Instituto Politécnico Nacional

Centro de Investigación en Computación

Secretaría de Investigación y Posgrado

TÍTULO DE LA TESIS

Diseño de Sistemas Difusos utilizando Operadores Parametrizables con una Arquitecturas Reconfigurable

T E S I S

QUE PARA OBTENER EL GRADO DE

MAESTRO EN CIENCIAS EN INGENIERÍA DE CÓMPUTO CON OPCIÓN EN SISTEMAS DIGITALES

P R E S E N T A

Ing. Prometeo Cortés Antonio

DIRECTOR (ES) DE TESIS:

Dr. Herón Molina Lozano
Dr. Ildar Batyrshin

MÉXICO, D.F.

SIP-14

INSTITUTO POLITÉCNICO NACIONAL
SECRETARÍA DE INVESTIGACIÓN Y POSGRADO

ACTA DE REVISIÓN DE TESIS

En la Ciudad de México, D.F. siendo las 13:00 horas del día 15 del mes de Abril de 2011 se reunieron los miembros de la Comisión Revisora de la Tesis, designada por el Colegio de Profesores de Estudios de Posgrado e Investigación del:

Centro de Investigación en Computación

para examinar la tesis titulada:

"DISEÑO DE SISTEMAS DIFUSOS UTILIZANDO OPERADORES PARAMETRIZABLES CON UNA ARQUITECTURA RECONFIGURABLE"

Presentada por el alumno:

CORTÉS	**ANTONIO**	**PROMETEO**
Apellido paterno	Apellido materno	Nombre(s)

Con registro: A 0 9 0 2 5 4

aspirante de: **MAESTRÍA EN CIENCIAS EN INGENIERÍA DE CÓMPUTO CON OPCIÓN EN SISTEMAS DIGITALES**

Después de intercambiar opiniones los miembros de la Comisión manifestaron ***APROBAR LA TESIS***, en virtud de que satisface los requisitos señalados por las disposiciones reglamentarias vigentes.

LA COMISIÓN REVISORA

Directores de Tesis

Dr. Herón Molina Lozano

Dr. Ildar Batyrshin

Dr. Luis Alfonso Villa Vargas

Dr. Víctor Hugo Ponce Ponce

Dr. José Luis Oropeza Rodríguez

PRESIDENTE DEL COLEGIO DE PROFESORES

Dr. Luis Alfonso Villa Vargas

INSTITUTO POLITÉCNICO NACIONAL
CENTRO DE INVESTIGACIÓN EN COMPUTACIÓN
DIRECCIÓN

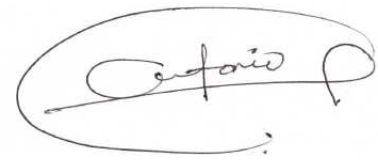
INSTITUTO POLITÉCNICO NACIONAL

SECRETARÍA DE INVESTIGACIÓN Y POSGRADO

CARTA DE CESIÓN DE DERECHOS

En la Ciudad de México, D.F., el día 16 del mes de Junio del año 2011, el que suscribe Prometeo Cortés Antonio alumno del Programa de Maestría en Ciencias en Ingeniería en Cómputo con Opción en Sistemas Digitales con número de registro A090254, adscrito a el Centro de Investigación en Computación, manifiesta que es autor intelectual del presente trabajo de la Tesis bajo la dirección de Dr. Herón Molina Lozano y del Dr. Ildar Batyrshin y cede los derechos del trabajo intitulado "Diseño de Sistemas Difusos utilizando Operadores Parametrizables con una Arquitectura Reconfigurable", al Instituto Politécnico Nacional para su difusión, con fines académicos y de investigación.

Los usuarios de la información no deben reproducir el contenido textual, gráficas o datos del trabajo sin el permiso expreso del autor y/o directores del trabajo. Este puede ser obtenido escribiendo a la siguiente dirección acorteo@hotmail.com. Si el permiso se otorga, el usuario deberá dar el agradecimiento correspondiente y citar la fuente del mismo.



