UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO
Escuela Nacional de Estudios Profesionales
" ARAGON "

ESTRUCTURAS LOGICAS PARA SISTEMAS DE CONTROL FUZZY LOGIC

T E S I S

QUE PARA OBTENER EL TITULO DE:

INGENIERO MECANICO ELECTRICISTA

P R E S E N T A N:

MEDINA ZURITA JORGE ALBERTO
VALERA ESPINOSA JORGE

Asesor Ing. David J. González Maxinez.

ENEP
ARAGON

SAN JUAN DE ARAGON, EDO. DE MEXICO **1994.**

UNAM – Dirección General de Bibliotecas Tesis Digitales Restricciones de uso

CONTENIDO

CAPITULO 2
METODOLOGIA DE DISEÑO PARA SISTEMAS FUZZY LOGIC.

CAPITULO 3
ESTRUCTURAS LOGICAS FUZZY LOGIC

INTRODUCCIÓN

El empleo de los Sistemas Automáticos de Control en los medios de producción ha logrado que la calidad y costos de los productos sean mas redituables, lo cual ha hecho que la complejidad de las aplicaciones de estos sistemas de control sea mayor. Es debido a esta creciente complejidad que los autómatas tengan costos mas altos y tanto su diseño como su mantenimiento se vuelve poco practico, por lo que una ligera modificación o mejora es difícil de implementar si es que el sistema no esta estructurado para tales modificaciones.

Por lo antes mencionado los diseñadores de sistemas de control se han visto en la necesidad de buscar nuevas alternativas para reducir la complejidad de los sistemas automáticos de control. De esta filosofía hemos observado que existen dos principales corrientes de diseño: una de estas es el aprovechamiento del sistema de control mediante la compatibilidad con futuros diseños para aplicaciones semejantes (concepto 'Evergreen'); la otra corriente es el proponer nuevas técnicas de diseño, una de las cuales es la Técnica de diseño mediante la Lógica Difusa (Fuzzy Logic).

La primera de las corrientes de diseño (Evergreen) es empleada en la actualidad por grandes empresas que tienen una estructura sólida en el mercado de producción de Circuitos Integrados (microprocesadores y microcontroladores), como es el caso de Motorola, Intel, National Semiconductors, etc. Donde tales circuitos son utilizados para el desarrollo de las aplicaciones de los sistemas de control, un ejemplo claro es la escalabilidad que se observa en las computadoras actualmente. Es de conocimiento general que en nuestro país las tecnologías de punta tardan un poco en llegar a las Universidades publicas, por tal motivo es difícil desarrollar a nivel académico un dispositivo o aplicación basado en circuitos comerciales (microprocesadores y/o microcontroladores) que pueda tener un futuro inmediato, ya que es muy probable que

durante el período de desarrollo y prueba los circuitos empleados tiendan a ser obsoletos debido a la presencia de una nueva generación de circuitos integrados dentro del mercado. Por este motivo enfocamos nuestra atención a las nuevas técnicas de diseño que prometen bajos costos e implementaciones flexibles, ya sea mediante Software y/o Hardware.

De las nuevas técnicas de diseño, la que más atrajo nuestra atención fue la llamada Fuzzy Logic (propuesta por Lofti A. Zadeh, a mediados de la década de los 60's), ya que propone modelos de control con respuestas semejantes al comportamiento de los operadores humanos que se requieren para ciertas aplicaciones, donde un control PID (como se les conoce de manera común a los Controladores Proporcionales, Diferenciales e Integrales, que ocupan dichas ecuaciones en su diseño para lograr el control de algún proceso) es complejo de implementar, además puede ser modelado para complementar sistemas PID o en caso de ser necesario sustituirlos, obteniéndose excelentes resultados. Aunado a la flexibilidad de su implementación, su bajo costo lo hace potencialmente accesible para su desarrollo e investigación en casi cualquier aplicación llevada a cabo por instituciones de diferente índole, tanto publicas como privadas.

El objetivo de desarrollar esta tesis es demostrar no solamente de forma teórica sino también de manera practica que con los conocimientos adquiridos durante nuestra formación profesional es posible implementar un proyecto con técnicas de reciente desarrollo, aunque este trabajo se haya diseñado con tecnologías convencionales (de desarrollo anterior). Aunado a lo anterior, se pretende plantear las bases de la Lógica Difusa de una forma sencilla y practica, ya que la mayor parte de las publicaciones enfocadas al tema lo expresan de una manera compleja y en muchos casos no se logra a apreciar la potencialidad de esta técnica.

El primer Capítulo permite conocer las premisas fundamentales de la Teoría Difusa de Conjuntos, con los principios que se muestran en dicho Capitulo es posible resolver problemas de control de una manera sencilla y sin involucrar planteamientos matemáticos complejos, de hecho se puede decir que las operaciones matemáticas se reducen a simples sumas, multiplicaciones y divisiones.

El Capítulo 2 se centra en plantear una metodología para resolver problemas de control. Para lograr este fin se dividió en dos partes al Capítulo 2, la primera de ellas explica los pasos que se deben seguir para poder atacar un problema de control; la segunda parte aplica dichos pasos en la solución de un ejemplo tomado de un artículo publicado por dos investigadores acreditados a nivel internacional.

El Tercer Capítulo es la esencia del trabajo de Tesis, en él se plantea el desarrollo de una arquitectura funcional cuyo diseño se cimienta en la Lógica Difusa. Para poder mostrar el diseño general de dicha arquitectura se plantean dos acercamientos previos, indicándose los problemas que se tuvieron para poder implementar tales arquitecturas. Se consideró importante mostrar las arquitecturas previas ya que, fue gracias a las limitantes que en ellas encontramos, y que nos impidieron construirlas físicamente, que pudimos llegar a una arquitectura sumamente compacta y flexible, la cual para su diseño tuvo que emplear Software programado y generado por nosotros a la vez de Sofware comercial.

Capítulo

Sistemas de Control Fuzzy Logic.

Introducción.

La Lógica Fuzzy es una técnica alterna a la Lógica Convencional que nos permite resolver problemas con menos pasos y recursos. La forma en que la Lógica Difusa encara un problema se asemeja en mucho a la manera en que los seres humanos razonamos y tomamos decisiones, de tal modo que cualquier problema resuelto mediante Fuzzy Logic puede ser entendido fácilmente, lo cual beneficia la manutención de los sistemas basados en dicha técnica. La gran ventaja que posee Fuzzy Logic frente a la Lógica convencional es que no requiere de información precisa, a Fuzzy Logic sólo le basta con saber una aproximación de los datos del problema para que tome una decisión. Aunque a simple vista esto parece incoherente desde un punto de vista convencional, esto no es así, ya que se han implementado bastantes aplicaciones basadas en ésta técnica, por ejemplo, en Japón se ha implementado un sistema automático que controla el proceso de acelerado y frenado de un tren de pasajeros, el tren Sendai. Lo cual es una prueba muy contundente de que Fuzzy Logic tiene un gran potencial de funcionalidad.

En el presente capítulo pretendemos mostrar las bases de la Lógica Fuzzy para poderlas aplicar a la solución de problemas de

control, comenzaremos por definir algunos conceptos fundamentales, que al ser aplicados correctamente pueden ayudarnos a manipular la información proporcionada por un problema de control. Así mismo explicaremos la "metodología" para aplicar a la Lógica Fuzzy en sistemas de control. Para concluir con un pequeño compendio de aplicaciones de la Lógica Difusa.

1.1 Definiciones Básicas.

El ser humano, por naturaleza, resuelve problemas lógicos en base a información imprecisa, inclusive la manera habitual de expresarse es mediante información imprecisa pudiendo de ésta manera transmitir una enorme cantidad de información muy "precisa", por ejemplo, cuando probamos un alimento y alguien nos pregunta ¿Está picoso? nosotros contestamos, Algo, o Poco, o Mas o menos, etc. Gracias a ésta respuesta nuestro interlocutor se da una idea de la condición del alimento; de la misma manera cuando ordenamos en un restaurante una carne asada le decimos al mesero, Démela medio cocida, entonces éste le transmite el cocinero el término en que deseamos la carne asada. Ya mencionamos que para nosotros es familiar el empleo de éste "lenguaje impreciso", pero ¿Podrá una `máquina' entender dicho lenguaje, para que de éste modo proporcione una respuesta adecuada? Con las técnicas de diseño convencionales seguramente no, sin embargo mediante el empleo de la Lógica Fuzzy esto si es posible.

Fuzzy Logic o Lógica Difusa es una técnica para resolver problemas de lógica cuyos valores de entrada no son exactos, es decir, "difusos" -de ahí su nombre-, que a pesar que sus entradas se definen en rangos de valores nos proporciona como resultado un dato que es el más adecuado al cuestionamiento dado.

La Lógica Fuzzy al igual que la Lógica Convencional se encuentra sustentada por desarrollos matemáticos, las bases de tales argumentos fueron planteadas por Lotfi A. Zadeh en 1965, a quien se le considera como el padre de la Lógica Fuzzy. Tales desarrollos matemáticos reciben el nombre de Teoría de Conjuntos de Zadeh o simplemente Matemáticas Fuzzy, cuyo equivalente en la

Lógica Convencional (Lógica Bivaluada) es la Teoría de Conjuntos y el Algebra Booleana.

Tanto en la Teoría de Conjuntos de Zadeh (TCZ) como en la Teoría de Conjuntos Convencional (TCC), cualquier situación se analiza dentro de un Universo limitante, dentro de éste se agrupan situaciones o eventos en Conjuntos, con la diferencia que en la TCZ un evento tiene un grado de veracidad para un conjunto dado, en cambio en la TCC un evento pertenece por completo o no pertenece a un conjunto dado, esto se ilustra mediante el siguiente ejemplo.

Ejemplo 1.1

Imaginemos un Amplificador de Audio que cuenta con 3 selectores de tono, Agudos, Medios y Graves. Sea "U" el universo de las frecuencias audibles para el ser humano, típicamente de 20 Hz a 20 KHz, "G" el conjunto de sonidos graves, identificados dentro del rango de 20 a 300 Hz, "M" representa a los tonos medios, desde 1 KHz hasta 5 KHz, y por último "A" es el conjunto de los tonos agudos, superiores a 10 KHz, los rangos indicados son las frecuencias no atenuadas por cada filtro de selección, por ejemplo para el conjunto "M" se cuenta con un filtro paso banda que atenúa las frecuencias menores a 1 KHz y las mayores a 5 KHz, (todos los rangos son aquellas frecuencias que no atenuada filtro de selección).

U = Frecs. Audibles para el ser humano
G = { 20, ..., 300 Hz}
M = { 1 K, ..., 5 KHz}
A = { 10 K, ..., 20 KHz}

"x" representa el tono de un sonido cuya frecuencia es de 100 Hz, el cual cae dentro del rango de "G" y por tanto podemos afirmar que "x" es un tono Grave, es decir:

$x \in G$ ('x' pertenece a "G", Fig. 1.1)
$x \notin M$ ('x' NO pertenece a M)
$x \notin A$ ('x' NO pertenece a A)

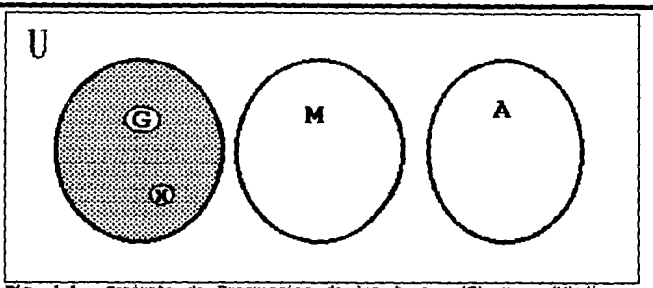


Fig. 1.1 Conjunto de Frecuencias de los tonos, (G)raves, (M)edios y (A)gudos.

En cuyo caso decimos que "x ε G" con un 100% de veracidad, que en Lógica Bivalente representa a un 1. Ahora bien, suponiendo que "x" toma un nuevo valor igual a 600 Hz, ¿a cuál de los 3 conjuntos pertenece "x"? La solución se puede derivar de la respuesta de los dispositivos que filtran a los tonos, (Fig. 1.2). Como se ve la frecuencia de 600 Hz se encuentra en un punto donde se intersectan dos bandas, sin embargo la señal se encuentra atenuada para ambas. Desde el punto de vista de la Lógica Bivalente la frecuencia de los 600 Hz se encuentra fuera de los rangos de "G" y de "M", para la Lógica Fuzzy, en cambio, esto *no* es así, ya que para ésta la frecuencia de 600 Hz pertenece a ambos conjuntos con un grado de verdad proporcional a la amplitud normalizada que representa dicha frecuencia en cada banda, es decir, la máxima amplitud alcanzada se toma como referencia para que de esta se analicen los valores inferiores como porcentajes de la misma, así una amplitud "G_1"=$G_{max}/2$ se puede considerar como 0.50 G_{max}.

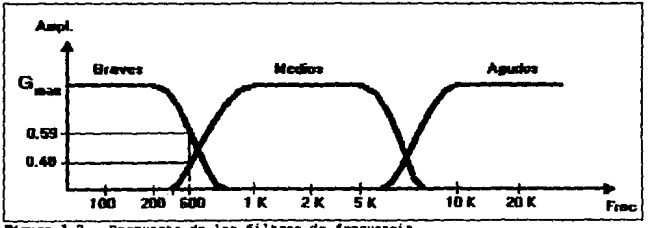


Figura 1.2 Respuesta de los filtros de frecuencia.

En la figura 1.2 se muestran las 3 gráficas de respuesta de los filtros, las cuales por conveniencia se han juntado en un solo plano coordenado. Debido a tal unión de gráficas es posible apreciar que existen zonas en las que se comparten algunas respuestas para una misma frecuencia, esta respuesta puede tener valores de amplitud diferentes para dos filtros adyacentes, observándose con ello que cada filtro actúa independientemente a un dato de entrada dado (en este caso es cualquier valor de frecuencia dentro del rango de 20 a 20 KHz).

1.1.1 Variable Lingüística.

Es aquella representativa de la entrada, por ejemplo, Edad, Velocidad, Posición, Fuerza, etc. Cuando se hace referencia a ella en notación generalizada es común representarla por las letras "X,Y o Z", indicándose con las letras minúsculas "x,y o z" cualquier valor tomado por dicha variable, así por ejemplo, para la variable lingüística X=Peso puede observarse un valor x=35Kg, o x=12 Kg, en donde observamos que X mayúscula indica el nombre de la variable y x minúscula es cualquier valor de la variable.

1.1.2 Universo en Cuestión.

Se denomina de éste modo al intervalo completo en que fluctúa una variable lingüística, la notación generalizada del Universo en Cuestión se hace mediante la letra "U o V". Debido a que en la mayor parte de los casos en el Universo en cuestión abarca todos los valores en que varía la variable lingüística, es común emplear de manera indistinta ambas notaciones.

1.1.3 Grado de Membresía.

También llamado Grado de Compatibilidad, Grado de Verdad o Grado de Pertenencia, es el valor de veracidad que adquiere un dato "x" dentro de un Fuzzy Set (Conjunto Difuso) en particular, dicho valor se encuentra dentro del intervalo de 0 a 1, pudiendo tomar cualquier valor dentro del rango, teniendo dos casos límite el 0, falso, y el 1, cierto. El grado de membresía puede considerarse como un porcentaje de certidumbre siendo 0 = 0% cierto y 1 = 100% cierto.

El grado de membresía se denota por la letra griega "μ", y su representación más común se escribe a continuación:

$$\mu_A(x) = \text{valor numérico entre 0 y 1.}$$

donde:

A - Es un Conjunto Fuzzy Cualquiera (Fuzzy Set).
x - valor cualquiera tomado por la variable lingüística.

Por ejemplo:

$\mu_{CALIENTE}(38°C)=0.23$

que se leería, "el grado de membresía del valor de Temperatura 38°C es igual a 0.23 para el Fuzzy Set CALIENTE".

1.1.4 Fuzzy Set.

También llamado Función de Pertenencia y cüya traducción literal al español es "Conjunto Difuso", es una reunión de datos de la variable lingüística los cuales cumplen con una condición de agrupamiento, tal condición puede ser un modelo matemático o una simple suposición de comportamiento los cuales satisfacen un mismo criterio de selección. Cuando se habla de ellos en forma generalizada se les etiqueta mediante las letras A, B, C, D, ... siempre que éstos pertenezcan al Universo en Cuestión.

Hemos dejado al último la definición de los Fuzzy Sets de manera intencionada, debido a que son éstos los que determinan el comportamiento difuso de la lógica fuzzy, ya que la acción que tienen sobre la variable lingüística logra una transición suave desde una No Pertenencia hasta una Total Pertenencia y viceversa, lo que es evidente en su aspecto gráfico mostrado en las figuras 1.3 y 1.4. El comportamiento difuso también lo determinan las etiquetas que van asociados a los Conjuntos Difusos (Fuzzy Sets).

Los Fuzzy Sets pueden generarse mediante una ecuación matemática que satisfaga un criterio de agrupamiento, cuando se conoce el comportamiento de dicho conjunto. Por ejemplo, una ecuación del tipo:

$$A = \{(x, \mu_A(x)) \mid x \in X\}$$

hace que se forme un Fuzzy Set de números reales cercanos a 10, teniendo una representación gráfica como se aprecia en la Fig. 1.3

donde

$$\mu_A(x) = (1 + (x - 10)^2)^{-1}$$

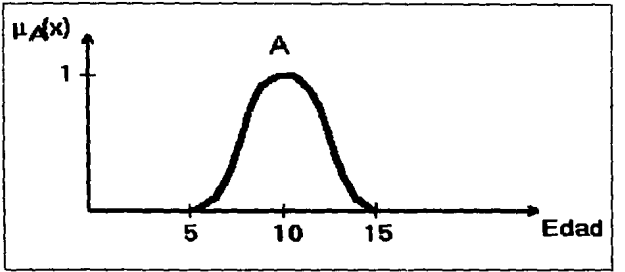


Figura 1.3 Representación gráfica de la ecuación del conjunto A de números cercanos a 10.

Sin embargo en problemas reales es difícil determinar una ecuación característica para cada Fuzzy Set, por lo que se ha optado por emplear modelos matemáticos de comportamiento menos complejos, esto se debe a que en procesos reales de control se requiere a cada Fuzzy Set en paralelo y si este posee una ecuación característica compleja se empleará un tiempo considerable en su análisis, los modelos más comunes son los que generan trapecios y triángulos (que se consideran como casos especiales de los trapecios.), los cuales permiten hacer un cambio gradual desde μA(x)=0 hasta μA(x)=1, además de poderse generar mediante ecuaciones de líneas rectas las cuales son simples de resolver, estas formas se aprecian en la figura 1.4. Debido a que se pueden generalizar se evita modelar cada Fuzzy Set.

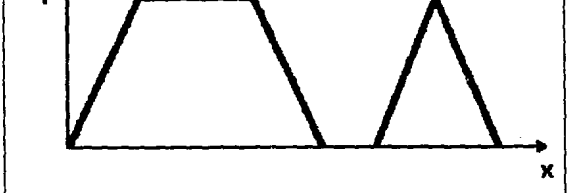


Figura 1.4 Representación más usual de los Fuzzy Sets.

Se puede observar que éstos permiten una transición suave y continua desde una no pertenencia hasta una total pertenencia, es decir desde $\mu(x) = 0$ hasta $\mu(x) = 1$, y viceversa.

Los Fuzzy Sets se disponen de tal forma que cubren todo el rango del Universo en Cuestión, la disposición de los mismos propicia el traslapamiento entre conjuntos adyacentes con ello se logra que un elemento pertenezca a dos conjuntos de manera simultánea aunque con su respectivo grado de membresía en cada conjunto. Gracias a este traslapamiento podemos recorrer todo el universo en cuestión sin perder información o truncarla drásticamente, como en el caso de la Lógica Bivalente. La figura 1.5 muestra la división de un Universo en Cuestión mediante Fuzzy Sets.

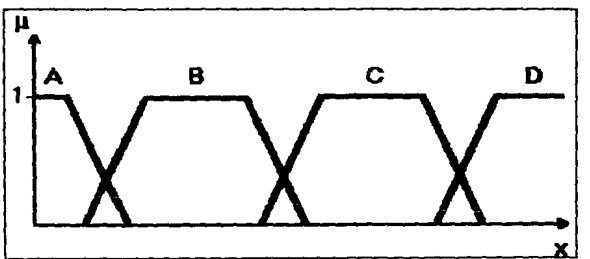


Figura 1.5 División del Universo en Cuestión mediante Fuzzy Sets.

Se habrá notado que las etiquetas o nombres de cada Fuzzy Set hasta el momento sólo se han indicado mediante notación generalizada, sin embargo éstos llevan implícita en su nombre la "importancia" del Fuzzy Set asociado, como se verá en la siguiente sección.

1.1.5 Etiquetas Fuzzy Logic.

También nombradas Valores Lingüísticos, son calificadores que se les asigna a cada uno de los fuzzy sets, los cuales son representativos de la condición de la variable dentro de su

universo, así por ejemplo tenemos que para la variable lingüística "Velocidad" podemos tener las siguientes etiquetas de fuzzy sets, 'Nula', 'Lenta', 'Media' y 'Alta'. Como vemos con el simple hecho de escuchar cualquiera de las etiquetas de un fuzzy set nos damos una idea de la condición de la velocidad en cualquier punto dentro del mismo fuzzy set, y como sabemos que todos los conjuntos forman al universo en cuestión podemos conocer cualquier condición de la variable dentro del universo gracias a las etiquetas fuzzy.

Todas las etiquetas son clasificadoras de la Variable Lingüística, pero éstas son susceptibles de alterarse mediante Modificadores o "Hedges"(contornos), no es lo mismo decir "la Velocidad de este automóvil es Alta" a decir que es "MUY Alta". Estos modificadores suelen ser de 2 clases fundamentales: (1) de Concentración, por ejemplo, "MUY"; y (2) de Dilatación, como es el caso de "MAS O MENOS". Los cuales se determinan mediante las siguientes definiciones:

$$\mu_{CON(A)}(u) = (\mu_A(u))^2$$
$$\mu_{DIL(A)}(u) = (\mu_A(u))^{1/2}$$

Pudiéndose apreciar su efecto en la figura 1.6.

