MATERIALIZACIÓN DE

CONTROLADORES DIFUSOS ACTIVOS

Nelson Acosta
INCA / INTIA
Departamento de Computación y Sistemas - Facultad de Ciencias Exactas - UNCPBA
Campus Universitario - 7000 TANDIL (Buenos Aires)
Email: nacosta@exa.unicen.edu.ar

Resumen:

Los controladores difusos (FLC) pueden ser implementados por hardware estándar, microtrontroladores dedicados o circuitos de aplicación específica. Históricamente, para obtener un FLC de altas prestaciones se baja en el nivel de abstracción de la materialización, así se obtienen controladores hardware a medida de la aplicación de muy altas prestaciones. En este artículo, se presentan técnicas que permiten el desarrollo de FLC que dupliquen (como mínimo) su frecuencia de funcionamiento sobre la misma plataforma. Estas técnicas se basan principalmente en reglas activas, y además se propone el concepto de fusificación y defusificación activa.

Palabras clave:

Control Difuso, Lógica Difusa, Controladores, Diseño de Controladores

1. INTRODUCCIÓN

Los controladores difusos pueden ser implementados por hardware estándar. Para procesar un alto número de reglas en tiempos muy cortos se puede utilizar un microcontroladore dedicado [Tog86; Wat90; Ung93; Sas93]. Otra posibilidad es usar un circuito específico sobre plataformas ASIC (Standard Cell o Full Custom), FPGA o PLD [Cos95; Men95; Des95; Hun95; Aco97]. Los diseñadores de FLC a medida de la aplicación han desarrollado un conjunto de técnicas que pueden ser utilizadas sobre cualquier plataforma para obtener un controlador de altas prestaciones. En este artículo, se presentan técnicas que permiten el desarrollo de FLC que optimicen la frecuencia de funcionamiento sobre la misma plataforma. Estas técnicas se basan principalmente en reglas activas [Aco98a; Ike91], y además se propone el concepto de fusificación y defusificación activa.

La sección 2 describe el funcionamiento de un algoritmo de control difuso. Las secciones 3 y 4 presentan respectivamente las técnicas de materialización de FLC clásico y utilizando las técnicas de FLC activos. La sección 5 presenta las conclusiones y futuros trabajos.

2. ALGORITMO DIFUSO

Un sistema de lógica difusa realiza la traducción de valores de entrada sobre valores de salida. Este proceso incluye cuatro componentes para todo FLC de *Mamdami*: reglas, fusificador, motor de inferencia difusa y defusificador. El fusificador traduce valores del mundo real a su representación en los conjuntos difusos, traducción necesaria para la activación de las reglas de inferencia. El motor de inferencia realiza la aplicación de todas las reglas a los valores difusos. El defusificador traduce los valores difusos obtenidos del motor de inferencias en valores reales que puedan representar una acción de control.

En este artículo se propone una metodología para el diseño de FLC, para demostrar la aplicación de la metodología se necesita de un problema práctico *bien* definido. Se decide utilizar como aplicación un ejemplo propuesto por D. Nguyen y B. Widrow en 1989 [Ngu89] y rescatado por Bart Kosko en su libro [Kos92]. Dicha aplicación es un clásico en la literatura de controladores difusos. La aplicación consiste en definir un FLC para el guiado automático en el estacionamiento marcha atrás de un camión en una dársena de carga. El camión se encuentra sobre una playa de estacionamiento sin obstáculos en alguna posición sobre los ejes *x-y*, con un ángulo sobre la vertical f (*phi*) dado.

2.1 - FUSIFICADOR.

Analizando el gráfico de todas las funciones de pertenencia de cada variable de entrada (Fig. 1 y Fig. 2), se determina que los valores fusificados que aportan información serán como máximo 2 por variable de entrada. Esto se debe a que hay una superposición máxima ***O*** de dos por variable.

