciones hardware o software del sistema de inferencia [11].

La implementación hardware de los módulos de inferencia difusos ha sido realizada usando la herramienta de síntesis del entorno *Xfuzzy* llamada *Xfvhdl* [12]. Esta herramienta proporciona una descripción VHDL del sistema de acuerdo con una arquitectura especialmente optimizada para la realización de sistemas difusos mediante hardware dedicado [13]. Los aspectos claves de esta arquitectura, que permiten aumentar la eficiencia de su realización digital, son: 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.3). Al ejecutar la herramienta de síntesis, el diseñador puede elegir distintas alternativas tanto arquitecturales como de implementación, de acuerdo con las particulares características de su problema. Respecto a la arquitectura, el diseñador puede seleccionar el tipo de generador de función de pertenencia (usando una técnica aritmética o una basada en memoria), el operador usado como conectivo (mínimo o producto), y el método de defuzzificación empleado. En cuanto a la implementación en la FPGA, el diseñador puede seleccionar el dispositivo concreto que va a ser usado y puede elegir la implementación de los diferentes componentes de las bases de conocimiento como lógica combinacional o por almacenamiento en memorias ROM o RAM. En este último caso, es posible elegir también entre el uso de RAM de tipo bloque o asociada a los CLBs de la FPGA [14]. Se obtiene como resultado una descripción VHDL del sistema que puede ser sintetizada con las herramientas del entorno ISE de Xilinx.

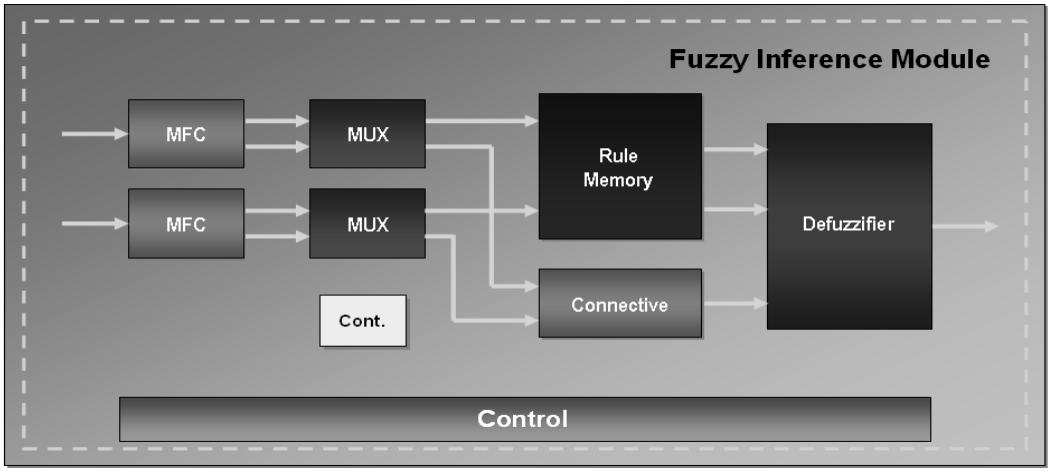


Figura 3. Diagrama de bloques de un módulo de inferencia difusa implementado mediante *Xfuzzy*

5. Ciclo de diseño del módulo IP

El entorno EDK (*Embedded Development Kit*) engloba un conjunto de componentes IP y herramientas de diseño que facilitan el desarrollo de sistemas de procesado empotrados sobre FPGAs de Xilinx. La interfaz gráfica de usuario XPS (*Xilinx Platform Studio*) proporciona, asimismo, una serie de plantillas que facilitan el desarrollo de periféricos conectables al bus OPB. Estas plantillas consisten en código VHDL que incluye dos componentes (Fig.4): IPIF (*Intellectual-Property Interface*), que realiza las funciones de interfaz con el bus OPB; y User_logic, que contiene la lógica desarrollada por el usuario. En nuestro caso, este último fichero incluirá la descripción VHDL del controlador difuso proporcionada por *Xfuzzy* más el código necesario para acceder al controlador a través de los registros del módulo-IP. Ambos componentes se comunican a través de la interfaz IPIC (*Intellectual-Property Interconnect*), que es independiente del bus periférico. Existen diferentes tipos de plantillas dependiendo del modo de operación del periférico (maestro/esclavo) y los servicios

