IMPLEMENTACIÓN DE PROCESADORES DIFUSOS TIPO-2 DE INTERVALO SOBRE EL DSP TMS320C6711 DE TEXAS INSTRUMENTS

Luis G. Marín[1] y Miguel A. Melgarejo[2]

Resumen: *Este artículo presenta una aproximación a la implementación hardware de sistemas difusos tipo-2 de intervalo sobre una clase particular de procesador de señales digitales. La implementación hardware de técnicas de inteligencia computacional se hace necesaria para tratar aplicaciones portables en las cuales restricciones de tiempo de cómputo y área deben ser satisfechas. Se considera un modelo computacional que incluye fuzzificación singleton, motor de inferencia Mamdani y los algoritmos de reducción de tipo más rápidos reportados en la literatura. Los resultados muestran que hay un error pequeño entre la salida de los procesadores implementados en el DSP y la salida de una solución software. Además, la evidencia experimental indica que se puede obtener tiempos de inferencia difusa de algunos milisegundos sobre esta clase de plataformas para sistemas difusos tipo-2 de intervalo relativamente complejos.*

Palabras clave: *Lógica difusa, Sistemas difusos tipo-2, Hardware difuso, Sistemas empotrados, Procesadores de señales digitales.*

Abstract: *This article presents a hardware implementation approach for interval type-2 fuzzy systems over a particular class of Digital Signal Processor. Hardware implementation of computational intelligence techniques is required for dealing with portable applications in which computing time and area constrains must be satisfied. A computational model composed of singleton fuzzification, mamdani inference engine and type reduction based on the fastest algorithms reported in literature is considered. Results show that there is a small error between the output of implemented processors and the output of software solution. Besides, experimental evidence points out that fuzzy inference times of some milliseconds can be achieved over this kind of platforms for relatively complex interval type-2 fuzzy systems.*

Key words: *Fuzzy logic, Type-2 fuzzy systems, Fuzzy hardware, Embedded Systems, Digital signal Processors.*

1 Ingeniero Electrónico, Facultad de Ingeniería, Universidad de la Amazonia, lgmarin@uniamazonia.edu.co

2 Magíster en Ingeniería Electrónica, Laboratorio de Automática, Microelectrónica e Inteligencia Computacional, Facultad de Ingeniería, Universidad Distrital Francisco José de Caldas, mmelgarejo@udistrital.edu.co

1. INTRODUCCIÓN

A finales de la década de los noventa surge un nuevo paradigma de los sistemas difusos, que introduce J. Mendel, al que llamó lógica difusa tipo-2. La lógica difusa tipo-2 ha venido ganando interés para el tratamiento de aplicaciones que tienen un comportamiento no lineal y estocástico (Karnik y Mendel, 1998; Mendel, 2001). Las aplicaciones realizadas con lógica difusa tipo-2 demuestran que éste paradigma ofrece mejores resultados que la lógica difusa tipo-1 (Mendel, 2007; Melgarejo y Peña, 2007).

Por ejemplo, en el área de control Lynch *et al.*, (2005), propusieron un sistema de control de velocidad para un propulsor marino y un motor diesel de tracción. El control difuso tipo-2 presentó una mejor respuesta que la lograda con un control PID y un control difuso tipo-1 utilizando menos reglas difusas. Por otro lado, Tan W. *et al.*, (2007) propusieron un sistema

difuso tipo-2 para la clasificación del latido arrítmico ECG. Tres tipos de señales de ECG, a saber, el ritmo sinusal normal (NSR), fibrilación ventricular (VF) y taquicardia ventricular (VT), se consideraron. Se demostró que el sistema difuso tipo-2 tiene una precisión del 90,91% para los eventos VT, 84% para los eventos VF y 100% para los eventos NSR. Yildirim y Yuksel (2007), realizaron un proyecto para el filtrado de imágenes digitales basado en lógica difusa tipo-2 para preservar y restaurar las imágenes corrompidas por ruido impulsivo. Los resultados experimentales presentaron un rendimiento superior con relación a técnicas convencionales y la eliminación del ruido en la imagen, al mismo tiempo que preserva eficazmente la información útil en la imagen.

La implementación de los sistemas difusos depende de la aplicación, el alcance de los mismos y las técnicas de implementación. Según el domino de aplicación, se define tanto la complejidad como la velocidad de procesamiento del sistema de inferencia difusa. Establecidas estas dos características, se define entonces la forma como el sistema se implementará. Note que la justificación de una solución hardware, en principio, se da cuando la aplicación requiere de un espacio reducido para la implementación. Sin embargo, otros elementos pueden hacer necesaria ésta solución, como por ejemplo, velocidad, consumo de potencia o incluso volumen de producción (Costa *et al.*, 1995; Baturone *et al.*, 2000).

