Consideraciones metodológicas para la síntesis automática de procesadores difusos en ANSI-C

Edgar Forero[1]
Edgar Garcia[2]
Miguel Melgarejo[3]

Resumen

Este documento presenta algunas consideraciones metodológicas para desarrollar una herramienta que genere automáticamente especificaciones de procesadores difusos, las cuales puedan ser implementadas en dispositivos de hardware empotrado. La descripción de las consideraciones se hace de manera general de tal modo que serviría de guía para desarrollar herramientas de síntesis automática de procesadores difusos sobre diferentes tecnologías hardware. Como caso específico se presenta una herramienta automática para generar código en lenguaje de programación ANSI C de modelos de procesadores difusos especificados en una herramienta de alto nivel. Además, se presentan algunos resultados prácticos obtenidos por medio de esta herramienta.

Palabras clave: lógica difusa, procesadores difusos, hardware difuso, diseño electrónico automático, diseño asistido por computador.

Methodological considerations for automatic synthesis of fuzzy processors in ANSI-C

Abstract

This paper presents some methodological considerations for developing an automatic tool that generates specifications of fuzzy processors. Our considerations are presented generally so that they could guide the development of automatic tools for synthesizing fuzzy processors over different hardware technologies. Additionally, It is presented a particular case of an automatic tool that synthesizes fuzzy processors specified in a high-level description tool. Some practical results obtained by using this tool are presented and discussed.

Key words: Fuzzy Logic, Fuzzy processors, Fuzzy hardware, Electronic Design Automation, Computer Assisted Design.

1. INTRODUCCIÓN

Los sistemas difusos son una técnica de inteligencia computacional que se ha venido desarrollando por más de 40 años. En especial, la lógica difusa se utiliza en la industria desde hace varios años con resultados satisfactorios [1,2,3]. Es común que en el contexto de la lógica difusa se utilicen plataformas de cómputo especializadas para realizar las implementaciones, dando lugar al concepto de procesador difuso [3,4]. Los procesadores difusos están presentes en aplicaciones donde el consumo de potencia, la velocidad de procesamiento y el espacio ocupado son parámetros importantes [5].

En general, se utilizan plataformas hardware que cumplen con las especificaciones del problema. Sin embargo, la implementación de los procesadores difusos en estas es un proceso lento y costoso si se aborda manualmente [4,12-15]. Esta particularidad se debe a que el proceso de diseño e implementación requiere de desarrolladores con conocimientos en lógica difusa y que además sean especialistas en desarrollo de hardware.

Lo anterior ocasiona que en aplicaciones, donde implementar un sistema difuso es una estrategia viable, se decida utilizar otra metodología, pues los costos se incrementan al necesitar mano de obra especializada y al elevar el tiempo de especificación de los sistemas. Al encontrarse actualmente algunas herramientas para la descripción de sistemas difusos [4], es evidente la posibilidad de desarrollar una herramienta que interprete los datos que se obtienen al realizar descripciones de sistemas difusos en alto nivel. Además esta interpretación permitiría realizar la generación automática de una especificación en un lenguaje de programación y que esta fuese llevada a un dispositivo hardware.

Por tanto, este trabajo busca presentar algunas consideraciones metodológicas para la síntesis automática de procesadores difusos en lenguaje ANSI C. En la formulación de estas consideraciones se toma como referencia el esquema general de un sistema difuso para proponer un modelo computacional de implementación. Luego, se describe cómo se puede sintetizar este modelo computacional en una especificación ANSI C a partir de un sistema difuso modelado por medio de una herramienta de diseño asistido.

En la actualidad es posible encontrar algunas herramientas que realizan síntesis automática de

[1] C'Ingeniero electrónico Universidad Distrital FJC
[2] Ingeniero electrónico Universidad Distrital FJC
[3] Profesor asistente, Facultad de Ingeniería, Investigador, LAMIC Universidad Distrital FJC

Artículo recibido en Junio de 2008, aprobado para publicación en Agosto de 2008

procesadores difusos [4,6]. Sin embargo estas fueron desarrolladas considerando casos específicos de plataformas hardware y no se aborda el problema con una propuesta general que pueda ser utilizada en la síntesis de procesadores difusos sobre diferentes plataformas hardware.

