View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by idUS. Depósito de Investigación Universidad de Sevilla

ARQUITECTURA EFICIENTE PARA LA IMPLEMENTACIÓN HARDWARE DE SISTEMAS DE INFERENCIA DIFUSOS

A. Cabrera[1], S. Sánchez-Solano[2], C. J. Jiménez[2], A. Barriga[2], I. Baturone[2]

[1] Dpto. Automática y Computación. Facultad de Ingeniería Eléctrica. Instituto Superior Politécnico "José Antonio Echeverría", Ciudad de la Habana, Cuba.

[2] Instituto de Microelectrónica de Sevilla - Centro Nacional de Microelectrónica
Avda. Reina Mercedes s/n, (Edif. CICA)
E-41012, Sevilla, Spain

Ingeniería Electrónica, Automática y Comunicaciones,
Vol. XXIII, No. 1, pp. 59-66, 2003.

This material is presented to ensure timely dissemination of scholarly and technical work. Copyright and all rights therein are retained by authors or by other copyright holders. All persons copying this information are expected to adhere to the terms and constraints invoked by each author's copyright. In most cases, these works may not be reposted without the explicit permission of the copyright holder.

Ingeniería Electrónica, Automática y Comunicaciones, Vol. XXIII, No. 1, 2003

Arquitectura eficiente para la implementación hardware de sistemas de inferencia difusos

A. J. Cabrera;[1] S. Sánchez-Solano;[2] C. J. Jiménez;[2] A. Barriga[2] e I. Baturone[2]

[1] Departamento de Automática y Computación. Facultad de Ingeniería Eléctrica, Instituto Superior Politécnico José Antonio Echeverría, Ciudad de La Habana, Cuba.
[2] Instituto de Microelectrónica de Sevilla, Centro Nacional de Microelectrónica, Sevilla, España.

RESUMEN / *ABSTRACT*

Se describen los elementos integrantes de una arquitectura de bajo costo y alto desempeño para la implementación hardware de sistemas de inferencia difusos, la cual se basa 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 defusificación simplificados. También se expone el entorno de desarrollo de sistemas difusos Xfuzzy, con énfasis en la herramienta **xfvhdl**, la cual permite la generación de código VHDL para los diferentes elementos de la arquitectura descrita.

Palabras clave: controlador difuso, Xfuzzy, VHDL, FPGA.

*This article describes a high performance and low cost architecture for hardware implementation of fuzzy inference systems. It is based on active rules processing, the overlapping degree of the inputs memberships functions limited to two and the use of simplified defuzzification methods. The Xfuzzy development environment is also exposed as well as the **xfvhdl** tool which lets the VHDL code generation for the different parts of the described architecture.*
Key words: fuzzy controller, Xfuzzy, VHDL, FPGA.

Recibido: octubre 2002 Aprobado: noviembre 2002

INTRODUCCIÓN

La capacidad de los sistemas difusos para describir la experiencia de un operador humano mediante reglas simples expresadas en lenguaje natural, junto con el hecho de que eliminan la necesidad de disponer de un modelo analítico del sistema a controlar, ha motivado un incremento considerable del número de aplicaciones de control que emplean técnicas de inferencia basadas en lógica difusa.[1]

Poco tiempo después de su formulación, a mediados de los años sesenta por el profesor Lofti Zadeh, comenzaron a desarrollarse aplicaciones de control basadas en lógica difusa, fundamentalmente en países de Asia y Europa. Los primeros trabajos sobre control difuso fueron desarrollados por Mamdani y Assilian.[2] Hoy, su aplicación se ha generalizado a escala mundial, desde grandes entornos industriales hasta pequeños electrodomésticos, constituyendo una de las ramas de mayor desarrollo dentro del llamado control inteligente.[3]

Existen diversas formas de implementar los controladores difusos, tanto basados en software como en hardware,[4] incluso utilizando técnicas de codiseño HW/SW. Cada una de las mismas presenta sus ventajas y limitaciones.

En el presente trabajo se expone una arquitectura que permite una implementación hardware eficiente en términos de velocidad y costo de un sistema de inferencia difuso, estando organizado de la siguiente forma: En primer lugar se presentan algunos elementos básicos de lógica difusa y la estructura general de un

controlador difuso. Seguidamente se abordan diferentes estrategias de implementación de controladores difusos, se exponen los detalles de la arquitectura propuesta y se describen las herramientas de desarrollo disponibles para esta.

FUNDAMENTOS DE CONTROL DIFUSO

Un sistema de control basado en lógica difusa (controlador difuso)[1] es capaz de evaluar un grupo de reglas del tipo *IF* < **antecedente** > *THEN* < **consecuente** >, muy similares a las utilizadas en el lenguaje natural, donde el **antecedente** y el **consecuente** están compuestos por una combinación de variables, conjuntos y operadores difusos. Ejemplos de estas reglas pueden ser las siguientes:

IF la Temperatura es **alta** *AND* el Nivel es **bajo**, *THEN* **aumentar mucho** el Flujo
IF la Temperatura es **baja** *AND* el Nivel es **alto**, *THEN* **reducir mucho** el Flujo

donde:
Temperatura, Nivel y Flujo: Variables lingüísticas que constituyen las entradas y la salida del controlador.
AND y *THEN:* Evaluados mediante operadores difusos.
alto, **bajo**, **aumentar mucho:** Representan conjuntos difusos de sus respectivas variables.

