Controladores Difusos de Altas Prestaciones: Sistema para la Generación Automática

Nelson Acosta
INCA - Departamento de Computación & Sistemas
Facultad de Ciencias Exactas - UNCPBA - Campus Universitario
7000 Tandil - Provincia de Buenos Aires - ARGENTINA
Email: nacosta@exa.unicen.edu.ar *http://www.exa.unicen.edu.ar/inca/*

Resumen:

Este artículo presenta un sistema de generación automática de controladores difusos de altas prestaciones que puedan ejecutarse sobre una arquitectura IBM-PC. Se define un lenguaje de especificación de controladores, que la herramienta toma para generar de forma automática los archivos que una vez compilados producen el ejecutable del FLC.

Palabras clave:

Lógica difusa, controladores difusos, arquitectura de controladores, CAE

1.-Introducción

Los controladores difusos (FLC) pueden ser materializados por software sobre hardware de propósito general, o sobre microcontroladores dedicados [1, 2, 3, 4] o mediante un circuito específico (ASIC, FPGA, PLD; [6, 9, 12]). Para cualquier forma de materialización es conveniente realizar una optimización del sistema, que involucre operaciones a medida de la aplicación: fusificación, defusificación y motor de inferencias a medida (a partir de las reglas que definen el comportamiento del FLC).

Una parte importante de la síntesis es la generación de un esquema y plataforma de cálculo para la evaluación de las expresiones reticulares y aritméticas (inferencia y defusificación). El requerimiento de velocidad para el cálculo de reglas en un FLC exige la migración de software a hardware, por ejemplo mediante técnicas de co-diseño hardware/software. En un hardware a medida se pueden aprovechar características propias de la aplicación (paralelismo [13] y tamaños de rutas de datos y operadores de cálculo a medida de los operandos) y técnicas de altas prestaciones (reglas activas y fusificación e inferencia paralela).

El objetivo global de éste proyecto es el desarrollo de herramientas de generación automática de FLC, que corran sobre plataformas PC, a partir de la especificación de las funciones de pertenencia y de las reglas de inferencia. Las herramientas en cuestión generan código que puede ser compilado para que ejecute sobre una arquitectura PC estándar, se generan archivos *Pascal* o *C*, siguiendo una metodología de inferencia de reglas activas, haciendo la fusificación por tablas o por funciones de cálculo en línea, y realizando la defusificación por medio del método del centroide. En suma, se aplican las técnicas de optimización utilizadas para el diseño de FLC hardware lo que permite obtener FLC software sobre PC que ejecuten a altas frecuencias.

2.-Propuesta General

El ambiente para la generación de FLC parte de una especificación, en lenguaje textual, que es procesado automáticamente [5, 8, 12] para la obtención de todos los parámetros del sistema y la posterior generación del código C o Pascal.

El lenguaje de definición de controladores difusos ha sido fuertemente influenciado, por un lado, del FIL (Fuzzy Inference Language) de Aptronix [10], y por el otro, de la implementación del primer prototipo, para lo cual se utilizó el lenguaje Prolog [11]. Así el lenguaje es una secuencia de "*hechos*" Prolog que definen: a) variables de entrada, b) funciones de pertenencia, c) variables de salida, d) funciones de decodificación, y e) reglas. A las variables se les define un dominio de trabajo, $\boldsymbol{y} = f(\boldsymbol{x})$, donde $\boldsymbol{x} \in [Xmin, Xmax]$ e $\boldsymbol{y} \in [Ymin, Ymax]$, expresado como la cuádrupla: [Xmin, Xmax, Ymin, Ymax]. Las funciones de pertenencia y decodificación trabajan en el dominio definido para su variable, y por polígonos. La sintaxis de las cinco sentencias es la siguiente:

invar("NomInVar", Xmin, Ymin, Xmax, Ymax)
f_p("NomFP", "NomInVar", Lista_de_Puntos_Polígono)
outvar("NomOutVar", Xmin, Ymin, Xmax, Ymax)
f_d("NomFD", "NomOutVar", Lista_de_Puntos _Polígono)

si([is(“NomInVar”, “NomFP”), is(...,...), ...], is(“NomOutVar”, “NomFD”))