El artículo está organizado como sigue: En la sección 2 se presenta el modelo computacional general para realizar la descripción de de los tres componentes de un sistema difuso. En la sección 3 se presentan las consideraciones metodológicas para generar automáticamente los componentes descritos en la sección 2. En la sección 4, como caso particular, se presenta una herramienta para la síntesis automática de sistemas de inferencia difusa en lenguaje ANSI C. Luego en la sección 5 se muestran y discuten algunos resultados prácticos obtenidos con la herramienta. Finalmente se dan algunas conclusiones y se comenta el trabajo en desarrollo en la sección 6.

2. MODELO COMPUTACIONAL

La estructura de un sistema difuso con fusificación y defusificación se presenta en la figura 1 [1-4]. El fusificador mapea valores puntuales en conjuntos difusos, luego el motor de inferencia se encarga de ejecutar una implicación de los conjuntos de entrada sobre los conjuntos de salida a partir de una base de reglas predefinida. Finalmente, el defusificador obtiene un valor puntual de salida a partir de la agregación de los conjuntos resultantes de la implicación.

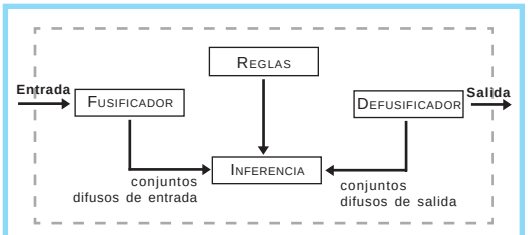

Figura 1. Esquema general de un sistema difuso tipo 1.

2.1. Fusificación

Tal como se muestra en la Figura 2, en el proceso de fusificación se mapean valores puntuales en los conjuntos difusos que describen los universos discursos de las entradas. Las funciones de pertenencia consideradas en este modelo computacional serán de tipo trapezoidal, mientras que la fusificación será de tipo singleton. Esto debido a que la combinación del los mismos conlleva a los más bajos costos de implementación independientemente de la plataforma de cómputo [4].

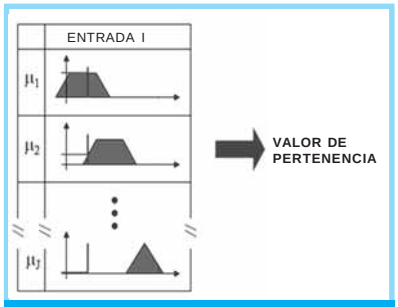

Figura 2. Fusificación singleton empleada en el modelo computacional.

Para la fusificación es necesario contar con las funciones de pertenencia de las entradas. Calcular una función de pertenencia tipo trapezoidal demanda el conocimiento de cuatro parámetros. Estos parámetros corresponden a los mostrados en la Figura 3. La función de pertenencia de tipo triangular es un caso especial de la función de pertenencia de tipo trapezoidal en donde los parámetros P2 y P3 son iguales.

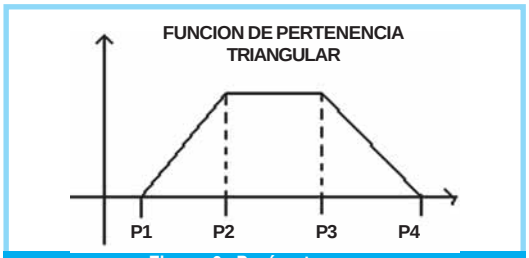

Figura 3. Parámetros que describen la función trapezoidal.

En el cálculo de los segmentos de recta que forman la función de pertenencia triangular se utilizan dos expresiones:

$$\mu_A(x) = \frac{-1}{p_4 - p_3}(x - p_3) \quad (1)$$

$$\mu_A(x) = \frac{1}{p_2 - p_1}(x - p_2) \quad (2)$$

donde x es representa la variable de entrada a fusificar.

Estas expresiones se evalúan dependiendo del intervalo en que se encuentre la variable con respecto a los cuatro parámetros que definen la función. Si el valor de la variable se encuentra entre P1 y P2, el valor de la función de pertenencia se calcula mediante la ecuación 1. Si el valor de la variable dentro del universo de discurso está entre P2 y P3, la función de pertenencia toma el valor de 1. Si el valor de la variable se encuentra entre P3 y P4, se utiliza la ecuación 2 para realizar el cálculo de la función de pertenencia en ese intervalo. En cualquier otro caso el valor de la función es cero.