Con la aparición de las primeras aplicaciones de los sistemas difusos, estos dejaron de ser una formulación teórica y pasaron a convertirse en una herramienta útil en la ingeniería. Se destacan trabajos donde proponen la implementación de sistemas difusos sobre diferentes plataformas, tales como: procesadores de propósito general, FPGA (Field Programmable Gate Array), ASIC (Circuito Integrado para Aplicaciones Específicas), Microcontroladores, DSC (Controladores de Señales Digitales) y DSP (Procesadores de Señales Digitales).

Hoy en día, el dominio de aplicación de los sistemas difusos tipo-2 ha crecido, tal como sucedió con sus antecesores (Coupland y Jhon, 2007). Se pueden mencionar varios trabajos relacionados con la implementación hardware de sistemas difusos tipo-2. Por ejemplo, Lin *et al.*, (2005) diseñaron un control basado en lógica difusa tipo-2 para conversión DC-DC. Los resultados experimentales mostraron que el control propuesto con lógica difusa tipo-2 es robusto frente a las variaciones del voltaje de entrada y resistencia de carga.

Por otro lado, Melgarejo y Peña (2004,2007), implementó un primer procesador difuso tipo-2 de intervalo en una FPGA el cual fue validado con la implementación de filtros difusos adaptativos y a través de estos filtros se llevó a cabo la ecualización de un canal de comunicaciones no lineal y variante en el tiempo. Sierra y Bulla (2008), realizaron una aproximación al desarrollo de un procesador difuso tipo-2 en un microcontrolador de 8 bits mostrando que es posible lograr resultados apropiados para ciertas aplicaciones. Finalmente, Leottau y Melgarejo (2010), realizaron la implementación de procesadores difusos tipo-2 de intervalo sobre DSC (Digital Signal Controller), donde se presenta la influencia de parámetros como la velocidad, uso de memoria, precisión y resolución en el rendimiento del procesador.

Lo anterior refleja, que la implementación hardware de sistemas difusos tipo-2 de intervalo (IT2-FLS, por sus siglas en inglés) serían de utilidad tanto en la industria como en la electrónica de consumo teniendo en cuenta que pueden brindar una solución adecuada en diferentes campos de aplicación (Wang, 1997; Babuska, 1998; Mendel, 2001). De igual manera, es de interés impulsar la implementación hardware de IT2-FLS en el ámbito académico, para que hagan parte de algunas áreas que forman los contenidos curriculares de las facultades de ingeniería en las universidades de Colombia y el mundo. Además, se destaca que en las últimas décadas, los dispositivos digitales con amplia escala de integración se han convertido en soluciones adecuadas para aplicaciones con

cierta complejidad (Ackenhusen, 1999). Como ejemplo, se encuentran los procesadores de señales digitales (DSP), los cuales permiten manejar una relación adecuada entre velocidad de proceso y capacidad de memoria (Barrero *et al.*, 2005).

Este trabajo presenta un estudio comparativo de la implementación hardware de diferentes sistemas difusos tipo-2 de intervalo (IT2-FLS) sobre la plataforma DSP TMS320c6711. Para la validación se implementaron los IT2-FLS haciendo uso de una toolbox de IT2-FLS (Castro *et al.*, 2007) y el DSP. Se implementaron IT2-FLS con 4, 9 y 25 reglas. En la reducción de tipo se utiliza el algoritmo iterativo con condición de parada (IASCO) (Melgarejo *et al.*, 2008) y el algoritmo mejorado de Karnik-Mendel (EKM) (Dongrui y Mendel, 2007). Para los conjuntos difusos tipo-2 de intervalo (IT2-FS, por sus siglas en inglés) de entrada se maneja una discretización de 128 niveles y para los conjuntos difusos tipo-2 de intervalo (IT2-FS) de salida se emplean diferentes niveles de discretización, parámetro de interés que se evalúa en la implementación.

Este artículo está organizado de la siguiente manera: la sección II presenta el modelo computacional propuesto para los IT2-FLS, en la sección III se describe los detalles de implementación de los IT2-FLS. Finalmente se exponen las conclusiones en la sección IV.

2. MODELO COMPUTACIONAL

Los IT2-FLS propuestos en éste artículo están caracterizados por *N* entradas, $M_i M_i$ conjuntos difusos tipo-2 de intervalo (IT2-FS) por entrada, $NR = M_1 \times M_2 \times \cdots \times M_N$ $NR = M_1 \times M_2 \times \cdots \times M_N$ reglas y *Mc* conjuntos difusos tipo-2 de intervalo (IT2-FS) en la variable de salida. D_a y D_c puntos de discretización para los antecedentes y consecuentes respectivamente. La estructura general de un sistema difuso tipo-2 se muestra en la figura 1. A continuación, se describen las cuatro partes consideradas para éste modelo computacional.

A. Fusificador

Los IT2-FLS que emplean fusificación singleton consideran incertidumbre acerca del antecedente y de los consecuentes en las reglas, pero no consideran explícitamente incertidumbre en las entradas al sistema (Mendel, 2007). Este método de fusificación es el empleado en este trabajo. La fusificación singleton es un método computacionalmente sencillo y especial para implementaciones hardware como los DSP, dado que simplifica el cálculo de la inferencia difusa para cualquier tipo de función de pertenencia (Mendel, 2001).