Ing. Prometeo Cortés Antonio

Resumen

En este trabajo se diseñan sistemas difusos de dos entradas y una salida, con funciones de membrecía, operadores de conjunción y reglas difusas parametrizables, para los modelos Mamdani, Sugeno y Tsukamoto, que son implementados en FPGA de Altera y Xilinx

También se presenta, una sección con lógica parcialmente reconfigurable, también conocida como lógica dinámicamente reconfigurable, para integrar los diferentes modelos de sistemas difusos en un solo diseño. Además los 18 operadores parametrizables utilizando el método de suma monotónica son modelados usando también este enfoque.

La implementación de la lógica parcialmente reconfigurable requiere de un parche, que se debe agregar en las herramientas de Xilinx. En esta tesis, se trabaja con las herramientas del Xilins ISE 9.2i, PlanAhead 10.1, en el sistema operativo Windows, y con el dispositivo, XC4VFX12, de la familia Virtex IV, usando el flujo de diseño, Early Access Partial Reconfiguration.

Se diseña una herramienta software para la manipulación de los parámetros del sistema difuso y el manejo de los archivos parciales de implementación, es decir, archivos bitstream parciales, usando el método de implementación de líneas de comandos.

Además se integra un anexo en el cual se explica paso a paso el flujo de diseño Early Access Partial Reconfiguration, con el objetivo, que este trabajo pueda reproducirse, modificar y agregar funcionalidad.

En la realización de esta tesis, se elaboraron los siguientes cuatro paper, sobre la implementación de operados de conjunción y los modelos de sistemas difusos. *"FPGA Implementation of (p)-Monotone Sum of Basic t-norms",* en el *ieee world congress on computational intelligence 2010 in Barcelona España. "FPGA implementation of fuzzy system with parametric membership functions and parametric conjunctions", en 9th Mexican International Conference on Artificial Intelligence, 2010 en Pachuca Hidalgo, México . FPGA implementation of parametric Fuzzy Mandani Systems en Ninth International Conference on Appplication of Fuzzy Systems and Soft Computing. Agosto de 2010. En Praga, Republica Checa. "FPGA implementation of (p, I-p) – Monotone Sum of Basic t-norm" en el World Conference on Soft Computiong in San Francisco California. 2011.*

Abstract

In this tesis, fuzzy systems are designed for two inputs and one output, with membership functions, operators of conjunction and fuzzy rules parameterized for the Mamdani, Sugeno and Tsukamoto models, which are implemented in Altera and Xilinx FPGA.

in this also presents a logical partially reconfigurable section, also called dynamic reconfigurable logic, for integrate the different fuzzy system models in a single design. Also, are implemented the 18 operators parameterized generated by monotonic sum method with this approach.

The implementation of partially reconfigurable logic requires a patch to be added in the tools Xilinx. In this thesis, working with tools Xilins ISE 9.2i, PlanAhead 10.1 on the Windows operating system, with the device XC4VFX12, of Virtex IV family, using design flow Early access Partial Reconfiguration.

Designing a software tool for handling parameters of the fuzzy system and partial files implementation called bitstream using the method of line commands.

It also integrates an appendix that explains step by step Early access Partial Reconfiguration design, with objective, this work can reproduces, modify and add functionality.

During this thesis, has developed the following four paper on the implementation of operations of conjunction and models of fuzzy systems. . *"FPGA Implementation of (p)-Monotone Sum of Basic t-norms"*, en el *ieee world congress on computational*

intelligence 2010 in Barcelona España. "FPGA implementation of fuzzy system with parametric membership functions and parametric conjunctions", en 9th Mexican International Conference on Artificial Intelligence, 2010 en Pachuca Hidalgo, México . FPGA implementation of parametric Fuzzy Mandani Systems en Ninth International Conference on Appplication of Fuzzy Systems and Soft Computing. Agosto de 2010. En Praga, Republica Checa. "FPGA implementation of (p, I-p) – Monotone Sum of Basic t-norm" en el World Conference on Soft Computiong in San Francisco California. 2011.

Índice de Contenido

Índice de Figuras

CAPÍTULO 1. INTRODUCCIÓN

La gran popularidad de implementación en hardware de sistemas difusos en la solución de problemas actuales en ramas como el control, Lund, Tipsuwanpom (2004), reconocimiento de patrones, Christos (2010), Nambakhsh(2008). Kawai, 2009, toma de decisiones, Chenn (2004), Chowdhury (2008) y Haider (2006), entre otros, ha creado la necesidad del desarrollo de herramientas de implementación de sistemas difusos que puedan ser fácilmente adoptados a diferentes aplicaciones y que puedan operar en diferentes ambientes.