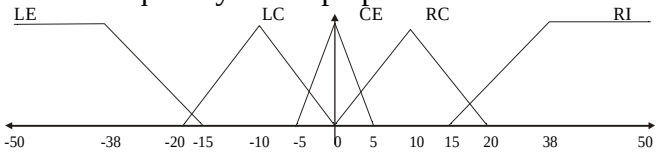

Fig. 1 – Funciones de pertenencia de la variable de entrada ***x***.

Así, en el ejemplo, tenemos que:

	LE	LC	**CE**	**RC**	RI
x=20	0	0	245	23	0

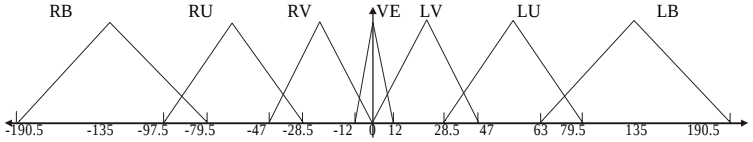

Fig. 2 – Funciones de pertenencia de la variable de entrada ***phi***.

	RB	**RU**	RV	VE	LV	LU	LB
phi=-90	60	9	0	0	0	0	

Dada la superposición ***O*** de los segmentos, se puede ahorrar:

- Memoria: al almacenar sólo los resultados diferentes de cero de las funciones de pertenencia.
- Tiempo de cálculo: al calcular sólo las funciones de pertenencia que dan resultados diferentes de cero.

2.2 - MOTOR DE INFERENCIA.

El conjunto de reglas que define el comportamiento del controlador puede ser representado por una estructura ***n***-dimensional (donde ***n*** representa la cantidad de variables de entrada). Por cada variable de entrada se activa sólo el conjunto de reglas ***O*** cuyas funciones de pertenencia tengan valor diferente de cero.

		X				
		LE	LC	**CE**	**RC**	RI
ϕ	**RB**	PS	PM	**PM**	**PB**	PB
	RU	NS	PS	**PM**	**PB**	PB
	RV	NM	NS	PS	PM	PB
	VE	NM	NM	ZE	PM	PM
	LV	NB	NM	NS	PS	PM
	LU	NB	NB	NM	NS	PS
	LB	NB	NB	NM	NM	NS

Fig. 3 - Reglas de inferencia

Así en el ejemplo (Fig. 3) tenemos que sólo se activan **CE** y **RC** para ***x***, y **RB** y **RU** para ***phi***. Esto nos lleva a determinar el conjunto de reglas que realmente aportan información a la inferencia difusa:

IF (***x*** is **CE**) and (***phi*** is **RB**) THEN ***theta*** is **PM**;
IF (***x*** is **CE**) and (***phi*** is **RU**) THEN ***theta*** is **PM**;
IF (***x*** is **RC**) and (***phi*** is **RB**) THEN ***theta*** is **PB**;
IF (***x*** is **RC**) and (***phi*** is **RU**) THEN ***theta*** is **PB**;

2.3 - DEFUSIFICADOR.

La selección de los consecuentes y de los valores difusos de los consecuentes realizada por el motor de inferencias determina que funciones de pertenencia de salida deben aplicarse a la defusificación. Entonces tenemos que las funciones de pertenencia de salida que aportan información son:

$$\prod O_{\text{variable de entrada}}.$$

En el ejemplo, la cantidad máxima de funciones de pertenencia de salida para la defusificación está dada por: $O_x * O_{phi} = 2 * 2 = 4$. Como se puede ver gráficamente, es la región representada por la tabla de reglas (Fig. 3, Fig. 4 y Fig. 5). Se debe destacar que este número es máximo, debido a que varias reglas pueden aportar información sobre el mismo consecuente.

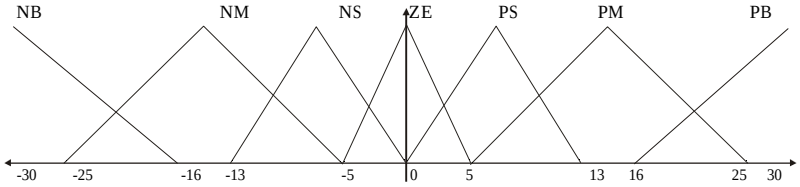

Fig. 4 – Funciones de pertenencia de la variable de salida THETA.

