Implementación de Sistemas Fuzzy Complejos sobre FPGAs

Garrigós-Guerrero F.J., Ruiz-Merino R.

Universidad Politécnica de Cartagena, Murcia, España,
javier.garrigos@upct.es, ramon.ruiz@upct.es

http://www.upct.es

Resumen. Las desventajas de las soluciones hardware dedicadas para la implementación de sistemas de inferencia fuzzy cuando se comparan con las estrategias basadas en software son principalmente la falta de flexibilidad y la complicación en el proceso de diseño. En este trabajo se presenta una arquitectura novedosa que permite la síntesis electrónica y la implementación hardware de sistemas expertos basados en conocimiento fuzzy. La definición de la arquitectura se basa en la descripción en forma de red de Petri de la base de reglas complejas, heredando de ella las características de modularidad y escalabilidad. Los componentes de nuestra arquitectura se definen entonces utilizando descripciones VHDL de alto nivel. Por ello, nuestra metodología de diseño proporciona flexibilidad, reusabilidad e independencia tanto de la tecnología electrónica como del tipo y tamaño de la aplicación, solucionando la mayoría de las limitaciones del hardware fuzzy.

1 Introducción

Desde que, en 1985, H. Watanabe y M. Togai desarrollaran el primer chip fuzzy con tecnología VLSI [1], un gran número de investigadores han dedicado sus esfuerzos al desarrollo de sistemas fuzzy en hardware, utilizando distintas soluciones, que se pueden clasificar en tres grandes grupos: extensiones fuzzy del juego de instrucciones de microprocesadores tradicionales, coprocesadores fuzzy, y hardware fuzzy dedicado. Algunos desarrollos interesantes pueden consultarse en [2].

Tradicionalmente, las soluciones propuestas para hardware fuzzy dedicado han sacrificado la riqueza expresiva del modelo en aras de una mayor simplicidad que facilite el diseño y reduzca los recursos. Sin embargo, además de mayor velocidad de cómputo, las nuevas aplicaciones necesitan sistemas fuzzy más complejos, con flexibilidad en el número de antecedentes y consecuentes de las reglas, número variable y formas arbitrarias de las funciones de pertenencia, bases de reglas jerárquicas con encadenamiento entre reglas, disponibilidad de distintos operadores de inferencia, etc. El desarrollo microelectrónico de este tipo de sistemas fuzzy complejos no puede realizarse sin la utilización de nuevas metodologías de diseño electrónico y estrategias de modelización.

Por otro lado, a pesar de ser la única alternativa viable cuando la velocidad de cómputo es elevada, las soluciones hardware en general, y las aplicaciones fuzzy no son una excepción, presentan tres dificultades principales: con frecuencia se ven limitadas por la complejidad del problema; presentan un alto tiempo y coste de desarrollo; y las soluciones adole-

cen de falta de flexibilidad una vez desarrolladas. La limitación por la complejidad de la aplicación está siendo minimizada por la creciente capacidad de los dispositivos electrónicos de aplicación específica que, junto con potentes herramientas de diseño, permiten abordar cada vez diseños más complejos, aunque para ello sea necesario mejorar la productividad del diseñador. La investigación en hardware fuzzy debe pues dirigirse, por un lado, hacia la automatización del proceso de diseño del hardware, para disminuir el tiempo de desarrollo. Por otro lado, estos nuevos diseños deben ser sencillos y fácilmente modificables, para que un sistema pueda adaptarse con rapidez y facilidad a un nuevo problema. Ello implica que han de utilizarse nuevas arquitecturas, más sencillas y flexibles, en el desarrollo de las próximas generaciones de hardware fuzzy para sistemas complejos.

En este artículo se propone una nueva arquitectura para la realización hardware de sistemas de inferencia fuzzy complejos que está basada en una descripción en forma de Red de Petri de la base de reglas, cuyas principales características son la independencia del tamaño y la complejidad de la base de reglas, y un diseño orientado a la reusabilidad.