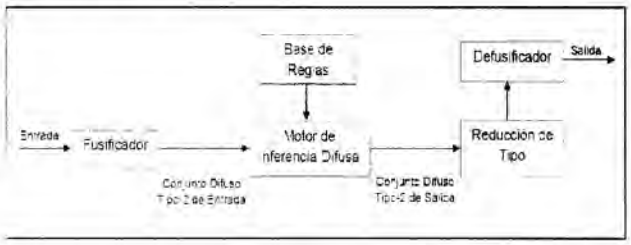

Figura 1. Estructura de un Sistema Difuso Tipo-2 (Tomado de Mendel, 2001).

A la unión de todas las funciones de pertenencia primarias de un conjunto difuso tipo-2 se le conoce como huella de incertidumbre (*FOU*) y estas huellas son caracterizadas por las funciones de pertenencia definidas para cada variable de entrada al sistema (Mendel, 2001). En la figura 2, se muestra un ejemplo de una *FOU* derivada de una función de pertenencia primaria tipo Gaussiana.

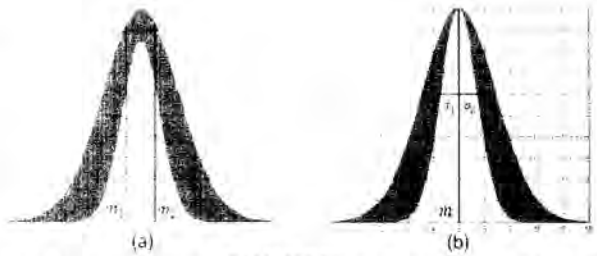

Figura 2. Ejemplos de $FOUFOU$ derivadas de una función de pertenencia primaria tipo Gaussiana (Tomado de Mendel, 2001).

Los valores que describen las funciones de pertenencia tipo-2 de intervalo para las variables

de entrada se almacenan en memoria. El tamaño de memoria necesario para almacenar éstos valores está dado por $2 \times (M_1 + M_2 + \cdots + M_N) \times D_a$ $2 \times (M_1 + M_2 + \cdots + M_N) \times D_a$ palabras. En la fusificación, solo es necesario hacer la lectura de la memoria para acceder a los valores de pertenencia superior e inferior y almacenarlos en un arreglo de tamaño $2 \times (M_1 + M_2 + \cdots + M_N)$ $2 \times (M_1 + M_2 + \cdots + M_N)$ palabras.

Esta estrategia de fusificación no impone limitaciones en el tipo y el número de funciones de pertenencia. Pero, la demanda de recursos de memoria aumenta rápidamente cuando el número de entradas o funciones de pertenencia crece. No obstante, este método es práctico cuando el número de entradas y salidas y sus grados de cuantización son bajos, de lo contrario el número total de palabras de almacenamiento sería demasiado grande (Mendel, 2001).

B. Base de Reglas

La base de reglas posee una estructura que no depende de la naturaleza de los conjuntos difusos. Por lo tanto, las reglas en un IT2-FLS se representan en la misma forma que en el caso de los sistemas difusos tipo-1 (Baturone *et al.*, 2000; Wang, 1997).

La información relacionada con la base de reglas se almacena previamente en memoria. En un arreglo de tamaño de memoria $NR \times N$ $NR \times N$ palabras se almacena la forma de cómo se relacionan los conjuntos difusos tipo-2 de intervalo de cada una de las entradas. A lo que se llama antecedentes de las reglas. Como es posible que estos antecedentes se conecten utilizando una unión o intersección difusa, esta información también es almacenada en un arreglo de tamaño de memoria *NR* palabras. Se llama *M* al número máximo de antecedentes que mapean al mismo consecuente. Por lo tanto, la información de cómo se mapean los antecedentes con los consecuentes, se almacena en un arreglo de memoria de tamaño $M_c \times M$ $M_c \times M$.

Por ejemplo, si se tiene un IT2-FLS con la siguiente base de reglas:

R^1: *Si* x_1 *es* $A1$ *y* x_2 *es* $B1$, *entonces y es* S^1 (1)
R^2: *Si* x_1 *es* $A1$ *y* x_2 *es* $B2$, *entonces y es* S^2 (2)
R^3: *Si* x_1 *es* $A2$ *y* x_2 *es* $B1$, *entonces y es* S^2 (3)
R^4: *Si* x_1 *es* $A2$ *y* x_2 *es* $B2$, *entonces y es* S^3 (4)

La información almacenada en la memoria del DSP relacionada con la base de reglas, entonces es:

Ante [4] [2] = {{1,1}, {1,2}, {2,1}, {2,2}} (5)
Cx [4]= {1, 1, 1, 1} (6)
Conse [3][2]={{1},{2,3},{4}} (7)

Los antecedentes de las reglas se codifican en el arreglo de la expresión (5) en orden ascendente. Por lo tanto, el primer par de elementos del arreglo, que corresponden a la primera regla, indica que el sistema tiene dos variables de entrada y que para ese caso particular el primer conjunto difuso tipo-2 de intervalo de la primera entrada se relaciona con el primer conjunto difuso tipo-2 de intervalo de la segunda entrada, tal como se puede ver expreso en (1).