El diseño de este tipo de herramientas pueden ser desarrolladas en dos niveles de abstracción: el modelado y simulación del sistema a nivel de software y la implementación del sistema a nivel de hardware.

Para modelar un sistema difuso generalizado, es conveniente dividir el sistema difuso en etapas y observar los parámetros que pueden ser ajustables. La etapa de la base de datos donde se definen los conjuntos difusos que modelan a las variables de entrada y salida, la etapa de la base de las reglas difusas modela el mecanismo de razonamiento del sistema difuso, la etapa de agregación, donde se obtiene un conjunto difuso (en el caso del sistema difuso tipo Mamdani) o un resultado certero (en el caso de un sistema difuso tipo Sugeno o Tsukamoto). Para sistemas Mamdani se considera otra etapa llamada defusificación para convertir un conjunto difuso a un valor certero.

Por lo general, la parametrización de las funciones de membresía es un enfoque común en la construcción de sistemas difusos, sin embargo la parametrización de los operadores difusos en el proceso de inferencia difusa y la agregación no ha sido muy

utilizada en la implementación de los sistemas difusos, a pesar de que existen varias investigaciones en el área, Batyrshin (1998, 1999), Cervinka (1996), Eskil (1999). En este trabajo se presentan ambos enfoques de parametrización, además de permitir la modificación de las reglas del sistema difuso en tiempo real, que permiten construir una herramienta de sistemas difusos altamente reconfigurable con altas posibilidades adaptativas.

Utilizando los principios de la lógica dinámicamente reconfigurable y la división en etapas de la lógica difusa, se utiliza una arquitectura reconfigurable que permita a la herramienta diseñar los tres tipos de modelos de sistemas difusos más utilizados. Además los operadores de conjunción serán implementados utilizando lógica dinámicamente reconfigurable, esto es para decrementar el tiempo de implementación, así como uso de componentes y energía en hardware.

1.1 ANTECEDENTES

Los sistemas difusos surgen como una alternativa para el modelado de sistemas no lineales, con incertidumbre, en donde se les da una solución aproximada, de la misma manera que lo hace el razonamiento humano. Este enfoque es sustentado en el seminario de Zadde (1965). En donde propone un modelo de lógica continua (*Fuzzy logic*), en la cual los elementos de un conjunto son miembros de este de forma gradual.

El enfoque de lógica difusa ha sido ampliamente utilizada en problemas de control, reconocimiento de patrones, toma de decisiones, entre otros, debido a que su modelación es muy simple, la cual se realiza formulando una base de reglas si-entonces que definen el comportamiento del sistema utilizando el conocimiento de un experto. A lo largo del tiempo han surgido varias propuestas de modelado de sistemas difusos, siendo los modelos Mamdani, Sugeno y Tsukamoto los más populares, Terano (1994), Ross(2004).

También se han desarrollado diferentes formas de modelar a los conjuntos difusos, tales como funciones triangulares, trapezoidales, gaussiana, sigmoidales, etc, siendo las funciones lineales las más utilizadas para la implementación en *Hardware*. Además han surgido muchas variantes de modelado de los operadores de conjunción (t-norma), disyunción (s-norma) y negación.

Actualmente se han desarrollado una gran variedad de herramientas que modelan sistemas difusos a nivel de hardware, principalmente para dispositivos FPGA, además se han desarrollado investigaciones en el diseño de herramientas para el modelado e implementación de sistemas difusos, que sean utilizados en diferente tipos de aplicaciones, sin embargo la mayoría de estos trabajos sólo consideran el ajuste en las funciones de membresía, fijando un solo operador de conjunción y disyunción en las etapas de inferencia y agregación, además que en todas las herramientas actuales fijan solo un modelo difuso en su aplicación, siendo el tipo Mamdani el más popular, utilizando conjuntos consecuentes singletons.

La lógica dinámicamente reconfigurable (también conocida como parcialmente reconfigurable) ha tomado gran aceptación en la implementación de sistemas sobre FPGA, y su principio es reconfigurar parte del dispositivo, mientas el resto del diseño sigue operando con normalidad, es decir regiones que han sido configuradas como parcialmente reconfigurable, pueden cambiar su funcionamiento, en tiempo real cargando un archivo de reconfiguración parcial (bitstream) que únicamente reconfigurará la región parcial, en lugar de reconfigurar todo el dispositivo.