El resto de este artículo se organiza como sigue. En la sección 2 se expone la problemática asociada a los sistemas fuzzy complejos, haciendo énfasis especial en los sistemas con encadenamientos entre reglas. En la sección 3 se aborda la representación de sistemas de inferencia fuzzy complejos mediante redes de Petri. En la sección 4 se desarrolla un formalismo adecuado para la representación tanto del proceso de razonamiento fuzzy como del sistema físico que lo implementará y se describe en detalle la arquitectura desarrollada. Finalmente, la sección 5 resume las conclusiones y principales aportaciones.

2 Sistemas Fuzzy complejos

Los modelos matemáticos para el desarrollo de Sistemas de Inferencia Fuzzy (FIS, en inglés) desarrollados han sido numerosos. Dado que nuestro objetivo es el de proporcionar un marco formal y una arquitectura hardware que soporten la realización electrónica de FIS de estructura compleja, nuestra primera tarea consistió en seleccionar de entre los diferentes modelos existentes aquéllas características, métodos, propiedades, operadores, etc. más adecuados, como un compromiso entre versatilidad, capacidad expresiva y de modelado, y complejidad computacional. A continuación realizamos una escueta revisión del modelo matemático subyacente en el que se fundamentará nuestra arquitectura, en la que omitiremos la justificación del modelo en aras de una mayor brevedad expositiva.

Podemos centrar nuestra exposición, sin pérdida de generalidad, en sistemas de inferencia fuzzy con bases de reglas ajustadas a la siguiente estructura:

$$R_n^r : \text{si } x_1 \text{ es } A_1^r \text{ y } x_2 \text{ es } A_2^r \text{ y} \ldots \text{y } x_M \text{ es } A_M^r \text{ entonces } y_n \text{ es } \mathrm{B_n^r} . \quad (1)$$

donde $n = 1, \ldots, N$, es el número de consecuentes, $r = 1, \ldots, R$ es el número de reglas, y $m = 1, \ldots, M$ es el número de antecedentes. A^r_m son M x R valores lingüísticos correspondientes a las variables lingüísticas x_m de las proposiciones antecedentes, que tienen asociado un conjunto fuzzy determinado por las funciones de pertenencia $\mu_{A_m^r}(x_m)$; B^r_n son N x R valores lingüísticos de las variables consecuentes, definidos por las funciones de pertenencia $\mu_{B_n^r}(y_m)$. Las variables de entrada (antecedentes), x_m, están definidas sobre los universos de discurso U_m, y las variables de salida (consecuentes), y_n, sobre los V_n.

Considerando una estrategia de ejecución dirigida por reglas, el operador *sup-min* para implementar la regla composicional de inferencia, y una t-norma como operador de impli-

cación, la conclusión de la regla *r-ésima* puede obtenerse como:

$$\mu_{B_n^{'r}}(y_n) = T_I\left\{T_c\left\{\bigvee_{x_1}\left[\left(\mu_{A_1^{'}}(x_1) * \mu_{A_1^{r}}(x_1)\right)\right],\ldots,\left[\bigvee_{x_M}\left(\mu_{A_M^{'}}(x_M) * \mu_{A_M^{r}}(x_M)\right)\right]\right\}, \mu_{B_n^{r}}(y_n)\right\}. \quad (2)$$

donde $\mu_{A_m^{'}}(x_m)$ es la función de pertenencia de la premisa asociada a la variable x_m, T_c denota al operador de conjunción seleccionado para la composición de los antecedentes y la *observación*, $\vee$ y $*$ definen al operador *sup-min*, y T_I representa la t-norma elegida como operador de implicación.

En la ecuación (2) cada uno de los términos entre corchetes define el *grado de compatibilidad* α^r_m entre un antecedente y la entrada fuzzy correspondiente (*observación*), o *grado de verificación* (GDV^r_m) de la proposición. Por tanto, expresando (2) de forma resumida:

$$\mu_{B_n^{'r}}(y_n) = T_I\left(\alpha^r, \mu_{B_n^{r}}(y_n)\right), \quad (3)$$

donde $\alpha^r = \underset{m}{T_c}(\alpha^r_m)$ es el *grado de verificación global* de la regla *r-ésima* (también denotado por GDV^r).