	NB	NM	NS	ZE	PS	PM	PB
Singleton	-28	-16	-6	0	6	16	28

Fig. 5 – Singleton de la variables de salida THETA

2.4 - PROCESO COMPLETO.

El proceso del controlador puede describirse o realizarse gráficamente. A la izquierda en la Fig. 6 se muestra el proceso de fusificación donde las gráficas de las funciones de pertenencia son cortadas por el valor de entrada (definiendo los trapecios de color). Las figuras (CE y RC para ***x*** y RB y RU para ***phi***) son utilizadas para aplicar las reglas de inferencia. Así, PM será determinado por el máximo entre los mínimos de CE-RB y CE-RU, mientras que PB será definido como el máximo entre los mínimos de RC-RB y RC-RU. Los máximos que definen las dos funciones de pertenencia de salida PM y PB definen las figuras sombreadas sobre el dominio de ***theta***. Para obtener el valor definitivo (o crisp) de la variable de salida se debe calcular el centro de gravedad de la gráfica formada por PM y PB.

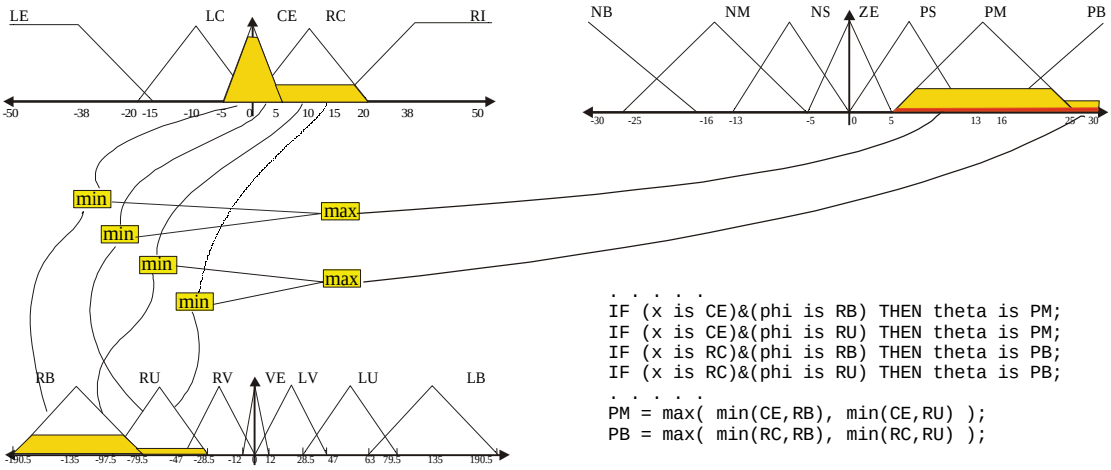

Fig. 6 – Proceso gráfico de inferencia.

3. ALGORITMO DIFUSO: ENFOQUE PASIVO

El enfoque pasivo de cálculo consta de las siguientes operaciones [Aco97, Cos95, Des96, Men95, Tog86, Wat90]:

a) **Fusificación**. Se debe aplicar a cada instancia de cada variable de entrada todas las funciones de pertenencia. Así, se tienen 5 valores fusificados de ***x*** y 7 de ***phi***. De esos 12 valores, sólo 4 como máximo tendrán valores diferentes de cero (en este caso son CE, RC, RB y RU).

b) **Inferencia**. Para cada conjunto de pertenencia de las funciones de salida, se determina su valor difuso de acuerdo a la regla MAX-MIN. Se calcula el máximo (agregación) de todos

los mínimos. En este ejemplo sólo tendrán valores mayores que cero las funciones de pertenencia PM y PB.

c) **Defusificación**. Se calcula el centro de gravedad de la unión de todas las gráficas de las funciones de pertenencia de salida (aunque en este caso solo PM y PB son diferentes de cero).