A diferencia de los conjuntos clásicos donde un elemento pertenece o no a un determinado conjunto, en la teoría de conjuntos difusos se establece un **grado de pertenencia** $\mu(x)$ de un elemento a un determinado conjunto difuso, el cual se expresa mediante un valor real en el intervalo [0,1].

En la figura 1 se ilustra este concepto donde se representan las funciones de pertenencia de los conjuntos difusos **baja**, **media** y **alta** para la variable Temperatura. Nótese que un valor de temperatura de 42º pertenece a más de un conjunto difuso con diferentes grados de pertenencia por lo que se desprende que en un sistema de control difuso varias reglas pueden estar activas simultáneamente.

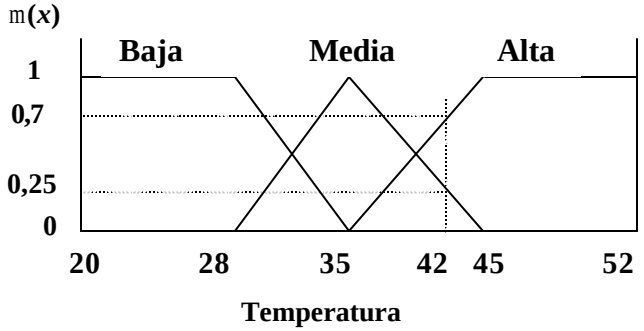

Figura 1
Conjuntos difusos de la variable Temperatura.

El número total de reglas en un sistema de control difuso dependerá del número de entradas del controlador y del número de conjuntos difusos de cada una de las mismas. En un controlador de *P* entradas, cada una con n_1, n_2, ...n_P conjuntos difusos, el número total de reglas *R* será:

$$R = n^1, n^2, \ldots n_P \quad \ldots(1)$$

Si todas las entradas poseen igual número de conjuntos difusos la expresión (1) se transforma en:

$$R = n^P \quad \ldots(2)$$

Obsérvese el crecimiento exponencial del número total de reglas a medida que se incrementa el número de entradas. Por ejemplo, un sistema de dos entradas con 8 funciones de pertenencia cada una tendría 64 reglas. Si se adiciona una tercera entrada este número se incrementa a 512.

Sin embargo, en un controlador difuso el número total de reglas que es realmente preciso evaluar en un momento dado (reglas activas) es significativamente inferior al número total de reglas.

Volviendo a la figura 1, si la temperatura toma un valor de 42º, solo es preciso evaluar aquellas reglas que contengan el antecedente de que la variable Temperatura es **media** o **alta** mientras que todas las reglas con el antecedente de que la temperatura es baja no aportarán nada para la determinación de la salida del controlador. De esta forma el número de reglas activas en un momento dado en un controlador difuso estará determinado por el grado de solapamiento de las funciones de pertenencia de los conjuntos difusos de las entradas del controlador. Si este grado de solapamiento es *K*, entonces el número de reglas activas quedará limitado a K^P. Así, retomando el mismo sistema del ejemplo anterior, si el grado de solapamiento es de 2, el número máximo de reglas activas en un momento dado será de 4 para el sistema de dos entradas y se incrementa a solo 8 si se añade una tercera entrada, cantidades considerablemente inferiores a las de los totales de reglas. Este aspecto posee una importancia crucial en la eficiencia de las diferentes implementaciones de los controladores difusos.

• Estructura general de un sistema de control basado en lógica difusa

La figura 2 muestra la estructura general de un sistema de control basado en lógica difusa. El mismo es muy similar a un sistema de control convencional en donde el bloque denominado sistema de inferencia hace las funciones del controlador, es decir, recibe los valores de las entradas y proporciona un valor en sus salidas. La diferencia fundamental estriba en que el control convencional obtiene el valor de las salidas evaluando diferentes funciones matemáticas mientras que el sistema de inferencia obtiene el valor de las salidas evaluando un conjunto de reglas.