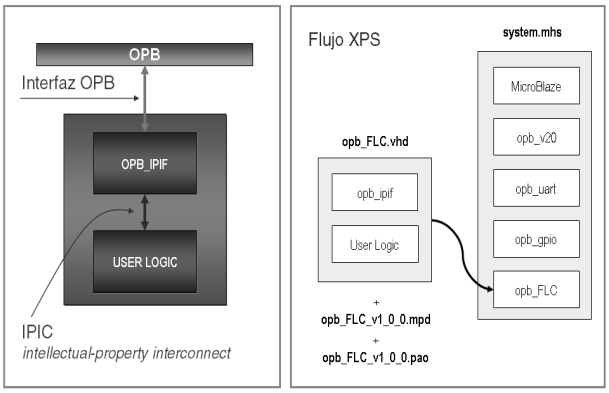


Figura 4. Conexión al bus OPB de periféricos de usuario

proporcionados por el bloque IPIF. La interfaz gráfica de usuario XPS también facilita la generación de los archivos ".mdp" (*microprocessor peripheral definition*) and ".pao" (*peripheral analyze order*). Estos ficheros son necesarios para el uso del módulo-IP en XPS como otros periféricos del sistema.

El ciclo de diseño de un controlador difuso como módulo-IP incluye los siguientes pasos. En primer lugar, mediante el asistente para creación de periféricos de XPS se indica el nombre y la versión del periférico, así como su lugar de destino. Seguidamente se selecciona el tipo de bus al que irá conectado. En tercer lugar, 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.). A continuación tiene lugar la implementación de la lógica de usuario. Finalmente, el último paso es la importación del periférico. El asistente para importar periféricos de XPS requiere la inclusión de las plantillas VHDL y los elementos de librería, así como la asignación de puertos y la definición de parámetros. Una vez completados estos pasos el módulo-IP puede utilizarse como cualquier otro periférico en EDK (Fig.5).

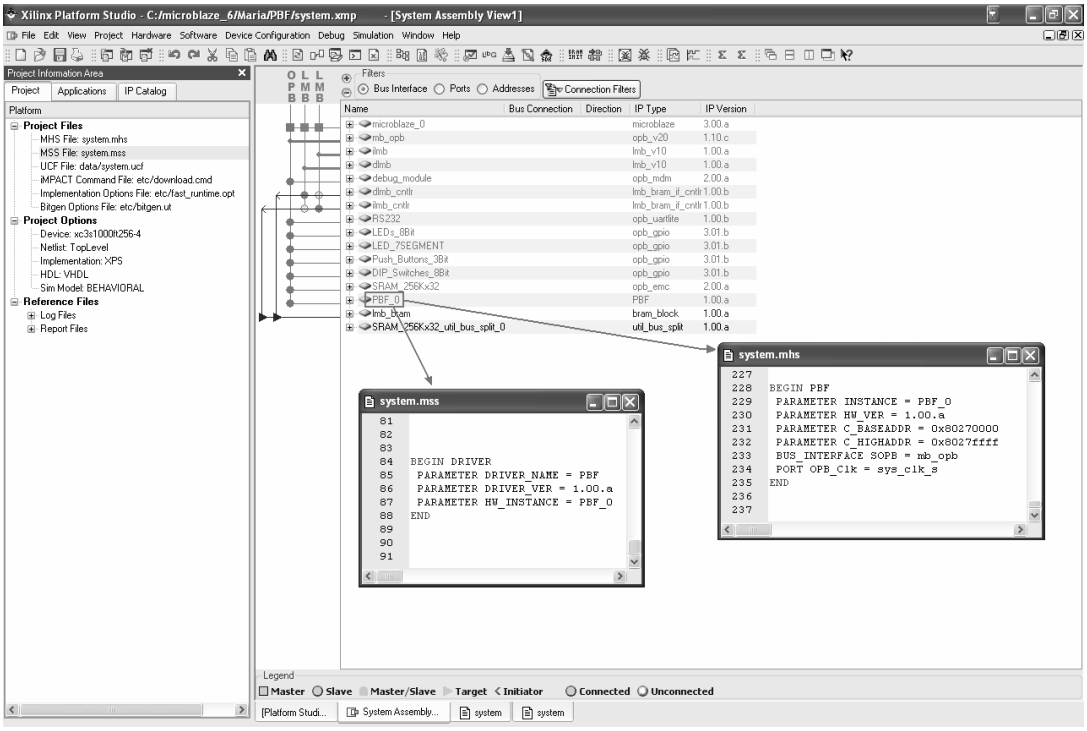


Figura 5. Uso del módulo-IP del controlador difuso como un periférico de MicroBlaze

6. Aplicaciones a la robótica móvil

Con objeto de validar la técnica de desarrollo propuesta, se ha empleado en el diseño de un sistema de control para el problema de aparcamiento de un vehículo autónomo. En este caso 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 atrás y hacia delante (Fig.6).