El sistema genera código fuente para cada una de las partes que forman el FLC basado en el modelo propuesto por ‘Mamdani’:

a) **Fusificador**. Se plantean tres enfoques principales: (1) se genera el contenido de las funciones de pertenencia (archivos de datos con las tablas) para materializar el código fuente de una función por cada función de pertenencia, (2) se generan tablas optimizadas en memoria, donde hay tantas tablas como funciones de pertenencia se superpongan por cada variable, o (3) se genera el código para el cálculo de las funciones de pertenencia en línea (código C, Pascal). Esta división de la generación, permite al diseñador mayor flexibilidad a la hora de definir su plataforma completa, permitiendo hacer todas las combinaciones que le sean posible de acuerdo a las necesidades y características de la aplicación.
b) **Motor de inferencias**. Se genera un esquema de inferencias difusas a partir del conjunto de reglas, que define el comportamiento del FLC, trabajando con reglas activas [12]. Cada valor de una variable de entrada es fusificado mediante la aplicación de las funciones de pertenencia. Esos valores de salida son diferentes de cero sólo para unos pocos conjuntos difusos. Por otro lado, sólo los valores diferentes de cero activan reglas del FLC, de modo que la mayoría de los consecuentes de salida estarán vacíos, y en el modelo de inferencia por reglas activas sólo se evalúan las reglas que aportan información relevante al modelo del controlador. Por la aplicación de este principio se logra reducir drásticamente el número total de operaciones, y consecuentemente, el tiempo de cálculo. Esta optimización sólo calcula un porcentaje muy pequeño del total de reglas, así en [12] de 35 reglas sólo se evalúan 4, en los ejemplos propuestos por [10]: en el servomotor de 62 reglas sólo se evalúan 4, en el control de temperatura de plasma de 234 sólo se evalúan 8, etc..
c) **Defusificador**. Las funciones de pertenencia de salida (o de defusificación) se representan por singleton (un único valor constante para cada función) y se defusifica utilizando el método de cálculo de centroide (simplificación del método Centro de Gravedad).

La simplificación de cálculo que impone trabajar con reglas activas representa la mayor ganancia en velocidad en el FLC, mientras que otra optimización relevante es el fusificador por segmentos [5], donde sólo se fusifica el número de funciones de pertenencia que se superpongan en todo el dominio de la variable.

3.-Sintetizador del controlador

El compilador de reglas, procesa el archivo con la definición de reglas para generar un esquema de cálculo capaz de realizar las inferencias difusas definidas. Así se obtiene un programa del motor de inferencia que materializa el FLC. A continuación se muestra un diagrama genérico de un controlador generado (en un estilo Pascal):

```
{--- (A) --- Fusificación por segmentos ---}
FOR cada variable de entrada V
   FOR cada función de pertenencia que se superpone F
      reg[V,F]:=fusificar( V, F );
{--- (B) --- Calcula Inferencia Difusa ---}
FOR cada variable de salida VO
   ind:=FAM[ VO, reg[V,perm], reg[V,perm] ];
   SOP[VO,ind]:=max( min(reg[V,perm],reg[V,perm]), SOP[VO,ind] );
{--- (C) --- Defusificacion por Centroide ---}
FOR cada variable de salida VO
   n[VO]:=n[VO]+defusifica( SOP[VO,ind] );
   d[VO]:=d[VO]+SOP[VO,ind];
   IF d[VO]<>0 THEN F[VO]:=n[VO] div d[VO];
```

Donde: Es necesario un reset global que lleve a un estado conocido a todos los registros, ***perm***, ***reg***, ***sop***, ***n***, y ***d***; ya que normalmente se opera acumulando los valores. Además: (A) La fusificación se realiza utilizando la técnica por segmentos, se hacen tantos calculos o consultas como la cantidad de variables por la cantidad de funciones de pertenencia que se superponen. (B) La inferencia difusa se calcula utilizando una memoria asociativa de reglas (en este caso de 2 variables), y reg[V,perm] es la consulta a los registros donde se almacenan los valores de la fusificación, permutando los valores (del producto cartesiano entre la cantidad de variables de entrada x el número de funciones de pertenencia que se superponen en dicha variable) en cada llamada. (C) Se realiza la defusificación por el método del centroide.