Si existen varias reglas con el mismo consecuente, B^r_n, deberemos agregar los resultados de cada una de ellas para obtener la conclusión global para esa variable, usualmente con el operador de "unión", aunque la intersección también puede ser utilizada.

La existencia de distintos grados de confianza asociados a cada una de las reglas justifica la utilización de un mecanismo de cualificación de las reglas que pondere el peso que la inferencia asociada a una regla tiene en el proceso de inferencia global. De esta forma, la conclusión de una regla genérica como la expresada en (1), donde τ^r es el grado de certeza asociado a dicha regla, se obtiene mediante la composición de la función asociada a la variable de verdad con la distribución de posibilidad asociada a la variable consecuente. Modificando la expresión (2), obtenemos:

$$\mu_{B_n^{'r}}(y_n) = \tau^r\left\{T_I\left\{T_c\left\{\left[\bigvee_{x_1}\left(\mu_{A_1^{'}}(x_1) * \mu_{A_1^{r}}(x_1)\right)\right],\ldots,\left[\bigvee_{x_M}\left(\mu_{A_M^{'}}(x_M) * \mu_{A_M^{r}}(x_M)\right)\right]\right\}, \mu_{B_n^{r}}(y_n)\right\}\right\}. \quad (4)$$

En algunos casos puede ser conveniente incluso establecer un mecanismo de cualificación de cada uno de los antecedentes de las reglas. Usualmente para ello se utiliza una ponderación mediante pesos de los grados de verificación (α^r_m) de cada antecedente.

En los sistemas complejos resulta difícil establecer una relación directa entre las variables de entrada y salida del sistema. Más bien, lo que ocurre es que entre las variables de entrada y salida existen un número de *variables intermedias*, y la relación de entrada/salida del sistema sólo puede ser expresada en forma de relaciones parciales de causalidad en etapas sucesivas de conocimiento en las que intervienen las variables intermedias. Este tipo de razonamiento jerárquico implica ciertas dependencias entre las reglas y una asimetría en la base de conocimiento [3].

Dado un subconjunto de la base de reglas R^I_n, ..., R^S_n, que realizan inferencias sobre una misma variable (consecuente) y_n, si esta variable figura a su vez en la parte antecedente de, al menos, una regla posterior R^T_l, decimos que tendrán lugar *S* encadenamientos simples en la misma variable y_n, que se denomina por tanto *variable de encadenamiento*. En este caso, el grado de verificación de la proposición y^T_n es B^T_n se determina mediante la expresión (5):

$$
\begin{aligned}
R_n^1 &: \text{si } x_1^1 \text{ es } A_1^1 \text{ y} \ldots \text{entonces } y_n^1 \text{ es } B_n^1 \quad (\tau^1).\\
R_n^2 &: \text{si } x_1^2 \text{ es } A_1^2 \text{ y} \ldots \text{entonces } y_n^2 \text{ es } B_n^2 \quad (\tau^2).\\
&\ldots\\
R_n^S &: \text{si } x_1^S \text{ es } A_1^S \text{ y} \ldots \text{entonces } y_n^S \text{ es } B_n^S \quad (\tau^S).\\
R_l^T &: \text{si } x_n^T \text{ es } B_n^T \text{ y} \ldots \text{entonces } y_l^T \text{ es } B_l^T \quad (\tau^T).
\end{aligned}
$$

$$
\alpha_n^T = \bigvee_{s=1}^{S} \left[\tau^s(\alpha^s) \overset{s}{\circ} \mu_{B_n^s, B_n^T} \right] \qquad (5)
$$

donde α^s es el grado de verificación de la regla *s* y el operador $\overset{s}{\circ}$ será el *mínimo* o el *máximo* dependiendo de que τ^S sea una función monótona creciente o decreciente respectivamente [4]. Genéricamente, $\mu_{B_n^s, B_n^T}$ representa el *Grado de Compatibilidad* (*GC*) existente entre la distribución de posibilidad asociada al valor lingüístico B^s_n y la distribución asociada a B^T_n.