La expresión (6), en este caso indica que los antecedentes están conectados con una operación t-norma. Esto se evidencia porque el vector llamado *Cx* está codificado solamente con unos. Si el vector tuviera como codificación un dos, esto indicaría que los antecedentes de esa regla en particular se conectan con una operación s-norma.

Por último, la expresión (7) brinda la información correspondiente al mapeo de los antecedentes con los conjuntos consecuentes. En este ejemplo, se está indicando que el antecedente de la regla 1 mapea al consecuente 1, que los antecedentes de las reglas 2 y 3 mapean al consecuente 2 y el antecedente de la regla 4 mapea al consecuente 3.

C. Motor de Inferencia

En este modelo computacional se utiliza motor de inferencia Mamdani, ya que su simplicidad hace que sea uno de los métodos de inferencia

más usados en aplicaciones hardware (Baturone *et al.*, 2000; Mendel, 2001; Camelo y Téllez, 2007). El motor de inferencia usa la t-norma mínimo en la implicación y la operación s-norma máximo en la agregación de las reglas.

Con los valores de pertenencia superior e inferior obtenidos del proceso de fusificación y la información previamente almacenada sobre la base de reglas, se calcula y almacena el nivel de activación superior e inferior de los antecedentes de cada regla en un arreglo de tamaño de memoria $2 \times NR\, 2 \times NR$ palabras. Seguidamente, considerando que en algunas aplicaciones, varios antecedentes pueden mapear al mismo consecuente, se realiza la agregación de consecuentes empleando la operación s-norma máximo y su resultado es almacenado en un arreglo de tamaño de memoria $2 \times M_c 2 \times M_c$ palabras. Los valores almacenados en éste arreglo son entonces los valores de los antecedentes de las reglas.

Los valores que describen las funciones de pertenencia tipo-2 de intervalo para la variable de salida, se almacenan previamente en un arreglo de memoria de tamaño $2 \times M_c \times D_c 2 \times M_c \times D_c$ palabras. Continuando con el proceso del motor de inferencia, se calcula el nivel de activación entre los valores de los antecedentes y todos los valores que describen la función de pertenencia del conjunto consecuente respectivo. Este proceso se realiza tanto para los niveles de activación superior como inferior de cada una de las reglas. El resultado de la agregación de las reglas se va almacenando en un arreglo de tamaño de memoria $2 \times D_c 2 \times D_c$ palabras. Las primeras $D_c D_c$ posiciones de memoria para la huella superior y las otras $D_c\, D_c$ posiciones de memoria para la huella inferior, así, queda conformado el conjunto difuso tipo-2 de intervalo de salida.

Como se puede analizar el proceso del motor de inferencia es computacionalmente costoso. La carga computacional se puede reducir mediante la reducción de los niveles de discretización en los consecuentes (Sierra y Bulla, 2008; Leottau y Melgarejo, 2010).

D. Procesamiento de Salida

El procesamiento de salida se hace en dos pasos. Primero, el conjunto difuso tipo-2 de intervalo de salida pasa por un reductor de tipo, en donde se obtiene un conjunto difuso tipo-1. Luego, éste conjunto difuso tipo-1 se defusifica para obtenerse el valor puntual de salida (Mendel, 2001).

El reductor de tipo calcula el centroide del conjunto difuso tipo-2 de intervalo de salida. Este conjunto de salida es una colección de infinitos conjuntos difusos tipo-1 interiores, cada uno con su correspondiente centroide (Mendel, 2007). Por lo tanto el centroide de un conjunto difuso tipo-2 de intervalo se puede ver como un intervalo $(C_L, C_R C_L, C_R)$ que contiene todos los centroides de todos los conjuntos difusos tipo-1 interiores.

En cuanto a los procedimientos computacionales para la reducción de tipo, se han desarrollado varios trabajos que buscan mejorar los tiempos de procesamiento. El primero de ellos se conoce como procedimiento iterativo de Karnik y Mendel (KM) (Mendel, 2001), el segundo es el algoritmo mejorado de Karnik-Mendel (EKM) (Dongrui y Mendel, 2007) y un tercer trabajo es el propuesto por Melgarejo llamado algoritmo iterativo con condición de parada (IASCO) (Melgarejo, *et al.*, 2008). En el presente trabajo se implementaron los algoritmos IASCO y EKM.

Por último, una vez que el reductor de tipo ha encontrado el intervalo $(C_L, C_R)\,(C_L, C_R)$, se lleva a cabo la defusificación como la media de dicho intervalo, siendo este valor puntual la salida del IT2-FLS.