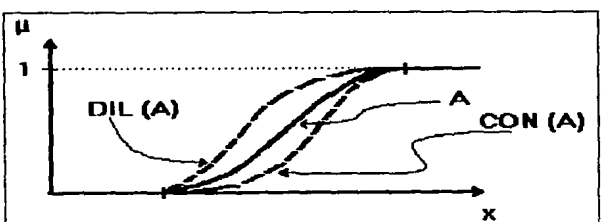


Figura 1.6 Acción de los Modificadores sobre un Fuzzy Set.

Es de esperarse que la Teoría de Conjuntos de Zadeh proporcione herramientas para la manipulación de los Conjuntos Fuzzy, tales herramientas permiten realizar mezclas entre fuzzy sets, al igual que en la Lógica Convencional, las operaciones básicas se describen a continuación.

1.1.6 Operaciones Básicas para Conjuntos Fuzzy.

El componente más importante para un fuzzy set es el grado de membresía (μ), por ello es obvio que las operaciones entre fuzzy sets se definan mediante éste. Las operaciones aquí mostradas son las primeras propuestas por Zadeh en 1965, mas no las únicas usadas por la teoría Fuzzy (como es el caso del producto cartesiano Fuzzy Logic, La suma algebraica, La suma limítrofe, etc.), estas operaciones parten de bases semejantes que las de la Teoría de Conjuntos Convencionales, razón por la cual reciben los mismos nombres, recordando que estamos hablando de una teoría de conjuntos extendida ya que emplean un criterio más amplio en el manejo de valores medios. En la Teoría de Conjuntos Fuzzy existen tres operadores principales que son la Intersección, Unión, y Complementación, los cuales se ilustran en la Figura 1.7 y se definen como sigue.

1.1.6.1 *Intersección.* La función de membresía $\mu_C(x)$ de la intersección $C = A \cap B$ se define por

$$\mu_C(x) = \min \{\mu_A(x), \mu_B(x)\}, \qquad x \in X$$

1.1.6.2 *Unión.* La función de membresía $\mu_D(x)$ de la unión $D = A \cup B$ se define por

$$\mu_D(x) = \max \{\mu_A(x), \mu_B(x)\}, \qquad x \in X$$

1.1.6.3 *Complemento.* La Función de membresía $\mu_{cA}(x)$ (complemento de A), está definida por

$$\mu_{cA}(x) = 1 - \mu_A(x), \qquad x \in X$$

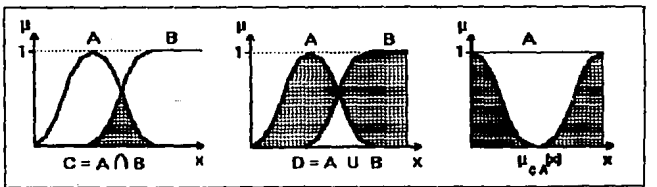


Figura 1.7 Ilustración del comportamiento de los operadores Intersección, Unión y Complemento.

A manera de ejemplo retomemos la figura 1.2, imaginemos que para la entrada "Frecuencia = 600 Hz" se obtienen los grados de pertenencia siguientes:

$$\mu_c(600) = 0.59 \text{ y } \mu_m(600) = 0.4$$

aplicando estos valores a las definiciones anteriores obtenemos

1. $C = A \cap B$ $\quad \mu_c(600) = \min\{0.59, 0.40\} = 0.40$

2. $D = A \cup B$ $\quad \mu_b(600) = \max\{0.59, 0.40\} = 0.59$

3. $\mu_{cA}(x)$ $\quad \mu_{cc}(600) = 1 - 0.59 = 0.41$

$\mu_{cm}(600) = 1 - 0.40 = 0.60$

La justificación de los resultados es la siguiente:

1. La intersección toma el valor común de los 2 valores.
2. La unión es la conjunción de ambos.
3. Es el faltante para completar el 100% de pertenencia.

Existen numerosos operadores en las Matemáticas Fuzzy, en donde la mayor parte de ellos son una mezcla entre la teoría de probabilidad y una extensión de la Teoría de Conjuntos Convencional, sin embargo aquí sólo mencionamos aquellos que consideramos más útiles para su aplicación en sistemas de control, en el inciso 1.3 se mencionarán otros operadores y su relación con el Control Fuzzy Logic. Por lo pronto describiremos las áreas en que Fuzzy Logic ha encontrado mejores desempeños.

1.2 Campo de Aplicación.

En la sección anterior se fijaron las bases de la Lógica Fuzzy desde el punto de vista de la teoría de conjuntos, la cual se puede ver como una extensión de la Teoría de Conjuntos Convencional que

trata con información imprecisa. Sin embargo, ésta información imprecisa se puede manipular de manera similar a la que el ser humano lleva a cabo, de tal modo que se logran obtener muy buenos resultados en procesos de control.

Es de nuestro conocimiento que la Teoría de Conjuntos Convencional ha dado lugar a procedimientos aritméticos y lógicos, tal es el caso de la Lógica Booleana (entre otras), la cual correctamente aplicada puede tomar las decisiones necesarias para llevar a cabo el control de procesos sumamente complejos, como en el caso de las computadoras aplicadas al control, no obstante ésta se ve limitada por las mismas restricciones que tiene la Teoría de Conjuntos Convencional de donde tiene su origen, tal teoría es la Teoría Bivalente que se limitada en la solución de problemas cuyas entradas sean imprecisas y por tanto no se pueden interpretar como 100% verdaderas o 100% falsas. La Lógica Difusa al ser una extensión de la Lógica Convencional no sólo rompe las barreras que limitan a ésta, sino que también puede ser aplicada en el campo del control, así como en todas las demás áreas donde la Teoría de Conjuntos Convencional ha sido aplicada, teniendo en muchos casos mejores resultados que ésta, inclusive se puede emplear en problemas donde la técnica convencional no había encontrado respuestas. Por ejemplo, hasta el empleo de la Lógica Difusa no se había automatizado un helicóptero, del cual la función más complicada (según los especialistas) es el lograr que el helicóptero este estático y suspendido en el aire.

No son pocos los usos de la Lógica Fuzzy que al ser comparados con sus "rivales" convencionales resultan superiores en desempeño e inferiores en costo, ya que para diseñar un sistema fuzzy no se requiere de modelos matemáticos, como en el caso de sistemas de control convencionales que requieren de modelos del sistema completo incluyendo entradas y salidas en vez de ello se emplean algoritmos del proceso para ser simulados en computadoras convencionales con lo cual se evita el invertir un largo período de tiempo, lo cual es un ahorro sin lugar a dudas. En vista de lo antes descrito resulta una pregunta válida y necesaria ¿Es la Lógica Difusa un reemplazo de la Lógica Convencional? Por supuesto que no, ya que Fuzzy Logic surge como una respuesta a las limitantes de la Lógica Convencional, es decir, Fuzzy Logic puede considerarse como una técnica que viene a reforzar las aplicaciones de la Lógica Convencional. Lo que si debe considerarse es la extensión del estudio de la Lógica incorporando a los aspectos

Difusos. En vista de que Fuzzy Logic es un refuerzo del control convencional podemos definir un área de aplicaciones donde la Lógica Difusa tiene un mejor desempeño, las cuales son:

1. Aplicaciones donde es difícil o casi imposible desarrollar un modelo matemático que cubra todo el comportamiento del sistema por diseñar. Generalmente las personas que desarrollan tales modelos son especialistas, de los cuales, por desgracia, nuestro país cuenta con muy pocos, por tal motivo podemos extender éste ramo de aplicaciones a: lugares donde se tenga carencia de recursos, tanto financieros como humanos.

2. Usos donde un experto humano sea quien tome decisiones. Obsérvese que no se intenta reemplazar a dicho experto, el fin del empleo de Fuzzy Logic en éste ramo es el proporcionar la asesoría necesaria que un experto pueda proveer en lugares donde no se cuente con dicha persona, pudiendo ser empleado hasta que el experto humano se presente y tome el mando del próblema, en cuyo caso se puede auxiliar del sistema fuzzy y con ello disminuir considerablemente ligeros errores que se puedan presentar.

3. Sistemas cuyas entradas y salidas sean moderadamente o muy complejamente continuas (o semicontinuas), por ejemplo en sistemas de control PID.

4. Sistemas con observaciones humanas que pueden ser empleadas como reglas de control y/o entradas.

5. Sistemas donde la vaguedad es común, tal es el caso de sistemas económicos, ciencias naturales, y ciencias del comportamiento.

Se intuye que el común denominador de los 5 puntos enumerados es la carencia de un modelo matemático, por ello se colocó en primer término. La manera de resolver un cuestionamiento mediante Fuzzy Logic se describe a continuación.

1.3 Estructura Básica de un Sistema Fuzzy Logic.

En ésta sección describiremos los pasos necesarios para que mediante el empleo de la Lógica Fuzzy podamos darle solución a un problema de control.

Cualquier problema puede ser resuelto en base a los tres pasos básicos de la Lógica Fuzzy, que son: Fuzzificación, Evaluación de Reglas, y Defuzzificación, éste conjunto de pasos recibe el nombre de *Inferencia Fuzzy*, que no es mas que el decidir cual es la mejor acción a tomar una decisión en base a los datos de entrada. La Inferencia Fuzzy se muestra esquemáticamente en la figura 1.8.

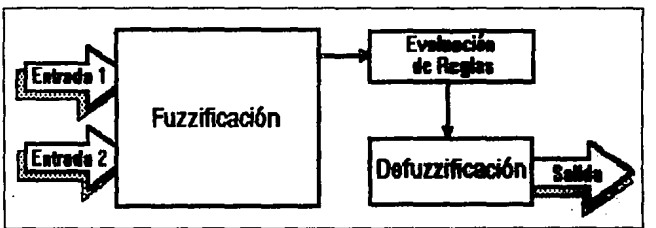


Figura 1.8 Diagrama a bloques de la secuencia para la solución de un problema empleando la Lógica Fuzzy.

Se notará que en la figura sólo se muestran dos entradas y una salida, esto es con el fin de simplificar la explicación, pudiéndose extender a "N" variables de entrada y "M" de salida. Cada uno de los bloques mostrados se describe a continuación.

Debemos aclarar que hasta el momento no se tiene conocimiento de algún texto especializado que formalice un método para tal fin, razón por la cual las personas encargadas del diseño del sistema deben decidir cual de las diversas operaciones es la que satisface mejor las necesidades del mismo, en cuyo caso el óptimo resultado dependerá del buen juicio o de la experiencia de los diseñadores. Por fortuna el desarrollo en las investigaciones que sobre Fuzzy Logic se han efectuado a lo largo de las últimas dos décadas nos indican que algunos operadores tienen un buen desempeño y son sencillos en su implementación, lo cual reduce considerablemente el problema de la elección de un método apropiado. También es

importante hacer notar que el procedimiento aquí descrito no sólo se aplica en el área de control sino que puede ser aplicado en la mayor parte de las áreas donde Fuzzy Logic tienen un buen desempeño.

1.3.1 Fuzzificación.

En realidad éste paso ya lo efectuamos en los ejemplos anteriores, éste consiste en determinar el grado de membresía que un dato de entrada tiene en un fuzzy set (o subset) en especial. La mayor parte de las entradas a nuestro sistema fuzzy son datos del mundo real, aquí les nombraremos datos físicos, que pueden tomar cualquier valor, ya sean positivos o negativos, por tal motivo es necesario proyectarlos dentro de un conjunto fuzzy y así conocer la pertenencia que dicho valor tiene para el conjunto en cuestión. De ésta manera podemos considerar al proceso de fuzzificación como una especie de normalización indispensable para poder trabajar con datos homogéneos.

El resultado que se obtenga de la proyección de un dato físico en un conjunto fuzzy depende directamente de la forma y el rango que tenga el fuzzy set (o subset), esto se puede observar en el ejemplo siguiente.

Ejemplo 1.2

Supongamos que tenemos como variable de entrada la condición de humedad de un asfalto (Como es un ejemplo ilustrativo para obtener el grado de membresía no se considera importante el método o medio de la obtención de la humedad, ya que esta se tomó como una variable cualquiera) y ésta se encuentra dividida como se muestra en la Figura 1.9. En un punto determinado tomamos una lectura que resultó ser de 5%, si nosotros proyectamos ésta lectura obtenemos un grado de pertenencia igual a 1 para el fuzzy set SE (Seco), cuya notación en conjuntos fuzzy es, $\mu SE(5\%) = 1$. Si ahora recibimos una entrada de 20%, al momento de proyectarla obtenemos.

$$\mu_{SE}(20\%) = 0.50$$

$$\mu_{HU}(20\%) = 0.25$$

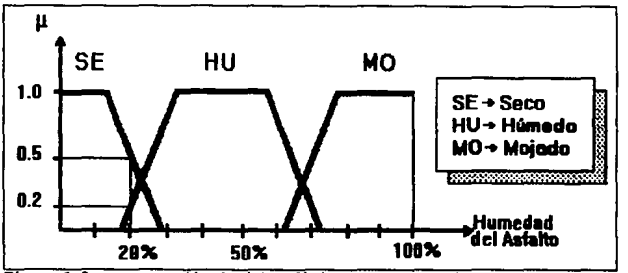


Figura 1.9 Proyección de datos físicos en Fuzzy Sets.

Como se observa el proceso de Fuzzificación es muy sencillo, simplemente evaluamos la lectura e indicamos el valor obtenido junto con su respectivo fuzzy set (o subset) que activa.

1.3.2 Evaluación de Reglas.

Este segundo paso es la esencia del proceso, es tan importante que en numerosas ocasiones se le designa PROCESO DE INFERENCIA, ya que en éste se efectúa toda la lógica necesaria para tomar una decisión, por tal motivo dicho "razonamiento" debe estar normado por ciertos criterios o REGLAS, éstas en su totalidad forman una base de datos, que en Fuzzy Logic se conoce como BASE DE CONOCIMIENTO, las reglas se estructuran de manera tal que cubren la totalidad de situaciones que en nuestro sistema puedan presentarse. El formato más común en que se presentan las reglas se ha tomado de los sistemas expertos, éste es de la manera IF...THEN..., el cual no debe tomarse en el sentido estricto de su significado bivalente (SI...ENTONCES...), sino considerarse como una regla que nos indica que SEGUN sea la proporción en que se presente la condición ENTONCES se ejecuta la acción correspondiente. La notación generalizada de cualquier regla fuzzy es:

IF Condición o Antecendente **THEN** Acción o Consecuencia

Todas las reglas fuzzy las podemos dividir en dos porciones, la correspondiente a IF y la porción THEN. La primera de ellas contiene las variables de entrada ya fuzzificadas, éstas pueden ser "N" variables, sin embargo, por simplicidad emplearemos una, ésta recibe el nombre genérico de Condición o Antecedente, por ejemplo:

IF (Asfalto es Seco) THEN ...

Como vemos el Antecedente tiene dos datos, uno es la variable de entrada, en éste caso Asfalto, y el otro es la condición del fuzzy set, en éste caso Seco. La porción THEN de la regla se constituye de igual manera, sólo que en ésta los datos son, primero la variable de salida y segundo la condición del fuzzy set, el conjunto de ambos datos recibe el nombre de Acción o Consecuencia.

... **THEN** (Vble. Salida es Fuzzy Set)

Estas reglas son fácilmente ampliadas para contener dos o más variables, tanto de entrada como de salida, gracias al operador AND o Intersección.

Como mencionamos anteriormente el conjunto de todas las reglas forman una base de conocimiento, existen dos maneras de representar dicha base de datos, una es mediante la tabulación de cada una de las reglas y la otra es mediante una matriz de reglas, como se muestra en el siguiente ejemplo.

Ejemplo 1.3

Representar la Base de Conocimiento para un problema de dos variables de entrada (E1 y E2) y una de salida (O), las variables de entrada se constituyen por los fuzzy sets, (A, B, C) y (D, E, F) respectivamente, mientras que la variable de salida cuenta con los fuzzy sets (J, K, L).La lógica propuesta para las reglas de control de este ejemplo, nos indica la siguiente base de conocimiento.

Representación Tabular

1.	IF	Antecedente A	AND	Antecedente D	THEN	Consecuencia J
2.	IF	Antecedente A	AND	Antecedente E	THEN	Consecuencia K
3.	IF	Antecedente A	AND	Antecedente F	THEN	Consecuencia L
4.	IF	Antecedente B	AND	Antecedente D	THEN	Consecuencia J
5.	IF	Antecedente B	AND	Antecedente E	THEN	Consecuencia K
6.	IF	Antecedente B	AND	Antecedente F	THEN	Consecuencia J
7.	IF	Antecedente C	AND	Antecedente D	THEN	Consecuencia L
8.	IF	Antecedente C	AND	Antecedente E	THEN	Consecuencia L
9.	IF	Antecedente C	AND	Antecedente F	THEN	Consecuencia K

Representación Matricial

Variable 1 \ Variable 2	D	E	F
A	J	K	L
B	J	K	J
C	L	L	K

En ambas representaciones se aprecia que es necesario incluir todas las posibles combinaciones de Antecedentes (9 en éste caso, ya que la Variable 1 tiene 3 Fuzzy Sets y la Variable 2 también tiene 3 Fuzzy sets, por lo tanto 3x3= 9 posibles combinaciones o reglas), sin embargo, las Consecuencias dependen de las funciones que se deseen realizar según sea el par de antecedentes del problema, por ejemplo, la Consecuencia "J" se encuentra en las reglas 1, 4 y 6 siendo evidente que no depende de un fuzzy set en particular sino de la lógica de la solución del problema.

En la representación matricial observamos que es mucho más simple identificar las reglas de manera matricial, sin embargo en éste arreglo se pueden trabajar fácilmente hasta 4 variables simultáneas, siendo el arreglo tabular el que mejor trabaja con "N" variables de entrada.

Hasta éste momento hemos podido apreciar que las 9 reglas están escritas en lenguaje común pero, ¿Cómo se realiza el proceso de Inferencia? En una regla se visualiza que una o más variables de entrada nos llevan a una salida. Recordemos que una variable ya sea de entrada o salida es una variable lingüística que determina un conjunto, por ejemplo: el conjunto de datos llamado HUMEDAD DEL ASFALTO. Este conjunto ya ha sido fuzzificado para una observación dada, por tanto sabemos que rangos están activos y en que proporción. Lo que necesitamos es aplicar dicha observación a cada una de las reglas para conocer la salida equivalente, esto se efectúa mediante cualquiera de los métodos de Composición existentes, que proyectan fuzzy sets en fuzzy sets (proyectar conjuntos de entrada en conjuntos de salida) explicaremos el método de Composición de máximos y mínimos o Composición max-min, que engloba otro método más simple y de resultados equivalentes.

- **Definición** -

Hagamos "R" una relación fuzzy que une el universo "U" y el universo "V", "x" es un subconjunto Fuzzy del universo "U", entonces el subconjunto Fuzzy "y" del universo "V" inducido por "x" al aplicarse la relación "R" se denota por:

$$y = x_0 R \qquad {}_0 \text{ - denota composición}$$

dicha relación se define en términos de membresía como:

$$\mu_y(v) = \max_{u \in U} \min\ (\mu_x(u),\ \mu_R(u,v))$$

en nuestro caso R es una regla fuzzy del tipo IF A THEN B (representado por A -> B), por lo cual podemos deducir que si "x = A" entonces

$$y = x_0(A \rightarrow B) = B$$

Es decir, cuando "x=A" se cumple por completo la regla "R" por lo cual se induce el resultado "B" en el universo "V" obteniendose "y = B".

La representación anterior puede ser aplicada a conjuntos que no son exactamente A, es decir, si recibimos una observación fuzzificada que se proyecte en A con $\mu < 1$, dicha observación pertenece a A pero no al 100%, pudiéndose identificar por A', si aplicamos ésta observación a la ecuación anterior con x = A' obtenemos

$$y = x_0 (A \rightarrow B) = A'_0 (A \rightarrow B) = B'$$

Para mayor claridad veamos el siguiente ejemplo:

R = A -> B	Los jitomates rojos están maduros
x = A'	Este jitomate está un poco rojo
y = x_0R = B'	Este jitomate está un poco maduro
= A'$_0$(A ->B)	
= B'	

donde:

A= los jitomates rojos	R= regla Fuzzy
B= están maduros	x= condición del jitomate en "U"
A'= jitomates un poco rojos	y= condición del jitomate en "V" dado x

En éste ejemplo aplicamos la observación a una sola regla, siendo que en la realidad ésta se debe aplicar a cada una de las reglas, obteniéndose tantas inducciones como reglas se tengan, de modo que debemos escoger el resultado óptimo a la combinación de todas las reglas, esto se hace mediante la Defuzzificación que se verá en la Sección 1.3.3, por lo pronto ampliaremos el uso de la Composición max-min a dos variables por medio del siguiente ejemplo.

Ejemplo 1.4

En un proceso cualquiera de dos variables de entrada y una de salida, que se encuentra gobernado por la base de conocimiento mostrada en el ejemplo 1.3, se obtuvieron las siguientes lecturas fuzzificadas en un momento dado: A' = 0.50, B' = 0.20 y D' = 0.30, encuéntrese la salida fuzzificada.

Observando la Base de Conocimiento nos damos cuenta que se cumplen las reglas 1 y 4 (en términos de Lógica Difusa, cuando una regla se cumple se dice que se activa o dispara, y cuando no se cumple se puede decir que esta inactiva o activa con un μ=0).

1. IF A AND D THEN J
4. IF B AND D THEN J

Resolvamos primero la regla 1. En éste caso debemos escoger el valor mínimo de los Antecedentes para activar la Consecuencia con el valor obtenido, resultando J = 0.30; esto se aplica de igual manera para la regla 4 obteniéndose J = 0.20, sin embargo ambas reglas nos activan al mismo fuzzy set de la consecuencia; ante tal estado elegimos el valor máximo de las 2 reglas para así proporcionar una sola decisión "J'". Este procedimiento puede apreciarse en la Figura 1.10.