Se han desarrollado herramientas para implementación de sistemas difusos, en donde se utiliza lógica dinámicamente reconfigurable, para el ajuste automático de las funciones de membresía, utilizando fundamentos de redes neuronales. Sin embargo estos sistemas son bastante rígidos, al no permitir la configuración del tipo de modelo difuso, la base de reglas y la parametrización de los operadores difusos. Por mencionar algunas otras herramientas más robustas tales como XFUZZY, AFAN, las

cuales generan código VHDL, por lo que todavía se tiene que elaborar el proceso de sintetización e implementación del sistema, utilizando de manera directa las herramientas del fabricante tales como Xilinx o Altera. Además si se quiere cambiar o modificar los parámetros o algunas de las etapas o alguno de los elementos antes descritos, se tiene que hacer nuevamente el proceso de las etapas de diseño, sintetización e implementación.

1.2 PLANTEAMIENTO DEL PROBLEMA

El proceso de diseño de los sistemas difusos se puede dividir en dos partes, el modelado de sistema software, y la implementación en hardware del sistema.

El modelado de un sistema difuso es muy sencillo, si inicia como todo modelo, definiendo las variables de entrada y salida, luego entonces, se definen el universo de discurso de las variables, posteriormente definir los conjuntos difusos que realizaran el mapeo (fusificación) del sistema. Una vez definidas las variables difusas, se establece un conjunto de reglas utilizando el conocimiento de un experto, para modelar el comportamiento del sistema, es decir se establece el razonamiento difuso. Por último se elige un modelo difuso a utilizar.

El siguiente paso es simular el sistema difuso en una herramienta de software, y se realizan ajustes tales como cambio en los parámetros o formas de los conjuntos difusos, operadores de conjunción y disyunción o de las reglas difusas de tal manera que se alcance alguna función objetivo o deseada.

Posteriormente se realiza el proceso de implementación en Hardware del sistema, el se divide en tres etapas fundamentales, la codificación, la síntesis y el proceso de implementación.

En la etapa de codificación el diseñador estable la topología de diseño del sistema, en donde se definen los módulos que componen el sistema, en la etapa de síntesis, con la

ayuda de las herramientas de diseño del fabricante, se realiza el *place and route* que se establecerá en el dispositivo, y en la etapa de implementación se establecen las restricciones del modelo.

Este proceso, es bastante lento y tedioso, aunado a esta limitante, cuando se quiere ajustar o calibrar el sistema difuso, es decir modificar algún parámetro o cambiar el funcionamiento de alguno de módulos, es necesario realizar todo el proceso de implementación nuevamente, lo cual hace que el proceso de calibración sea realmente engorroso.

Por lo que para los ingenieros es muy factible una herramienta que agilice este proceso, se plantea una solución utilizando lógica parcialmente reconfigurable, que permita hacer cambios operacionales en partes del dispositivo.

1.3 JUSTIFICACIÓN

Actualmente las herramientas de diseño e implementación de sistemas difusos a nivel de hardware, no han alcanzado el alto grado de reconfiguración como en el modelado a nivel de software, que permitan implementar diferentes tipos de sistemas difusos, con funciones de membresía, operadores difusos y base de reglas modificables. Además un problema de los sistemas a nivel de Hardware que permiten la parametrización de algunos componentes, es que este proceso se realiza a nivel de código, por lo que el proceso de simulación, sintetización e implementación en hardware tiene que realizase, lo cual hace al proceso muy lento y tedioso.

En este trabajo se realiza una herramienta de diseño en la cual se permita modelar e implementar sistemas difusos, eliminando el proceso de codificación síntesis e implementación a nivel hardware. Realizando módulos parametrizables, que sean dinámicamente reconfigurables, de tal modo que el software cargue únicamente los módulos al dispositivo.

Con esta herramienta se logra que la etapa de implementación sea más rápida, y no sea necesario utilizar de manera directa las herramientas de diseño del fabricante. Sin embargo, para poder lograr una herramienta con alto grado de reconfiguración, es necesario, restringir el fabricante y el tipo de dispositivo a utilizar.