3. IMPLEMENTACIÓN DE LOS IT2-FLS

El número seleccionado de entradas y conjuntos difusos por cada entrada, está basado en el número promedio utilizado para diferentes aplicaciones hardware. Por ejemplo, en las aplicaciones tales como: sistemas de posición de vehículos, sistemas de control de fluidos, predictor de series de tiempo caóticas, sistemas

de control de velocidad y posición de motores, control de sistemas térmicos, sistemas de control en procesos químicos, entre otros, las entradas están entre dos y cuatro y el número de conjuntos difusos por entrada no es mayor a siete (Camelo y Téllez, 2009; Baturone *et al.*, 2000). Lo anterior debido a que si el número de conjuntos difusos es mayor y si se considera una base de reglas completa el número de reglas crece exponencialmente complicando la implementación del sistema y haciendo lenta su operación. Por otro lado, cuando el número de conjuntos difusos por entrada y de reglas excede cierto punto, la significancia lingüística de las reglas se pierde. En conclusión, la elección del número de términos lingüísticos por variables debe ser un compromiso entre simplicidad de implementación y detalle de descripción (Baturone *et al.*, 2000).

Los IT2-FLS implementados se caracterizaran por tener dos variables de entrada con 128 puntos de discretización, 4, 9 y 25 reglas y una variable de salida. En la reducción de tipo se emplean los algoritmos IASCO y EKM. En los conjuntos consecuentes se emplean 32, 128 y 512 puntos de discretización. Un total de 18 IT2-FLS se implementan en la plataforma DSK C6711. Las características de los IT2-FLS se resumen en la tabla1.

Tabla 1. Características de los IT2-FLS

Número de Reglas	Número de IT2-FS por entrada	Número de IT2-FS en la salida
4	2	4
9	3	5
25	5	5

A continuación se describen las características del sistema de desarrollo utilizado para la implementación, seguidamente, se muestra la metodología empleada, el procedimiento y pruebas y finalmente, los resultados y discusión.

A. Sistema de Desarrollo DSP

Los IT2-FLS se implementan en el kit de inicio de desarrollo (DSK) C6711. El DSK está compuesto básicamente por el procesador de punto flotante TMS320C6711, dos relojes, memoria SDRAM externa de 16 Mbytes, memoria Flash externa de 128Kbytes, un ADC de 16bits y un controlador JTAG, que provee una fácil emulación y depuración (Barrero *et al.*, 2005).

El procesador tiene seis unidades aritmético-lógicas (ALU's) y dos multiplicadores. Cuenta con ocho bits de guarda para evitar el *overflow* aritmético y 32 registros de 32 bits de propósito general. El ciclo máquina de este procesador es de 6.7ns y es capaz de ejecutar 8 instrucciones en un solo ciclo de reloj. Puede manejar buses de datos de 8, 16 y 32 bits y buses de direcciones de 32 bits. Tiene una arquitectura de memoria interna basada en L1 y L2. La memoria de caché L1 se divide en dos bloques: 4Kbyte son para memoria de programa y 4Kbyte para memoria de datos. La memoria L2 tiene una capacidad 64Kbyte unificada para programa y datos. La arquitectura interna del procesador TMS320c6711 se puede ver en detalle en el Datasheet (2000).

La memoria externa SDRAM se puede manejar como un banco de memoria de 4M para palabras de 32 bits con un tiempo de acceso de 10ns. En cada aplicación se debe configurar la memoria para indicar como queda distribuida físicamente. Las secciones para memoria de programa, memoria de datos inicializados o constantes y datos no inicializados, se deben definir.

B. Metodología

Los IT2-FLS inicialmente se implementan haciendo uso de una toolbox de IT2-FLS desarrollada por el Instituto Tecnológico de Tijuana (Castro *et al.*, 2007). Una vez validados los modelos computacionales se procedió a escribir el código C apropiado para la plataforma hardware. El código C construido es compilado, ensamblado, depurado y cargado al DSP utilizado el software de evaluación específico Code Composer Studio (CCS). Una comunicación DSP-PC se utiliza por medio del puerto paralelo y el controlador JTAG para poder obtener los datos utilizados en el proceso de validación. Posteriormente, se calcula el Error Cuadrático Medio (ECM) y el Error Absoluto Medio (EAM).

También se registran los tiempos promedios consumidos por el procesamiento de salida. Finalmente, se obtiene el tiempo promedio de la inferencia difusa tipo-2.