4. ALGORITMO DIFUSO: ENFOQUE ACTIVO

Los 3 bloques principales que componen los FLC son diseñados a medida de la aplicación, generando un controlador de altas prestaciones [Aco98a, Aco98b, Chi92, Ike91]. El objetivo principal es que el FLC no ejecute instrucciones que no aportan información al sistema.

4.1 - FUSIFICACIÓN ACTIVA

Los valores de las variables de entrada (valores crisp) deben ser traducidos a su representación difusa, para lo cual deben aplicarse todas las funciones de pertenencia de cada variable al valor de su instancia. El resultado de la aplicación de las funciones de pertenencia dará valores diferentes de cero sólo para algunas pocas funciones (como se demostró en 2.1). Así el problema se puede enfocar utilizando, principalmente dos técnicas: usando tablas de memoria (LUTs) o calculando los valores en línea.

La fusificación toma los valores de las variables y les aplica las funciones de pertenencia para traducir el valor a su representación difusa. El fusificador sólo calcula los valores difusos, se supone un registro externo que almacene los valores (en éste caso, REGx y REGp para los valores difusos y además IDFx e IDFp para los identificadores de las funciones de pertenencia activadas).

4.1.1 - Tablas de memoria (LUT).

En 1992 Chiueh [Chi92] propuso una técnica de compresión que permite almacenar todas las funciones de pertenencia en memoria, sin restricciones con respecto a la complejidad de las funciones utilizadas y además con un alto incremento de la velocidad de consulta al minimizar los accesos.

La solución estándar desperdicia gran cantidad de memoria al almacenar todos los elementos, incluyendo los valores nulos. Además, como contrapartida debe acceder a la memoria tantas veces como funciones de pertenencia se quiera aplicar, lo que provoca un retraso importante en la velocidad de respuesta del FLC.

Para utilizar la técnica propuesta por Chiueh se debe tener en cuenta la superposición de funciones de pertenencia por cada una de las variables. Dicho factor ***O*** determina la cantidad de bancos de memoria necesarios para almacenar todas las tablas. Así se dividen todas las funciones de pertenencia en ***O*** bancos de tal forma que no haya superposición, evitando la pérdida de información. En la Fig. 7 puede verse que la variable ***X***, con 3 funciones de pertenencia (***bajo***, ***medio*** y ***alto***) tiene una superposición de 2; por tal razón se debe utilizar el banco ***A*** (que almacena las funciones ***bajo*** y ***alto***) y el banco ***B*** de memoria (que almacena la función ***medio***).

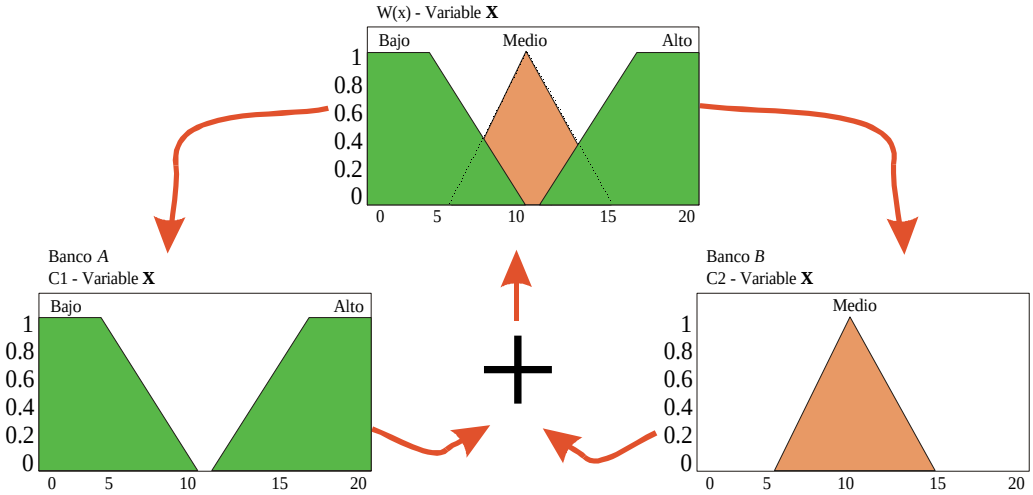

Fig. 7 – Aplicación de O bancos de memoria para mantener las funciones de pertenencia

Debe también tenerse en cuenta que dado que se tiene en cada banco de memoria un conjunto de ***O*** funciones, se debe agregar otro campo dentro del mismo banco para determinar cual es la función de pertenencia que se ha activado. Con esta técnica sólo se necesitan ***O*** accesos a memoria por cada variable de entrada.