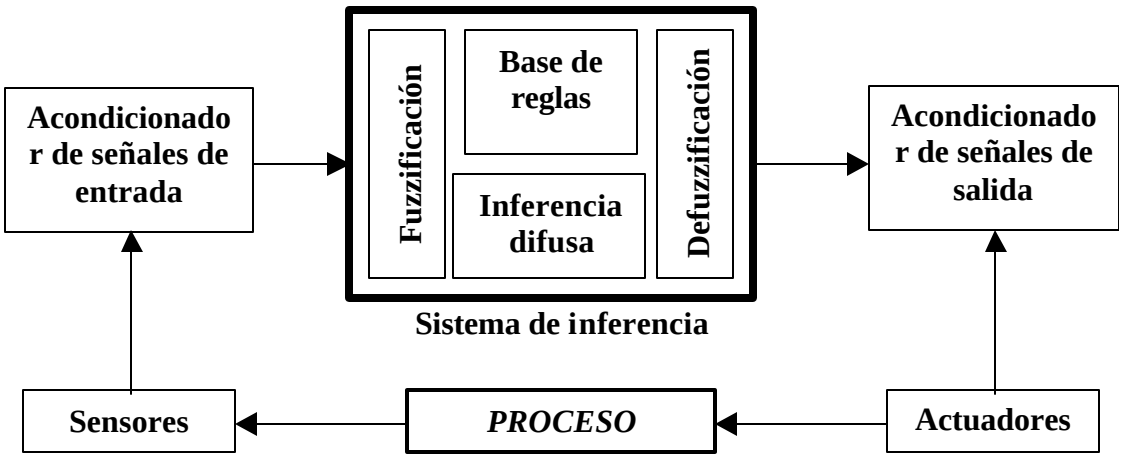

Figura 2
Estructura general de un sistema de control basado en lógica difusa.

Las variables provenientes del proceso son capturadas por sensores y transformadas en una serie de señales eléctricas representadas por tensiones o intensidades. La etapa de acondicionamiento de señales de entradas realiza las tareas de amplificación de la señal, multiplexado y conversión análogo-digital (A/D), así como diferentes algoritmos de preprocesado necesarios para calcular las entradas al **sistema de inferencia**.

Este sistema de inferencia recibe los valores de las entradas y, evaluando las diferentes reglas que describen el proceso, ofrece un valor concreto en sus salidas. Para ello y dado que en un sistema de control las entradas provenientes de los sensores representan valores concretos, es preciso determinar los valores de los grados de pertenencia de dichas entradas a los conjuntos difusos correspondientes. Esta es la función de la etapa de **fuzzificación**. Una vez fuzzificadas las entradas se accede a la **base de reglas** que describen el comportamiento del proceso y, mediante un proceso de **inferencia difusa**, se obtiene el aporte de cada regla a los conjuntos difusos que representan la acción de control de las salidas. Finalmente la etapa de **defuzzificación** proporciona valores concretos para cada una de las salidas del sistema de inferencia.

Estas salidas son procesadas por los circuitos acondicionadores de las señales de salida, los cuales usualmente realizan el proceso de conversión digital-analógico (D/A), llevando sus salidas hacia los actuadores del proceso.

En el diseño de un sistema de inferencia difuso intervienen diversos elementos, entre los cuales se encuentran los siguientes:[4]

(a) Número y tipo de las funciones de pertenencia de las entradas y salidas.

(b) Universo de discurso de las funciones de pertenencia

(c) Operadores difusos utilizados para los conectivos de antecedentes.

(d) Función de implicación utilizada.

(e) Operador de agregación de reglas.

(f) Método de defuzzificación utilizado.

Nótese la gran diversidad de operadores difusos existentes, por lo que son múltiples las posibles variantes de implementación de los sistemas de inferencia difusos.

ALTERNATIVAS DE IMPLEMENTACIÓN DE SISTEMAS DE INFERENCIA DIFUSOS

Independientemente de los operadores difusos utilizados, las alternativas de implementación de los sistemas de inferencia difusos pueden ser divididas en dos grandes grupos: implementación software o hardware.

La primera consiste en que todo el proceso de fuzzificación, inferencia y defuzzificación se realiza mediante un programa que se ejecuta sobre una determinada plataforma. La plataforma de ejecución puede variar desde una computadora personal hasta un microcontrolador, pero en cualquier variante siempre será la ejecución de un programa el encargado de fuzzificar las entradas, evaluar las diferentes reglas y defuzzificar las salidas. De aquí que este tipo de implementación se caracterice por su gran versatilidad dado que es posible seleccionar cualquier tipo de operador difuso para las diferentes etapas, independientemente de que la programación de algunos operadores pueda resultar más compleja que la de otros.

Sin embargo, el propio hecho de ejecutar un programa, caracterizado por su ejecución secuencial, contribuye a su mayor limitación: su reducida velocidad, normalmente del orden de los milisegundos y superiores. Por supuesto, esta velocidad estará determinada por la potencialidad (y costo) de la plataforma. Pero en cualquier variante es más lento que las alternativas de implementación hardware.

A esta baja velocidad contribuyen frecuentemente la forma en que se implementan los diferentes algoritmos. Por ejemplo, es usual que la evaluación de las reglas se realice de forma iterativa evaluando toda la base de reglas en lugar de las reglas activas. También, es muy frecuente la utilización de métodos de defuzzificación convencionales que requieren recorrer todo el universo de discurso de las variables de salida para obtener los valores concretos de las mismas, lo cual implica nuevamente procesos iterativos inherentemente lentos. Por ejemplo, el método más utilizado por la mayoría de las implementaciones software es el del centro de gravedad.