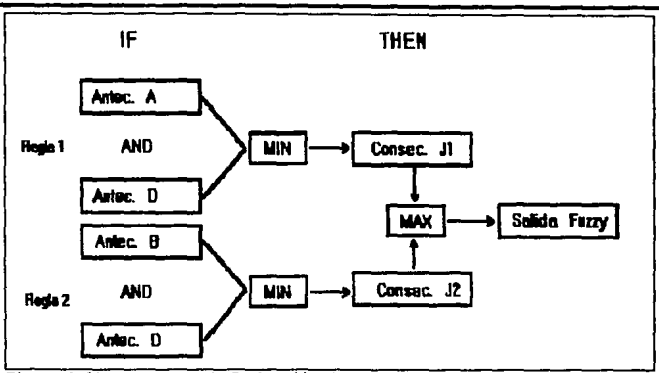


Figura 1.10 Esquema de Evaluación de Reglas.

En el ejemplo anterior, el proceso descrito es suficiente como para proporcionar una salida que aunque fuzzificada es representativa de la inferencia, esto es debido a que sólo se activo un subset fuzzy de la variable de salida (el conjunto "J"), sin embargo éste es un caso atípico en un sistema fuzzy, lo común es que se activen varios sets simultáneos, por lo que debemos mezclar de alguna manera los valores de cada uno para obtener un resultado final, esto se hace mediante la Defuzzificación.

1.3.3 Defuzzificación.

El último paso dentro de la Inferencia Fuzzy, es el encargado de evaluar todas las consecuencias activas y de entregar una salida "traducida" a dispositivos que no trabajan con técnica fuzzy, por ejemplo a un motor no le podemos entregar un valor de 0.25% Rápido como lo proporcionaría Fuzzy Logic, la defuzzificación se encargará de dar un valor en Amperes (o Voltaje, para cualquiera que sea la variable de control) correspondiente a 25% Rápido. Existen diferentes métodos para defuzzificar la información de salida, la elección del método más conveniente depende del medio de inferencia empleado en la Evaluación de Reglas. A continuación mencionaremos tres métodos compatibles tanto con la Composición max-min como con la Composición de mínimos.

1.3.3.1 OR. Imaginemos que se tienen las siguientes salidas J = 0.30, K = 0.50 y L = 0.75, en tal caso para escoger una salida les aplicamos el operador OR obteniéndose como resultado μ_{MAX} = 0.75, éste valor se proyecta de manera semejante que en la Fuzzificación en el Fuzzy Set L (que es su Fuzzy Set correspondiente) y se obtiene un valor "Ys" de salida efectiva o defuzzificada. Como se aprecia en la Figura 1.11.

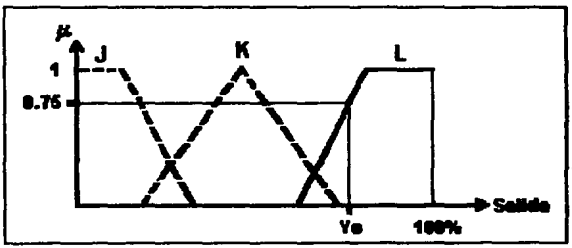


Figura 1.11 Representación Gráfica del Método OR.

1.3.3.2 Promedio de Pesos. Este método no excluye ninguna acción como el método anterior, que sólo ejecuta la acción cuya membresía sea la mayor, para el método de Promedio de Pesos todas las consecuencias y sus pares de salida deben de ser ejecutadas con un valor resultante que involucre todas las acciones, el valor resultante puede verse como un valor promedio y se calcula mediante la siguiente ecuación.

$$O = \frac{\sum_{i=1}^{N} \mu_{y_i}(v)\, y_i}{\sum_{i=1}^{N} \mu_{y_i}(v)}$$

donde:

$\mu_{Yi}(v)$	es el grado de membresía del Fuzzy Set activo "i"
Y_i	es el Fuzzy Set activo "i"
N	es el Número de Fuzzy Sets contenidos en el Universo de Salida
O	Salida Defuzzificada.

1.3.3.3 Centroide. Esta operación es similar a la anterior, sólo que aquí la proyección del grado de membresía determina el tamaño real del área del Fuzzy Set que activa, como se aprecia en la Figura 1.12, el área resultante del fuzzy set involucrado se multiplica por su par de salida correspondiente (centro de masa o centroide que es constante para el Fuzzy Set activo, "un ejemplo de la solución del método del centroide se puede apreciar en el Capítulo 2 Sección 2.3"), la ecuación que describe el método del centroide se escribe a continuación.

$$O = \frac{\sum_{i=1}^{N} (\text{área}_i)(c_i)}{\sum_{i=1}^{N} \text{área}_i}$$

donde:

área$_i$ = Área resultante del Fuzzy Set activo "i".
c_i = Valor del Centroide del Fuzzy Set activo "i".
O = Salida Defuzzificada.
N = Número de Fuzzy Sets

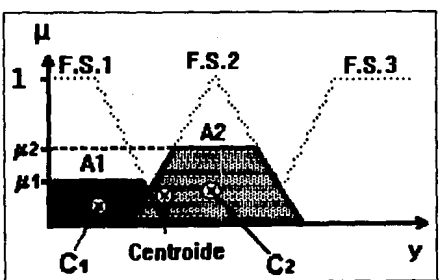


Figura 1.12 Ilustración del Cálculo del Centroide.

En la Figura 1.12 se muestran tres Fuzzy sets de salida, de los cuales sólo están activos dos (cuyo grado de membresía es diferente de cero). Los Fuzzy Sets activos son los que están etiquetados como F.S.1 y F.S.2, con sus respectivos grados de membresía (μ1 y μ2) los cuales truncan a los Fuzzy Sets originales, obteniéndose de esta forma las áreas resultantes (A1 y A2). Las etiquetas marcadas como C1 y C2 indican la posición en que se encuentran los centroides (constantes) de los Fuzzy Sets activos, de esta forma se tienen los valores necesarios para aplicar la ecuación del Método del Centroide.

De los tres métodos descritos, el de Promedio de Pesos es el más exacto, sin embargo, al igual que el método OR, sufre de ambigüedad, en la Fig. 1.13 se ven dos fuzzy sets, uno de ellos (triangular) activo en un 50%, es decir, μ = 0.5 el problema surge en determinar que orilla del fuzzy es la que nos proporciona el mejor valor de salida, la orilla izquierda nos da un valor "y1" y la orilla derecha uno "y2", claramente se ve que "y2 > y1", por tanto, la salida "O2" es mayor que la salida "O1". El mismo problema se acentúa para una función de membresía de forma trapezoidal cuando μ = 1, la cual también se muestra en la figura 1.13 en cuyo caso podemos tomar un extenso rango de valores de salida que van desde "Y1" hasta "Yn". Debido a la ambigüedad que presentan tanto el Método de Promedio de Pesos como el Método OR y la gran complejidad que presenta la implementación del Método del Centroide en Hardware se decidió emplear una variante de este último, que se conoce como Método de Singletons.

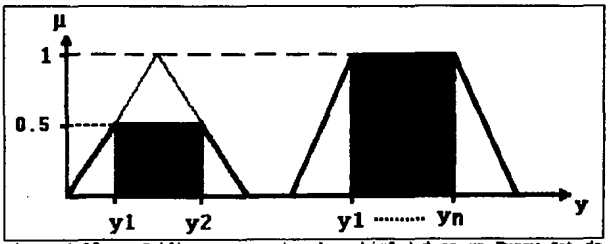


Figura 1.13 Gráfica que muestra la ambigüedad en un Fuzzy Set de Salida.

1.3.3.4 Método de Singletons.

El Método de Singletons es una variante del Método del Centroide, en donde se evita el cálculo de una área bajo la curva del Fuzzy Set activo, para ello se cambia la forma del Fuzzy Set (sin importar si es Triangular o Trapezoidal) por una función unitaria llamada **SINGLETON**, ubicada en el lugar del centro de masas del Fuzzy Set reemplazado, forzando de esta manera a que el área formada por el Fuzzy Set sea igual a la unidad, como se muestra en la Figura 1.14.

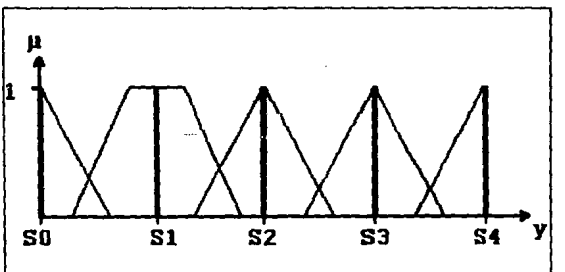


Figura 1.14 Muestra de la posición de los SINGLETONS respecto a los Fuzzy Sets de salida.

Si aplicamos la nueva área con valor fijo igual a la unidad en la ecuación del Método del Centroide, se evita el cálculo del área bajo la curva del Fuzzy Set original, por lo que la ecuación del Método del centroide queda expresada de la siguiente forma:

$$0 = \frac{\sum_{i=1}^{N} \mu_i(y)\, S_i}{\sum_{i=1}^{N} \mu_i(y)}$$

donde:

$\mu i(y)$ = Grado de Membresía del Fuzzy Set "i".
Si = Singleton representativo del Fuzzy Set "y"
(función unitaria multiplicada por el centroide).
O = Salida Defuzzificada.

De tal modo que cuando se proyecta un grado de membresía hacia la función unitaria representativa del Fuzzy Set en cuestión, sólo tenemos que multiplicar dicho grado de membresía por el valor del centro de masas del Fuzzy Set, obteniéndose de esta forma el valor del Singleton (Si) que se sustituirá en la ecuación anterior.

La defuzzificación puede ampliarse para "M" variables de salida, tomando en cuenta que cada variable deberá defuzzificarse independientemente, por tanto se tendrán "M" número de ecuaciones, una por cada salida.

De esta manera podemos obtener una respuesta adecuada a las variables de entrada, las cuales pueden ser empleadas con cualquier propósito, como en el control automático, diagnósticos médicos, etc.

1.4 Aplicaciones.

En la actualidad se han implementado diversas aplicaciones siguiendo métodos como el antes descrito, obteniéndose muy buenos resultados. Las aplicaciones que mas sobresalen son las siguientes, resaltando que la mayor parte de ellas se han desarrollado en Japón, ya que es este país es el que mayor atención le ha prestado a esta técnica de resolución de problemas de control.

Al principio del capítulo mencionamos que en la actualidad el tren Sendai de Japón se encuentra operado por medio de un sistema de control fuzzy logic, el tren funciona bajo modo automático en las horas pico, mientras que en el resto de la jornada los operadores lo conducen. El control fuzzy se implementó mediante Software y gobierna todos los aspectos de aceleración, frenado y detención del tren, tal implementación fue patrocinada por la Hitachi Corp.'s System Development Laboratory en Kawasaki, y la

idea fue concebida por Seiji Yasunobu y Soji Miyamoto. Durante el desarrollo del sistema se realizaron comparaciones con otros controladores Proporcionales, Integrales y Diferenciales (PID), observándose una mejora de la precisión en la detención sobre las plataformas, mejor confort al viajar (aceleración y frenado más suave), y un menor consumo de potencia eléctrica.

Mientras que el controlador del tren Sendai fue implementado en software que corre en computadoras ordinarias Takeshi Yamakawa junto con Miyamoto implementaron un problema clásico de control automático en hardware. El problema es balancear una pértiga que se encuentra anclada sobre una plataforma mediante un perno que permite movimientos hacia la izquierda o derecha, en base a las lecturas de su posición y velocidad se mueve la plataforma para mantener a la pértiga en una posición vertical, (éste es el mismo juego que todos experimentamos cuando tratamos de balancear un palo sobre la palma de la mano). El chip diseñado por estos dos investigadores emplea información analógica y corre en paralelo todas las reglas, empleando otro chip que defuzzifica las observaciones. El éxito de su proyecto fue tal que sobre la punta libre de la pértiga colocó una plataforma en la cual puso un vaso con vino, el cual no se derramó, e inclusive puso un pequeño ratón vivo que movía aleatoriamente al péndulo, siendo que el controlador compensaba satisfactoriamente todos los movimientos del ratón.

En Japón es común el encontrar productos fuzzy logic, estos cubren toda una gama de aplicaciones, de entre las cuales están, un controlador en el mecanismo de autofoco en una cámara de 8 mm de la Cannon Inc.. Panasonic por su parte ha implementado la Lógica Fuzzy en una video cámara para controlar la estabilización de imágenes. Se han implementado lavadoras automáticas que regulan el ciclo de lavado de acuerdo a la carga, suciedad y tipo de prendas de manera automática, teniéndose también los modelos industriales. En la Fuji Photo Film se emplea fuzzy logic para controlar el flujo de polvos. Hitachi lo emplea para bancos de elevadores. Mitsubishi Electric en aires acondicionados. Nissan Motors en transmisiones automáticas. Toshiba Systems en sistemas de ventilación para trenes subterráneos. También se ha empleado en ventiladores, limpiadores al vacío, y calentadores de gas o parrillas. La cantidad de aplicaciones continúa incrementándose y está lejos de saturarse.

Además de las aplicaciones de control, fuzzy logic también se ha empleado en otras áreas del que hacer humano, como es en la Medicina, en ésta aplicación se emplea la Lógica Fuzzy para establecer diagnósticos y tratamientos de primeros auxilios en una sala de emergencias en hospitales, ya sea mientras un especialista se presente o como auxiliar al especialista, ya que debido a la premura y tensión de una sala de emergencias un error es inevitable. Fuzzy Logic también se ha empleado para filtraje de señales, reconocimiento de caracteres, toma de decisiones, predicción de fenómenos meteorológicos y en una cantidad enorme de aplicaciones ajenas al control.

Como se ve, Fuzzy Logic es una técnica para manejar conjuntos muy simple pero demasiado efectiva, mediante la cual las aplicaciones cada día serán más complejas y con mejores resultados.

Capítulo

Metodología de Diseño para Sistemas Fuzzy Logic.

Introducción.

Cuando se intenta desarrollar un sistema capaz de controlar un proceso específico siempre surge la interrogante ¿Cómo lo hacemos?, para llevar a cabo el fin propuesto debemos realizar un estudio de cuales son los datos de que disponemos, así como de aquellos que podemos obtener y los que definitivamente no podemos obtener. La decisión de la manera de manipular tales datos dependerá del

"medio" que empleemos para procesar la información. El "medio" es la herramienta 'lógica' que tomará las conclusiones necesarias para controlar al sistema.

La herramienta empleada en éste reporte es la Lógica Fuzzy, de la que hasta el momento ya se ha descrito como evalúa la información que recibe, pero todo el análisis visto partió del supuesto que el problema y su comportamiento lógico ya se había definido con precisión. El objetivo de éste capítulo es mostrar una manera de identificar los parámetros del problema a resolver.

2.1 Metodología de Diseño.

La aplicación de la Composición *max-min* o la Composición *min* es muy efectiva para llevar a cabo la Inferencia de un problema, en donde la efectividad de la decisión depende directamente del buen juicio con que se ataque al problema. Nosotros proponemos una secuencia de pasos para facilitar la implementación de un sistema Fuzzy Logic, aunque es necesario recalcar que esto no es una metodología formal, que hasta el momento no la hay, o una "receta de cocina", los pasos aquí mencionados sirven como base para el planteamiento y desarrollo de un Control Fuzzy, sin embargo puede mejorarse o adaptarse a las necesidades específicas de un diseño cualquiera.

Cualquier aplicación de Fuzzy Logic enfocada al control se puede implementar mediante los siguientes pasos:

a) Identificación de variables de entrada, así como de sus rangos de acción.
b) Identificación de las variables de salida y sus rangos.
c) Creación de las funciones de membresía para las variables.
d) Formular la Base de Conocimiento que gobernará el comportamiento del sistema.
e) Determinar cómo se combinarán las acciones para proporcionar una salida precisa.

Cada uno de los pasos se verá a continuación.

a. Identificación de variables de entrada, y sus rangos.

Cuando vamos a resolver cualquier problema primero debemos conocer los datos de que disponemos, la forma en que los encontramos e inclusive buscar aquellos que nos hacen falta. De todos los datos que obtengamos tenemos que escoger los que sean más representativos del problema. Una vez identificados, los tenemos que etiquetar con un nombre que refleje su importancia para el problema, ya que el nombre que se le asigne se empleará en las reglas de funcionamiento.

Cada variable de entrada fluctúa sobre un rango de valores o estados, sin embargo para nuestro problema los rangos deben limitarse con valores extremos de modo que se tengan valores máximos y mínimos. En ocasiones los sensores empleados para tomar la lectura de los datos pueden imponer los límites de las variables, aunque en numerosas ocasiones nosotros debemos normalizar las lecturas para hacerlas compatibles con el controlador, con lo cual las limitamos, incluso en algunas ocasiones debemos calcular una nueva variable con las lecturas obtenidas.

Debido a que un Controlador Fuzzy Logic es muy flexible no debe ser de gran preocupación el escoger las variables correctas (junto con sus rangos) en primera instancia, esto se efectúa en un proceso de ajuste iterativo, durante el proceso de ajuste se deben descartar las variables no usadas e incorporar aquellas no previstas en los primeros acercamientos.

b. Identificación de Variables de Salida y sus Rangos.

Realmente es muy similar al punto anterior, de modo que la mayor parte de los criterios aplicables a las entradas son válidos para las salidas, aunque en éste caso el proceso de ajuste es un poco más sencillo ya que tenemos bien determinado el objetivo a conseguir.

c. Creación de las Funciones de Membresía.

Una vez determinadas las variables involucradas, así como sus rangos, debemos dividir cada rango en fuzzy sets (o subsets, según se vea), la mayor dificultad durante éste proceso es crear una

función que cubra correctamente el significado del fuzzy set asociado.

El método más recomendado para determinar los límites de los fuzzy sets es el interrogar a humanos expertos, ya sea a ingenieros en sistemas, operadores experimentados e inclusive usuarios finales. Esto es porque cada persona tiene un concepto diferente de las situaciones y debemos llegar a un concepto promedio.

Las funciones de membresía deben estar traslapadas entre sí, cuidando que el traslape no sea ni excesivo ni muy ligero, si nosotros sobreponemos en demasía un fuzzy set con otro no podemos identificar claramente a cual de ellos pertenece un dato, y en el caso contrario si el traslape es muy pobre estaríamos cayendo en el carácter excluyente de la lógica convencional, siendo el último caso el más grave de los dos ya que la solidez de la Lógica Fuzzy reside precisamente en el compartimento de datos entre fuzzy sets.

Otro aspecto que debemos cuidar es el dividir un universo en cuestión en un número "bajo" de fuzzy sets, recordemos que Fuzzy Logic resuelve problemas gracias a datos imprecisos, y si nosotros asignamos un número alto de fuzzy sets estamos haciendo que los datos de entrada y/o salida sean cada vez más precisos, alejándonos con ello del proceso de inferencia de la Lógica Fuzzy.

La forma de las funciones de membresía sobre un mismo universo no deben ser estrictamente iguales, es decir, los rangos cubiertos por cada una de ellas pueden o no ser los mismos. Es recomendable que modifiquemos el rango de las mismas para hacer más fina la respuesta del sistema en el punto deseado, como se muestra en la Figura 2.1.

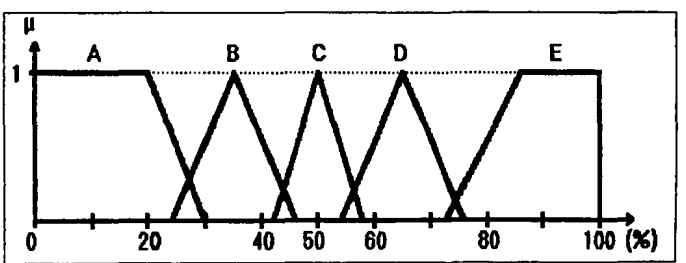


Fig. 2.1 Ejemplo de Fuzzy Sets con rangos y formas diferentes.

En la figura 2.1 apreciamos que el sistema es más sensible alrededor del 50% del Universo de la variable, esto es en el fuzzy set "C", mientras que en el resto del mismo la sensibilidad del sistema no es tan precisa, por ejemplo, en el fuzzy set "A" obtenemos el mismo grado de membresía (igual a la unida) si sensamos un dato cuyo valor sea igual a 5% o uno de 15%.

d. Formular la Base de Conocimiento.

Todo el peso de la inferencia del sistema Fuzzy cae sobre la Base de Conocimiento, que no es más que el cúmulo de todas las reglas que gobiernan las decisiones del controlador.

Las reglas son relaciones entre Antecedentes y Consecuencias. Teóricamente, el número de acciones que se deben definir para un sistema con "N" entradas y cada una con "M" fuzzy sets se determina mediante:

$$\text{Número de Acciones} = \prod_{i=1}^{N} M_i$$

donde:

N - Es el número de entradas.

M - Es el número de conjuntos fuzzy de cada entrada además, M1,2, ..,N, pueden ser o no iguales entre sí.

El operador π, denota multiplicación.

Por ejemplo, un sistema con dos variables, donde la primera (X1) tiene 5 fuzzy sets y la segunda (X2) tiene 3 fuzzy sets tendrá:

No. Acciones = 5 x 3 = 15 acciones.

(las cuales son seleccionadas según el fuzzy set de salida que activen, de tal modo que aunque sean 15 acciones diferentes éstas son limitadas al número de fuzzy sets que se tenga a la salida, por lo que algunas acciones pueden coincidir con otras, según sea la lógica del problema)

La manera más común para asegurarnos que todas las acciones posibles se han cubierto mediante la Base de Conocimiento es por medio de la Matriz de Reglas como a continuación se muestra:

X2 \ X1	FS1	FS2	FS3	FS4	FS5
FS1	A1	A2	A3	A4	A5
FS2	A6	A7	A8	A9	A10
FS3	A11	A12	A13	A14	A15

donde:

X1,X2 - Son las Variables de entrada
FS1,...,FS5 - Son los Fuzzy Sets de cada variable
A1,...,A15 - Son las acciones de salida.

Sin embargo, tanto la ecuación anterior como la matriz de reglas indican todas las posibles acciones; del total de ellas debemos descartar aquellas que por lógica son imposibles de ocurrir, a la vez de marcar y/o modificar aquellas que no son imposibles de ocurrir sino poco probables, dejando por último las reglas que realmente gobiernan al controlador. Por ejemplo, supongamos que de la matriz anterior las Acciones 3, 4, 5, 9, 10, y 11 son imposibles de ocurrir y las Acciones 6, 7, 12, y 13 son poco probables, entonces la matriz queda de la siguiente manera.