A continuación se muestra el pseudo-código estilo pascal que implementa la fusificación por LUT, donde *ivar* es la variable a fusificar, *inc_base* define el banco (uno de los *O*) que se desean calcular.

```
banco := inc_base;
valor_MF := ROM_ib1[ banco, ivar ];
id_MF    := ROM_ib1[ banco, ivar ];
```

4.1.2 - Cálculo de funciones de pertenencia en línea.

La cantidad de funciones de pertenencia que se deben calcular es ***O***, ahora hay que determinar cuales son y como se representan las funciones. Si se restringen las funciones de pertenencia a polinomios lineales genéricos (linear-piecewise), cada función se representa por una serie de segmentos lineales. Donde cada segmento se calcula utilizando la ecuación de la recta.

Los parámetros necesarios para el cálculo de cualquier función de pertenencia con tales características son (Fig. 8): posición horizontal inicial del segmento ***x0***, posición vertical final del segmento ***y0***, y pendiente del segmento ***Constante***. Se debe además tener en cuenta el identificador de función de pertenencia (***#tabla***), que vincula a cada segmento con la función de pertenencia a la que pertenece.

En este caso, el incremento del segmento base (*inc_base*) se ha implementado como un puntero al siguiente segmento, que es calculado fuera de línea y almacenado en la misma tabla de ROM. Se usan las variables internas de cálculo *v*1, *v*2 y *v*3. Los rangos de la variable de entrada se tomaron de la variable *x* del ejemplo de Kosko.

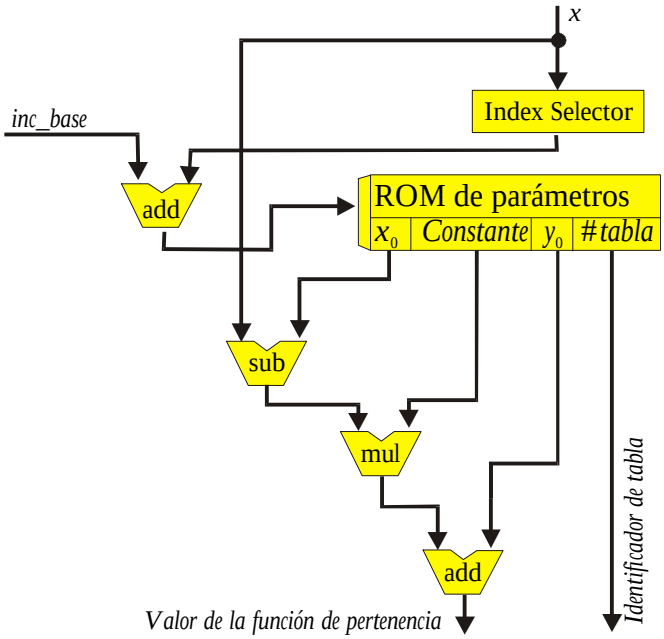

Fig. 8 – Cálculo de las funciones de pertenencia

```
IF (ivar>=-500) THEN v1:=1 { Selector de indice }
ELSE IF (ivar>=-380) THEN v1:=2
ELSE IF (ivar>=-200) THEN v1:=3
ELSE IF (ivar>=-100) THEN v1:=4
ELSE IF (ivar>= -50) THEN v1:=5
ELSE IF (ivar>=   0) THEN v1:=6
ELSE IF (ivar>=   0) THEN v1:=7
ELSE IF (ivar>= 100) THEN v1:=8
ELSE IF (ivar>= 120) THEN v1:=9
ELSE IF (ivar>= 380) THEN v1:=10;
CASE inc_base OF       { inc_base }
   0 : v2 := v1 -1;
   1 : v2 := ROM_ib1[ v1 -1 ];
   END; { case inc_base }
v3 := ivar - ROM_x0[ v2 ];
valor_MF := v3 * ROM_ang[ v2 ];
id_MF    := ROM_tabla[ v2 ];
```