C. Procedimiento y pruebas

El procedimiento utilizado para validar los resultados de salida de los IT2-FLS, se describe a continuación:

1. Se generan N=100 pares ordenados como entradas para evaluar el IT2-FLS.
2. El valor de salida obtenido de la toolbox de IT2-FLS con Matlab® se guardan como out_{ref} out_{ref} y el valor de salida del mismo sistema obtenido de la plataforma DSP se guarda como out_{DSP} out_{DSP}.
3. Se calcula el ECM y EAM como sigue:

$$ECM = \frac{1}{N} \cdot \sum_{i=1}^{N} (out_{ref}(i) - out_{DSP}(i))^2$$
$$ECM = \frac{1}{N} \cdot \sum_{i=1}^{N} (out_{ref}(i) - out_{DSP}(i))^2 \quad (8)$$

$$EAM = \frac{1}{N} \cdot \sum_{i=1}^{N} abs(out_{ref}(i) - out_{DSP}(i))$$
$$EAM = \frac{1}{N} \cdot \sum_{i=1}^{N} abs(out_{ref}(i) - out_{DSP}(i)) \quad (9)$$

Este procedimiento se repite para los 18 IT2-FLS implementados.

Para el cronometraje de los tiempos de inferencia de los IT2-FLS, se emplea el siguiente procedimiento:

1. Se genera un número aleatorio como entrada al IT2-FLS.
2. Se registran los tiempos consumidos por el procesamiento de salida y el tiempo de la inferencia difusa tipo-2.
3. Se repite 1 y 2 para 10 números aleatorios de entrada.
4. Se calculan los tiempos promedios y las desviaciones estándar.

D. Resultados y discusión

Los resultados obtenidos se presentan de la siguiente manera: en las tablas 2, 3 y 4 se presentan los ECM y EAM de los 18 IT2-FLS implementados, en la tabla 5, 6 y 7 se presentan los tiempos promedios consumidos por el procesamiento de salida y finalmente, en la tabla 8, 9 y 10 se presenta el tiempo promedio demandado para una inferencia difusa. A partir de las tablas, se puede notar que:

Los dos algoritmos de reducción de tipo tienen la misma precisión. El ECM y EAM son valores pequeños para los 18 IT2-FLS implementados. Esto indica que la diferencia en promedio es pequeña entre los valores obtenidos de la toolbox y el DSP. Al incrementar el nivel de discretización de los conjuntos consecuentes la precisión se mejora aproximadamente en 0.75% con respecto al anterior.

Tabla 2. Resultados del ECM y EAM para un IT2-FLS de 4 reglas.

Niveles de Discretización del Consecuente	Algoritmo de Reducción de Tipo			
	IASCO		EKM	
	ECM	EAM	ECM	EAM
32	0.0022	0.0279	0.0022	0.0279
128	0.0022	0.0277	0.0022	0.0277
512	0.0022	0.0277	0.0022	0.0277

Tabla 3. Resultados del ECM y EAM para un IT2-FLS de 9 reglas.

Niveles de Discretización del Consecuente	Algoritmo de Reducción de Tipo			
	IASCO		EKM	
	ECM	EAM	ECM	EAM
32	0.0015	0.0304	0.0015	0.0304
128	0.0015	0.0298	0.0015	0.0298
512	0.0014	0.0296	0.0014	0.0296

Tabla 4. Resultados del ECM y EAM para un IT2-FLS de 25 reglas.

Niveles de Discretización del Consecuente	Algoritmo de Reducción de Tipo			
	IASCO		EKM	
	ECM	EAM	ECM	EAM
32	0.0094	0.0578	0.0094	0.0578
128	0.0092	0.0564	0.0092	0.0564
512	0.0092	0.0562	0.0092	0.0562

El tiempo demandado por el procesamiento de salida depende de los niveles de discretización de

los consecuentes y no del número de reglas del IT2-FLS. Por el contrario, el tiempo de inferencia difusa depende del número de reglas del IT2-FLS y del valor *DC* utilizado en los conjuntos consecuentes.

El aumento de los niveles de discretización en los conjuntos del consecuente es un factor influyente en el tiempo de inferencia difusa. Por ejemplo, el IT2-FLS de 9 reglas con algoritmo de procesamiento de salida IASCO, con *DC=32* es *3.7* veces más rápido que el IT2-FLS con *DC=128* y *15* veces más rápido que el IT2-FLS con DC=512 y en los tres sistemas se obtiene la misma precisión.

En el procesamiento de salida el algoritmo EKM converge más rápido que el algoritmo IASCO y la desviación estándar de EKM es más pequeña que la de IASCO, indicando que los tiempos de procesamiento están menos dispersos sin importar los datos de entrada. Estos resultados se deben a que el algoritmo IASCO requiere de más operaciones de división que el algoritmo EKM y el DSP empleado solamente tiene multiplicadores hardware. Por lo tanto, las divisiones se calculan por medio de las subrutinas de software, que suelen ser más lentas que las unidades de hardware dedicado.

El mayor tiempo de inferencia difusa tipo-2 se obtiene en el caso de 25 reglas con 512 puntos de discretización en el conjunto consecuente. Este tiempo está alrededor de 3.1523ms para el algoritmo IASCO y de 1.8323ms para el algoritmo EKM. Pero es importante resaltar que éste mismo sistema con 32 puntos de discretización en el conjunto consecuente consume 0.2214ms con algoritmo IASCO y 0.1537 con algoritmo EKM. Esto sin mayor pérdida precisión con respecto al sistema con 512 puntos de discretización.