3 Representación de FIS mediante redes de Petri

Las estrategias de ejecución basada en diferentes tipos de representaciones matriciales [5] presentan importantes inconvenientes cuando se trata de bases complejas, con encadenamiento entre reglas. Las Redes de Petri (RP) [6] han demostrado ser una herramienta eficaz en la representación y análisis de sistemas dinámicos complejos tales como los basados en reglas, los sistemas productivos, los relativos a la planificación de tareas, etc. Este formalismo tiene como principales características una alta capacidad de representación y la posibilidad de modelar procesos paralelos y concurrentes.

Las extensiones del formalismo original, con la incorporación de nuevos elementos y comportamientos que permiten la representación del concepto de "borrosidad" y operaciones relacionadas, se referencian en la bibliografía como un nuevo modelo de representación, denominado Red de Petri Fuzzy, o FPN (*Fuzzy Petri Net*) [7].

La utilización de un formalismo perfectamente establecido, como son las RP, para modelizar sistemas fuzzy complejos presenta las siguientes ventajas:

1) permite expresar tanto el comportamiento estático del sistema (estructura de la base de reglas), como el dinámico (ejecución de la base de reglas, razonamiento);

2) el formalismo FPN es a menudo una extensión únicamente a nivel interpretativo, por lo que se conservan las propiedades de las RP tradicionales, lo que permite realizar análisis de la consistencia de la Base de Reglas;

3) la descripción FPN está formada por un número limitado de elementos (componentes) que se combinan entre sí, lo que favorece la reusabilidad;

4) admite topologías sin límites en cuanto a complejidad, lo que permite abordar aplicaciones de cualquier tipo y tamaño;

5) permite la simulación y el análisis bajo distintas condiciones de operación;

6) da lugar a arquitecturas modulares y escalables, lo que facilita su proyección hardware.

Sin embargo, los formalismos desarrollados hasta la fecha adolecen de una o varias deficiencias importantes: factores de certeza numéricos en lugar de lingüísticos, encadenamiento únicamente a nivel de valor y no de variable, no especifican mecanismos de agregación de variables intermedias, y, por último, usualmente no se realiza una descripción formal de la topología y el comportamiento de las redes a las que cada modelo puede dar lugar, de vital importancia en nuestro caso para la detección de situaciones de paralelismo,

máxima concurrencia, conflictos, etc. Para superar estas limitaciones se hizo necesaria la elaboración de un modelo FPN propio (8) y perfectamente orientado para su realización hardware, cuyos componentes se enumeran el siguiente apartado.

4 Arquitectura FIS modular basada en FPNs

En este apartado se describe la metodología de representación de sistemas fuzzy basada en FPNs propuesta, que denominaremos representación *hardware-FPN*.

A nivel lógico, en nuestro modelo *hardware-FPN* deberemos distinguir entre lugares de entrada (*LEM*), de salida (*LSM*) e intermedios (*LIM*), y entre transiciones ordinarias (*TOM*) y de encadenamiento (*TEM*). En cuanto a su representación gráfica, los distintos tipos de lugares no ofrecen confusión, debido a su distribución en la red, por lo que utilizaremos el símbolo tradicional de las RP para todos ellos. Sin embargo, en el caso de las transiciones se hace necesaria la definición de una simbología que permita distinguirlas, pues su disposición en una red es arbitraria. Utilizaremos la tradicional "barra" para las transiciones ordinarias, asociadas al disparo de reglas, y un rectángulo más ancho para las transiciones de encadenamiento.