4.2 - INFERENCIA DIFUSA POR REGLAS ACTIVAS

La base de reglas o banco de reglas suele ser denominado FAM (fuzzy associative memory). La asociación (A, B) relaciona el conjunto difuso de salida B (de los valores controlados) con el conjunto difuso de entrada A (de valores de entrada), donde el par representa las asociaciones antecedente/consecuente de cláusulas IF-THEN.

Cada valor de variable de entrada es fusificado por medio de la aplicación de las funciones de pertenencia. Estos valores son diferentes de cero sólo para unos pocos conjuntos difusos, dependiendo de la figura de la función de pertenencia. Sólo los valores diferentes de cero activan reglas, así la mayoría de los conjuntos consecuentes estarán vacíos. Mediante la aplicación de este principio es posible desarrollar una arquitectura que sólo calcule los consecuentes de las reglas que se han activado (diferentes de vacío). Este principio reduce el número total de operaciones, y

consecuentemente el tiempo de cálculo. En el ejemplo, el controlador usa como máximo 4 reglas sobre un total de 35 reglas de la FAM.

El motor de inferencia por reglas activas (Fig. 9) tiene como entrada dos bancos de ***O*** registros por cada variable de entrada (REGx e IDFx para la variable *x* y REGp e IDFp para la variable *phi*. El bloque "construye dirección" selecciona la regla que debe activarse del banco de reglas en función de las funciones de pertenencia que son antecedentes de la regla (registros IDF), y determina el consecuente que es utilizado como dirección para acceder al banco de registros SOP. El valor almacenado en SOP es determinado por el máximo valor de los mínimos de los antecedentes. Como se trabaja en forma acumulativa, debido a que varias reglas pueden aportar sobre el mismo consecuente, se debe poner a cero el registro SOP antes de cada iteración.

```
{--- Variable de salida: teta ---}
FAM_teta( IDFx[0], IDFphi[0], indice );
SOPteta[indice]:=max( min(REGx[0], REGphi[0] ), SOPteta[indice]);
FAM_teta( IDFx[1], IDFphi[0], indice );
SOPteta[indice]:=max( min(REGx[1], REGphi[0] ), SOPteta[indice]);
FAM_teta( IDFx[0], IDFphi[1], indice );
SOPteta[indice]:=max( min(REGx[0], REGphi[1] ), SOPteta[indice]);
FAM_teta( IDFx[1], IDFphi[1], indice );
SOPteta[indice]:=max( min(REGx[1], REGphi[1] ), SOPteta[indice]);
```