Por tanto, computacionalmente hablando, la fusificación singleton consiste en ubicar el valor puntual de entrada dentro del universo de discurso y aplicar luego el cálculo de la función de pertenencia para este punto a través de una rutina [5].

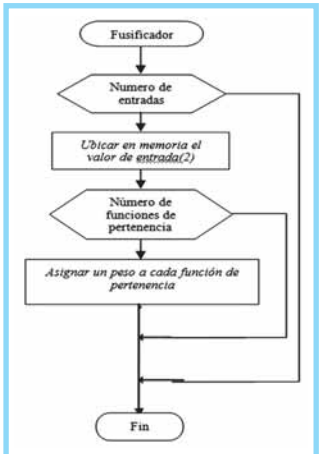

Figura 4. Algoritmo para la realización de un fusificador singlenton.

El proceso de fusificación, por el método descrito anteriormente, se aplica a todas las funciones de pertenencia de todos los universos de discurso de las entradas, tal como se describe en el diagrama de flujo presentado en la Figura 4. De esta forma se obtienen todos los valores de pertenencia con los cuales operará el motor de inferencia que es la siguiente etapa en el modelo computacional.

2.2. Motor de inferencia

El motor de inferencia se desarrolla a partir de las características encontradas en la literatura para motores de inferencia tipo Mandami [1-4].

Para modelar el motor de inferencia se propone un esquema basado en dos etapas: un proceso de implicación y un proceso de agregación. De esta manera, a la salida se obtiene un único conjunto difuso. Para la implicación y la agregación se emplea un algoritmo basado en sentencias condicionales "si" y bucles "para".

En la implicación se direccionan los valores de pertenencia almacenados según lo especificado en la base de reglas, es decir, a partir de los antecedentes, consecuentes y conectores de cada regla se realiza la operación de mínimo o máximo según se requiera[2]. Además se necesita saber el grado de activación de cada función de pertenencia de acuerdo con la entrada del sistema.

En la agregación se toman los conjuntos resultantes de evaluar cada una de las reglas y se realiza una operación máximo entre ellos. Los conjuntos involucrados en esta operación dependen de las reglas que se activen en el proceso de implicación. El resultado de este proceso es un único conjunto de salida, el cual es la entrada al proceso de defusificación. La figura 5 muestra de manera resumida este proceso.

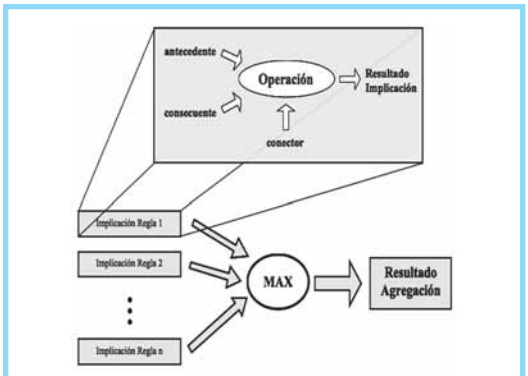

Figura 5. Representación del proceso de implicación y agregación.

2.3. Defusificación

La defusificación es el mapeo de un conjunto difuso a un valor puntual [1]. Existen varios métodos para realizar la defusificación en un sistema difuso [1-4].

En este modelo computacional se utiliza el método del centroide: Este método consiste en tomar la sumatoria de los valores del universo de discurso discretizado ponderados por el valor de pertenencia en cada uno de esos puntos. Este valor se divide entre la sumatoria del valor de pertenencia en los puntos del universo de discurso discretizado [1,2]. El conjunto difuso que se opera en el proceso de defusificación es el resultante de realizar la agregación de reglas en el proceso de inferencia. Se realiza un algoritmo, que se presenta en la figura 6, para evaluar estos valores siguiendo

$$Y = \frac{\sum(\mu(x_i)\cdot x_i)}{\sum\mu(x_i)} \qquad (3)$$

Donde x_i es el punto evaluado dentro del universo de discurso de la salida y $m(x_i)$ es el valor de pertenencia del conjunto difuso resultante.