Por lo tanto, teniendo en cuenta los tiempos de inferencia para esta clase particular de DSP, se puede considerar la implementación de aplicaciones donde la captura de datos se realice desde un ADC con frecuencias de muestreo entre 5Khz y 10kHz para lograr procesamiento en tiempo real. Comparativamente hablando, Sierra y Bulla (2008), presentan la implementación de un IT2-FLS sobre un microcontrolador con frecuencia de operación de 32Mhz, mostrando que los tiempos de inferencia están en el orden de los 5ms con 32 puntos de discretización en el consecuente. Leottau y Melgarejo (2010), presentan la implementación de IT2-FLS sobre un DSC con frecuencia de operación de 32Mhz, mostrando que los tiempos de inferencia están en el orden de los 4.8ms con 100 puntos de discretización en el consecuente. Por lo tanto, se puede decir que los tiempos obtenidos sobre éste DSP son adecuados para ciertas aplicaciones.

Tabla 5. Resultados Tiempo del Procesamiento de Salida para un IT2-FLS de 4 reglas.

Niveles de Discretización del Consecuente	Algoritmo de Reducción de Tipo			
	IASCO		EKM	
	Tiempo Promedio (ms)	Desviación Estándar (ms)	Tiempo Promedio (ms)	Desviación Estándar (ms)
32	0.0579	0.0358	0.0201	0.0042
128	0.2186	0.1424	0.0326	0.005
512	0.8634	0.5694	0.0806	0.0163

Tabla 6. Resultados Tiempo del Procesamiento de Salida para IT2-FLS de 9 reglas.

Niveles de Discretización del Consecuente	Algoritmo de Reducción de Tipo			
	IASCO		EKM	
	Tiempo Promedio (ms)	Desviación Estándar (ms)	Tiempo Promedio (ms)	Desviación Estándar (ms)
32	0.0814	0.0259	0.0195	0.003
128	0.3193	0.1088	0.0455	0.0071
512	1.2634	0.4369	0.1128	0.0181

Tabla 7. Resultados Tiempo del Procesamiento de Salida para un IT2-FLS de 25 reglas.

Niveles de Discretización del Consecuente	Algoritmo de Reducción de Tipo			
	IASCO		EKM	
	Tiempo Promedio (ms)	Desviación Estándar (ms)	Tiempo Promedio (ms)	Desviación Estándar (ms)
32	0.0894	0.0067	0.0209	0.0032
128	0.3568	0.0271	0.0375	0.0070
512	1.3783	0.0783	0.1025	0.0100

4. CONCLUSIONES

Este trabajo ha presentado los resultados de implementación de varios IT2-FLS sobre la plataforma hardware DSP TMS320c6711 de Texas Instruments. Los IT2-FLS se caracterizaron por tener 4, 9 y 25 reglas, se utilizaron tres niveles de discretización de los conjuntos consecuentes y se emplearon los algoritmos IASCO y EKM en el procesamiento de salida.

Para ésta plataforma DSP, se pueden emplear diferentes puntos de discretización en el conjunto consecuente sin pérdida de precisión. Además, se pueden obtener tiempos de inferencia difusa adecuados para ciertas aplicaciones, como por ejemplo en el campo de control empotrado.

Los resultados obtenidos muestran que es posible implementar IT2-FLS en esta clase particular de DSP, con tiempos de procesamiento en el orden de los milisegundos. Siendo un tiempo atractivo para ciertas aplicaciones. Se sugiere la exploración de otros modelos computacionales para la implementación de los IT2-FLS y si es necesario disminuir los tiempos de inferencia difusa tipo-2 se puede considerar la programación de los DSP en código ensamblador.

Finalmente, se considera como trabajo futuro el desarrollo de una metodología para la síntesis automática de IT2-FLS en DSP y la implementación de algunas aplicaciones de interés en el campo de control.

Tabla 8. Resultados Tiempo de Inferencia Difusa para un IT2-FLS de 4 reglas.

Niveles de Discretización del Consecuente	Algoritmo de Reducción de Tipo			
	IASCO		EKM	
	Tiempo Promedio (ms)	Desviación Estándar (ms)	Tiempo Promedio (ms)	Desviación Estándar (ms)
32	0.1480	0.0380	0.1090	0.0036
128	0.5399	0.1520	0.3597	0.0186
512	2.1599	0.6008	1.3853	0.1093

Tabla 9. Resultados Tiempo de Inferencia Difusa para un IT2-FLS de 9 reglas.

Niveles de Discretización del Consecuente	Algoritmo de Reducción de Tipo			
	IASCO		EKM	
	Tiempo Promedio (ms)	Desviación Estándar (ms)	Tiempo Promedio (ms)	Desviación Estándar (ms)
32	0.1962	0.0276	0.1374	0.0056
128	0.7324	0.1153	0.4664	0.0233
512	2.9281	0.4677	1.7971	0.1113

Tabla 10. Resultados Tiempo de Inferencia Difusa para un IT2-FLS de 25 reglas.