Se utiliza un FPGA Virtex IV, debido a que es el dispositivo con mayores recursos en hardware que soporta el flujo de diseño de reconfiguración parcial *Early Access Partial Reconfiguración*, que a la vez, es la última versión de flujos de diseño de Xilinx. Además con este se proyecto se da continuidad a la tesis "Desarrollo de una herramienta para el diseño de sistemas reconfigurables" realizada en esta institución.

Esta herramienta permite que los iniciadores en la lógica difusas puedan implementar sistemas a nivel de hardware sin necesidad de conocer un leguaje de programación HDL.

1.4 OBJETIVOS

1.4.1 OBJETIVO GENERAL

Desarrollar una herramienta para el diseño e implementación en hardware de sistemas difusos con una arquitectura reconfigurable para los modelos Mamdani, Sugeno y Tsukamoto, utilizando la tarjeta de desarrollo de Virtex 4 de Xilinx.

1.4.2 OBJETIVOS PARTICULARES

- Implementar en lenguajes de descripción de hardware las funciones de membresía de los conjuntos difusos, operadores de conjunción parametrizables usando el método se suma monotónica (p) y (p,I-p) en los modelos difusos Mamdani, Sugeno, Tsukamoto.
- Diseñar módulos parcialmente reconfigurables, para describir los operadores de conjunción

- Integrar los modelos modelos difusos Mamdani, Sugeno y Tsukamoto, en un sólo sistema difuso, utilizando reconfiguración dinámica

- Desarrollar la interfaz de usuario (software) de la herramienta de diseño que sirva para manipular la implementación en hardware

1.5 ORGANIZACIÓN DEL TRABAJO

El contenido de esta tesis de divide en 5 capítulos, el presente capítulo se han presentado, una breve introducción sobre las herramientas de diseño de sistemas difusos, el planteamiento, la justificación de problema y se establecen los objetivos de este trabajo.

En el capítulo 2 se presentan los fundamentos para el desarrollo de este trabajo, que se dividen en dos partes, los fundamentos de la lógica difusa y los fundamentos de la lógica parcialmente reconfigurable, donde se definen las propiedades de los sistemas dinámicamente reconfigurables, para el flujo de diseño EA_PR.

En el capítulo 3, se establece el modelado y arquitectura que se emplea para el diseño de la herramienta de modelado e implementación de sistemas difusos, que satisfaga los objetivos presentados en la presente tesis. Así como el desarrollo del sistema, donde se explica paso a paso el diseño de los módulos que conforman el sistema difuso, mostrando el pseudocódigo que los describen, los diagramas lógicos, así como los resultados que genera cada módulo.

Además este trabajo, se presenta un ejemplo que explica el flujo de diseño de un sistema parcialmente reconfigurable, esto para lectores que deseen modificar o anexar módulos a la herramienta.

CAPÍTULO 2. FUNDAMENTOS.

Introducción

La lógica difusa se fundamenta en dos principios. La teoría de conjuntos difusos que se refiere a la vaguedad que se encuentran en la semántica, y la lógica difusa que incluye el principio de las reglas difusas y el razonamiento difuso. En este capítulo se introducirán las definiciones básicas y los conceptos fundamentales de la teoría de conjuntos y del razonamiento difuso, de la misma manera mencionaran los conceptos fundamentales de la lógica reconfigurable y las características y restricciones de flujo de diseño "*Early Access Partial Reconfiguration*"

2.1 Teoría de conjuntos difusos

La teoría de conjuntos difusos es una extensión de la teoría de conjuntos clásica, con un cambio de enfoque al considerar que un elemento de un conjunto no necesariamente pertenece o no un conjunto, si no que un elemento tiene un grado de pertenencia o membresía a un conjunto, a continuación se presentan las definiciones de la teoría de conjuntos, Jang (1997).

2.2 Universo de discurso

El universo de discurso, es el espacio que contiene toda la información disponible para una variable en un problema dado. Una vez definido el universo de discurso de una variable, podemos definir eventos certeros sobre el espacio de la información, es decir, es posible obtener abstracciones del universo de discurso, tales como conjuntos.