Por otra parte, las alternativas de implementaciones hardware de los sistemas de inferencia difusos, al no realizar la ejecución

secuencial de un programa sino al propagar señales eléctricas a través de los diferentes circuitos que lo componen, con el paralelismo propio de los mismos, resultan mucho más rápidas, con tiempos de inferencia del orden de los microsegundos e inferiores, de aquí que sea la solución para aquellas aplicaciones de control que requieran una muy elevada velocidad de respuesta.[5]

Sin embargo, la limitación fundamental de estas variantes de implementación radica en su carencia de flexibilidad. Mientras que en las alternativas software es relativamente sencillo modificar cualquier operador difuso, una vez fabricado el circuito integrado que le da soporte al sistema de inferencia resulta imposible su modificación. Incluso, aun cuando este haya sido implementado sobre un dispositivo programable (FPGA o CPLD), su modificación requiere de un nuevo proceso de síntesis del mismo.

Las variantes de implementaciones hardware pueden ser divididas en dos grandes grupos: las analógicas y las digitales. Las primeras, a pesar de tener una interfaz natural con los procesos, presentan los inconvenientes relativos a la falta de resolución y la pobre inmunidad al ruido. Adicionalmente, la carencia de herramientas de CAD (Computer Aided Design) para los desarrollos analógicos hace más complejas sus realizaciones, de aquí que la gran mayoría de las implementaciones hardware sean digitales.

Existen múltiples variantes de implementaciones mediante hardware digital de sistemas de inferencia difusos, las cuales corresponden a diferentes arquitecturas. Algunas de ellas, por su gran complejidad solo implementan parte del sistema de inferencia, mientras otras lo incorporan en su totalidad.

Debe tenerse muy presente que en las realizaciones hardware es fundamental reducir al mínimo el área del circuito integrado resultante, reduciendo así su costo. De aquí que sea preciso recurrir en ocasiones a estrategias de realización totalmente diferentes a las utilizadas en las implementaciones software. Por esta razón, las arquitecturas más eficientes en términos de costo y velocidad son aquellas que se basan en las restricciones siguientes:

(a) El procesamiento de las reglas activas solamente.[6] Al no tener que evaluar todas las reglas posibles del sistema (la mayoría de las cuales no aportan a las salidas) sino solo aquellas reglas que realmente contribuyen a las salidas, se reduce considerablemente el tiempo de inferencia

(b) La limitación del grado de solapamiento de las funciones de pertenencia de las entradas ya que así se reduce el número posible de reglas activas. La mayoría de las aplicaciones prácticas limita este solapamiento a 2.

(c) La utilización de métodos de defuzzificación simplificados.[7] Estos métodos se basan en sustituir la información de los consecuentes difusos de cada regla por una serie de parámetros que los caracterizan, de forma tal que los procesos de inferencia y defuzzificación pueden ser simultaneados, no siendo así necesario recorrer todo el universo de discurso de las variables de salida para efectuar la defuzzificación. Ejemplos de estos métodos son los de la media difusa (Fuzzy Mean, FM), el de la media difusa ponderada (Weighted Fuzzy Mean, WFM), el método de calidad (QM), el método del centro de sumas cuando la inferencia se realiza mediante el operador mínimo (CoSm) y el de Yager (YM) entre otros.

Además de las consideraciones anteriores, es también importante disponer de herramientas de CAD que permitan automatizar el proceso de síntesis del sistema de inferencia difusa, por ejemplo, disponer de herramientas que posibiliten la generación de código sintetizable en algún lenguaje de descripción de hardware para la arquitectura propuesta.

DESCRIPCIÓN DE LA ARQUITECTURA

La arquitectura[5,8] que se expone permite la realización de sistemas de inferencia difusos del tipo SISC (Singleton Input Singleton Consequent) característico de muchos sistemas de control donde las entradas, provenientes de los sensores, representan valores concretos que pueden ser representados por conjuntos difusos tipo singleton. En un sistema de inferencia SISC cada regla propone una determinada conclusión, con una "fuerza" definida por su correspondiente grado de activación.

La arquitectura SISC que se muestra en la figura 3 está basada en la consideraciones expuestas en el epígrafe anterior con la limitación a dos del grado de solapamiento de las funciones de pertenencia de las entradas. Por simplicidad se ha representado un sistema de inferencia de dos entradas y una salida aunque no existen limitaciones para el número de entradas. Esta arquitectura posee alternativas de implementación para las etapas de fuzzificación y defuzzificación y se caracteriza por su excelente relación velocidad/costo.