Los pesos (w_{pi}) de cada antecedente se expresarán junto a la entrada correspondiente de cada transición ordinaria. De igual forma, los grados de certeza de cada regla (τ) se mostrarán junto a la salida correspondiente de la transición. En las transiciones de encadenamiento no existen pesos ni coeficientes de certeza, pero deberemos definir los grados de compatibilidad entre cada antecedente y el consecuente ($\mu_{pi,pj}$), cuyo valor se expresará sobre el símbolo gráfico de estas transiciones. Por último, en el interior de cada lugar se deberá especificar el valor lingüístico de la proposición a la que se encuentra asociado. Con estas especificaciones, y a modo de ejemplo, la representación en el formalismo *hardware-FPN* de la siguiente base de reglas:

$$\begin{aligned} R^1 &: \text{si } A \text{ es } A_1 \text{ y } C \text{ es } C_1 \text{ entonces } B \text{ es } B_1 \quad (\tau^1), w_{A_1}, w_{C_1} \\ R^2 &: \text{si } D \text{ es } D_1 \text{ entonces } B \text{ es } B_1 \text{ y } E \text{ es } E_2 \quad (\tau^2), (\tau^3), w_{D_1} \\ R^3 &: \text{si } B \text{ es } B_1 \text{ y } E \text{ es } E_3 \text{ entonces } F \text{ es } F_1 \quad (\tau^4), w_{B_1}, w_{E_3} \end{aligned} \tag{6}$$

será la mostrada en la figura 1.

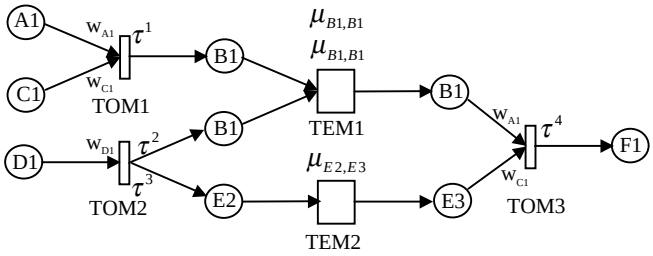

Fig. 1. Estructura gráfica general de una base de reglas en el formalismo *hardware-FPN*

4.1 Estructura general de un módulo

Con las consideraciones realizadas en los apartados anteriores, podemos establecer la estructura general de todos los módulos de nuestro formalismo. Siguiendo los estándares de diseño orientado hacia la reusabilidad [9], las entradas y salidas del módulo deberán ser almacenadas en registros. En su interior se dispondrán dos unidades operativas: la primera de ellas, la unidad de procesamiento, *f(x)*, realizará las operaciones aritmético-lógicas necesarias para llevar a cabo la funcionalidad del módulo (según la sección 2); la segunda, una unidad de control, *g(m)*, tendrá el cometido de controlar el flujo de datos y propagar la marca (figura 2).

Con este esquema, una vez que recibe la marca, un lugar permanece marcado hasta que finaliza su procesamiento sobre los datos actuales (sea cual fuere el número de ciclos necesarios para ello, que puede variar entre los diferentes elementos de una RP). En ese momento transfiere (borra) su marca a la siguiente etapa de inferencia, indicando que el ítem de información que representa es válido en el instante actual. Cuando los lugares de salida reciben una marca y finalizan el procesamiento de sus datos, deben generar nuevas marcas en los lugares de entrada, indicando el comienzo de un nuevo ciclo de razonamiento.

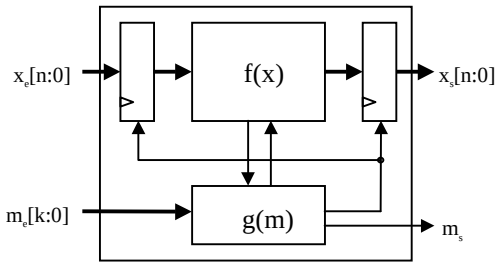

Fig. 2. Estructura general de los módulos del formalismo *hardware-FPN*

4.2 Componentes adicionales del modelo hardware-FPN

Además de los módulos anteriores, relacionados con la etapa de inferencia, la realización de un FIS completo necesita una etapa de *desfuzzyficación*. Para ello se añadieron dos nuevos componentes:

- Módulo Etapa de Agregación (*EA)*. Tras obtener las conclusiones parciales de cada regla, es necesario componerlas (agregación) y obtener una única función de pertenencia para la variable sobre todo su universo de discurso, utilizando este módulo.
- Módulo de *Desfuzzyficación* de Mamdani (*CGM*) o de Takagi (*CGT*). De entre los numerosos métodos de *desfuzzyficación*, el Centro de Gravedad (CG) el más utilizado en la bibliografía, puesto que proporciona resultados coherentes semánticamente para la mayoría de las aplicaciones, y será el método elegido para proporcionar un valor numérico para las variables de salida.

La biblioteca de módulos *FPN_LIB* (*Fuzzy Petri Net Library*) contiene todos los componentes desarrollados en los apartados anteriores, además de otros elementos auxiliares y los necesarios parámetros de configuración. Cabe destacar que los diferentes componentes son independientes de la tecnología de implementación, y en su diseño se ha puesto especial énfasis para que sus componentes puedan ser elaborados sin problemas por las herramientas de síntesis más habituales (*Xilinx®*, *Synopsys®*, *Mentor Graphics®*).

En resumen, y según nuestra opinión, a nivel de arquitectura las claves para la realización de hardware fuzzy son dos: procesamiento distribuido y máximo paralelismo. Es decir, para ser competitivo, el hardware fuzzy debe plasmar también las características que definen al propio formalismo. El procesamiento distribuido significa que la arquitectura deberá estar formada por nodos localmente distribuidos que realicen operaciones a nivel fuzzy, no a nivel de bit, y con un control local, minimizando de esta forma la comunicación entre ellos. Igualmente, deberá evitarse la compartición de las rutas de datos, definiéndose líneas dedicadas en cada caso. Por otro lado, para conseguir la máxima velocidad es obligado realizar el procesamiento en paralelo de la base de reglas, en la medida que las relaciones entre las variables lo permitan, con etapas en *pipeline* cuando sea conveniente.

4.3 Aplicación a la aproximación de funciones no lineales

Una de las primeras aplicaciones utilizadas para validar nuestra arquitectura fue la aproximación de la función:

$$y = \left(1 + x_1^{0,5} + x_2^{-1} + x_3^{-1,5}\right)^2 \tag{7}$$

para lo que se utilizó un sistema fuzzy de Takagi y Sugeno de orden 0 con 27 reglas. Los resultados obtenidos al implementar el sistema sobre una FPGA XC4013XLHT144-08 de *Xilinx Inc.* se muestran en la tabla 1, donde J_1 y J_2 representan respectivamente el error relativo medio del aproximador fuzzy simulado por software, y el del circuito microelectrónico real. Según esta tabla, nuestra metodología presenta una precisión superior a la de otros modelos hardware que consumen más del triple de área. Con respecto a otros aproximadores tradicionales, nuestro modelo tiene la mitad del error de una aproximador lineal, y aproximadamente el mismo que un aproximador polinómico no lineal con un costo computacional notablemente superior al de un sistema fuzzy. Los mejores resultados, como es de esperar, los consiguen sistemas de inferencia neuro-fuzzy realizados en software, a costa de una menor velocidad de cómputo, o de una mayor área de circuito, en caso de su realización hardware.

5 Conclusiones

En este artículo se han analizado las distintas soluciones para la realización de sistemas de inferencia fuzzy, relacionando cada alternativa con su ámbito de aplicación. En particular, nos hemos centrado en las aplicaciones más exigentes, que requieren el desarrollo de hardware fuzzy de propósito específico, resultando de la revisión una serie de propiedades y características que el hardware fuzzy debe satisfacer para enfrentarse a los ámbitos de aplicación con prestaciones más exigentes.