X2 \ X1	FS1	FS2	FS3	FS4	FS5
FS1	A1	A2			
FS2	(A6)	(A7)	A8		
FS3		(A12)	(A13)	A14	A15

Como vemos eliminamos las acciones que son imposibles de ocurrir y modificamos las poco probables de suceder, por tanto las acciones no marcadas son las que regulan el comportamiento del controlador bajo condiciones normales de operación, las acciones marcadas (con paréntesis) son las que actúan bajo condiciones anormales y su efecto es el regresar al sistema a las condiciones normales de operación. De modo que el número de acciones (y sus reglas asociadas) se redujo de 15 a, 5 normales mas 4 de "seguridad", es decir 9 en total.

Para generar la Base de Conocimiento también podemos auxiliarnos por expertos, al igual que en el caso anterior.

e. Determinar la Combinación de la Salida Unica.

Tal vez éste paso sea el menos complicado, en él debemos elegir un método para obtener una Acción resultante en el momento en que dos o más acciones se activen al mismo tiempo. En el capítulo anterior mencionamos tres métodos.

En resumen, podemos decir que los pasos 'a, b, y c' nos determinan la manera de Fuzzificar los datos de entrada; el paso 'd' nos ayuda a encontrar el medio de inferencia, y el paso 'e' el método para Defuzzificar el resultado inferido.

2.2 Planteamiento de un problema fuzzy.

En la sección anterior describimos una secuencia de pasos para aplicar la Lógica Fuzzy en la solución de un problema. En la presente sección resolveremos un problema aplicando los pasos antes descritos. En ésta ocasión citamos la primera parte del artículo "Fuzzy-logic system solves control problem" de David I. Brubaker y Cedric Sheerer, editado por la EDN en Junio de 1992, no incluimos íntegramente al artículo sino sólo extractos que son de nuestro interés.

El objetivo del artículo mencionado es el implementar un semáforo que regule el acceso a una vía rápida en el Norte de California, "actualmente en éstas vías rápidas existen semáforos convencionales que regulan el acceso mediante periodos fijos de luz verde y roja (pasando por la amarilla), una luz adicional indica que sólo puede ingresar un vehículo con cada luz verde. Se activa automáticamente un contador fijo durante los períodos que generalmente tienen tráfico pesado."

Pretenden controlar el retardo del encendido de las luces verde y roja según estén las condiciones de velocidad y densidad del tráfico. A continuación aplicaremos a éste problema los pasos descritos en la Sección 2.1, los cuales serán separados en secciones para su mejor identificación.

2.2.a Identificación de Variables de Entrada y sus Rangos.

Pese a que existen diferentes variables para dar solución al planteamiento dado se eligieron dos para actuar como entradas independientes, éstas son:

"VELOCIDAD - el promedio de la velocidad real del tráfico de la vía rápida, y
DENSIDAD - el promedio de la densidad real de tráfico sobre la vía rápida."

La Velocidad nos indica que tan rápido circula el tráfico y la Densidad la separación existente entre autos, de los cuales sus rangos de acción son:

Velocidad	0	a	67	MPH
Densidad	0.0	a	1.0	Autos/Espacio Unitario

2.2.b Identificación de Variables de Salida y sus Rangos.

Las salidas estaban claramente definidas desde el planteamiento del problema, lo delicado es escoger el modo en presentar la salida hacia el dispositivo final. Las salidas empleadas son:

"1. LUZ VERDE - La duración en segundos del estado de luz-verde, durante el cual los autos pueden ingresar a la vía rápida.

2. LUZ ROJA - La duración, también en segundos del estado de luz-roja (que incluirá un período constante de estado de luz-amarilla), durante el cual los autos sobre la rampa de ingreso no pueden entrar a la vía rápida"

A modo de abreviar designaremos las salidas LUZ VERDE como LVERDE y LUZ ROJA como LROJA. Tanto la primera como la segunda varían dentro de un Universo de 0 a 20 segundos.

2.2.c. Creación de las Funciones de Membresía.

"A la Variable DENSIDAD se le asignarán 3 valores fuzzy: PESADA -separación entre autos mínima, MEDIO_PESADA - separación entre autos nominal, y LIGERA - separación entre autos máxima."

Los rangos finales (aproximados por nosotros) de cada Fuzzy Set y su forma gráfica se muestran a continuación.

LIGERA	0.0	a	0.3	(LI)
MEDIO_PESADA	0.1	a	0.5	(MP)
PESADA	0.3	a	1.0	(PE)

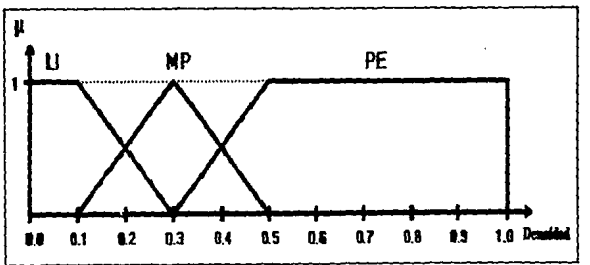


Fig. 2.2 Rangos de acción de la variable Densidad.

"La variable fuzzy VELOCIDAD también tendrá 3 valores fuzzy: LENTA -tráfico moviéndose lentamente, MEDIO_RAPIDA - tráfico moviéndose a velocidad nominal por debajo del límite de velocidad, y RAPIDA -tráfico moviéndose a o por encima del límite de velocidad."

Nuevamente suponemos los rangos finales de los fuzzy sets, éstos junto con su forma gráfica se muestran a continuación.

LENTA	0	a	20 MPH	(LE)
MEDIO_RAPIDA	5	a	55 MPH	(MR)
RAPIDA	37	a	67 MPH	(RA)

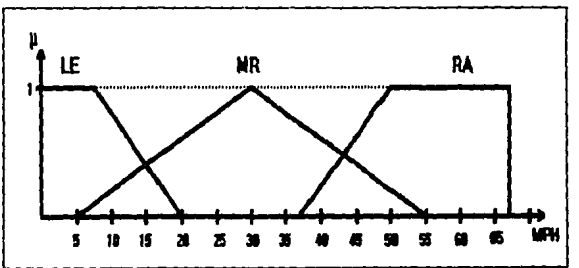


Fig. 2.3 Universo en cuestión de la variable Velocidad.

"Tres valores fuzzy se manejan de modo casi idéntico a las 2 salidas: CORTA - la luz dada se enciende por un corto tiempo, MEDIO_LARGA - la luz dada se enciende un período medio de tiempo, LARGA - la luz dada se mantiene encendida un período largo de tiempo. Aunado a éstos tres valores, LUZ_VERDE tiene un valor adicional: ENCENDIDO_CONSTANTE - la luz verde está encendida continuamente."

CORTA	0	a	6 seg.	(CO)
MEDIO_LARGA	2	a	10 seg.	(ML)
LARGA	6	a	20 seg.	(LA)

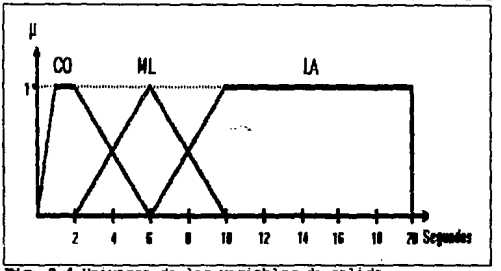


Fig. 2.4 Universo de las variables de salida.

2.2.d. Formular la Base de Conocimiento.

"Después de haber asignado los fuzzy sets debemos proyectar cada una de las 9 posibles condiciones de entrada a una salida. Esta proyección se hace mediante la base de reglas. La forma más común es:

IF Condición **THEN** Acción"

Por ejemplo,

IF (DENSIDAD es PE AND VELOCIDAD es LE)
THEN (LVERDE es CO, LROJA es LA)

De éste modo se generan todas las reglas posibles las cuales se enlistan a continuación:

1 If (Densidad es PE and Velocidad es LE) Then (LROJA es LA, LVERDE es CO)
2 If (Densidad es PE and Velocidad es MR) Then (LROJA es ML, LVERDE es ML)
3 If (Densidad es PE and Velocidad es RA) Then (LROJA es CO, LVERDE es LA)
4 If (Densidad es MP and Velocidad es LE) Then (LROJA es ML, LVERDE es ML)
5 If (Densidad es MP and Velocidad es MR) Then (LROJA es CO, LVERDE es LA)
6 If (Densidad es MP and Velocidad es RA) Then (LROJA es CO, LVERDE es Const.)
7 If (Densidad es LI and Velocidad es LE) Then (LROJA es CO, LVERDE es Const.)
8 If (Densidad es LI and Velocidad es MR) Then (LROJA es CO, LVERDE es Const.)
9 If (Densidad es LI and Velocidad es RA) Then (LROJA es CO, LVERDE es Const.)

Para asegurarnos que todas las posibles acciones se cubrieron podemos implementar las matrices de la Figura 2.5.

LUZ_ROJA

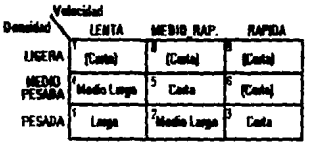


Fig. 2.5.a Matriz de la LUZ-ROJA

LUZ_VERDE

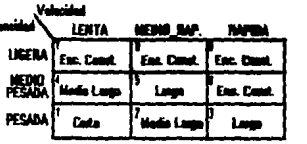


Fig. 2.5.b Matriz de la LUZ-VERDE

En la matriz de la Figura 2.5.b se muestra un fuzzy set de salida denominado *Enc. Const.* que no se muestra en las gráficas de las funciones de membresía de salida, sin embargo el criterio a seguir para entender la función de tal salida es simple, cuando las condiciones de tráfico fluido permanezcan constantes la LUZ_VERDE permanecerá encendida hasta el momento en que cambien las condiciones del tráfico.

Las reglas indican el método de inferencia, de modo que para ilustrarlo se asignarán valores a los datos de entrada. Se asume: DENSIDAD = 0.35 y VELOCIDAD = 17 MPH. La densidad proyecta un grado de membresía en los fuzzy sets MEDIO_PESADA (MP) y PESADA (PE), de manera similar la velocidad de 17 MPH proyecta grados de membresía en los fuzzy sets MEDIO_RAPIDA (MR) y LENTA (LE). Tales valores proyectados son:

$$\mu MP(0.35) = 0.80$$
$$\mu PE(0.35) = 0.28$$
$$\mu MR(17) = 0.50$$
$$\mu LE(17) = 0.22$$

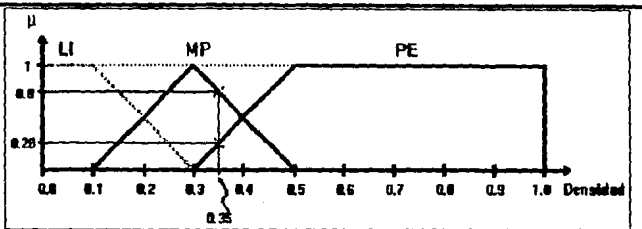


Fig. 2.6 Proyecciones de el dato de entrada 0.35 para la variable Densidad.

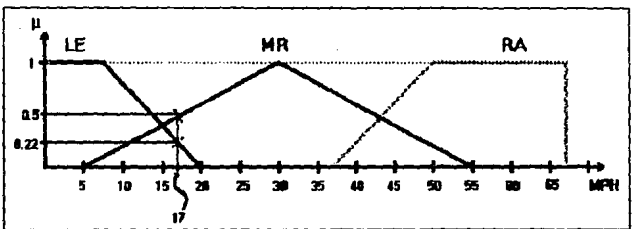


Fig. 2.7 Proyecciones del dato de entrada 17 MPH para la variable Velocidad.

Aplicando éstos 4 grados de membresía diferentes de cero a la Base de Reglas observamos que sólo se activan 4 reglas, es decir, las reglas 1, 2, 4, y 5 (permaneciendo inactivas las reglas restantes). Por ejemplo la Regla 1 nos indica:

IF (DENSIDAD es PE AND VELOCIDAD es LE)
THEN (LROJA es LA, LVERDE es CO)

es decir,

$$\mu LA(LROJA) = \mu CO(LVERDE) = \min(\mu PE(0.35), \mu LE(17))$$
$$= \min(0.28, 0.22)$$
$$= 0.22 \qquad \text{Regla 1}$$

de igual manera:

$$\mu ML(LROJA) = \mu ML(LVERDE) = \min(\mu PE(0.35) = 0.28, \mu MR(17) = 0.50) = 0.28 \quad \text{Regla 2}$$

$$\mu ML(LROJA) = \mu ML(LVERDE) = \min(\mu MP(0.35) = 0.80, \mu LE(17) = 0.22) = 0.22 \quad \text{Regla 4}$$

$$\mu CO(LROJA) = \mu LA(LVERDE) = \min(\mu MP(0.35) = 0.80, \mu MR(17) = 0.50) = 0.50 \quad \text{Regla 5}$$

2.2.e Determinar acciones de salida.

Realizada la inferencia se debe tomar una acción resultante de las 4 obtenidas, en éste artículo emplean el método del "Centroide" (ver nota al final del capítulo).

"Tal y como lo muestra la Figura" (Figura 2.8 en nuestro caso) "empleamos la técnica del Centroide para calcular la duración de la LUZ_ROJA, la acción de la regla 4 da un valor de verdad μML = 0.22. El centroide del área resultante se encuentra en LUZ_ROJA Regla 4 = 6 seg."

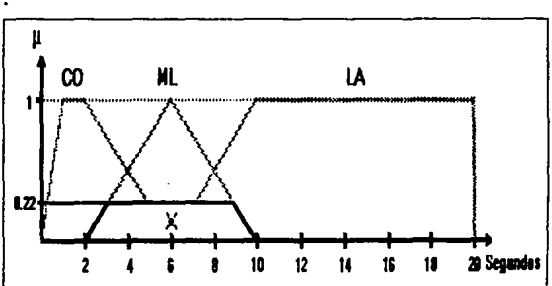


Fig. 2.8 Muestra gráfica del Centroide del Fuzzy Set ML.

Para encontrar el centroide de un triángulo el método gráfico es el más simple, ya que el centroide se encuentra en el punto de intersección de las medianas del triángulo, para el caso de un trapezoide la altura del centroide se determina mediante la ecuación siguiente:

$$y = (h/3)*((a + 2b) / (a + b))$$

donde:

- h - es la altura del trapezoide (la cual es el valor de membresía obtenido por la inferencia).
- a - es la base superior del trapezoide.
- b - la base inferior del trapezoide

El valor de la altura obtenido nos sirve como factor para reducir la distancia entre los puntos medios de las bases, el valor reducido se suma o resta al punto medio de la base inferior (dependiendo de la posición del punto medio de la base superior), para encontrar la posición del centroide.

Por ejemplo calculemos el centroide del fuzzy set "CO" del universo de salida de la LUZ_ROJA, para ello fijaremos los valores de los vértices del trapezoide en 4 puntos coordenados, la base inferior la forman los puntos:

P1 = (0,0)
P2 = (6,0)

mientras que la base superior está formada por los puntos:

P3 = (2,1)
P4 = (3,1)

La longitud de la base superior es = 1 unidad (x4 - x3)
La longitud de la base inferior es = 6 unidades (x2 - x1)
El grado de membresía es = 1 unidad(toda la figura)

La altura "y" del centroide es por tanto:

$$y = (1/3)*((1 + 12)/(1 + 6))$$
$$y = 0.619$$

El punto medio de la base superior (xs) se encuentra en:

xs = 2.5

El punto medio de la base inferior (xi) se encuentra en:

xi = 3

La diferencia entre los dos puntos medios es de 0.5 unidades, por lo cual aplicamos el factor encontrado en "y" a la distancia entre los puntos medios dando un valor al cual llamaremos "xd"

$$xd = y * (xi - xs)$$
$$xd = 0.619 * 0.5$$
$$xd = 0.3095$$

El valor de "xd" se lo restamos al valor del punto medio de la base inferior (xi) ya que el valor del punto medio de la base superior (xs) se encuentra mas a la izquierda, por lo que obtenemos la distancia "x" del centroide respecto al origen:

$$x = xi - xd$$
$$x = 3 - 0.3095$$
$$x = 2.6905$$

Por lo tanto la posición del centroide dentro del fuzzy set "CO" se encuentra en:

$$Pc = (x , y)$$
$$Pc = (2.6905 , 0.619)$$

donde:

Pc - es el punto coordenado del centroide.

Podrán observarse diferencias entre el resultado calculado y el resultado proporcionado en el articulo, pero éstas se deben a la falta de los valores exactos en que se encuentran los vértices del trapezoide.

"De modo similar, calculando los centroides para las áreas de salida correspondientes a las acciones 1, 2, y 5 se obtiene: LUZ_ROJA Regla 1 = 13.2 seg., LUZ_ROJA Regla 2 = 6 seg., LUZ_ROJA Regla 5 = 2.75 seg. Un método de promedio de pesos, donde los pesos son el respectivo valor de verdad, combina todos esos centroides. La salida final es:"

$$t_{LROJA} = ((13.2)(0.22) + (6)(0.28) + (6)(0.22) + (2.75)(0.50)) / (0.22 + 0.28 + 0.22 + 0.50)$$
$$= 5.96 \text{ segundos.}$$

NOTA:

Se habrá notado que al principio de ésta sección indicamos Centroide entre comillas, cuando mencionamos el método de defuzzificación, esto es debido a que en realidad están empleando Singletons, si emplearan el Centroide como dicen tendrían que haber reemplazado todos los valores de "μ" por el área bajo la curva.

Como vemos el proceso de plantear y resolver un problema mediante la técnica de la Lógica Difusa no es difícil, sino mas bien laborioso ya que para realizar todos los pasos y corregirlos se tiene que hacer iterativamente, lo cual hace ideal a una computadora para efectuar tal proceso de refinamiento de parámetros.

Capítulo

Estructuras Lógicas Fuzzy-Logic.

3.1 Herramientas Lógicas.

Como pudimos ver en los capítulos anteriores el empleo de la Teoría de Conjuntos de Zadeh nos permite resolver problemas lógicos de un modo poco complejo y fiable, por ello problemas complejos de cualquier índole, no solo de control, se reducen a unos cuantos pasos muy repetitivos, esto es, las entradas de los problemas varían dentro de su universo respectivo y pueden tomar cualquier valor en un instante determinado obteniéndose en ése momento un par

de entradas "P1" que infieren un resultado "R1", como en el ejemplo anterior, en otro instante puede obtenerse un par "P2" diferente de "P1", por tanto "R2" es inferida, éste proceso continúa indefinidamente por lo que si tenemos "n" pares de entrada obtendremos "n" inferencias, en vista de lo expuesto es deseable contar con herramientas que nos auxilien en el proceso de inferencia, siendo una de las metas el que tales herramientas puedan tomar decisiones por sí solas, que en otras palabras no es más que implementar un autómata dedicado a resolver o controlar un problema.

Desde el punto de vista de sistemas de control estos autómatas dedicados reciben el nombre de Controladores, que implementados sobre bases de la Teoría de Conjuntos de Zadeh se nombran Controladores Fuzzy Logic o simplemente FLC's. Al igual que las implementaciones convencionales los FLC's pueden realizarse en 3 medios básicos:

1. Software.
2. Software_Hardware.
3. Hardware.

que obviamente deben ser más simples que los convencionales, debido al menor número de operaciones y el menor grado de complejidad de las mismas. A continuación se describe brevemente cada una de ellas.

3.1.1 Software

A lo largo de éste trabajo de investigación hemos podido apreciar que existen 2 características dentro del proceso de inferencia de la Lógica Fuzzy que hacen a dicho proceso apto para ser aplicado en computadoras convencionales, éstas características son: (1) el formato de las Reglas que Gobiernan al proceso y (2) el carácter iterativo de la Lógica Fuzzy. Por éstas razones no es de extrañarse que las primeras aplicaciones se hubieran implementado en Software. La primera de ellas, el formato de las reglas, es fácil de comprender, ya que la estructura IF...THEN... de las reglas ha sido "copiado" de los sistemas expertos que corren en computadoras convencionales. La segunda característica es un poco más complicada de visualizar, por lo que la veremos al final de

ésta sección, por lo pronto daremos una breve muestra de las funciones que algunos paquetes ofrecen.

En la actualidad, el diseñador puede elegir el software adecuado a sus necesidades y/o presupuesto, así podemos encontrar paquetes sencillos que nos permiten graficar en dos dimensiones la acción tomada o bien emplear aquellos que de un modo tridimensional trazan las acciones de todo el proceso pudiéndose observar simultáneamente la intersección de los antecedentes que generan una acción, tal es el caso de 'CubiCalc', existe también software de este tipo con un mayor grado de especialización que permite una vez introducidas las funciones de pertenencia modificarlas desde una pantalla gráfica ya sea por medio de un mouse o por otro periférico y con ello alterar las acciones ejecutadas, haciendo más ágil el diseño de un sistema, por ejemplo 'Fuldek' que corre bajo ambiente Windows. También existe software especialmente diseñado para realizar todo el proceso de inferencia y proporcionar la respuesta correcta en el instante requerido, como en el caso de estimaciones del comportamiento de la bolsa de valores, del estado del tiempo, etc. Sin embargo éstos programas sólo responden a las entradas de las computadoras y proporcionan un resultado que no acciona ningún dispositivo de salida, es decir, sólo trabajan dentro de las computadoras. Por último encontramos el Software que simula las condiciones de operación de un control fuzzy logic, gracias a éstos programas podemos "revisar" y ajustar el comportamiento del FLC (Controlador Fuzzy Logic), éste proceso, como ya mencionamos, es iterativo por tanto los Simuladores son de gran ayuda, aunado a esto se tiene el hecho que un FLC al no contar con un modelo matemático debe ser simulado repetidamente hasta lograr la respuesta adecuada a todas las condiciones de operación. Por ésta razón mencionamos al principio de la sección que la Lógica Difusa cuenta con un carácter iterativo.