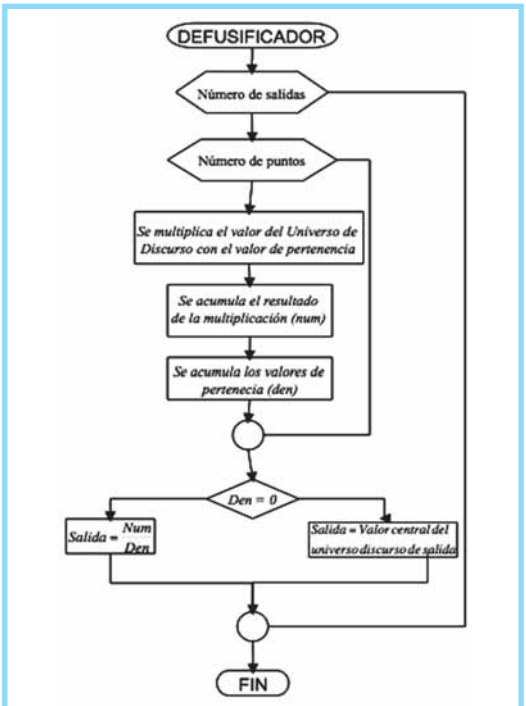

Figura 6. Algoritmo de defusificación.

3. CONSIDERACIONES METODOLÓGICAS PARA LA SÍNTESIS AUTOMÁTICA DE PROCESADORES DIFUSOS

3.1. Consideraciones generales

Para generar automáticamente especificaciones de sistemas difusos en un lenguaje de programación, se propone inicialmente especificar una estructura general que represente el modelo computacional de un sistema difuso. La estructura del modelo computacional de los módulos para la fusificación, inferencia y defusificación se considera como un

modelo estático. Esto es general e independiente para cualquier sistema, pues siempre se tiene la misma estructura que va a interpretar y procesar los datos para generar una especificación en un lenguaje de programación.

Se presenta dinamismo al considerar los sistemas de entrada, puesto que las características particulares de cada sistema difuso a trabajar son diferentes. Esto no cambia el modelo computacional como tal, sin embargo, dependiendo de estas características el desempeño en tiempo y ocupación de memoria del modelo varían. Al tener este modelo, es necesario especificar las variables que se necesitan en cada etapa.

Inicialmente se utiliza una herramienta de descripción de sistemas difusos de alto nivel para modelar el sistema de interés. Es necesario que esta herramienta permita la descripción de los sistemas de acuerdo con las características y limitaciones que hayan sido establecidas. De esta descripción se obtiene una especificación que contiene todas las características del sistema difuso descrito y los datos necesarios para realizar la síntesis automática. Las variables especificadas dependen de los datos adquiridos del modelo desarrollado en la herramienta de alto nivel y los tipos de datos se conocen dependiendo de las características del sistema difuso que se esté trabajando.

Para el proceso de generación automática que se plantea, se debe en primer lugar extraer los parámetros de la especificación en alto nivel de manera correcta. Para esto se necesita interpretar la estructura de la especificación y ubicar todos los parámetros necesarios. Luego se deben generar todas las variables que se van a utilizar en el algoritmo. Estas variables definen el sistema difuso dando características de las entradas y salidas, de los operadores difusos utilizados, del tipo de funciones de pertenencia, de la base de reglas, del tipo de fusificación, etc. El tamaño y tipo de estas variables se especifica de acuerdo con los valores extraídos en la especificación de alto nivel. Un esquema que representa las etapas y los elementos necesarios se presenta en la figura 7.

Cuando se consideran lenguajes de programación de plataformas hardware, es necesario realizar la generación de sentencias que sean características del lenguaje y variables que se utilizan en el cuerpo de un programa estándar, pero que no son propias del procesador que se quiere generar. Por ejemplo, los archivos cabecera necesarios para el uso de bibliotecas, variables de configuración de puertos, etc. Esta consideración varía dependiendo del tipo de hardware al que se enfoque el desarrollo. Inclusive, si son plataformas con un mismo lenguaje de programación, en general la configuración del dispositivo cambia dependiendo del fabricante [16,17].

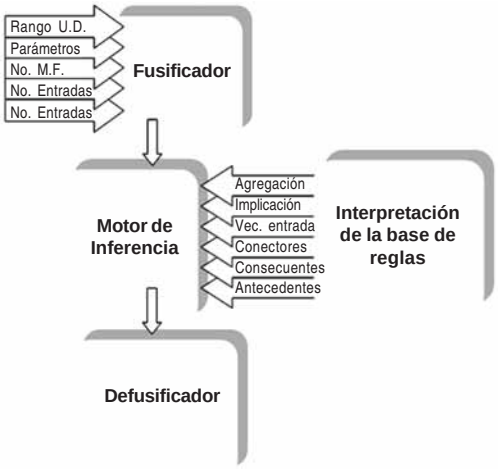

Figura 7. Componentes generados automáticamente y datos necesarios.