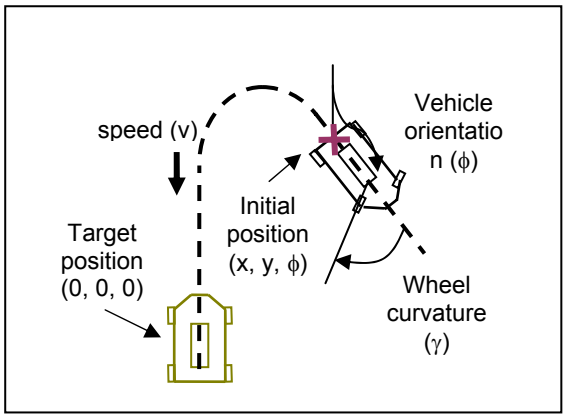


Figura 6. Ilustración del problema de aparcamiento

La plataforma de desarrollo descrita anteriormente ha sido aplicada en la realización de un sistema de control difuso para el aparcamiento del vehículo Romeo 4R. Dicho vehículo, construido y diseñado en la Escuela Superior de Ingenieros (ESI) de la Universidad de Sevilla (Fig.7), es un robot eléctrico y autónomo que está dotado de una serie de sensores entre los que destacan, para nuestro objetivo de control, encoders de tracción, de dirección y un giróscopo. 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. Un procesador digital de señal (DSP) TMS-320LF de Texas Instruments es el encargado del control a bajo nivel de los motores, así como de la adquisición de la información procedente de los sensores y su procesado para determinar la velocidad, curvatura, orientación y posición del vehículo. Por tanto, el controlador de alto nivel queda liberado de las tareas de adquisición y procesado de la información procedente de los sensores así como del control directo de los motores. De esta forma, sus tareas se centran en 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 ser incorporado también en el programa que se ejecute en el controlador difuso.

Figura 7. Romeo 4R

De acuerdo con la estrategia de codiseño HW/SW empleada, 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 empleadas por un conductor humano.

El sistema de control propuesto utiliza 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) (Fig.8). Las bases de conocimiento *Position, Planning y Direction* 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 *Planning* para obtener la planificación del aparcamiento (plan). Esta base toma la 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 *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 conocimiento *Celerity, Backward y Forward* corresponden a estructuras de control. La base de conocimiento *Celerity* determina el valor absoluto de la velocidad (cel), mientras que las bases *Backward* y *Forward* determinan la curvatura de Romeo 4R en un sentido u otro (bw, fw). Es posible la unificación de *Backward* y *Forward* en un mismo módulo debido a que comparten las mismas entradas. La salida final del controlador (v) 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) a partir de un multiplicador como se ilustra en Fig. 8. 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) tal y como se muestra en Fig. 8.