5.-Generador de funciones de pertenencia y decodificación

De acuerdo a las funciones de pertenencia y decodificación, definidas con el lenguaje de especificación (cada uno de los conjuntos difusos con polígonos [11]), son generadas las funciones (para el cálculo en línea o mediante tablas). También es posible definir el conjunto de variables, de entrada y salida, con diferentes tamaños de palabra (precisión), tanto para las abscisas como para las ordenadas de las funciones (rango de valores que adopta). Las funciones de decodificación, especifican una tabla que contiene los centros de gravedad de la función cortada por el valor de ingreso (el peso calculado). Opcionalmente, dichos valores son multiplicados por el peso, de tal forma que el programa no necesite realizar dicha operación en línea (obviamente a costo de aumentar la memoria necesaria).

6.-Conclusiones

La herramienta descrita en éste artículo genera un FLC de altas prestaciones automáticamente, basándose en reglas activas y un fusificador por segmentos. Estos diseños permiten evaluar sólo las funciones de pertenencia y reglas que aporten información al sistema de control. La comparación de tales sistemas permite una ganancia del orden de 40% (y en algunos casos bastante más del 90%) de la frecuencia de muestreo. Esa ganancia depende directamente de la relación entre: a) la cantidad de funciones de pertenencia con respecto a las funciones de pertenencia que no son nulas. b) la cantidad de reglas totales sobre la cantidad de reglas que se activan como máximo.

Actualmente se está trabajando en la generación automática de circuitos que materialicen tales diseños sobre plataformas FPGA utilizando VHDL como lenguaje de síntesis.

Referencias

[1] "*Expert System on a Chip: An Engine for Real-Time Approximate Reasoning*", M. Togai & H. Watanabe. IEEE EXPERT, vol. 1, nº 3, pp. 55 - 62, August 1986.

[2] "*A VLSI Fuzzy Logic Controller with Reconfigurable, Cascadable Architecture*", H. Watanabe, W. D. Dettlof & K. E. Yount. IEEE Journal on Solid-State Circuits, vol. 25, nº 2, April 1990.

[3] "*Architecture of a PDM VLSI Fuzzy Logic Controller with Pipelining and Optimized Chip area*", A. P. Ungering, K. Thuener & K. Goser. Proc. of IEEE Int. Conf. on Fuzzy Systems, pp. 447 - 452, 1993.

[4] "*7.5 MFLIPS Fuzzy Microprocessor Using SIMD and Logic-in-Memory Structure*", M. Sasaki, F. Ueno & T. Inoue. Proc. of IEEE Int. Conf. on Fuzzy Systems, pp. 527 - 534, 93.

[5] "*Segment representation for membership functions*", N. Acosta, J. Garrido y J-P Deschamps. Int. Symp. on Engineering of Intelligent Systems (EIS'98), Tennerife, España. Feb'98..

[6] "*Hardware Solution for Fuzzy Control*", A. Costa, A. De Gloria, P. Faraboschi, A. Pagni & G. Rizzotto. Proceedings of the IEEE, vol. 83, nº 3, March 1995, pp. 422 - 434.

[7] "*Fuzzy Logic Systems for engineering: A Tutorial*", J.M.Mendel. Proceedings of the IEEE, vol. 83,nº 3, March 1995, pp. 345 - 377.

[8] "*Metodología para la Materialización de Algoritmos Borrosos: de una Especificación de Alto Nivel a un ASIC*", J-P Deschamps y J. I. Martínez Torre. V Congreso Español sobre Tecnología y Lógica Fuzzy (ESTYLF'95), Murcia 20 - 22 de Septiembre de 1995, pp. 251 - 256.

[9] "*Dedicated Digital Fuzzy Hardware*", D. L. Hung. IEEE Micro, vol. 15, nº 4, August 1995, pp.31 - 39.

[10] "*Ejemplos del ambiente FuzzyNET*", Aptronix. http://www.aptronix.com.

[11] "*Automatic program generator for customized fuzzy logic controllers*", N. Acosta, J-P Deschamps y G. Sutter. International Symposium on Algorithms and Architectures for real-time control (AART'97), organizado por la IFAC (International Federation of Automatic Control), Vilamoura, Portugal. Abril'97..

[12] "*Optimized active rule fuzzy logic custom controller*", N. Acosta, J-P Deschamps y J. Garrido. International Symposium on Engineering of Intelligent Systems (EIS'98), Tennerife, España. Feb'98.

[13] "*Motores de Inferencia para Controladores Difusos: análisis de materializaciones hardware*". N. Acosta & H. Curti. Congreso Internacional en Ing. Informática (ICIE). UBA, Bs. As. 26-28 abril 2000.