3.1.2 Software_Hardware.

También llamado Control Fuzzy Logic Integrado a Computadoras, es un grupo de aplicaciones orientadas al control que se encuentra dentro del tenue límite que existe entre el Software y el Hardware y que emplea a ambos para poder trabajar. En éstos casos el fabricante de este tipo de controles proporciona una tarjeta compatible a la plataforma empleada y el Software necesario para programar dicha tarjeta.

Al igual que el Software, las tarjetas son de diferentes costos, y por tanto, con diferente capacidad. De ésta manera podemos encontrar desde kits didácticos, de US$ 195, hasta herramientas muy potentes, para resolver problemas de control en tiempo real, como las provistas por Togai Infralogic con costos de hasta US$ 18,500.00. Generalmente éstas tarjetas son provistas de un Kernel Fuzzy, el cual es un esqueleto de código programable (en donde el usuario sustituye sus datos de control) y en donde se realizan los 3 pasos básicos de la lógica fuzzy -Fuzzificación, Evaluación de Reglas y Defuzzificación-, en éste especificamos tanto los rangos de entrada como los de salida, como en el caso del Neuron Chip de Echeleon empleado por Motorola en su microcontrolador MC143150, el cual se programa por medio de un código específico denominado "Neuron C". Motorola también provee Kernel's para 2 de sus microcontroladores de 8 bits, el MC68HC11 y el HC05, así como para el controlador de 16 bits HC16 y para la familia de 32 bits, los de la serie 68000.

El que un dispositivo de control se encuentre integrado a una computadora tiene la enorme ventaja de poder ser programado para realizar procesos diversos sin que para ello se tenga que emplear mucho tiempo en su programación, siendo su mayor desventaja el estar ligado a la computadora, lo que los hace en algunos casos inaplicables, ya que dependiendo del proceso por controlar es más simple emplear un controlador específico que una computadora dedicada, tal y como sucede con los PLC's (Controladores Lógicos Programables).

3.1.3 Hardware.

El país que ha prestado mayor atención y apoyo al desarrollo de tecnología fuzzy es Japón, por ello los mayores adelantos han aflorado en este país. Los primeros desarrollos de Hardware en Japón, y en el mundo, se guardaron celosamente hasta que fueron patentados a nivel internacional, no fue sino hasta entonces que comenzaron a publicarse algunas arquitecturas básicas y hasta la fecha son pocos los artículos que hablan de Hardware, la mayor parte de las publicaciones se basa en simulaciones efectuadas en Software.

En las aplicaciones dependientes del Software, como el tren Sendai en Japón, se tiene la deficiencia de no ejecutar en paralelo a las reglas de control, como lo requiere la Inferencia Fuzzy, esto se debe a que las computadoras comunes basan su funcionamiento en la Lógica Booleana, por lo que realizan los procesos secuencialmente, salvo en lo casos en que se cuente con circuitos específicos para tal proceso, como en el caso de los paquetes Togai Infralogic que incorporan chips fuzzy en sus tarjetas. Por ello resulta obvio que una arquitectura especialmente desarrollada para trabajar la Inferencia Fuzzy tenga mejores respuestas que el software corriendo en plataformas convencionales. Es por ello que los resultados de las investigaciones han aportado Hardware Inferencial, el cual puede ser de 2 tipos: Analógico y Digital, siendo de vital importancia para ambos el empleo de computadoras para el diseño y simulación de los mismos.

Cualquiera de las dos clases de Hardware son diseñados de modo tal que reciban la información de entrada y la Fuzzifiquen, posteriormente se activan o analizan las reglas de comportamiento y por último se Defuzzifica una Consecuencia. La exactitud del proceso y el modo de implementar los tres pasos mencionados depende del método de Composición empleado, variando en cada aplicación.

Debido a la naturaleza de la Lógica Fuzzy, las arquitecturas diseñadas en bases Analógicas responden mejor a el manejo de la información imprecisa que las arquitecturas Digitales, por tanto responden mejor. Sin embargo, la ventaja de los circuitos analógicos es relativa debido a que (1) los cálculos de inferencia en circuitos digitales son menos complejos y (2) es común que los circuitos analógicos sean de propósito específico mientras que uno digital posee la flexibilidad de poderse programar para diferentes aplicaciones, es decir, un Hardware Digital Fuzzy Logic puede hacerse de propósito general. La mayor desventaja con que se enfrentan ambas implementaciones es el excesivo requerimiento de componentes debido al paralelismo del proceso de inferencia, pero ello se compensa con la velocidad de respuesta del circuito.

El objetivo que se fijó para este trabajo de investigación es el desarrollar una arquitectura acorde a nuestros recursos, ya que no contamos con la tecnología y la capacitación como para diseñar circuitos integrados a nivel de microelectrónica. En las secciones siguientes se describe el desarrollo de la arquitectura propuesta, que aunque se maneja como tres arquitecturas separadas se trata en realidad de la evolución de la tercera arquitectura.

3.2 Estructura Fuzzy Logic de Propósito Específico. (Arquitectura 1)

Como se mencionó en la sección precedente, existen 2 maneras posibles de desarrollar arquitecturas fuzzy logic, aquellas Analógicas y las Digitales, por lo cual nuestro primer paso fue decidir entre ellas. Para poder tomar una decisión tuvimos que hacer un recuento de los recursos disponibles y de los que de alguna manera podíamos obtener, resultando nuestra inclinación hacia medios digitales por diversos motivos, de los que destacan: (1) los cálculos de inferencia en circuitos digitales son más simples, (2) la flexibilidad para poder programar los CI's (Circuitos Integrados) digitales los hacen ideales para aplicaciones de propósito general, esto aunado a que estamos más familiarizados con el diseño de aplicaciones digitales y el gran número de componentes que se requieren hacen imprescindible que una arquitectura analógica sea implementada mediante microelectrónica.

La segunda interrogante que nos surgió fue ¿Cómo realizamos el Proceso de Inferencia? La respuesta resultó bastante obvia, ya que debíamos implementar una arquitectura que efectuara la inferencia apegándose paso a paso a la metodología antes vista. De la teoría mostrada con anterioridad nos dimos cuenta que se requiere de una circuitería capaz de realizar la Composición de Máximos y Mínimos, pero para llevar a cabo esta operación se necesita codificar de alguna manera las entradas para así representar la fuzzificación de las mismas. De igual manera se debe decodificar o defuzzificar todas las acciones simultáneas que se generaran durante la Composición Máx-min, por tanto, decidimos desarrollar una arquitectura compuesta por los 3 bloques fundamentales del proceso inferencial:

a) Bloque dedicado a la Fuzzificación
b) Bloque de Composición de Máximos y Mínimos
c) Bloque para la Defuzzificación

los cuales se muestran en la Figura 3.1.

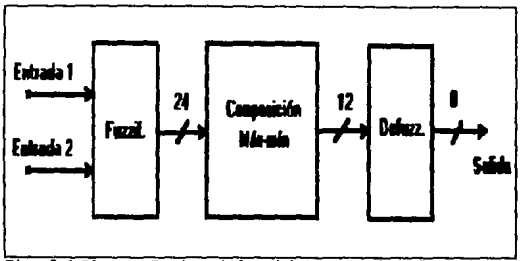


Fig. 3.1 Bloques Fundamentales del Proceso de Inferencia.

En la figura 3.1 se muestra con flechas la transmisión de los datos de un problema de control cualquiera, las diagonales que cruzan a las flechas indican el número de bits en que codificamos cada compartimento de datos entre bloques, de tal modo que se respetan los estándares de los sistemas digitales en donde un bus de datos se indica con una sola línea cruzada por una diagonal indicando el número de bits transmitidos.

La idea original era realizar una arquitectura generalizada para poder aplicarla a la mayor parte de los problemas de control de 2 variables de entrada y una de salida, lo cual resultó ser más complicado de lo que suponíamos, ya que cada uno de los 3 bloques fundamentales depende directamente de las características particulares del problema por resolver, ésta dependencia se indica a continuación:

1. La Fuzzificación depende del número de fuzzy sets en que se divida cada variable de entrada, ya que a su salida se debe entregar los grados de membresía de todos los fuzzy sets de ambas entradas simultáneamente.

2. La composición máx-min depende de tres aspectos:
   - El número de fuzzy sets de cada variable de entrada
   - El número de fuzzy sets de la variable de salida
   - La combinación de fuzzy sets de entrada que activan a cada fuzzy set de salida.

   Es por ello que éste bloque es el que mayor cuidado debe tenerse al diseñarse y además es el que requiere un

número de componentes creciente según los universos de las variables del problema.

3. La defuzzificación además de estar sujeta al número de fuzzy sets de salida esta supeditada al método empleado para tal fin.

Por las razones antes mencionadas tuvimos que proponer una arquitectura específica para un problema que se ajustara al método de inferencia que encontramos más conveniente para ser desarrollado en hardware. El problema a implementar lo pudimos haber propuesto nosotros, pero decidimos no hacerlo así ya que ¿De qué manera comprobamos si estamos obteniendo resultados correctos? De modo que optamos por implementar el problemas descrito en el capítulo 2 sección 2, que al haber aparecido en una publicación internacional es factible el hecho de que sus resultados sean correctos, además que se ajusta exactamente a el tipo de aplicación que deseamos desarrollar, aunque con ciertas modificaciones para poderlo adaptar a nuestro prototipo.

3.2.1 Bloque Fuzzificador.

(La explicación aquí dada se refiere a una sola variable de entrada, sin embargo los mismos criterios mencionados se extienden para la segunda variable de entrada).

Recordemos que la variable Densidad de tráfico del problema mencionado en la sección 2.2 es sensada en un rango de 0 a 1, y ésta mide el espacio existente entre autos, ése mismo Universo es empleado por nuestro sistema sólo que al muestrearlo y digitalizarlo en 8 bits, proporcionados por el convertidor Analógico-Digital ADC0800, obtenemos un Universo equivalente que varía de 0 a 255, debido a que el convertidor proporciona una respuesta lineal a la entrada el 0 analógico o valor mínimo equivale a la combinación 0 y el 1 analógico o valor máximo equivale a la combinación 255, como se muestra en la Fig. 3.2.a. En donde se muestra el cambio de ejes que se efectúa para poder interpretar gráficamente la digitalización de las entradas, en las que además se proporcionan valores intermedios entre el 0 y el 255, podemos mostrar valores intermedios debido a la linealidad del convertidor los cuales se obtienen por medio de una regla de tres. De la misma manera el grado de pertenencia se decidió muestrearlo con 4 bits, lo cual nos da 16 valores diferentes entre μ=0 y μ=1, siendo un 0 analógico igual a un 0 decimal y un 1 analógico a un 15 decimal. Fig. 3.2.b.

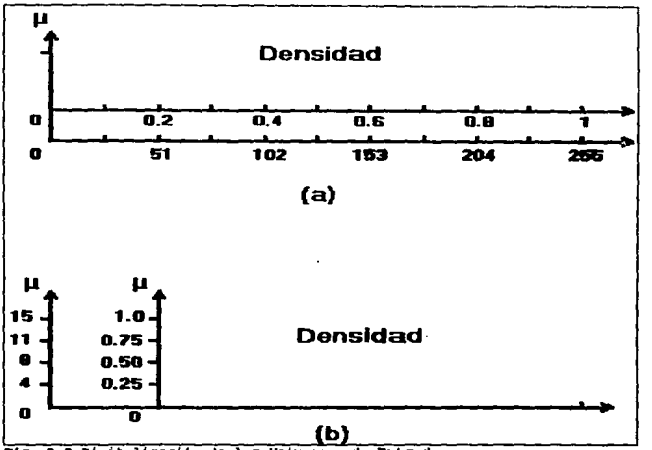


Fig. 3.2 Digitalización de los Universos de Entrada

La razón por la que se codificó el grado de membresía en 4 bits se debe a que comercialmente la mayor parte de Circuitos Integrados (CI's) digitales emplea 8 bits como palabra de trabajo y nosotros requerimos procesar 2 valores de membresía simultáneos, como se verá más adelante.

El Universo de la Densidad de tráfico se encuentra cubierto por 3 fuzzy sets; LI (0 -- 0.3), MP (0.1 -- 0.5) y PE (0.3 -- 1), ver la Figura 3.2.1, de modo que cualquier valor de entrada en un instante dado proyecta simultáneamente un grado de pertenencia en cada fuzzy set, por ejemplo, una entrada de densidad igual a 0.156 proyecta los valores siguientes:

Valores Analógicos	**Valores Digitales**
$\mu LI(0.156) = 0.733$	$\mu LI(40_d) = 11_d = B_H$
$\mu MP(0.156) = 0.267$	$\mu MP(40_d) = 4_d = 4_H$
$\mu PE(0.156) = 0.0$	$\mu PE(40_d) = 0_d = 0_H$

donde: los subindices d y H, significan valores Decimales y Hexadecimales respectivamente.

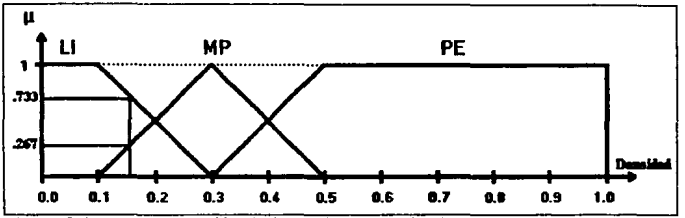


Fig. 3.2.1 Rangos de acción de la variable Densidad.

Como se observa, el valor de entrada sólo activó a los fuzzy sets LI y MP, sin embargo, debemos aplicar la entrada a los 3 fuzzy sets, ya que en un proceso real donde la información es aleatoria no se puede prever cual de los tres fuzzy sets se activará, por ello un valor de membresia diferente de cero nos indica activación del fuzzy set. De éste modo es fácil visualizar que requerimos de tres circuitos que emulen el comportamiento de cada uno de los fuzzy sets, a los que denominamos: GLI, GMP y GPE según el fuzzy set que represente. Esto se muestra en la Figura 3.3.

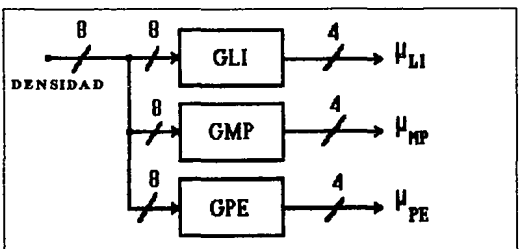


Fig. 3.3 Bloques de los Circuitos Fuzzificadores de la Variable DENSIDAD.

En la Figura 3.3 se muestran los tres circuitos que se encargan de realizar la fuzzificación de la entrada ya convertida a un código binario de 8 bits (en este caso la entrada es el valor de la Densidad de Tráfico), los circuitos están etiquetados con los nombres:

"GLI" - GAL para el Fuzzy Set Ligero
"GMP" - GAL para el Fuzzy Set Medio Pesado
"GPE" - GAL para el Fuzzy Set Pesado

Cuyas salidas son los valores de membresía codificados en 4 bits, y que reciben las etiquetas:

"μLI" - Grado de Membresía en el Fuzzy Set Ligero
"μMP" - Grado de Membresía en el Fuzzy Set Medio Pesado
"μPE" - Grado de Membresía en el Fuzzy Set Pesado

El funcionamiento del Bloque de los circuitos fuzzificadores (Figura 3.3) es el siguiente. La entrada se alimenta en paralelo a los tres circuitos y cada uno de ellos entrega un valor de grado de membresía respectivo al fuzzy set que representa, si retomamos el ejemplo anterior en donde la combinación de la entrada codificada en 8 bits, que tuvo un valor de "40d", activó a los fuzzy sets "LI" y "MP", aplicando esta entrada al circuito mostrado se tendrá el mismo efecto, activando a los elementos del circuito "GLI" y "GMP", los cuales proporcionan una salida de 4 bits que representa el valor de la membresía que para el valor de entrada se tiene en cada Fuzzy Set. La manera en que fuzzifican la entrada es una simple codificación de un dato binario de entrada de 8 bits en uno de salida de 4 bits.

Como se mencionó en las líneas anteriores la fuzzificación de la entrada la realizan GAL's (que son Circuitos Lógicos Programables mediante el simulador PALASM2), los cuales son sensibles sólo al rango que cubre cada fuzzy set, por ejemplo, para el circuito "GLI" proponemos un GAL que entregue resultados desde la combinación de entrada 00H hasta 4DH, es decir desde un 0 analógico hasta un 0.3 analógico que es el rango de valores que cubre el fuzzy set LI, siendo insensible desde 4EH hasta FFH (desde el 0.31 analógico hasta el 1.0 analógico), tal y como se muestra en la tabla siguiente.

Tabla de acción del circuito GLI.

Entrada	Salida	Entrada	Salida
00	F	34	7
:	:	:	:
1C	E	37	6
:	:	:	:
20	D	3B	5
:	:	:	:
23	C	3E	4
:	:	:	:
26	B	42	3
:	:	:	:
2A	A	45	2
:	:	:	:
2D	9	48	1
:	:	:	:
31	8	4C	0
:	:	:	:
		FF	0

La representación gráfica de la tabla para el Fuzzy Set LI se muestra en la figura 3.4. Donde se observa una escala decimal que va desde la combinación 0 hasta la 255, lo cual equivale a la codificación del Universo de entrada en 8 bits, (los valores que se aprecian abajo de los datos decimales son los valores equivalentes hexadecimales), como se aprecia en la figura el Fuzzy set LI se encuentra dividido en 16 valores de membresía que van desde el 0 hasta el 1, los cuales son las 16 posibles combinaciones que podrían tener los 4 bits del código de membresía, de modo que si vamos graficando cada valor de entrada junto con su valor de salida que proporciona el circuito GLI podemos obtener la gráfica 3.4.

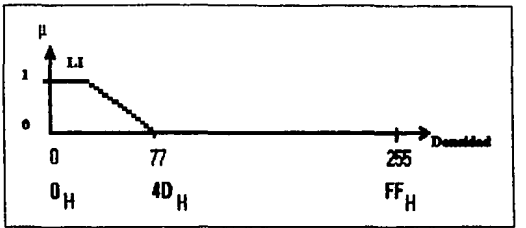


Fig. 3.4 Forma gráfica de la respuesta del Circuito GLI.

Teóricamente debería haber respuesta en 4C y 4D, pero los valores de entrada son tan pequeños que el convertidor los considera como cero. De una manera similar se crean los circuitos para fuzzificar MP y PE, tomando en cuenta los límites de cada uno de los Fuzzy sets de.

3.2.2 Bloque Composicional.

Los 6 grados de pertenencia entregados por el bloque fuzzificador, 3 de la entrada Densidad y 3 de la entrada Velocidad, son procesados por el bloque composicional aplicándoles la Composición max-min gobernada por la matriz de reglas siguiente, misma que se mostró en la Figura 2.5.a.

Matriz de Reglas para la Luz Roja.

DE \ VE	LE	MR	RA
LI	CO	CO	CO
MP	ML	CO	CO
PE	LA	ML	CO

La composición la realizamos siguiendo paso a paso el método visto en la sección 2.2, es decir, mediante un par de entradas que se aplica a cada una de las reglas detectamos cuales se activan, la

o las reglas activadas mediante el par de datos antecedente ejecutan una acción según sea el mínimo grado de membresía antecedente. En éste punto añadimos un paso extra, para nuestra arquitectura requerimos el emplear el máximo grado de pertenencia de todas las reglas que activen un mismo fuzzy set de salida, esto es debido a que empleamos Singletons como funciones de pertenencia para la salida. La Figura 3.5 muestra las partes que constituyen al Bloque Composicional, el cual se compone a su vez por dos bloques, el Bloque de Mínimos (MIN) y el Bloque de Máximos (MAX).

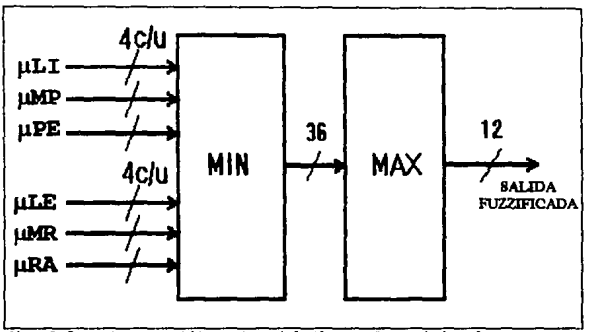


Fig. 3.5 Partes constituyentes del Bloque Composicional.

De la matriz de reglas claramente se puede observar que existe un total de 9 reglas, cada una de ellas formada por un par antecedente el cual activo o no infiere una respuesta; como mencionamos anteriormente, debemos obtener el valor mínimo del par antecedente en cada una de las reglas (para lo cual se diseñó el circuito de mínimos), con ésta operación podría surgir una pregunta ¿Qué consecuencia se tiene si el circuito de mínimos escoge al cero como resultado? En un principio se pensó que podría alterarse la respuesta de la inferencia, sin embargo analizando más detenidamente al proceso de inferencia llegamos a la conclusión de que no afecta en nada el tener a un cero como resultado de la acción de una regla, debido a que un cero a la salida del circuito de mínimos indica que ésa regla no se encuentra activa.

La salida de 4 bits de cada uno de los Circuitos Mínimos se conecta físicamente a la entrada del circuito de Máximos que indica la matriz de reglas, es éste alambrado fijo el que le da el carácter de "Controlador de propósito específico" para la aplicación propuesta, en la Fig. 3.6 se puede ver la implementación de la matriz de reglas mediante las conexiones existentes entre los circuitos de mínimos y máximos.