3.2. Propuesta de síntesis automática

Aquí se propone realizar la especificación del fusificador de la siguiente manera: Se generan las variables necesarias para calcular las funciones de pertenencia de entradas y salidas del sistema difuso. Luego se procesan estas variables y se generan las funciones de pertenencia mediante el modelo presentado en la segunda sección de este documento. Para esto se utiliza un módulo generador de funciones de pertenencia para las entradas y las salidas.

El módulo de generación de funciones de pertenencia trabaja como sigue: Inicialmente se toman los valores referentes a las entradas y las salidas del sistema difuso. Estos valores son para cada entrada o salida, universo de discurso, número de funciones de pertenencia y parámetros de cada una de estas funciones. El resultado del generador es un código que calcula las funciones de pertenencia de acuerdo con los parámetros de entrada. Para realizar la síntesis de las funciones de pertenencia de la entrada se especifican dos rutinas. Una exterior que referencia al número de entrada que se está trabajando y otra interior que referencia la función de pertenencia que se está calculando. Lo mismo se hace para generar las funciones de pertenencia en las variables de salida.

En la síntesis del motor de inferencia se sigue el método descrito en la sección dos de este documento. Para la generación de este módulo no es necesario conocer la arquitectura del dispositivo hardware en donde va a ser empotrado el procesador. Esto debido a que se busca un modelo general de generación automática y el entrar en detalles de la arquitectura del hardware particulariza el modelo. El motor de inferencia se reduce a la generación de un código que siga el modelo planteado. Este modelo se basa en sentencias de bajo nivel encontradas en los lenguajes de programación utilizados en varías familias de procesadores [16,17]. En la defusificación se genera

automáticamente el algoritmo que se presenta en la sección dos para el cálculo del centroide.

4. HERRAMIENTAS PARA LA GENERACIÓN AUTOMÁTICA DE SISTEMAS DIFUSOS EN ANSI-C

Como caso específico de aplicación de las consideraciones metodológicas descritas, se describe una herramienta de síntesis automática de procesadores difusos en ANSI C, considerando lo siguiente: una caja de herramientas para sistemas difusos se utiliza como interfase de especificación de sistemas difusos de alto nivel. Esta plataforma igualmente se utiliza para hacer la especificación de la herramienta de generación automática. Finalmente se escoge generar código ANSI C, el cual es un lenguaje de programación utilizado en varias tecnologías que se encuentran en el mercado como micro controladores y procesadores digitales de señales [11,16].

Para realizar esto, inicialmente se realiza la generación automática de sistemas difusos en lenguaje de especificación alto nivel. Luego se utilizó el modelo computacional descrito para generar especificaciones en C. La generación automática de los sistemas difusos, se realizó utilizando comandos especiales de la plataforma empleada para este propósito [8]. La tabla 1 muestra las características de la herramienta. Las restricciones del número de entradas, salidas y reglas dependen del hardware donde se proyecte la implementación.

Tabla I. Características de la herramienta de generación de código ANSI-C.

Fusificación	singlenton	**Numero entradas**	Sin restriccion
Numero de reglas	Sin restriccion	**Numero de salidas**	Sin restriccion
Resolución	32 bits	**Tipo de funciones**	Trapezoidales
defusificacion	Centro de gravedad	**Implicacion**	Mandami max-min
Tipo de datos	Float		

5. RESULTADOS

Para validar la herramienta se especificaron algunos sistemas difusos en la caja de herramientas de sistemas difusos de la plataforma de desarrollo.

Luego de esto se utilizó la herramienta para generar la especificación C de tales sistemas. Finalmente, se evaluaron los sistemas, el especificado en la caja de herramientas y el generado automáticamente comparándose los resultados.