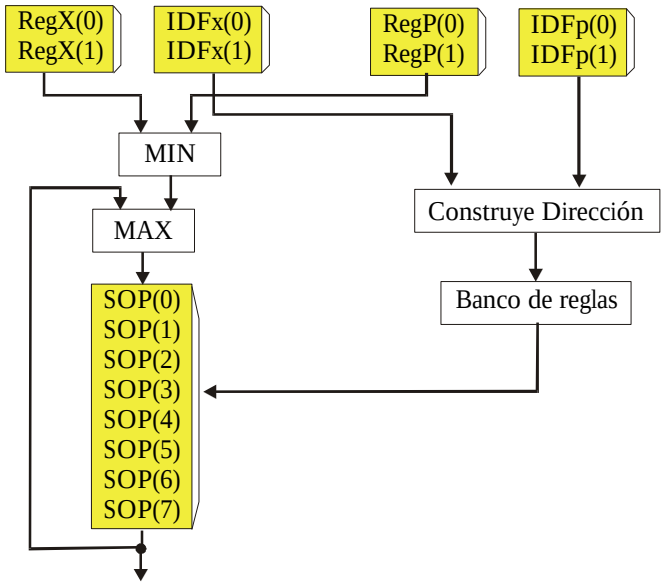

Fig. 9 – Diagrama de materialización del motor de Inferencia por reglas activas

4.3 - DEFUSIFICACION ACTIVA

Los resultados de la inferencia difusa son almacenados en un subconjunto de registros del banco SOP de acuerdo a los consecuentes que fueron activados por las reglas. La cantidad de registros que potencialmente se activan puede calcularse como el producto de la cantidad de funciones de pertenencia que se superponen. Durante la inferencia, al accederse al banco SOP se almacena qué registros tienen información, así durante la defusificación sólo se procesan dichos registros (en lugar de todo el banco o todo el subconjunto de activación potencial).

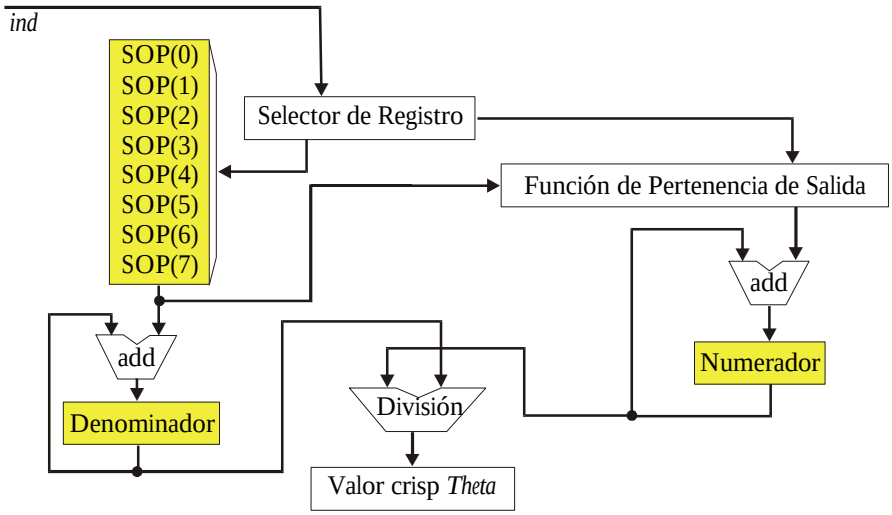

Fig. 10 – Diagrama de materialización del defusificador activo

Para materializar este mecanismo (Fig. 10), el bloque "*selector de registro*" genera los índices correctos de acuerdo a la señal *ind*, que selecciona los registros del subconjunto a los que se debe acceder. La estructura que permite este comportamiento puede ser implementada como una pila en la que se almacenan los registros accedidos; luego de acuerdo a lo que *ind* señale obtener dichos valores para ser procesados. Otra posible materialización puede basarse en un mapa de bits que se marca cuando el valor es modificado (tipo *dirty-bit*), que luego durante la defusificación se consulta para determinar las direcciones de los registros del banco.

La mejora de los tiempos de cálculo es directamente proporcional a la cantidad de funciones de pertenencia de salida que tiene cada variable de salida. Para cada variable se obtienen optimizaciones de acuerdo a la diferencia entre el número total de funciones de pertenencia y dos factores definidos como:

- **Reducción Fija**. La diferencia entre la cantidad de funciones de pertenencia y el subconjunto de funciones de pertenencia que se ha activado.
- **Reducción Variable**. La diferencia entre el conjunto de activación potencial y el conjunto de consecuentes que se ha activado realmente. Esta característica sucede debido a un mismo consecuente que se activa por múltiples reglas (lo cual reduce la cantidad de operaciones del subconjunto a calcular). Como promedio este valor es como máximo aproximadamente la mitad del tamaño del conjunto potencial de reglas que se activa.

5. CONCLUSIONES Y TRABAJOS FUTUROS

En este artículo se han presentado diferentes técnicas para materializar FLC sobre plataformas genéricas sobre software estándar. Como aporte, se destaca el concepto de defusificación activa, en plena fase de desarrollo, para lograr optimizar los FLC en hardware a medida de la aplicación. La optimización en la defusificación tiene un impacto significativo en la calidad de los FLC, debido a que permite definir mayor cantidad de conjuntos difusos de salida sin aumentar el tiempo de cálculo.