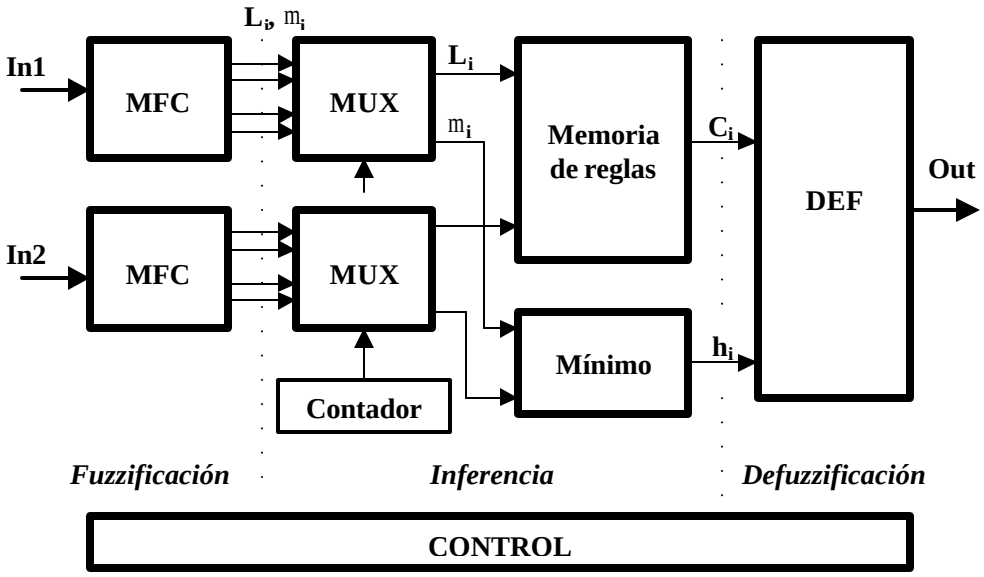

Figura 3
Diagrama en bloques de la arquitectura.

La misma consta de los siguientes bloques:
(a) Circuitos generadores de funciones de pertenencia (MFC).
(b) Arreglo de multiplexores (MUX).
(c) Memoria de reglas.
(d) Bloque de obtención del valor mínimo.
(e) Etapa de defuzzificación (DEF).
(f) Bloque de control.

Operación general

Para cada valor de las entradas, los circuitos generadores de funciones de pertenencias (MFC), encargados de la fuzzificación de las entradas, suministran tantos pares etiqueta-grado de pertenencia (L_i,μ_i) como grado de solapamiento se haya previsto en el sistema (dos en este caso). La siguiente etapa consiste en procesar secuencialmente cada una de las reglas que se activen, utilizándose para ello un arreglo de multiplexores controlados por un contador (el cual realmente forma parte del bloque de control).

En cada ciclo del contador los grados de pertenencia μ_i de cada una de las entradas son combinados a través del operador mínimo para calcular el grado de activación de la regla (h_i), mientras que las respectivas etiquetas de los antecedentes direccionan la posición de memoria que contiene su correspondiente consecuente C_i. De esta forma, en cada ciclo del contador se accede a la memoria de reglas y se procesa la regla activada.

Finalmente se encuentra la etapa de defuzzificación encargada de procesar los consecuentes de cada regla (C_i) con sus diferentes grados de activación (h_i) según el método simplificado que se utilice, destacándose que el proceso de defuzzificación se va realizando simultáneamente con el procesamiento de las reglas que se activan.

Una explicación mas detallada de las diferentes etapas se realiza a continuación.

- **Etapa de fuzzificación**

Los circuitos generadores de funciones de pertenencia (MFC) encargados de la fuzzificación de las entradas proporcionan los grados de pertenencia μ_i de un elemento del universo de discurso de las entradas a los conjuntos difusos representados por su etiqueta lingüística L_i.

Su implementación microelectrónica puede realizarse de dos formas: mediante almacenamiento en memoria o mediante cálculo aritmético.

MFC mediante almacenamiento en memoria

La primera variante consiste en almacenar en una memoria los valores de las etiquetas y de los distintos grados de pertenencia. La fuzzificación se realiza direccionando dicha memoria con la palabra binaria correspondiente al valor de la entrada, tal como se ilustra en la figura 4.

En dicha figura, para una entrada de valor $X1$ corresponde una etiqueta L_a con grado de pertenencia μ_a y una etiqueta L_b con grado de pertenencia μ_b. Por lo tanto en la localización de memoria correspondiente al valor binario de la entrada $X1$ deben almacenarse estos valores. Obsérvese que dado que la codificación de las etiquetas es consecutiva, solo es necesario almacenar en memoria una de ellas.

Nótese que el tamaño de esta memoria depende de los niveles de discretización de las entradas, así como del número de bits utilizado para codificar el grado de pertenencia, así como los números de bits necesarios para la codificación de las etiquetas. Por ejemplo, si la resolución de las entradas es de 6 bit, se codifica el grado de pertenencia con 5 bit y se utilizan 3 funciones de pertenencia, la memoria de antecedentes requerida en cada una de las entradas será de $64 \cdot 12$ bit [$2^6 \cdot (2 + 5 + 5)$].

La limitación fundamental de esta variante de fuzzificación se encuentra en el crecimiento exponencial de la memoria a medida que aumenta el número de bits de resolución de las entradas y de los grados de pertenencia. Sus ventajas radican en que puede representarse cualquier tipo de función de pertenencia así como en que la velocidad de la fuzzificación se reduce a un solo acceso a memoria.