Teniendo en cuenta las propiedades enunciadas, se ha desarrollado una metodología de representación de sistemas de inferencia fuzzy complejos sobre redes de Petri y una arquitectura hardware que facilitan el diseño de procesadores fuzzy de propósito específico, al proporcionar un marco formal de representación y un mecanismo de proyección del modelo fuzzy sobre la tecnología electrónica seleccionada. Las principales características de nuestra metodología y de la arquitectura hardware asociada son: modularidad, escalabilidad, reusabilidad, e independencia tanto de la tecnología de implementación como del tamaño y complejidad de la base de reglas.

Método		Descripción	J_1(%)	J_2(%)	Área (CLB)	Velocidad (FLIPS)
Blake [10]	FL	8 reglas, T&S orden 0	55,40	51,97	286	-
(hardware)	NN	3-8-1, feedforward	13,05	20,29	582	-
	FNN	4-6-1 T&S orden 0	4,43	10,38	1290	-
Garrigós	FPN	27 reglas, T&S orden 0	**5,79**	**5,74**	**351**	**170.998**
Sugeno [11]	linear	aproximador lineal	12,7	11,1	-	-
(software)	GMDH	apr. polinom. no lineal	4,7	5,7	-	-
	Fuzzy I	3 reglas, T&S orden 1	1,5	2,1	-	-
	Fuzzy II	4 reglas, T&S orden 1	0,59	3,4	-	-
Furuhashi [12]	FNN I	8 reglas, T&S orden 0	0,84	1,22	-	-
(software)	FNN II	4 reglas, T&S orden 1	0,73	1,28	-	-
	FNN III	8 reglas, Mamdani	0,63	1,25	-	-

Tabla 1. Diferentes realizaciones de aproximadores de la función expresada en (7)

Referencias

[1] Togai, M., Watanabe, H.: Expert system on a chip: an engine for real-time approximate reasoning. IEEE Expert (1986) 55-62.

[2] Baturone, I., Barriga, A., Sánchez-Solano, S., Jiménez-Fernández, C.J., López, D.R.: Microelectronic Design of Fuzzy Logic-Based Systems. CRC Press LLC (2000).

[3] Driankov, D., Hellendoorn, H.: Chaining of fuzzy IF-THEN rules in Mamdani-Controllers. Proceedings of 1995 IEEE International Conference on Fuzzy Systems (1995) Vol. 3, 103-108.

[4] Bugarín, A.J., Barro, S.: Reasoning with Truth Values on Compacted Fuzzy Chained Rules. IEEE Transactions on Systems, Man and Cybernetics--Part B (1998) Vol. 28, No. 1 34-46.

[5] Chen, S.M.: A new approach to inexact reasoning for Rule-Based Systems. Cybernetics and Systems (1992) Vol. 23, 561-582.

[6] Murata, T.: Petri Nets: Properties, Analysis and Applications. Proceedings of the IEEE (1989) Vol. 77, No. 4, 541-580.

[7] Cardoso, J., Camargo, H.: Fuzzyness in Petri Nets. Springer-Verlag (1999).

[8] Garrigós, F.J.: Solución Automatizada Basada en una Arquitectura Modular para la Síntesis Electrónica de Sistemas de Inferencia Fuzzy. Tesis Doctoral (2002).

[9] Keating, M., Bricaud, P.: Reuse Methodology Manual, Second Edition. Design Reuse Partnership - Synopsys Inc. and Mentor Graphics Corp. Kluwer Academic Publishers (1999).

[10] Blake, J.J., Maguire, L.P., McGinnity, T.M., Roche, B., McDaid, L.J.: The implementation of fuzzy systems, neural networks and fuzzy neural networks using FPGAs. Information Sciences (1998) Vol. 112, 151-168.

[11] Sugeno, M., Kang, G.T.: Structure Identification of Fuzzy Model. Fuzzy Sets and Systems (1988) Vol. 28 15-33.

[12] Horikawa, S., Furuhashi, T., Uchikawa, Y.: On Fuzzy Modeling Using Fuzzy Neural Networks with Back-Propagation Algorithm. IEEE Trans. on Neural Networks (1992) Vol.3, 801-806.