Los futuros trabajos incluyen un estudio más pormenorizado del método de defusificación, y una comparación más detallada de los márgenes de incremento de velocidad de cálculo por medio de su uso. En las versiones hardware es posible un estudio de paralelización temporal (mediante un pipelining).

REFERENCIAS

[Aco97] Acosta N., Deschamps JP, Sutter G: "*Herramientas para la materialización de controladores difusos µprogramados*", Iberchip, DF Mexico, '97.

[Aco98a] Acosta N., Deschamps JP y Garrido J.: *'Optimized active rule fuzzy logic custom controller*', International Symposium on Engineering of Intelligent Systems (EIS'98), organizado por la IFAC (International Federation of Automatic Control), Tennerife, España. Feb'98.

[Aco98b] Acosta N., Garrido J. y Deschamps JP: *'Segment representation for membership functions*', International Symposium on Engineering of Intelligent Systems (EIS'98), organizado por la IFAC (International Federation of Automatic Control), Tennerife, España. Feb'98.

[Cos95] Costa A., De Gloria A., P.Faraboschi, A.Pagni and G.Rizzotto, "*Hardware Solution for Fuzzy Control*", Proceedings of the IEEE, vol. 83, nº 3, March 1995, pp. 422 - 434.

[Chi92] Chiueh T. C., "*Optimization of Fuzzy Logic Inference Architecture*", Computer, pp. 67-71, May 1992.

[Des95] Deschamps J.-P. y Martínez Torre J.I., "*Metodología para la materialización de algoritmos borrosos: de una especificación de alto nivel a un ASIC*", ESTYLF'95, Murcia, sep.'95, pp. 251-256.

[Des96] Deschamps J.-P. y Acosta N., "*Algoritmo de optimización de funciones reticulares aplicado al diseño de controladores difusos*", 25as JAIIO, Buenos Aires, sep.'96.

[Hun95] Hung L., "*Dedicated digital fuzzy hardware*", IEEE Micro, v15, nº 4, Aug 95, pp.31-39.

[Ike91] Ikeda H., Hiramoto Y. and Kisu N., "*A Fuzzy Inference Processor with an Active-Rule-Driven Architecture*", in Symposium on VLSI Circuits, Oiso, Japan, May 1991.

[Kos92] Kosko B., "*Neural network and fuzzy systems*", Prentice Hall, 1992.

[Men95] Mendel M., "*Fuzzy logic systems for engineering: a tutorial*", Proc. of the IEEE, vol. 83,nº 3, March 1995, pp. 345-377.

[Ngu89] "*The truck backer-upper: an example of self-learning in neural networks*", Nguyen D. Y Widrow B. Proceedings of International Joint Conference on Neural Networks (IJCNN'89), vol. 2, pp: 357-363, June 1989.

[Sas93] Sasaki Mamuro, Fumio Ueno & Takahiro Inoue, "*7.5 MFLIPS fuzzy microp. using SIMD and logic-in-memory structure*", Proc. IEEE Int.Conf. Fuzzy Systems, pp.527-534, '93.

[Tog86] Togai Masaki and Watanabe Hiroyuki, "*Expert system on a chip: an engine for real-time approximate reasoning*", IEEE EXPERT, vol.1, nº3, pp.55-62, Aug'86.

[Ung93] Ungering A.P., Thuener Karsten and Goser Karl, "*Architecture of a PDM VLSI fuzzy logic controller with pipelining and optimized chip area*", Proc. IEEE Int. Conf. on Fuzzy Systems, pp. 447-452, 1993.

[Wat90] Watanabe Hiroyuki, Dettlof D. and Kathy Yount, "*A VLSI Fuzzy Logic Controller with Reconfigurable, Cascadable Architecture*", IEEE Journal on Solid-State Circuits, vol. 25, nº 2, April 1990.