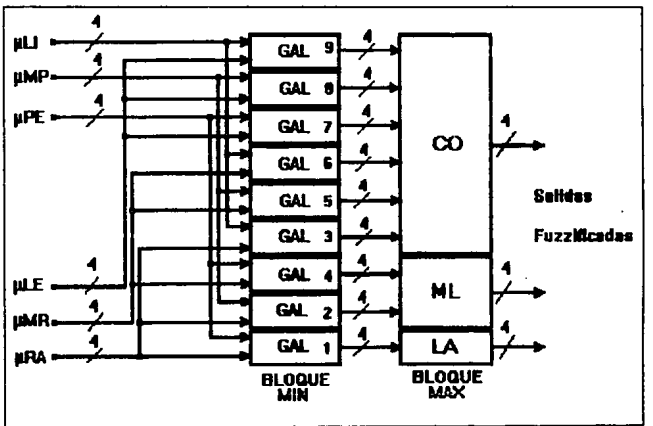


Fig. 3.6 Diagrama completo del Circuito Composicional de Propósito Específico

Para poder explicar el funcionamiento de los bloques de Mínimos y Máximos, tomaremos como referencia los valores de entrada siguientes: Velocidad = 17 MPH, Densidad = 0.35 autos/pie que son los mismos valores que se citaron en el ejemplo 2.2.d. Los valores fuzzificados que se obtienen de las entradas citadas son:

DENSIDAD	VELOCIDAD
μMP(0.35) = 0.80	μMR(17) = 0.50
μPE(0.35) = 0.28	μLE(17) = 0.22

Se recordará que con esos valores se activan las reglas 1, 2, 4 y 5, de manera similar se activan los GAL's GAL1, GAL2, GAL4 y GAL5, en donde el número del GAL activado coincide con la regla activada. Si nos fijamos en la matriz de reglas del mismo ejemplo

veremos que las acciones correspondientes a cada una de las reglas activas son:

Para la Regla 1 la acción LARGA (LA)
Para la Regla 2 la acción MEDIO LARGA (ML)
Para la Regla 4 la acción MEDIO LARGA (ML)
Para la Regla 5 la acción CORTA (CO)

En la figura 3.6 se observa que la salida de los GAL's coincide con la acción correspondiente a efectuar por la regla activa, siendo evidente que dicha acción se realiza debido a que la salida de cada GAL del bloque mínimo se encuentra unido por medio de un bus de datos exclusivo para la acción de salida deseada.

Los circuitos de mínimos y máximos son comparadores de magnitud, sin embargo, en el mercado los comparadores de magnitud sólo cuentan con salidas que indican cual de los 2 números es mayor, menor o si son iguales, lo cual cubre en parte nuestras necesidades mas no las satisface totalmente, ya que requerimos que aunado a determinar que entrada es la mayor o menor el circuito debe ser capaz de transmitir dicho valor a su salida, por tal razón tuvimos que diseñar en un GAL un circuito que comparara 2 números y q mostrara en sus salidas el menor o mayor de ellos, según se re iriera.

Los circuitos máximos entregan una única salida de 4 bits para ca consecuencia inferida, de modo que cuando una misma co cuencia recibe más de 2 antecedentes debe conectarse en ca da circuitos máximos de dos entradas o bien, diseñar un ci uito con mayor número de entradas.

3.2.3 Bloque Defuzzificador.

El último de los tres bloques fundamentales es el encargado de procesar una acción resultante, la finalidad de éste bloque es el implementar la ecuación:

$$\text{Salida} = \frac{\sum_{i=1}^{n} \mu_i S_i}{\sum_{i=1}^{n} \mu_i} \qquad (1)$$

donde:

μ - es el grado de membresía
S_i - es el Singleton de salida

Siendo la manera propuesta la mostrada en la figura 3.7. Aunque a primera vista parece un circuito simple en realidad es bastante complejo, por ejemplo, el circuito más sencillo es el etiquetado como "$\Sigma\mu$", quien suma los 3 grados de pertenencia de 4 bits que resultan del Bloque de Máximos, para poder realizar ésta simple suma requerimos diseñar 2 circuitos sumadores completos de datos de 4 bits. Por otra parte el multiplicador "MULT" es en realidad un circuito compuesto por 3 bloques, cada uno de los cuales está destinado a multiplicar su grado de membresía con su respectivo Singleton, proporcionando a su salida un dato de hasta 11 bits, y hasta donde hemos visto no existe un dispositivo programable (PLD) de 11 bits de salida. Que decir del sumador de datos multiplicados "ΣM" que recibe 33 bits y entrega 14 en el caso más extremo.

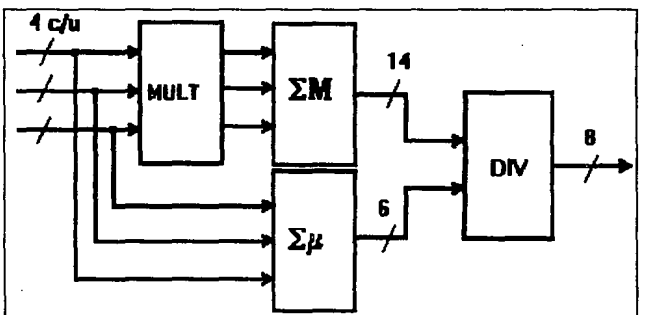


Fig. 3.7 Diagrama a Bloques de la Circuitería planteada para Defuzzificar las consecuencias del Bloque Inferencial.

Se puede apreciar que el Bloque Defuzzificador es el que presenta un mayor problema para poder desarrollarse en la arquitectura propuesta, por ello dejamos al criterio del lector interesado la optimación de dicho bloque de salida, pudiendo desarrollarse un arreglo de circuitos para simplificar la función

deseada, o bien emplear una Unidad Aritmética-Lógica (ALU) si se desea obtener un resultado digital, o un Convertidor Digital-Analógico (DAC) en cada una de las tres salidas del bloque de máximos y así realizar la defuzzificación analógicamente.

Otro aspecto que nos obligó a buscar otra alternativa y abandonar el desarrollo de ésta arquitectura fue el elevado número de componentes que aumenta el costo del circuito. El número total de componentes requeridos, en caso de llegar a implementarse éste diseño, se ha estimado en tener un mínimo de 34 CI's.

Lógicamente el número de componentes aumenta proporcionalmente al número de fuzzy sets de entrada y salida, ya que mayor Número de fuzzy sets de entrada implica más circuitos fuzzificadores y una matriz de reglas más grande, debido a que el número de reglas "NR" está dada por la siguiente expresión:

$$NR = N \times M$$

donde N es el número de fuzzy sets de la variable 1, y M el de la variable 2. De igual modo las entradas del bloque defuzzificador aumentarán según sea el incremento del número de fuzzy sets de la salida.

La arquitectura completa del circuito de propósito específico se muestra en la Figura 3.8.

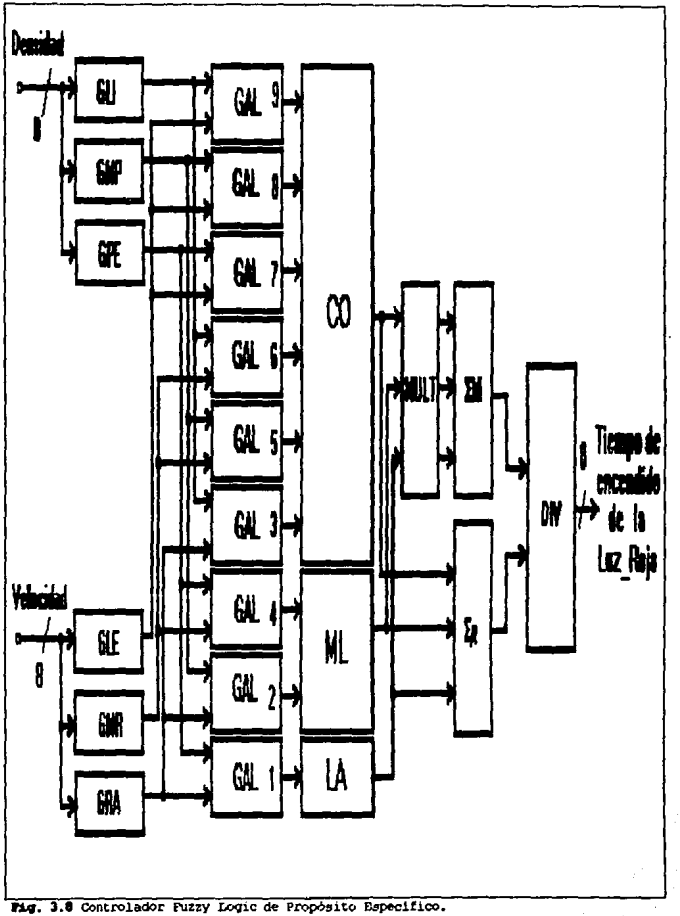


Fig. 3.8 Controlador Fuzzy Logic de Propósito Específico.

3.3 Estructuras Lógicas Fuzzy Logic de Propósito General.

Como mencionamos, el mayor problema que se tuvo al plantear la arquitectura antes descrita fue la defuzzificación, aunado a esto el elevado número de componentes requerido hace a dicho circuito incosteable e impráctico para ser aplicado en una tarea, teniendo la ventaja, en caso de funcionar, que todo el proceso inferencial se realiza en paralelo. Para su correcto funcionamiento requeriría de un sencillo controlador que regule la sincronía del flujo de información a través del circuito. Afortunadamente fueron estas razones las que nos llevaron a buscar otras alternativas en el diseño del circuito inferencial, con lo cual no sólo se logró reducir el diseño anterior, sino que vimos que se podía extender a "casi" cualquier otra aplicación de dos entradas y una salida. En las secciones siguientes se explican las dos arquitecturas fundamentales de propósito general que se obtuvieron.

3.3.1 Controlador Fuzzy Logic de Propósito General -Restringido. (Arquitectura 2).

Esta primera arquitectura de propósito general es una mejora directa de la antes descrita, sin embargo, a pesar de que en ella resolvemos al problema de la defuzzificación sigue estando conformada por los tres bloques fundamentales a los cuales les adicionamos uno de control. La inclusión de el bloque de control en esta arquitectura nos permite aplicarla a cualquier problema de dos entradas y una salida, La relación entre ellos se muestra en la Fig. 3.9

Como se mencionó con anterioridad la Arquitectura 2 esta compuesta por cuatro Bloques fundamentales: Bloque Fuzzificador, Bloque de Composición (o simplemente MAX-MIN), Bloque Defuzzificador y por el Bloque de Control. En la Figura 3.9 se puede ver la distribución de los cuatro bloques y el número de bits que comparten entre ellos. Basándonos en la figura y en los nombres de los bloques podemos suponer su funcionamiento, el Bloque Fuzzificador es quien recibe las entradas ya convertidas a un código de 8 bits y se encarga de codificar la membresía de los valores de entrada en un formato de 4 bits, los valores de membresía se transmiten por parejas de 4 bits (8 en total) hacia el Bloque MAX-MIN junto con 3 bits de control, en este segundo bloque

se realiza la inferencia de cada una de las acciones correspondientes a la información de entrada, recordemos que en el problema de la luz de tráfico se tienen 3 fuzzy sets de salida, sin embargo, como ésta ya es una arquitectura de propósito general ampliamos el número de Fuzzy Sets de salida a 5, el valor de cada acción de salida, que va en un formato de 4 bits, se transmite al Bloque Defuzzificador, quien "promedia" la acción resultante de los 5 fuzzy sets de salida. El funcionamiento de cada uno de los bloques se encuentra gobernado por el Bloque de Control, quien regula de un modo secuencial no sólo la transmisión de datos entre bloques sino también el comienzo de la conversión Analógica a Digital.

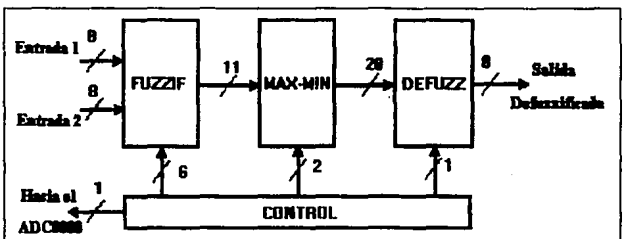


Fig. 3.9 Diagrama a bloques del Procesador Inferencial Restringido.

En la figura 3.9 se aprecia una clara reducción de 5 bits transmitidos entre los tres bloques fundamentales, no obstante la presencia del bloque de control lo hace ver más complejo, si se compara con la estructura mostrada en la figura 3.1. En las secciones siguientes se describe cada uno de los bloques.

3.3.1.1 Bloque Fuzzificador.

Para fuzzificar las entradas digitales se siguió el mismo criterio que el descrito en la sección 3.2.1, esto es, el Universo de cada variable de entrada se encuentra codificado por 256 valores diferentes mientras que su grado de pertenencia proyectado puede tomar cualquiera de los 16 valores diferentes que le es posible adquirir, lo que en otras palabras quiere decir que cada muestra

del Universo de entrada correspondiente convertida a un código de 8 bits proporciona un dato de 4 bits representativo de la pertenencia. Recordemos que en el bloque fuzzificador de la sección 3.2.1 se tiene un circuito específico para cada fuzzy set del universo de la variable, como lo muestra la Fig. 3.3, representando de un modo independiente y simultáneo un grado de pertenencia, sin embargo, todos los circuitos responden a la misma entrada, por ésta razón decidimos agrupar en un mismo circuito a todos los fuzzy sets del Universo de cada variable de entrada. De tal modo que se requieren sólo dos circuitos para fuzzificar ambas entradas, lo cual nos indica que tenemos un circuito fuzzificador por entrada en vez de 6, como en el caso anterior, ya que como mencionamos con anterioridad empleamos un circuito por cada fuzzy set de las entradas y como tenemos tres fuzzy sets en cada una de las dos entradas requerimos de 6 circuitos para fuzzificar los dos universos. De igual manera si las condiciones de un problema de control nos indican que debemos dividir en 7 fuzzy sets cada una da

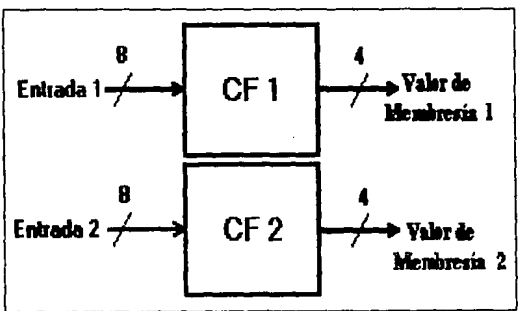
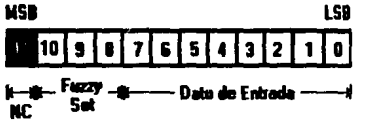


Fig. 3.10 Circuitos Fuzzificadores Simplificados.

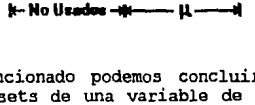
las dos entradas tendríamos que emplear 14 circuitos. Los dos circuitos fuzzificadores se muestran en la Figura 3.10

El Bloque Fuzzificador recibe las señales de entrada provenientes de dos ADC0800, cada uno de los cuales proporciona una palabra cuya extensión es de 1 byte, a la salida del Bloque Fuzzificador se obtienen dos valores de membresía simultáneos, dichos valores de pertenencia son la proyección de las entradas en cada uno de los Fuzzy Sets que componen a sus Universos respectivos, para ello se accesa de manera secuencial a cada Fuzzy Set dentro del circuito fuzzificador que le corresponda, como se verá en posteriormente.

Para poder explicar como se logró tal reducción analizaremos al Circuito Fuzzificador 1 (CF1). En éste circuito en vez de usar uno de 8 entradas, como en la arquitectura anterior, empleamos uno de 12 entradas, esto se hace con el fin que 8 bits representen el dato de entrada muestreado y digitalizado, mientras que 3 de los 4 bits restantes indiquen el fuzzy set por analizar de la variable de entrada correspondiente, por tanto la palabra de accionamiento del circuito CF1 se compone de la siguiente manera, los 8 bits menos significativos indican el valor del universo de entrada, los 4 bits más significativos indican que fuzzy set se evalúa, pudiéndose apreciar dicha palabra de entrada al CF1 en la siguiente forma.



La salida es un dato de 4 bits que representa el grado de pertenencia del fuzzy set seleccionado por la entrada, y aunque el circuito CF1 entrega una palabra de 8 bits sólo aprovechamos cuatro bits, que son los 4 bits menos significativos representando el grado de membresía de la entrada. Esquematizados en la ilustración siguiente.



De lo antes mencionado podemos concluir que, los rangos de acción de los fuzzy sets de una variable de entrada se encuentran contenidos en el circuito fuzzificador, pero debido a que los fuzzy sets deben ser accesados de manera independiente se cuenta con los 3 pines de selección de Fuzzy Set que van a multiplexar al Fuzzy Set por evaluar, pudiendo multiplexar de ésta manera hasta 7 fuzzy sets distintos por variable de entrada, de los cuales el primer

fuzzy set se elige con el dato 1H (0001) en los 4 bits más significativos de la entrada, el segundo fuzzy set con el dato 2H (0010) y así sucesivamente hasta llegar al fuzzy set 7 dato 7H (0111), el código de solección 0H (0000) está reservado para indicarle al circuito el momento en que una acción es nula, es decir, que regla de la matriz de reglas es imposible de ocurrir por tanto no es activada. La manera de almacenar las funciones de membresía se puede ejemplificar en la Figura 3.11, la cual también muestra de modo esquemático un Circuito Fuzzificador general.

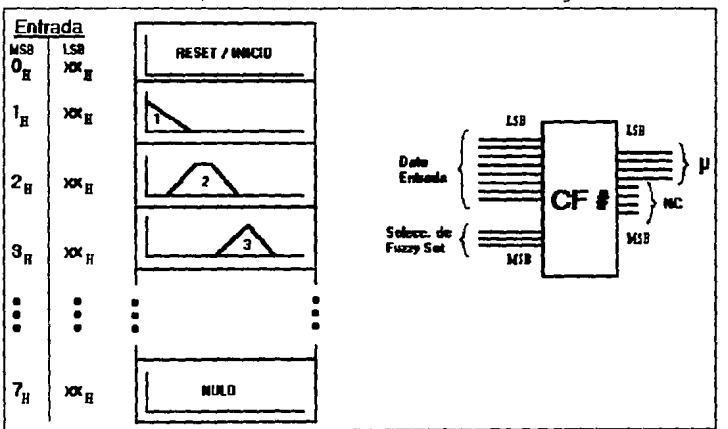


Fig. 3.11 Representación gráfica del almacenamiento de fuzzy sets.

La Figura 3.11 en su extremo izquierdo muestra una tabla que representa la palabra de entrada de 12 bits, la cual, por simplicidad, se ha codificado en formato Hexadecimal (por esta razón se muestran tres valores uno numérico y dos "XX", el dato numérico indica el Fuzzy Set que se selecciona con dicho valor (es la palabra de multiplexaje) y las "XX" indican los 8 bits aleatorios del dato de entrada, así por ejemplo con el dato "1H-XXH" se selecciona al Fuzzy Set 1 contenido dentro del Circuito Fuzzificador, con ello se le habilita para que evalúe la entrada y pueda entregar un grado de membresía respectivo al Fuzzy Set que representa. En el extremo derecho de la Figura 3.11 se puede ver a

un Circuito Fuzzificador generalizado que es similar a los de la Figura 3.10 sólo que con sus buses de datos expandidos.

Aunado a los dos CF's de las entradas se tiene un circuito que contiene la matriz de reglas de comportamiento, es éste circuito el que se encarga de direccionar el par de grados de pertenencia provenientes de los CF's hacia su Consecuencia específica, la matriz de reglas se encuentra programada tratando de imitar la matriz original, para ello empleamos un GAL que entrega una dirección para habilitar una salida según sea el dato de 6 bits que le llegue a sus entradas, por ejemplo, supongamos que la primera variable de entrada se divide en 4 fuzzy sets y la segunda en 3 fuzzy sets, denominados 1A, 1B, 1C, 1D y 2A, 2B, 2C respectivamente, la matriz de reglas es la mostrada en la figura 3.12.a y la variable de salida consta de 4 fuzzy sets.

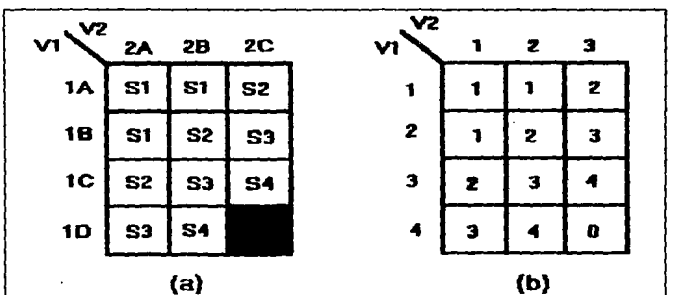


Fig. 3.12 Matrices de Reglas, (a) Matriz original, (b) Matriz codificada para el circuito de reglas fuzzy.

El número total de reglas es: NR = 4 x 3 = 12, las cuales tienen acciones independientes al implementarse en el circuito, lo cual es "contrario" a la teoría donde se activan simultáneamente. Decimos que las acciones son independientes debido a que las reglas se accesan secuenciálmente.

Para poder programar la matriz de reglas es necesario transformar la matriz (a) en la matriz (b), la matriz (a) contiene las etiquetas de los fuzzy sets de entrada y salida, tales etiquetas deben cambiarse a su número de fuzzy set equivalente, como se aprecia en la figura 3.11, donde se le asigna el número 1

al fuzzy set que cubre el rango inferior del Universo, desplazándonos hacia el valor máximo le asignamos el número 2 al siguiente fuzzy set, continuamos de ésta manera hasta cubrir todos los conjuntos de datos, así para el fuzzy set 1A, se asigna al valor $1_d = 1_H = (0001)_2$, labor que se hace a todos los fuzzy sets hasta obtener la matriz (b). Esta matriz contiene toda la información que se le programará al GAL. Se puede observar que existe una regla cuya consecuencia es nula, al transformarla se asigna un 0H, lo cual no direcciona ninguna salida. De la matriz (b) se pasan todos sus datos hacia la tabla de verdad del circuito de Matriz de Reglas "MR", por ejemplo:

V1	V2	SAL
1H	1H	1H
:	:	:
3H	2H	3H
:	:	:
4H	4H	0H

Es importante señalar que aunque el circuito MR accesa secuenciálmente cada una de las reglas de inferencia, estas se encuentran sincronizadas con los Fuzzy Sets que podrían activarlas, las cuales provienen de los circuitos CF1 y CF2, debido a que las entradas de selección de los mismos se encuentran alimentadas en paralelo como se muestra en la Fig. 3.13

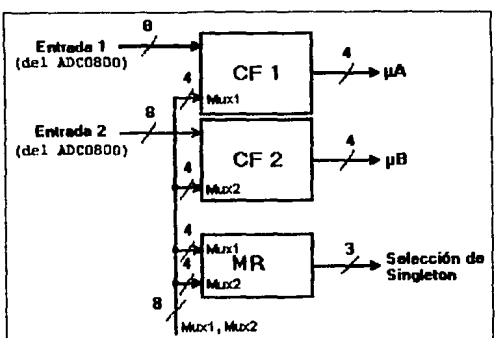


Fig. 3.13 Circuitos Fuzzificadores accesados en paralelo.