5.1. Metodología de la validación

Se realizaron algunos experimentos para validar la herramienta desarrollada con las consideraciones propuestas en este trabajo. Cada prueba se realizó de la siguiente manera:

1) Se describe un sistema difuso con la herramienta de especificación de alto nivel.
2) Se genera automáticamente el modelo en C con la herramienta desarrollada.
3) Se mide el tiempo de síntesis para 1000 generaciones del mismo sistema.
4) Se evalúan los dos sistemas difusos en distintos puntos sobre todo el universo discurso. Para el caso en que los sistemas de interés tengan más de dos entradas, solo se toman dos de ellas para la realización del proceso de validación.
5) Luego se normaliza el universo de discurso [UDN] de salida del sistema difuso al intervalo [0 1].
6) Finalmente se calcula el error cuadrático medio [ECM] de los valores de salida entregados por la herramienta para generar la especificación C con respecto a los valores de salida entregados por la caja de herramientas de sistemas difusos. Para los casos de interés, el ECM se halló mediante (5).

$$ECM = \frac{1}{n_2 n_1} \sum_{i_2=1}^{n_2} \sum_{i_1=1}^{n_1} \left(X_f(i_1, i_2) - X_c(i_1, i_2) \right)^2 \qquad (4)$$

donde, $X_f(i_1,i_2)$ es el valor de salida del sistema difuso realizado en la caja de herramientas de sistemas difusos para los valores de i_1,i_2, $X_c(i_1,i_2)$ es el valor de salida del sistema difuso realizado con la herramienta de síntesis automática para los valores de i_1,i_2 y n_1,n_2son el número de puntos evaluados para la entrada 1 y 2 respectivamente.

5.2. Resultados

La tabla 2 muestra el error cuadrático medio, el error cuadrático medio con el universo de discurso de la salida normalizado y los tiempos de síntesis de algunos sistemas considerados.

Al considerar cada aplicación referenciada para validar la herramienta se observa que el error obtenido con respecto a la simulación realizada no es significativo. Esto se constata si se compara una desviación de la magnitud del error en el valor de salida del sistema.

El tiempo de síntesis varía de sistema en sistema pues depende del número de reglas y de funciones de pertenencia de todo el sistema. Además el

Tabla II. Resultados validación de la herramienta, donde (a). Sistema de posicionamiento de un vehículo [1], (b) Controlador de un generador síncrono [9], (c) Predictor de serie de tiempo caótica [1], (d) Tratamiento de aguas residuales [10].

Sis.	No. entradas	No. salidas	ECM reglas	ECM	UDM	Tiempo de síntesis del sistema	
						Valor Medio[s]	Desviación estándar[s]
a	2	1	27	2.23E-10	3.49E-14	0.022262	0.0038
b	2	1	49	2.105E-13	2.33E-14	0.0273895	0.0062
c	4	1	57	3.83E-7	1.288E-7	0.0344045	0.0046
d	2	2	25	1.78E-10	1.306E-15	0.022798	0.0047

computador donde se realiza la generación también influye. El tiempo de generación presentado en la tabla 2 fue el resultado de generar mil veces el mismo sistema, tomar el tiempo implementado y dividir el resultado entre mil. Se realizó de esta manera con el fin de tener una validación estadística del tiempo de síntesis.

6. CONCLUSIONES

Se presentaron algunas consideraciones metodológicas acerca del desarrollo de herramientas para la síntesis automática de sistemas difusos a partir de una descripción en un lenguaje de alto nivel. Se tuvieron en cuenta estas consideraciones para desarrollar una herramienta de generación automática de procesadores difusos en lenguaje de programación ANSI C.

Con la creación de una herramienta para la síntesis automática de sistemas difusos en procesadores comerciales, se reduce de manera significativa el tiempo, la complejidad y los costos en la implementación de este tipo de sistemas. Es por esto que se hace posible el uso de la herramienta en aplicaciones industriales y académicas.

Actualmente, la herramienta se está extendiendo para generar código C adaptado a la arquitectura varias arquitecturas de procesadores comerciales de bajo costo[16]. El propósito a largo plazo es integrar la mayor cantidad de arquitecturas posibles en un único entorno de desarrollo de síntesis automática.

REFERENCIAS BIBLIOGRÁFICAS

[1] L. X. Wang, *A course in Fuzzy Systems and Control*, Prentice Hall, 1997.

[2] R. Babuska, *Fuzzy modeling for control*, Kluwer academics, 1998.

[3] J. Mendel, *Uncertain rule based fuzzy logic systems*, Prentice Hall, 2000.