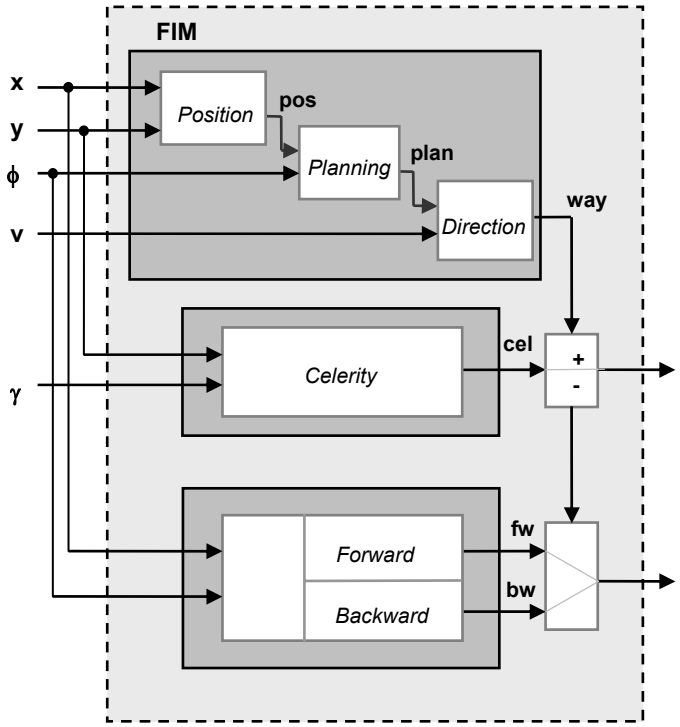
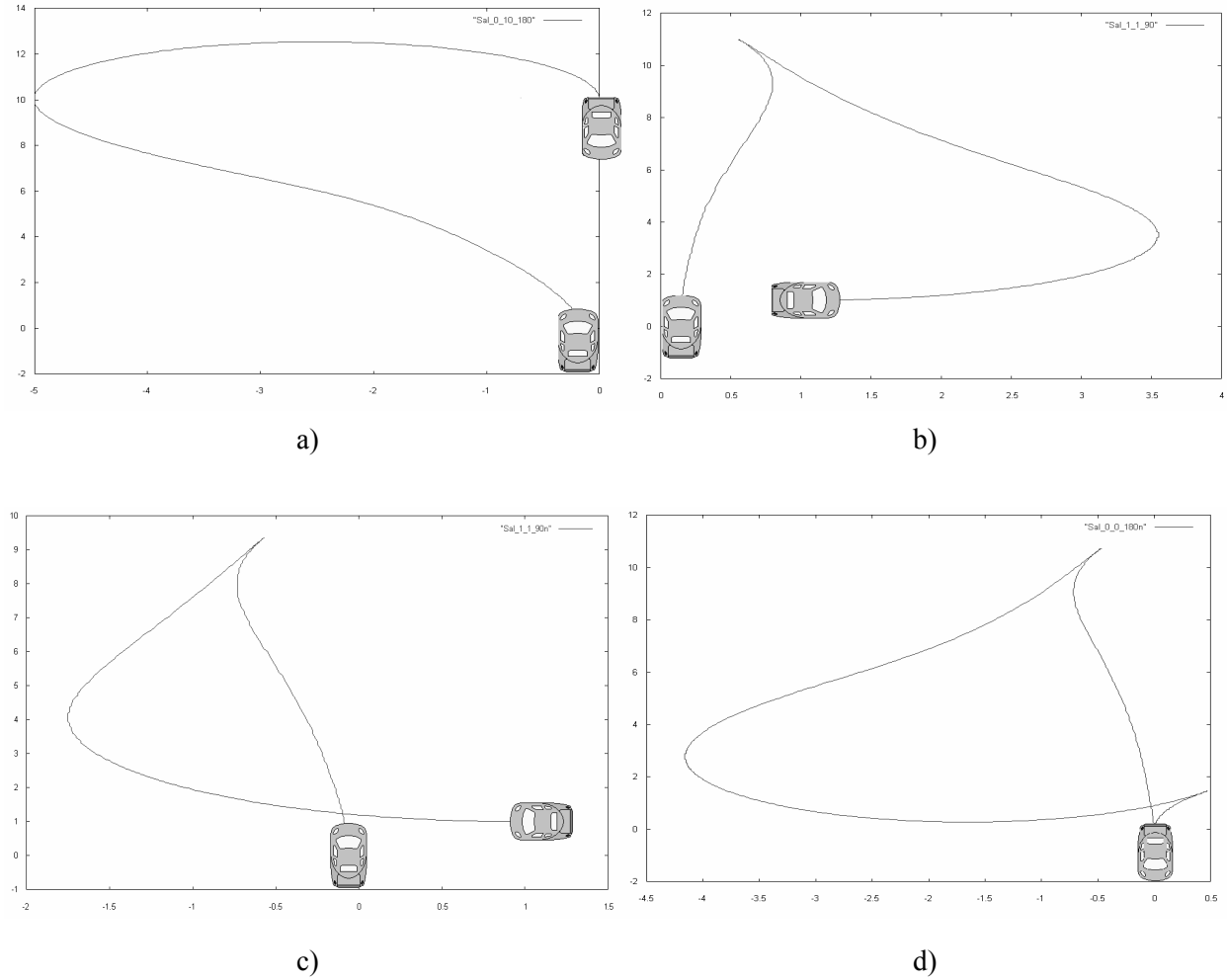