3.3.1.2 Bloque Composicional.

Como se indicó en la sección anterior, la integración de todos los fuzzy sets de una variable en un sólo circuito y su posterior acceso en serie logran que sólo sea procesado un par Antecedente a la vez, lo cual nos lleva a una reducción del Bloque de Mínimos, el número de circuitos que compone a este se reduce (de 9, uno para cada par antecedente de la arquitectura anterior) a un sólo circuito de dos entradas, por las razones antes descritas. En éste caso el diseño del circuito comparador que selecciona el mínimo valor de los antecedentes que le llegan es exactamente el mismo de la arquitectura anterior.

El resultado obtenido por el circuito MIN (circuito de selección de Mínimos) se alimenta al bloque de máximos, que esta formado por 5 circuitos MAX que comparan la magnitud de sus dos entradas, teniendo la particularidad de poder habilitarse para la entrada de sus datos, característica que los hace selectivos, el diseño de los circuitos MAX logra retener la información anterior hasta que éstos sean reinicializados. En la Figura 3.14 se muestra el Bloque Composicional.

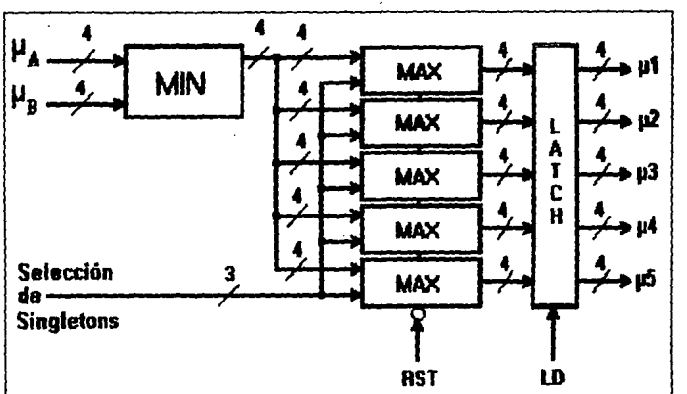


Fig. 3.14 Circuitos constituyentes del Bloque Inferencial.

En la Figura 3.14 observamos que a los circuitos MAX solamente le llega una línea de datos del grado de membresía, la marcada como bus de 4 bits, esto se debe a que internamente se realimenta el valor más grande, por lo cual se debe inicializar a los circuitos cada vez que entran nuevos datos, esta operación se hace mediante la entrada de reset (RST). Las salidas de los máximos pasan a un LATCH, o circuito retenedor de datos, para asegurar sincronía en la transmisión de resultados, el control de la carga de datos en el LATCH se efectúa por medio de la entrada LD.

Los grados de membresía mostrados en la Figura 3.14 tienen el siguiente significado:

μA - Membresía proveniente de CF1
μB - Membresía proveniente de CF2
μ1 - Grado de Membresía con que se activa el Singleton 1
μ2 - Grado de Membresía con que se activa el Singleton 2
μ3 - Grado de Membresía con que se activa el Singleton 3
μ4 - Grado de Membresía con que se activa el Singleton 4
μ5 - Grado de Membresía con que se activa el Singleton 5

El objetivo de contar con 5 circuitos máximos reside en que empleamos 5 Singletons que cubren al Universo de salida, de modo que un máximo está destinado para darle información únicamente a su Singleton asociado. La decisión de emplear 5 Singletons a la salida se fundamenta en que con 5 fuzzy sets triangulares traslapados 50% entre sí podemos cubrir de manera completa al Universo de salida sin perder información, ver Fig. 3.15.

En ella podemos ver que el centroide del primero de los fuzzy sets no se encuentra en el 0H, se decidió la exclusión de éste fuzzy set "Nulo" ya que al aplicar el Singleton 1 (S1) éste varía desde 0 hasta 20%.

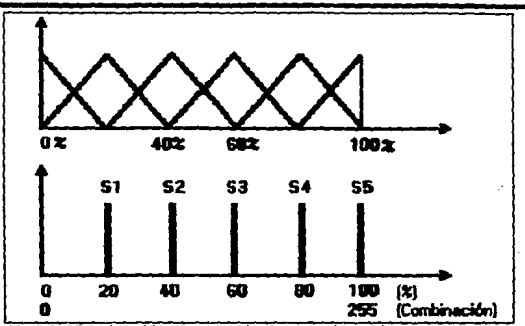


Fig. 3.15 Ubicación de los Singletons de Salida.

3.3.1.3 Bloque Defuzzificador.

La gran complejidad del bloque defuzzificador descrito en la sección 3.2.3 nos obligó a proponer otras opciones en el diseño del mismo. La primera parte de la defuzzificación que tratamos de cambiar fue la manera de resolver la ecuación de pesos_promediados, la ecuación original desarrollada para 5 singletons se muestra a continuación.

$$\text{Salida} = \frac{\mu_1 S_1 + \mu_2 S_2 + \mu_3 S_3 + \mu_4 S_4 + \mu_5 S_5}{\mu_1 + \mu_2 + \mu_3 + \mu_4 + \mu_5} \qquad (2)$$

donde:

μ1 - Grado de membresía de activación del Singleton 1 (S1)
μ2 - Grado de membresía de activación del Singleton 1 (S2)
μ3 - Grado de membresía de activación del Singleton 1 (S3)

$\mu 4$ - Grado de membresía de activación del Singleton 1 (S4)
$\mu 5$ - Grado de membresía de activación del Singleton 1 (S5)

que se puede reescribir de la siguiente manera.

$$\text{Salida} = \frac{\mu_1 S_1}{\sum \mu_i} + \frac{\mu_2 S_2}{\sum \mu_i} + \frac{\mu_3 S_3}{\sum \mu_i} + \frac{\mu_4 S_4}{\sum \mu_i} + \frac{\mu_5 S_5}{\sum \mu_i} \quad (3)$$

donde:

$\mu 1, \ldots, \mu 5$ - Grados de membresía de su respectivo Singleton
$S1, \ldots, S5$ - Valor del Singleton de Salida
$\Sigma \mu i$ - Sumatoria de los Grados de membresía consecuentes.

La ecuación (3) parece ser más compleja que la ecuación (2) ya que implica 5 divisiones independientes en vez de una general, sin embargo, esta permite visualizarla de manera diferente lo cual logra simplificar la operación de defuzzificación, ya que con ella se aprecia que se requieren cinco componentes de circuito que realicen parte a parte la defuzzificación. En la figura 3.16 se muestra la implementación de la ecuación (3) en hardware. En ella vemos que a cada circuito "MD" (Multiplicador Divisor) le llega el resultado de la suma de los grados de pertenencia, la cual en el más extremo de los casos nunca excede de 6 bits, los que al juntarse con los 4 bits del grado de membresía respectivo forman un código de 10 bits, los posibles valores tomados por el código de 10 bits pueden ser estimados y por tanto simulados y programados.

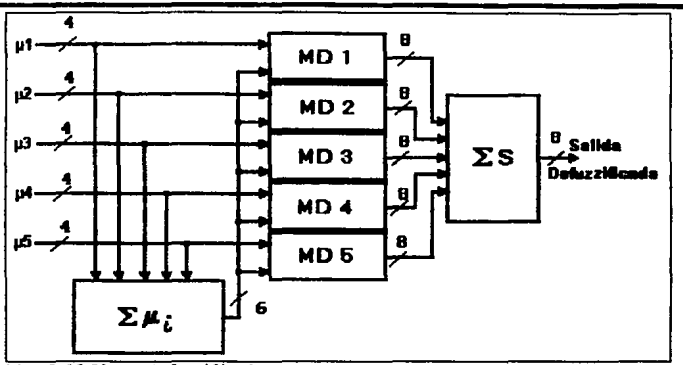


Fig. 3.16 Bloque Defuzzificador.

Por ejemplo, veamos un valor de entrada al circuito MD1, el circuito MD1 es quien representa al Singleton 1, por lo que el máximo valor que puede tomar es 20% de la salida total cuando recibe un μ1 = 1. Supongamos que la composición realizada por el circuito de la Figura 3.14, nos indica que sólo se activa dicho singleton (S1), de modo que el resultado de Σμi = μ1, en una proporción AH = 10d ≈ 0.67, con ése valor la salida debe ser: SAL = (0.67)(20%) = 13.4 % ≈ 34d = 22H. En el caso que se activen dos o más singletons se tiene que aplicar la ecuación (3).

Ahora suponiendo que además de S1 se activa S2 con μ2 = 0.3 que equivaldría a tener μ2 = 5d = 5H, tendremos:

SAL = (0.67x20%)/0.97 + (0.3x40%)/0.97 = 26.18% o 43H (a)

o bien

SAL = (AH)(33H)/FH + (5H)(66H)/FH = 22H + 22H = 44H (b)
= (10d)(51d)/15d + (5d)(102d)/15d = 68d = 44H (c)

La ecuación (a) es la forma en que se llevaría a cabo de manera ordinaria, sin embargo la simulación de la misma emplea los valores mostrados en las ecuaciones (b) y (c), lo cual equivaldría

a trabajar con los datos digitalizados. La forma física en que entrarían los datos a los circuitos antes mencionados se muestra en la figura 3.17. En donde se ven tres valores entrando a dos de los circuitos que conforman al bloque defuzzificador, dichos valores son:

$$\mu 1 = AH$$
$$\mu 2 = 5H$$
$$\Sigma\mu_i = 0FH$$

se puede ver que al circuito MD1 le llegan los datos "AH" y "0FH", entregando a su salida el dato "22H", el cual es el resultado de la multiplicación y división del primer miembro de la ecuación (b), este resultado se obtiene gracias a que todas las posibles combinaciones de datos de entrada a defuzzificar para el Singleton 1 han sido simulados por computadora y grabados en el circuito MD1, de manera semejante se simulan los casos posibles para los restantes 4 singletons.

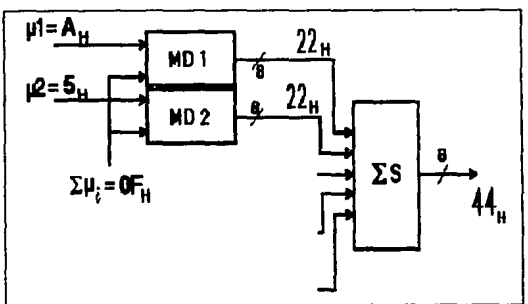


Fig. 3.17 Ejemplo de la defuzzificación de dos datos de entrada.

En éste bloque defuzzificador la parte más voluminosa es la de los sumadores, que en el caso de $\Sigma\mu i$ se emplean 4 sumadores completos de 4 bits como mínimo y en ΣS al menos 4 sumadores de 8 bits.

3.3.1.4 Bloque de Control.

El último de los bloques es el de control, es éste quien nos va a regular el flujo de información y el procesamiento de la misma. Este se compone por dos circuitos uno es un circuito secuenciador que entrega las salidas de carga y lectura de valores sincronizando todo el proceso, por ello le denominamos CONTROL, la salida del circuito de control es de 8 bits, 6 para la selección de fuzzy set y 2 de sincronía. El otro circuito (denominado RESET) sirve para restablecer al bloque composicional, en la figura 3.18 se puede observar que el circuito de reseteo depende del estado del circuito de control, esto se debe a que el diseño original del circuito de control contenía internamente al circuito de reseteo, sin embargo, la implementación de este excedía la capacidad de almacenamiento de compuertas de los GAL's 16V8 lo que nos obligó a crear un circuito externo de reseteo. El circuito de control es un contador o secuenciador que tiene la capacidad de cambiar de estado dependiendo de las condiciones de sus entradas, de modo que podemos decirle que cuente, se mantenga o se reinicialize con las entradas Cuenta o Espera (C/$\overline{H}$), y Reset (RST) respectivamente.

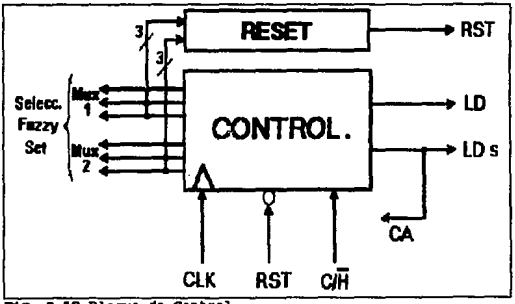


Fig. 3.18 Bloque de Control.

La descripción del funcionamiento del bloque de control mostrado en la figura 3.18 es la siguiente. El circuito de control es un contador que requiere para su funcionamiento de tres entradas que gobiernan la secuencia del conteo, la primera de las entradas es la etiquetada con las letras CLK, esta es la entrada del pulso de reloj que marca la frecuencia a la que va a trabajar el contador, el contador requiere de aproximadamente 70 pulsos de reloj para realizar un ciclo de control completo, esto es en el caso de que los circuitos fuzzificadores contuvieran 14 fuzzy sets en total. La siguiente entrada de control es la marcada como RST, que se encarga de restablecer a las condiciones preestablecidas para iniciar el control del proceso inferencial, como se ve en la figura 3.18 la entrada esta prevista para activarse en niveles lógicos bajos. La tercera entrada de control es la que recibe la etiqueta C/H, la cual permite que el controlador continúe con su cuenta normal o bien que permanezca en un estado de espera hasta que se le indique que puede seguir con el control del proceso, la función de espera fue ideada para acoplar a dos o más circuitos de control juntos, ya que en aplicaciones de más de dos entradas se requeriría al menos un Procesador inferencial adicional y el control de todo el proceso estaría distribuido entre los circuitos de control de cada procesador inferencial.

Las salidas del circuito de control son 8 como se mencionó con anterioridad, 6 de estas tienen el propósito de multiplexar tanto a los circuitos CF como al circuito de matriz de reglas MR, con ello se logra controlar la secuencia de las reglas de entrada, estas mismas salidas resetean a los circuitos de máximos que se encuentran dentro del bloque inferencial, el reseteo se lleva a cabo mediante el circuito de RESET externo, el cual está conectado en paralelo a las salidas MUX1 y MUX2. La séptima salida del circuito de control es la marcada como LD, que tiene la función de indicarle al LATCH de salida del bloque inferencial el momento en que debe cargar los datos que le llegan a sus entradas. La última salida (LDs) tiene una doble función, habilita la carga de datos en el LATCH de salida y al mismo tiempo le indica a los Convertidores Analógicos-Digitales de entrada el momento en que deben tomar una muestra de la señal analógica de entrada.

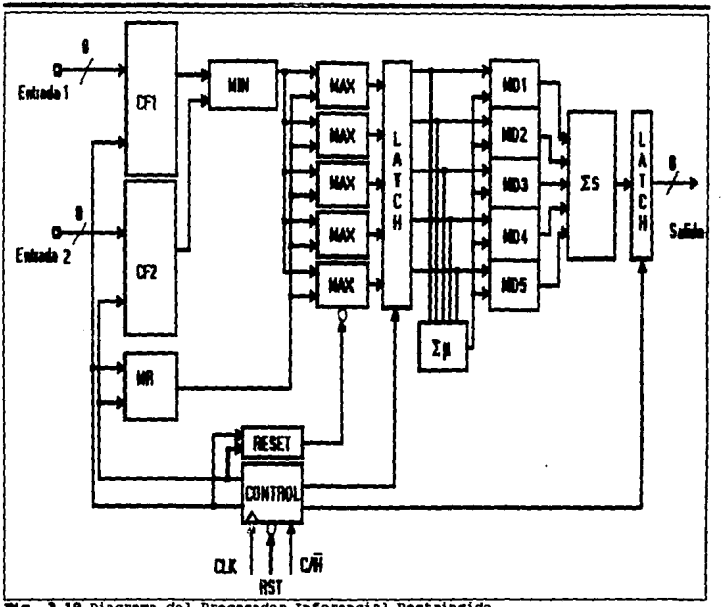


Fig. 3.19 Diagrama del Procesador Inferencial Restringido.

El circuito completo de la Fig. 3.19 requiere de 28 CI's de los cuales 21 son GAL's y 7 EPROM's, a pesar que sigue siendo muy voluminoso es ya un controlador fuzzy Logic de propósito general, al cual para modificarle sus respuestas y aplicarlo a otro problema sólo debe cambiársele el Bloque Fuzzificador, es decir sólo 3 CI's, no obstante la sencillez de su programación es un procesador limitado, ya que solamente admite hasta 7 fuzzy sets por variable de entrada y 5 para la salida; los fuzzy sets de entrada pueden extenderse, pero para ello requerimos que el Bloque de Control nos entregue una palabra de control de mayor extensión, por otro lado el diseño presentado lee todos los fuzzy sets secuenciálmente, lo cual en aplicaciones con menos de 7 fuzzy sets en cualquiera de las entradas implica una pérdida de tiempo, para resolver ése problema se puede cambiar al circuito de control que es un contador con secuencia fija por un contador "programable" en el Bloque de

Control, el cual se pueda programar por medio de Switches y así seleccionar el momento de llamar los fuzzy sets empleados y brincar los fuzzy sets sin uso. Otro defecto que puede mejorarse en éste diseño es la palabra de salida de los Circuitos de Fuzzificación o (CF's) ya que sólo se emplea 4 de los 8 bits disponibles, una opción de mejora podría ser aumentar el tamaño del código del grado de membresía o bien insertar simultáneamente dos grados de pertenencia de 4 bits y con ello indicar el traslapamiento entre funciones de pertenencia y procesar simultáneamente 4 valores. La mayor limitante en éste circuito sigue siendo la defuzzificación, ya que se emplean 5 Singletons fijos, por ello nombramos a éste circuito FLC-PG Restringido, razón por la que todo el proceso de inferencia ha de calcularse en base a éste Universo de salida limitado. Suponiendo que la salida de 5 Singletons se ajusta a nuestros propósitos, pero requerimos que el Singleton 1 (S1) en vez de estar al 20% del Universo de salida deba estar al 10%, la manera más sencilla de resolver parcialmente éste dilema es emplear un circuito que en vez de contar con 10 entradas posea 12, lo cual nos da la holgura de contar con 4 valores diferentes en la simulación de cada MD, y por medio de switches elegir el que mejor se ajuste.

En la sección siguiente se describe una estructura que resuelve la mayor parte de los inconvenientes de la descrita en la presente sección.

3.3.2 Controlador Fuzzy Logic de Propósito General Ajustable. (Arquitectura 3)

Las limitaciones en el Universo de salida de la arquitectura mostrada en la Figura 3.19 nos hicieron pensar en un medio más simple y eficaz para implementar un procesador inferencial, recordemos que los circuitos 'MD' de dicha arquitectura deben ser simulados para implementarse y que dicha simulación se basa en los datos de entrada esperados, basándonos en éste planteamiento nos surgió la interrogante ¿Se podrá simular toda la Defuzzificación? La respuesta seguramente es un si, pero no hay que perder de vista que las entradas para el Bloque de Defuzzificación son 5 y que cada una de ellas esta codificada en 4 bits, lo que representa un total de 20 bits que nos llevan a tener un total de 1,048,576

combinaciones diferentes, de llevarse a cabo tal simulación resultaría un reducción de componentes sustancial, la arquitectura completa podría reducirse de 28 a 15 CI's, pero seguiría presente el inconveniente de emplear 5 Singletons de salida fijos, sin embargo existiría la flexibilidad de escoger el valor exacto de cada Singleton, pero, si llegamos hasta ése grado de ajuste a la salida ¿Por que no tratar de simular todo el proceso de inferencia? La pregunta era forzosa ya que para desarrollar los CF's y MD's requeríamos de programar un Software para tal fin y si ése software era capaz de fuzzificar y aunque parcialmente defuzzificar ¿Por que no incorporarle la composición de máximos y mínimos para que se simulara por completo un problema específico?

Como resultado a esas inquietudes surgió un programa que realiza todo el proceso de inferencia, de una manera muy cercana a la teórica, recordemos que en la teoría las 2 entradas se fuzzifican y se obtiene un máximo de 4 grados de pertenencia diferentes de cero simultáneamente, éstos se aplican a las reglas de funcionamiento y se infieren hasta 4 acciones, de ellas se obtiene una consecuencia final por medio de la defuzzificación. La intención fue que el programa trabajara de esta manera, pero debido a nuestros escasos conocimientos sobre lenguajes de programación sólo pudimos elaborar un proceso secuencial, siendo que se requiere de un proceso en paralelo para inferir resultados, el proceso secuencial resultante se describe a continuación.

Todos los datos del problema de control aplicados al programa de Inferencia Fuzzy pueden ser ajustados a los valores deseados, es decir, las dimensiones de los Fuzzy Sets de los dos Universos de Entrada, la matriz de reglas y el valor de los Singletons pueden adaptarse a las condiciones reales del problema de control. La Lógica del programa es la siguiente: como primer paso se introducen uno a uno los valores de los Universos de entrada y sus respectivos Fuzzy Sets, una vez que se tienen los datos completos de los dos universos, se requiere introducir la combinación de las reglas de control Fuzzy o matriz de reglas, por último el programa pide la posición de los Singletons y el tamaño del Universo de Salida. Tanto los Universos de Entrada como el de Salida son convertidos a una escala decimal de 64 combinaciones y 256 combinaciones respectivamente, lo cual equivale a tener codificados los Universos de Entrada en 6 bits y el Universo de Salida en 8 bits. El objetivo del programa es combinar cada uno de los valores de entrada y evaluar la salida que resultaría de esta combinación, según lo indique la matriz de reglas. Por tanto el programa al combinar uno a uno los posibles valores de entrada realiza un total de 4096

cálculos de acciones de salida, evaluando todo el proceso inferencial en cada una de las combinaciones simuladas. Como parte adicional a la simulación se implementó una sección lógica dentro del programa que genera los códigos necesarios para que los datos simulados se guarden en un archivo y por medio de este se pueda programar un Circuito Integrado mediante un Programador de Circuitos Digitales.