Niveles de Discretización del Consecuente	Algoritmo de Reducción de Tipo			
	IASCO		EKM	
	Tiempo Promedio (ms)	Desviación Estándar (ms)	Tiempo Promedio (ms)	Desviación Estándar (ms)
32	0.2214	0.0099	0.1537	0.0075
128	0.7988	0.0368	0.4797	0.0295
512	3.1523	0.1278	1.8323	0.1419

REFERENCIAS

Ackenhusen J. (1999). *Real-Time Signal Processing: The design and implementation of signal processing systems*. New York, Prentice Hall.

Babuska R. (1998). *Fuzzy Modeling for Control.* United States of America, Kluwer Academic Publishers.

Barrero F, Ruiz M, Marín T. (2005). *Procesadores Digitales de Señal de Altas Prestaciones de Texas Instruments.* España, McGraw-Hill.

Baturone I, Barriga A, Jimenez C, Sanchez S, Lopez D. (2000).*Microelectronic Design of Fuzzy Logic-Based Systems*". Sevilla, España, CRC Press.

Camelo Z, Téllez A. (2009). *Diseño e Implementación de una Herramienta Software para la Síntesis Automática de Procesadores Difusos con Orientación a la Arquitectura de la*

Familia DSPIC33F. Trabajo de grado (Ingeniería). Bogotá, Colombia, Universidad Distrital Francisco José de Caldas.

Castro J, Castillo O, Melin P. (2007). An Interval Type-2 Fuzzy Logic Toolbox for control Applications. Proc. FUZZ-IEEE. 1-6.

Costa A, De Gloria A, Faraboschi P, Pagni A, Rizzotto G. (1995). Hardware solutions for fuzzy control. *Proceedings of the IEEE, vol. 83.* 422-434.

Coupland S, Jhon R. (2007). Type-2 Fuzzy Logic: A Historical View. *IEEE Computational Intelligence Magazine, Vol. 2, No 1.* 57 – 62.

Datasheet TMS320c6711. (2000). SPRS073D Texas Instruments, August 2000.

Dongrui W, Mendel J. (2007). Enhanced Karnick-Mendel Algorithms for Interval Type-2 Fuzzy Sets and Systems. *Proceedings of Nafips.* 184-189.

Karnik N, Mendel J. (1998). Introduction to Type-2 Fuzzy Logic Systems. *Fuzzy Systems FUZZ-IEEE 98.* 915–920.

Leottau L, Melgarejo M. (2010). A simple approach for designing a type-2 fuzzy controller for a mobile robot application. *Proc. of the Annual Meeting of the North American Fuzzy Information Processing Society (NAFIPS).*1-6.

Leottau L, Melgarejo M. (2010). A Methodological Proposal for Implementing Interval Type-2 Fuzzy Processors over Digital Signal Controllers. *Journal of applied computer science methods, v.*2.61 – 81.

Lin P, Hsu C, Lee T. (2005). Type-2 Fuzzy Logic Controller Design for Buck DC-DC Converters. *IEEE International Conference on Fuzzy Systems.* 365- 370.

Lynch C, Hagras H, Callaghan V. (2005). Embedded Type-2 FLC for Real-Time Speed Control of Marine & Traction Diesel Engines. *IEEE International Conference on Fuzzy Systems.* 347- 352.

Melgarejo M, Peña C. (2004). Pro-Two: a hardware based platform for real time type-2 fuzzy inference. *IEEE International Conference on Fuzzy Systems,* 977- 982.

Melgarejo M, Peña C. (2007). Implementing interval type-2 fuzzy processors. *IEEE computational intelligence magazine, Vol. 2, No 1.* 63-71.

Melgarejo M, Duran K, Bernal H. (2008). Improved Iterative Algorithm for Computing the Generalized Centroid of an Interval Type-2 Fuzzy Set. *Proceedings of Nafips.* 1-5.

Mendel J. (2001). *Uncertain Rule-Based Fuzzy Logic Systems, introduction and new directions.* Los Angeles, CA, Prentice Hall PTR, Upper Saddle River.

Mendel J. (2007). Type-2 Fuzzy Sets and Systems: An Overview. *IEEE Computational Intelligence Magazine, Vol. 2.* 20 – 29.

Sierra G, Bulla J. (2008). Implementing a simple microcontroller-based interval type-2 fuzzy processor. *Procceedings of MWSCAS.* 69 – 72.

Tan W, Foo F, Chua T. (2007). Type-2 Fuzzy System for ECG Arrhythmic Classification. *Proc. FUZZ-IEEE 2007.* 859 – 864.

Wang L. (1997). *A course on Fuzzy Systems and Control.* New Jersey, Prentice Hall PTR, Upper Saddle River.

Yildirim M, Yuksel M. (2007). A Type-2 Fuzzy Logic Filter for Detail-Preserving Restoration of Digital Images Corrupted by Impulse Noise. *FUZZ-IEEE 2007.* 1-6.