[4] A. Baturone, S. Barriga, S.Sanchez, C. Jimenez and D. Lopez, *Microelectronic Design of Fuzzy Logic-Based Systems*, CRC Press, 2000

[5] M. Melgarejo y C. Peña, Implementing interval type-2 fuzzy processors, *IEEE computacional Intelligence Magazine*, Vol 2 No 1, Feb. 2007, pp 63-72.

[6] Instituto de Microelectrónica de Sevilla , XFUZZ, http://www.imse.cnm.es/Xfuzzy/Xfuzzy_3.0/xfl/packages_sp.html.

[7] C. H. Chen, *Fuzzy Logic and Neural Networks Handbook* , McGraw-Hill, 1996, pp. 23.12.

[8] A. Gaona y D. Olea, *Desarrollo de un Sistema de Inferencia Difusa Sobre FPGA*, Universidad Distrital FJC, 2003.

[9] J. Brock , "Design and implementation of a fuzzy logic-based voltage controller for voltage regulation of a synchronous generator", Technical Report, Montana State University. 2004.

[10] O. Pires et al ,"A fuzzy-logic based expert system for diagnosis and control of an integrated wastewater treatment", *Proceedings of the 2nd Mercosur Congress on Chemical Engineering – ENPROMER and 4th Mercosur Congress on Process Systems Engineering* , Rio de Janeiro, 2005

[11] M. Brand and A. Massa, *Programming embedded systems with C and GNU tools*, 2nd edition, O'Reilly, 2006.

[12] R. Rovatti, "High Speed Implementation picewise – quadratic Takagi-Sugeno System with memory saving", in proc. *IEEE Int. Conference on Fuzzy Systems, comunicaions and Computers*, pp 6451 – 6454. Athenas, 1999.

[13] R. Rovatti, M. Vittuari, "Linear and Fuzzy piecewise – linear signal processing with an extended DSP archictecture", in proc. *IEEE Int. Conference on Fuzzy System*, pp 1082 – 1087, Anchorage, 1998.

[14] A. Gaona, D. Olea and M. Melgarejo, "Sequential Fuzzy Inference system Based on distributed Arithmetic", *International simposium on computacional Inteligence for Messurement system and Alications*, Lugano, Switzrland, 2003, PP 125 – 129.

[15] A. Gaona, D. Olea and M. Melgarejo, "Distributed Arithmetic in the Desing of High Speed Hardware Fuzzy Inference System", *22nd International Conference of the North American Fuzzy Information Processing Society – NAFIPS*, 2003, p. 116 – 120.

[16] Z. Camelo, F. Tellez y M. Melgarejo," Modelo computacional de inferencia difusa para la arquitectura DSPIC33F", *·3rd IEEE Colombian Workshop on Circuits and Systems CWCAS´09*, Bogota, Colombia, 2009.

Edgar Forero

Ingeniero Electrónico de la Universidad Distrital Francisco José de Caldas (2008). Líder de la línea de tecnologías Virtuales orientadas a el desarrollo de aplicaciones para televisión digital y desarrollo de sistemas mediante el uso de inteligencia artificial en TecnoParque Colombia nodo Central. Paralelamente se desempeña como CTO de network TV, una empresa que realiza soluciones de televisión sobre plataformas WEB. **eforero@sena.edu.co**

Edgar García

Ingeniero Electrónico de la Universidad Distrital Francisco José de Caldas (2008). Se desempeña actualmente como ingeniero de automatización y control para la empresa RAYCO LTDA.

Miguel Melgarejo

Ingeniero electrónico de la Univerisdad distrital Francisco José de Caldas, Magister en Ingeniería electrónica y computadores, de la Universidad de los Andes, Bogotá Colombia. Ha sido investigador del Centro de Microelectrónica de la Universidad de los Andes, Colombia e investigador invitado del Logic Systems Laboratory de la Ecolé Polytechnique Federale de Lausanne, Suiza. Actualmente es profesor asistente de la facultad de ingeniería de la Universidad Distrital Francisco José de Caldas, Colombia e investigador del Laboratorio de Automática, Microelectrónica e Inteligencia Computacional (LAMIC) en la misma universidad. Ha publicado 45 artículos técnicos y dos capítulos de libro. **mmelgarejo@udistrital.edu.co**