Debido a la simulación todo el proceso de inferencia se logra integrar en un sólo CHIP, lo cual reduce drásticamente el número de componentes requerido por cualquiera de las dos arquitecturas anteriores, por tal motivo proponemos una circuitería completa, que muestra desde la entrada de datos analógicos hasta la salida de valores digitales, tal arquitectura se divide en 4 bloques.

a. Adquisición de Datos (ADQ)
b. Bloque Inferencial (BI)
c. Entrega de Resultados (ENR)
d. Administrador del Proceso (ADMP)

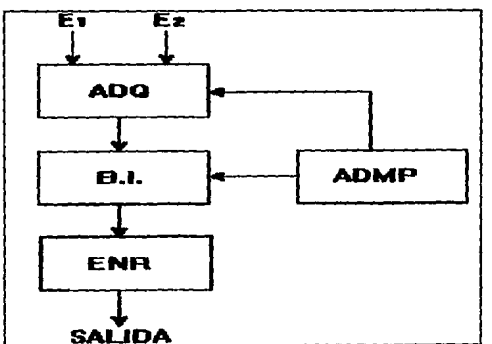


Fig. 3.20 Bloques funcionales del Controlador Ajustable.

a. Adquisición de Datos.

El primero de los bloques se muestra en la Figura 3.21 y es quien traduce la información Analógica en Digital, por medio de 2 ADC's, éstos entregan 8 bits representativos de la entrada analógica, sin embargo por las características del diseño aquí descrito sólo se emplean datos de **6 bits**, por ésta razón empleamos un circuito que reduce la longitud de palabra del convertidor a la vez que sincroniza las entradas, es por ello que denominamos a dicho circuito como Circuito de Retardo y Sincronía "R&S". Para su correcto funcionamiento se requiere de terminales de control, la señal de Control 1 (CTRL 1) se alimenta en paralelo a los dos convertidores les indica el momento en que deben comenzar la conversión de las entradas analógicas, mientras que la señal CTRL 2 logra que los circuitos "R&S" carguen los datos de entrada y los muestren a sus salidas.

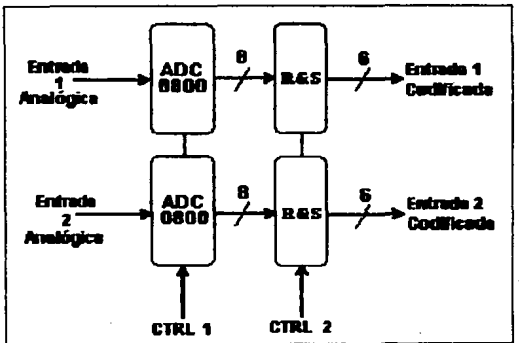


Fig. 3.21 Bloque de Adquisición de Datos.

b. Bloque Inferencial.

El segundo Bloque es el que contiene al proceso inferencial simulado y compactado, se habrá notado que las entradas muestreadas llegan en un código de 6 bits para cada una de ellas, lo cual difiere de las arquitecturas anteriores, ello se debe a que durante el desarrollo del Software se tuvo que emplear el menor número de combinaciones posibles para poder afinar detalles en la programación, encontrando que un código de 6 bits puede ser lo suficientemente corto como para ser analizado rápidamente y lo suficientemente largo como para representar fuzzy sets con resolución media. Para compensar un poco la burda resolución empleada se utilizaron 8 bits a la salida. Figura 3.22.

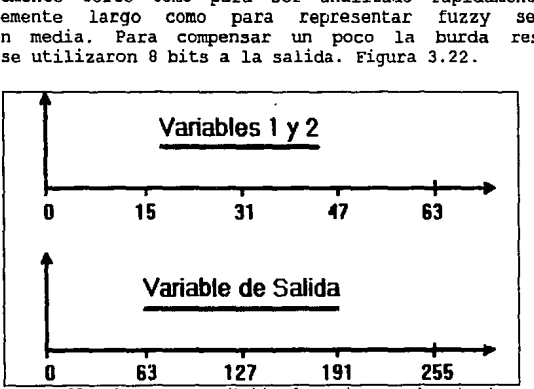


Fig. 3.22 Modo en que se dividen los universos de entrada y salida para la tercera arquitectura.

Como ya mencionamos, se introducen al programa los datos detallados del problema y éste se encarga de combinarlos e inferir un resultado para cualquier situación. El Software genera toda la información necesaria para poder programar al bloque inferencial, debido a que la función de éste bloque es proyectar una salida según sea la combinación de entrada decidimos nombrarlo Centro de Proyecciones Fuzzy "CPF". Como se muestra en la Figura 3.23. En donde además del circuito CPF, se muestra un Cerrojo o LATCH, que cuenta con una entrada RST, que sirve para inicializar las salidas así como una entrada de Control (CTRL), dicha entrada le indica en

que momento debe accesar los datos resultantes del Circuito de Proyecciones Fuzzy (CPF). Tanto las señales RST como CTRL provienen del circuito Administrador del Proceso (ADMP).

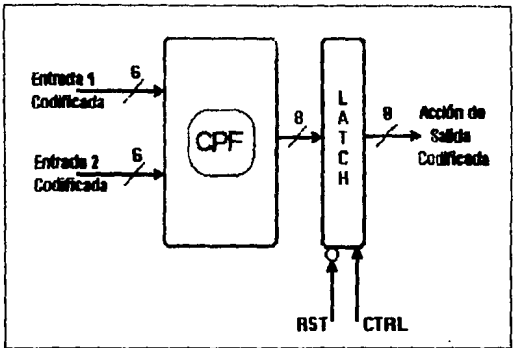


Fig. 3.23 Bloque Inferencial.

c. Entrega de Resultados.

El resultado inferido debe ser transformado para su utilización en circuitos analógicos convencionales o bien dejarlo como un dato defuzzificado de 8 bits, para ello el Bloque de Entrega de Resultados dispone de dos salidas, la primera empleada para entregar datos analógicos y la segunda para mostrar el mismo valor digitalizado.

d. Administrador del Proceso.

El Administrador del Proceso (ADMP) como lo indica su nombre administra o regula el funcionamiento de todo el circuito, éste no es más que un secuenciador que indica la carga de datos y su posterior salida, a la vez que inicializa a los circuitos que requieran de ésta operación. El ADMP fue programado en un GAL16V8 lo cual nos permite integrar en un sólo CHIP a un contador programable junto con sus multiplexores de función. En caso de no emplearse un sólo GAL como ADMP, este puede ser implementado mediante un simple controlador como lo es el 74LS161, que es un contador programable, sin embargo para poder lograr que este funcione con la secuencia que requerimos es necesario añadirle al menos tres circuitos integrados más (como se puede encontrar en cualquier configuración básica del 74LS161 como controlador). El diagrama esquemático del GAL16V8 Administrador del Proceso se muestra en la Figura 3.24.

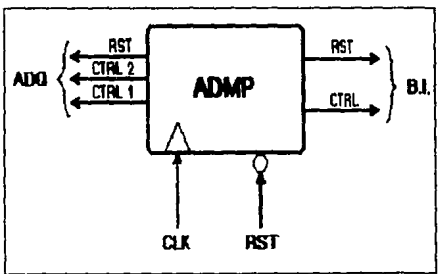


Fig. 3.24 Diagrama del Administrador del Proceso Inferencial.

El funcionamiento del ADMP es el siguiente. Los pulsos de reloj que se le alimentan mediante la entrada CLK hacen que los FLIP-FLOP'S internos del ADMP generen una secuencia predeterminada, que indica como primera función el inicio de la conversión en el bloque ADQ mediante la salida CTRL 1, posterior a la conversión se tiene una salida CTRL 2 que activa la carga de los datos

convertidos en los circuitos R&S, cuando ya se cargaron los datos convertidos se tiene un período de espera para que el circuito CPF direccione su salida y por último la tercera salida (CTRL) hace que el LATCH de salida cargue el resultado de la inferencia para que con ello se asegure el mantenimiento de la misma durante los ciclos de carga anteriores. Como se ve la función del control es muy simple y requiere de sólo 4 ciclos de reloj.

En la Figura 3.24 se pueden ver dos salidas adicionales a las mencionadas en el párrafo anterior, las cuales se han etiquetado como RST, las cuales no son salidas reales del controlador hacia los circuitos sino sólo un puente entre la entrada de RESETEO del ADMP y las entradas de RESET de los cerrojos de aseguramiento de información (LATCH's).

El circuito completo para el controlador de propósito general ajustable no requiere de tantos componentes como las arquitecturas anteriores, ésta reducción en componentes lo hace costeable para ser implementado tanto en aplicaciones didácticas, como en controles industriales, lo cual dependerá de la calidad de los conocimientos que se tengan sobre Fuzzy Logic, así como de la habilidad con que se cuente para visualizar la solución de los problemas. Como ya se mencionó ésta arquitectura sirve de propósito general, sólo que para poder ser programada para otra aplicación es indispensable utilizar el Software asociado; la ventaja de éste diseño respecto a los anteriores es el poco espacio empleado, que también se hubiera logrado de poder implementar las arquitecturas anteriores mediante microelectrónica, sin embargo, esto elevaría su costo, el cual no excede de US$ 40.00 para la arquitectura de la Fig. 3.25 (si las partes se compran a nivel de componente). El pequeño inconveniente del circuito es la baja resolución de sus entradas la cual puede mejorarse incrementándola a 8 bits por variable, lo cual no hemos hecho hasta el momento debido a que la inferencia efectuada con las herramientas de que disponemos (una PC XT compatible y el software programado en BASIC), tardaría de 2 hrs. a 2½ hrs., tiempo que es excesivo cuando se trata de afinar detalles en el diseño de fuzzy sets, además el software actual ha mostrado dar buenos resultados en aplicaciones donde las entradas tengan hasta 5 fuzzy sets y su tamaño sea proporcional, así como los rangos de valores que cubran sean equitativos.

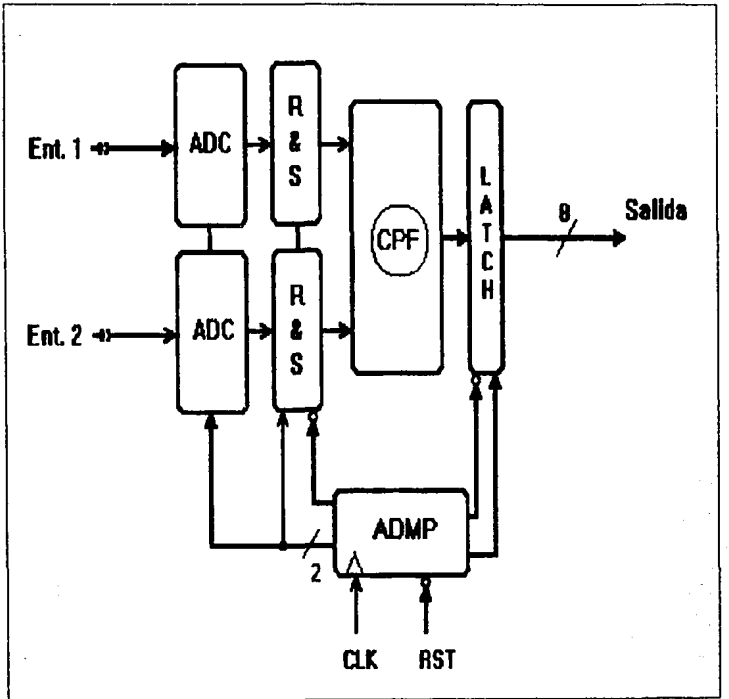


Fig. 3.25 Arquitectura de Propósito General Programable mediante Software.

CONCLUSIONES.

Durante el período de investigación observamos que la Técnica denominada Fuzzy Logic o Lógica Difusa empleada para la solución de problemas de control es aún desconocida por la mayor parte de los profesionales relacionados con los sistemas de control industrial, el escaso número de conocedores del tema son, en su mayoría, investigadores, docentes y personas encargadas de la comercialización de los pocos productos que las grandes compañías comienzan a lanzar al mercado.

La gran ventaja del empleo de esta técnica es que no existe una metodología formal para el diseño y desarrollo de arquitecturas de sistemas de control Fuzzy Logic, ya que la mayor parte de las publicaciones que los investigadores ponen al alcance de los lectores, muestran más frecuentemente ejemplos teóricos que ejemplos de arquitecturas funcionales.

Gracias a la libertad de diseño para las arquitecturas, es como se pudo desarrollar esta tesis, la cual consistió en proponer una estructura lógica funcional para sistemas de control fuzzy logic que contara con dos variables de entrada y una de salida, ya que en el mercado se han comercializado arquitecturas que aplican la lógica difusa para resolver problemas de control, teniendo el inconveniente de que manejan elementos PID que usualmente procesan la información en forma secuencial, desaprovechando con ello una de las mayores ventajas del Proceso Inferencial Fuzzy Logic que es el procesamiento en paralelo, lo que nos motivó para el desarrollo de tales diseños.

Como se mencionó esta técnica de diseño puede ser implementada tanto en Software como en Hardware, o ambas, con ello se logra una gran flexibilidad. La implementación en Software se lleva a cabo mediante un lenguaje de programación de últimas generaciones como

lo sería el lenguaje 'C', para lo cual se requiere un gran conocimiento y manejo de este, además de interfases, tanto gráficas dentro del programa como de control, diseñadas especialmente para tal fin. En vista de que para aplicar el Software se requiere forzosamente un hardware asociado, se decidió implementar un controlador desarrollado en Hardware, el cual se podía desarrollar mediante circuitos analógicos o digitales. Debido a las necesidades de procesamiento en paralelo de la Lógica Difusa y la versatilidad que deseábamos obtener, nos dimos cuenta que de realizarse mediante una arquitectura analógica hubiéramos requerido de un gran numero de componentes, lo cual no es practico ni costeable. Sin embargo, la tecnología digital, a pesar de tener bases teóricas diferentes, puede acoplarse al manejo de la teoría difusa. Por lo cual se decidió hacer el desarrollo de este proyecto mediante la implementación de dispositivos lógicos programables que nos permiten reducir circuiteria, lo cual en el caso de electrónica analógica solamente se lograría mediante microelectrónica.

Durante el diseño y desarrollo de este proyecto nos dimos cuenta de que el empleo de Software especializado en simulación y programación de PLD's era indispensable. Por otra parte también se requería el desarrollo de un Software para la simulación, corrección y ajuste del proceso de decisión requerido por la Lógica Difusa, que aunque en el mercado existen paquetes para cubrir con tales necesidades nuestro acceso a ellos es casi nulo.

Para poder verificar que nuestra propuesta puede ser funcional decidimos implementar la última estructura lógica, empleándola para dar solución al ejemplo propuesto por David I. Brubaker y expuesto en el Capítulo 2. Durante el período de simulación y ajuste del sistema llegamos a la conclusión de que si es posible llevar a cabo la implementación de dicha arquitectura, ya que las muestras de los resultados se aproximan al comportamiento que la matriz de reglas del ejemplo plantea. Aunque no tenemos mas que un valor comparativo que los autores del artículo proporcionan, éste se encuentra muy próximo al obtenido por nuestro controlador, como se puede apreciar en los siguientes resultados:

Resultado Proporcionado por los autores	Resultado obtenido por el FLC-PG Ajustable
t_{ENOJA} = 5.96 seg.	t_{ENOJA} = 5.90 seg.

La ligera diferencia que se aprecia en los valores mostrados, se debe a que la resolución empleada por el muestreo es media, como se mencionó en el Capítulo 3. A modo de comprobación de que la estructura lógica es funcional, se cita a continuación la matriz de reglas para el comportamiento de la luz roja junto con un muestreo aleatorio de datos de entrada, con los resultados obtenidos por el Controlador Fuzzy Logic de Propósito General.

LUZ_ROJA

Densidad \ Velocidad	LENTA	MEDIO RAP.	RAPIDA
LIGERA	7 (Corta)	8 (Corta)	9 (Corta)
MEDIO PESADA	4 Media Larga	5 Corta	6 (Corta)
PESADA	1 Larga	2 Media Larga	3 Corta

Matriz de Reglas para el comportamiento del encendido de la luz roja.

Densidad autos/esp.	Velocidad MPH	Salida seg.
0.05	10	1.97
0.17	30	1.71
0.25	60	2.12
0.31	19	4.14
0.43	25	4.87
0.5	50	3.31
0.03	4	2.74
0.3	30	2.64
0.7	60	2.74
0.9	3	13.19
0.6	10	11.75
0.47	12	9.62
0.35	**17**	**5.90**

En esta tabla se indican tres columnas, la primera de ellas, etiquetada como 'Densidad autos/esp', y la segunda, cuya etiqueta es 'Velocidad MPH' representan valores aleatorios de los Universos de entrada, la tercer columna es el tiempo de encendido de la Luz Roja del semáforo de acceso a la vía rápida citado con anterioridad y que representa la Salida del controlador. Los valores de entrada aplicados al controlador hacen que este genere una salida cuyo valor es un código binario de 8 bits que interpreta un circuito contador y con el genera el tiempo de encendido deseado, que es el valor real mostrado en la tercer columna, así para las primeras entradas (Densidad = 0.05 autos/esp y Velocidad = 10 MPH)que se le aplicaron al controlador, este nos entregó un tiempo de encendido de luz roja de 1.97 segundos. De manera semejante se aplicaron los siguientes 11 valores listados que fueron aleatorios, el último valor era forzoso introducir al controlador ya que este es nuestra referencia de comparación con los cálculos teóricos y la salida práctica.

Debido a que a nuestro criterio los resultados obtenidos durante las pruebas de la arquitectura final fueron satisfactorios, no dudamos en confirmar la funcionalidad de la Lógica Difusa en sistemas de Control, pudiendo aventurarnos a decir que en un futuro próximo se contará con Sistemas de Control Fuzzy en la mayor parte de los controles industriales y domésticos, ya que como comprobamos son de bajo costo, por ello consideramos que el campo de desarrollo de los Controladores de Lógica Difusa es abundante.

Bibliografía.

1. **FUZZY EXPERT SYSTEMS.**
   Abraham Kandel;
   CRC Press, 1992.

2. **NEURAL NETWORKS AND FUZZY SYSTEMS - A Dynamical Systems Aproach to Machine Intelligence.**
   Bart Kosko;
   Prentice Hall, 1992.

3. **SPECIAL TOOLS AMDS CHIPS MAKE FUZZY LOGIC SIMPLE.**
   Gari Legg;
   EDN, July 6 1992.

4. **ADVANCED FUZZY LOGIC CONTROL TECHNOLOGIES IN AUTOMOTIVE APLICATIONS.**
   C. von Altrock, B. Krause and H. J. Zimmermann;
   IEEE, 1992

5. **PERFORMANCE AND EVALUATION OF A SELF-TUNNING FUZZY CONTROLLER.**
   Walter Daugherity, Balaji Rathakrishnan and John Yen;
   IEEE San Diego, 1992.

6. **SELF-ORGANIZIING FUZZY CONTROL.**
   Mark D. Spinrad;
   ISA, 1991.

7. **SINGLE LOOP FUZZY CONTROLLERS.**
   M. Tanaka, A. Patani and K. Yamada;
   ISA, 1991.

8. **FUZZY MODELIND.**
   Amin Patani and Koichi Yamada;
   ISA, 1991.

9. **FUZZY FUNDAMENTALS.**
   Earl Cox;
   IEEE Spectrum, October 1992.

10. **FUZZY-LOGIC BASICS: INTUITIVE RULES REPLACE COMPLEX MATH.**
David I. Brubaker;
EDN Press, 1992.

11. **FUZZY LOGIC IN C - Creating an Fuzzy-Based Inference Engine.**
Greg Viot;
Dr. Dobbs Journal, Frebruary 1993.

12. **FUZZY LOGIC AND ITS APLICATION IN CONTROL SYSTEMS.**
Pat Murphy;
ISA, 1991.

13. **FUZZY LOGIC FLOWERS IN JAPAN.**
Daniel G. Schwartz and George J. Klir;
IEEE Spectrum, July 1992.

14. **FUZZY SETS THEORY AND ITS APLICATIONS.**
H. J. Zimmermann;
Kluwer Academic Publishers, 1990.

15. **FUZZY LOGIC.**
Lotfi A. Zadeh;
Reprinted from IEEE Computer Mag., April 1988.

16. **FUZZY OUTPUTS.**
David I. Brubaker;
The Huntington Group, 18 January 1991.

17. **FUZZY OUTPUTS II - Discontinuity and Ambiguity.**
David I. Brubaker;
The Huntington Group, 15 Frebruary 1991.

18. **SINGLETON OUTPUTS.**
David I. Brubaker;
The Huntington Group, 15 July 1991.

19. **ALTERNATE RULE REPRESENTATIONS.**
David I. Brubaker;
The Huntington Group, 19 August 1991.

20. **FUZZY CENTROIDS AND SINGLETONS.**
David I. Brubaker;
The Huntington Group, 18 November 1991.

21. **ORDERING, DISTANCE AND CLOSENESS OF FUZZY SETS.**
Lázló T. Kóczy;
Elsevier Science Publisher, 1993.

22. **FUZZY LOGIC AND THE NEURON CHIP.**
Motorola Inc., 1993.

23. **FUZZY LOGIC SOLVES CONTROL PROBLEM.**
David I. Brubaker and Cedric Scheerer;
EDN Press, June 18, 1992.

24. **DATA ACQUISITION LINEAR DEVICES.**
National Semiconductor; 1989.

25. **AMPLIFICADORES OPERACIONALES Y CIRCUITOS INTEGRADOS LINEALES.**
Robert F. Coughlin and Frederic F. Driscoll;
Prentice Hall; 1993.

26. **FAST AND LS TTL DATA.**
Motorola Inc.; 1992.

27. **PROGRAMMABLE LOGIC DEVICES.**
National Semiconductor; 1989.

28. **SISTEMAS ELECTRONICOS DIGITALES.**
Enrique Mandado;
Alfaomega Marcombo, 1992.

29. **LOGICA DIGITAL Y DISEÑO DE COMPUTADORES.**
M. Morris Mano;
Prentice Hall, 1982.

30. **EPROM - Data Book.**
Intel; 1990.

31. **ECG - Semiconductor Master Replacement Guide.**
Philips ECG; January 1989.