- **MFC mediante cálculo aritmético**

La fuzzificación mediante cálculo aritmético consiste en ir calculando el grado de pertenencia de los antecedentes. Este método tiene la restricción de utilizar solo funciones de pertenencia de formas triangulares y normalizadas (en cualquier punto la suma de sus grados de pertenencia será igual a la unidad), almacenándose en una memoria (común para todas las entradas) solo los puntos notables (intercepto y pendiente) de las diferentes

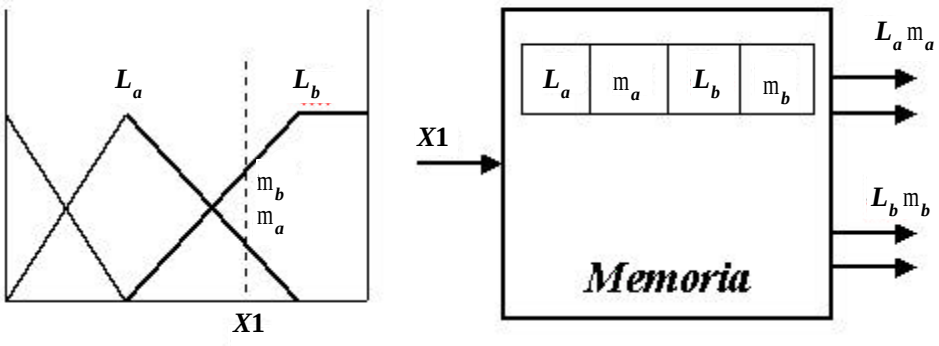

Figura 4
MFC mediante antecedentes almacenados en memoria

porciones de líneas rectas de los antecedentes. Luego, mediante un circuito aritmético (uno para cada entrada) se resuelve la ecuación de la recta correspondiente para obtener el grado de pertenencia.

La figura 5 ilustra el proceso descrito anteriormente. Un contador va recorriendo la memoria donde se almacenan los valores de los puntos de intersección (interceptos) y las pendientes de cada tramo de recta de cada una de las entradas, y los circuitos aritméticos van calculando y almacenando los grados de pertenencia mientras el valor de la entrada sea mayor que el valor del intercepto. De esta forma, el último valor calculado se corresponderá con el grado de pertenencia correspondiente y la salida del contador con la etiqueta de la función de pertenencia. Una vez obtenido uno de los grados de pertenencia, el otro se calcula inmediatamente a través de su complemento.

Esta variante de fuzzificación permite disminuir el área del circuito resultante con relación a la variante anterior en aquellos casos en que se requieran varias entradas con niveles de discretización elevados (8 o más bits).

- **Etapa de inferencia**

La etapa de inferencia está compuesta por un arreglo de multiplexores controlado por un contador (circuito de selección de reglas activas), además de un circuito para la obtención del valor mínimo, el cual sirve para obtener el grado de activación de la regla h_i. También forma parte del mismo una memoria de reglas que almacena los parámetros que definen los consecuentes C_i de las reglas.

En cada ciclo del contador los multiplexores ofrecen a su salida un par etiqueta-grado de pertenencia (L_i, μ_i). La combinación de las etiquetas de salidas de cada uno de los multiplexores se utiliza para el direccionado de la memoria de reglas, de donde se obtiene el correspondiente consecuente C_i. Simultáneamente, a la salida del bloque MINIMO se obtiene el grado de activación de la regla h_i, resultante del valor mínimo entre los grados de pertenencia μ_i de las entradas

Nótese que el número de reglas potencialmente activas queda determinado por el grado de solapamiento de los conjuntos difusos de las entradas y el número de entradas del sistema de inferencia.

- **Etapa de defuzzificación**

Al igual que ocurre con la etapa de fuzzificación, también existen alternativas de implementación de la etapa de defuzzificación. La salida del sistema de inferencia se puede realizar mediante diferentes métodos de defuzzificación simplificados, algunos de los cuales fueron citados con anterioridad. Entre estos métodos se encuentra el de la media difusa (FM), que consiste en elegir como conclusión parcial de cada regla el punto de máxima pertenencia del conjunto difuso de salida.

Esta se obtiene como la suma de las conclusiones parciales de las diferentes reglas C_i ponderada por los grados de activación h_i de las mismas, según la expresión (3).

$$y = \frac{\sum C_i \times h_i}{\sum h_i} \qquad ...(3)$$

Este método es el más simple de implementar entre todos los métodos simplificados al requerir solo una etapa de multiplicación, tal como se puede apreciar en el esquema circuital de la figura 7. Los restantes métodos soportados por la arquitectura (QM, YM y CoSm) incorporan un segundo multiplicador.

A medida que se realiza la inferencia se va accediendo a la memoria de reglas para obtener su consecuente C_i y se evalúa el grado de activación de la regla h_i, se introducen ambos valores al circuito de la figura 6, el cual cuantifica la expresión (3) y almacena temporalmente los resultados del numerador y denominador de dicha expresión. Al concluir el proceso de inferencia (evaluación de todas las reglas activas) solo será necesario una operación de división para obtener el valor defuzzificado de la salida.

- **Bloque de control**

Este bloque es capaz de generar todas las señales que permiten controlar secuencialmente las operaciones de las diferentes etapas, como por ejemplo, el contador que gobierna los multiplexores, un segundo contador para el acceso a la memoria de puntos notables en caso de realizar la fuzzificación mediante cálculo aritmético, etcétera.