Figura 8. Estructura del sistema de control para el problema de aparcamiento

Los resultados experimentales mostrados en Fig. 9 demuestran que la técnica de desarrollo propuesta es válida. En todos los casos la posición de aparcamiento objetivo es (0, 0, 0), de forma que el vehículo describe trayectos hacia delante y hacia atrás hasta alcanzarla. La Fig. 9-a corresponde a un caso en el que el vehículo se encuentra en la vertical x=0 situado en frente de la posición de aparcamiento objetivo. Como está lo suficientemente lejos del lugar de aparcamiento, consigue alcanzar esta posición conduciendo sólo hacia atrás. Las Fig. 9-b y 9-c corresponden a casos en los que el vehículo se encuentra en una posición muy cercana al lugar de aparcamiento pero con una orientación perpendicular a éste. El controlador difuso conduce al robot hacia delante con unos valores de curvatura máximos en un principio y luego con valores más pequeños hasta alcanzar la vertical x=0 con orientación Φ=0. A partir de este punto, Romeo 4R es conducido hacia atrás con curvatura prácticamente nula hasta alcanzar el aparcamiento objetivo. La trayectoria mostrada en Fig. 9-d corresponde a un caso más complejo compuesto de tres tramos en el que el vehículo se encuentra situado originalmente en el lugar de aparcamiento pero con orientación opuesta a la deseada. En este caso, comienza circulando hacia atrás con curvatura máxima hasta alcanzar un punto en el que empieza a ser conducido hacia delante, siguiendo a partir de entonces un camino similar al descrito en la Fig. 9-c.



a) b)

c) d)

Figura 9. Resultados experimentales

7. Conclusiones

En esta comunicación se ha propuesto una técnica de desarrollo de módulos-IP de controladores difusos que actúan como periféricos OPB del procesador MicroBlaze de Xilinx. Esta técnica ha sido validada demostrándose su comportamiento correcto para el problema de aparcamiento diagonal de un vehículo autónomo. Debido al importante y actual uso de los dispositivos lógicos programables en técnicas de diseño, consideramos que el tema tratado en la comunicación supone un trabajo muy interesante a desarrollar como caso práctico en un curso de postgrado de diseño de sistemas empotrados sobre FPGAs. De esta forma, los alumnos pueden reforzar sus conocimientos dentro de un campo de importancia actual, familiarizándose con el entorno EDK y el uso de las FPGAs.

Referencias

[1] Zadeh, L. A., *Outline of a new approach to the analysis of complex systems and decision processes. IEEE Transactions on Systems, Man, and Cybernetics,* Vol. 3, N. 1, 28-44 (1973)

[2] Driankov, D., Hellendoorn, H., Reinfrank, M., *An Introduction to Fuzzy Control. Springer-Verlag* (1993)

[3] Ross, T. *Fuzzy logic with Engineering Aplication. Mc Graw Hill* (1995)

[4] Baturone, I., Sánchez-Solano, S., Barriga, A., Huertas, J. L. *Implementation of inference/defuzzification methods via continuous-time analog circuits. IFSA World Congress.*623-626 (1995)

[5] Baturone, I., Barriga, A., Sánchez-Solano, S., Jiménez, C.J., López, D. *Microelectronics Design of Fuzzy Logia-Based Systems. CRC Press*(2000)

[6] L. Reyneri. *Implementation issues of neuro-fuzzy hardware: going toward HW/SW codesign. IEEE-Transactions-on-Neural-Networks.* Vol. 14, Nº1, 176-194 (2003)

[7] Ferrari, A., Sangiovanni-Vincentelli, A., *System Design: traditional concepts and new paradigms. International Conference on Computer Design. 2-12 (1999)*

[8] W. Savage, J. Chilton y R. Camposano. *IP reuse in the System on a Chip Era*. *13th International Symposium on System Synthesis*. 2-7 (2000)

[9] M. Keating. The reuse methodology manual: *Toward a reuse discipline. Design, Automation and Test in Europe* (DATE).141-145 (1998)

[10] *MicroBlaze Reference Guide*. Xilinx Inc. (2002)

[11] F. J. Moreno-Velo, I. Baturone, S. Sánchez-Solano, A. Barriga. *Rapid Design of Complex Systems with Xfuzzy. Proc. IEEE Int. Conf. On Fuzzy systems.*342-347 (2003)

[12] *Xfuzzy: Herramientas de CAD para Lógica Difusa* (http://www.imse.cnm.es/Xfuzzy/)

[13] S. Sánchez –Solano, A. Barriga, C. J. Jiménez, J. L. Huertas. *Design and Applications of Digital Fuzzy Controllers*. *IEEE Int. Conf. on Fuzzy Systems*. 869-874 (1997)

[14] E. Lago, C. J. Jiménez, D.R. López, S. Sánchez-Solano, A. Barriga. *Xfvhdl: A tool for the Synthesis of Fuzzy Logic Controllers. Design, Automation and Test in Europe* (DATE'98). 102-107 (1998)