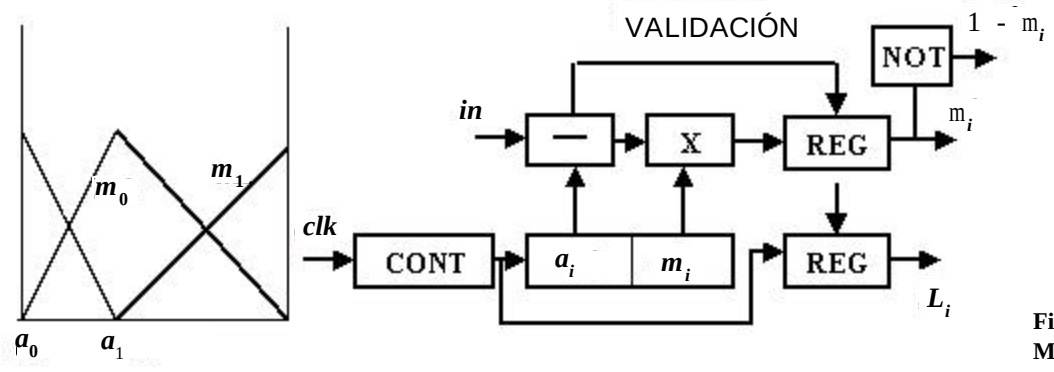

Figura 5
MFC mediante cálculo aritmético.

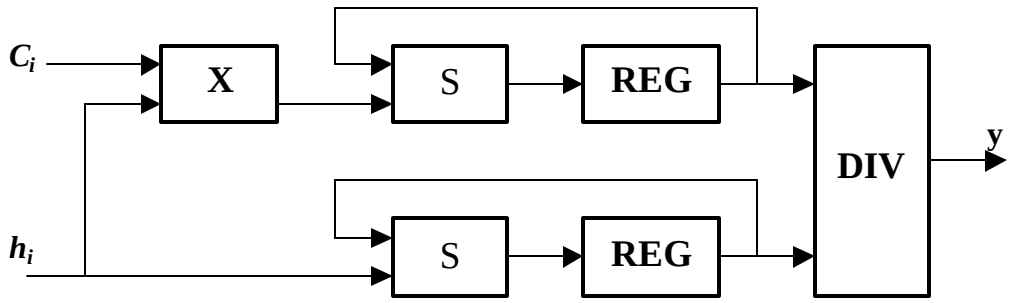

Figura 6
Esquema circuital de un defuzzificador Fuzzy Mean.

Una característica muy importante de esta arquitectura es que incluye también etapas intermedias de *pipeline* que permiten simultanear los procesos de fuzzificación, inferencia y defuzzificación, reduciendo así el tiempo total para obtener la salida del sistema de inferencia. De esta forma, si $T1$, $T2$ y $T3$ son los tiempos consumidos por cada una de las etapas, el tiempo total para la realización de una inferencia será el valor máximo entre los mismos en lugar de su suma.

HERRAMIENTAS DE DESARROLLO DE SISTEMAS DIFUSOS

Un elemento de trascendencia vital a la hora de acometer el desarrollo de un sistema de inferencia difuso, independientemente de que sea realizado mediante software o hardware, consiste en la disponibilidad de herramientas de CAD que permitan simplificar las diferentes etapas de diseño y desarrollo de este.

Existen múltiples herramientas de CAD para el desarrollo de sistemas difusos entre las que se encuentran Matlab, FuzzyTECH, FIDE, MicroFPL y Xfuzzy.[5] Las mismas permiten diferentes facilidades que van desde la descripción del sistema difuso (especificación de los conjuntos y operadores difusos y establecimiento de la base de reglas), la simulación y hasta la síntesis del sistema de inferencia.

Sin embargo, la gran mayoría de las herramientas existentes son solo capaces de realizar una síntesis software del sistema de inferencia (usualmente mediante la generación de codigo C). La principal característica que distingue al entorno de desarrollo Xfuzzy[9] de los restantes es su capacidad de realizar procesos de síntesis hardware. Una de sus opciones de síntesis hardware consiste precisamente en la generación de código VHDL (Very High Speed Hardware Description Language) sintetizable el cual soporta las diferentes opciones de la arquitectura descrita en la sección anterior.

Realmente Xfuzzy, desarrollado por especialistas del Instituto de Microelectrónica de Sevilla y de libre distribución (http://www.imse.cnm.es/xfuzzy/) es un entorno de desarrollo que agrupa a varias herramientas. Una de estas herramientas, xfVHDL,[10] es precisamente la encargada de facilitar la síntesis hardware en correspondencia con la arquitectura expuesta.

Una vez especificado el sistema difuso, se puede invocar a xfVHDL y se genera automáticamente un grupo de ficheros que contienen las descripciones VHDL de los diferentes bloques de la arquitectura, por lo que los mismos pueden ser posteriormente procesados por una herramienta de síntesis de dispositivos hardware para obtener la realización microelectrónica del sistema de inferencia.

Existen experiencias muy interesantes de realizaciones de controladores difusos, tanto software como de codiseño hardware/software (donde el sistema de inferencia se implementa en hardware mediante un FPGA), desarrolladas con la ayuda de Xfuzzy[11,12] y que permiten avalar las muy buenas prestaciones de la arquitectura expuesta en este trabajo.

CONCLUSIONES

Los sistemas de inferencia difusos pueden ser implementados mediante software o mediante hardware. Las implementaciones software se caracterizan por su gran versatilidad en cuanto a la elección de los conjuntos y operadores difusos pero están limitadas en velocidad debido a la ejecución secuencial de los programas que le dan soporte.

Las implementaciones hardware posibilitan una mejor representación del paralelismo propio de los sistemas difusos por lo cual presentan velocidades de inferencia muy superiores a las realizaciones software. Sin embargo, carecen de la flexibilidad de aquellas e, incluso, su implementación eficiente en términos de área obliga a la utilización de restricciones en las arquitecturas.

Se han expuesto los distintos bloques de una arquitectura que presenta muy buenas prestaciones a un costo relativamente bajo, basada en el procesamiento de reglas activas, la limitación a dos del grado de solapamiento de las funciones de pertenencia de las entradas y la utilización de métodos de defuzzificación simplificados. Para la materialización de la misma se cuenta con el entorno de desarrollo Xfuzzy el cual posibilita la generación de código VHDL que representa dicha arquitectura.

Los resultados de las implementaciones hardware de sistemas de inferencia difusos basados en la arquitectura descrita avalan las muy buenas prestaciones de esta.

REFERENCIAS

1. **PASSINO, K. M. & S. YURKOVICH:** *Fuzzy Control*, Addison-Wesley, 1998.
2. **MAMDANI, E. H.:** "Applications of Fuzzy Algorithm for Control of a Simple Dinamic Plant", *Proc. IEE*, Vol. 12, 1974.
3. **YEN, J.; R. LANGARI & L. ZADEH:** "Industrial Applications of Fuzy Logic and Intelligent Systems", *Ed. IEEE Press*, 1995
4. **REZNIK, L.:** *Fuzzy Controllers*, Ed. Newness, 1997.
5. **BATURONE, I.** *et al*.**:** "Microelectronic Design of Fuzzy Logic-Based Systems", *CRC Press*, 2000.
6. **IKEDA, H** *et al*.**:** "A Fuzzy Inference Coprocesor Using a Flexible Active-Rule-Driven Architecture", Proc. *IEEE ICFS'92*, pp. 537-544, San Diego, 1992.
7. **BATURONE, I.** *et.al*.**:** "Implementations of Inference/ Defuzzification Methods Via Continuous-Time Analog Circuits", *Proc. IFSA World Congress*, pp. 623-626, Sao Paulo, July, 1995.
8. **SÁNCHEZ-SOLANO, S.** *et al*.**:** "Design and Applications of Digital Fuzzy Controllers", *Proc. Sixth IEEE International Conference on Fuzzy Systems* (FUZZ-IEEE'97), Vol. 2, pp. 869-874, Barcelona, July, 1997.
9. **LÓPEZ, D. R.** *et al*.**:** "Xfuzzy: A Design Environment for Fuzzy Systems", *Proc Seventh IEEE International Conference on Fuzzy Systems*, pp. 1060-1065, Anchorage, May, 1998.
10. **LAGO, E.** *et al*.**:** "xfVHDL: A Tool for the Synthesis of Fuzzy Logic Controllers", *Proc. Design, Automation and Test in Europe* (DATE´98), pp. 102-107, Paris, February, 1998.
11. **CABRERA, A.** *et al*.**:** "Development of Level Controllers Based on Fuzzy Logic", *Proc. First ICSC-NAISO International Congress on Neuro-Fuzzy Technologies*, La Habana, January, 2002.
12. **CABRERA, A.** *et al*.**:** "Hardware/Software Codesign Methodology for Fuzzy Controllers Implementation", *IEEE International Conference on Fuzzy Systems* (FUZZ-IEEE2002), Honolulu, May, 2002.

AUTORES

Alejandro José Cabrera Sarmiento
Ingeniero Electricista, Master en Ciencias en Sistemas Digitales, Profesor Auxiliar. Línea de investigación: Sistemas digitales de alto nivel.
Correo electrónico: alex@electrica.cujae.edu.cu

Santiago Sánchez-Solano
Licenciado en Física Electrónica, Doctor en Ciencias Físicas, Investigador Titular. Línea de investigación: Sistemas digitales de alto nivel.
Correo electrónico: santiago@imse.cnm.es

Carlos Jesús Jiménez
Licenciado en Física Electrónica, Doctor en Ciencias Físicas; Profesor Titular. Línea de investigación: Sistemas digitales de alto nivel.

Ángel Barriga
Licenciado en Física Electrónica, Doctor en Ciencias Físicas, Profesor Titular. Línea de investigación: Sistemas digitales de alto nivel.

Iluminada Baturone
Licenciada en Física Electrónica, Doctora en Ciencias Físicas, Profesora Titular. Línea de investigación: Sistemas digitales de alto nivel.

IX Workshop IBERCHIP, IWS-2003

26-28 de Marzo de 2003
Hotel Palco - Palacio de Convenciones
La Habana, Cuba