2.2.1 Conjunto Difuso

Sea *X* una colección de objetos denotada generalmente por *x*, entonces un conjunto difuso *A* en X está definido como el conjunto de pares ordenados siguiente:

$$A = \{(x, \mu_A(x)) \mid x \epsilon X \ \ y \ \mu_A(x) : U \rightarrow [0,1]\}$$

Donde $\mu_A(x)$ es llamada función de membresía del conjunto difuso A.

2.2.1 Función de membresía

Una función de membresía mapea cada elemento de X a un grado de membresía o pertenecía entre 0 y 1, es decir, el es espacio de [0,1].

Los conjuntos difusos, pueden ser definidos para espacios continuos o discretos, una notación alternativa para definir los conjuntos difusos es:

Espacio discreto

$$A = \sum_{x_i \in X} \frac{\mu_A(x_i)}{x_i}$$

Espacio continuo

$$A = \int_X \frac{\mu_A(x)}{x}$$

Donde $\sum$ y el signo de integral establecen la unión de los grados de membresía; el símbolo "/" establece un marcador y no implica la división. Siendo el espacio discreto el que se va a utilizar en este proyecto. El universo de discurso, es usualmente particionado en conjuntos difusos.

2.3 Funciones de membresía comunes

Como se ha mencionado las funciones de membresía representan el grado de pertenencia de los elementos del universo de discurso en un conjunto difuso. En la bibliografía se han propuesto diversas ecuaciones matemáticas que modelen a los conjuntos difusos. Las funciones de membresía más usadas para representar a las funciones de pertenencia son las funciones triangular, trapezoidal, gaussiana y singleton, las cuales son definidas con algunos parámetros.

2.3.1 Función de membresía Triangular

La función triangular es definida por tres parámetros {a,b,c}, que dividen el espacio en tres comportamientos.

$$\mu_{Triangular}(x;a,b,c) = \begin{cases} 0 & x \leq a \\ \dfrac{x-a}{b-a} & a \leq x \leq b \\ \dfrac{c-x}{c-b} & b \leq x \leq c \\ 0 & c \leq x \end{cases}$$

Una forma alternativa de definir la función triangular es usando las operaciones máximo y mínimo como sigue.

$$\mu_{Triangular}(x;a,b,c) = \max\left(\min\left(\frac{x-a}{b-a}, \frac{c-x}{c-b}\right), 0\right)$$

En la figura 2.1 se muestra la función triangular con parámetros (x; 40,80, 90), y como se describe en la ecuación, la función vale cero para valores de x menores a 40 y mayores a 90. Para $40 \leq x \leq 80$ la ecuación es descrita por una recta con pendiente positiva, y para valores de $40 \leq x \leq 80$ se describe por la ecuación de la recta con pendiente negativa.

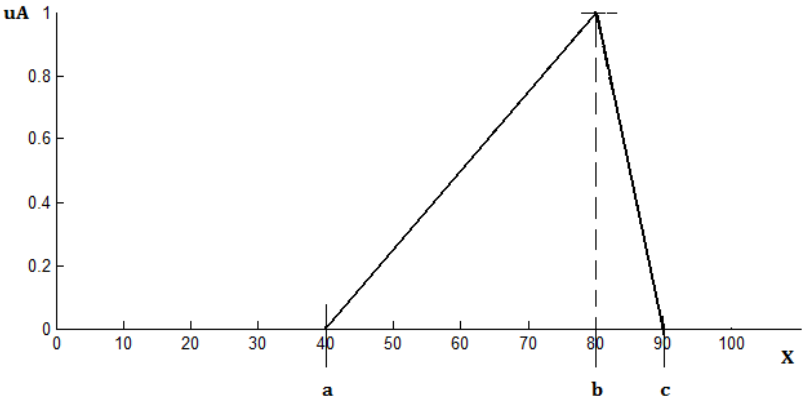


Figura 2.1 Función de membresía triangular parametrizada. Con parámetros a, b, c

2.3.2 Función de membresía Trapezoidal

La función trapezoidal se define a partir de cuatro parámetros

$$\mu_{Trapezoidal}(x;a,b,c,d) = \begin{cases} 0 & x \leq a \\ \dfrac{x-a}{b-a} & a \leq x \leq b \\ 1 & b \leq x \leq c \\ \dfrac{d-x}{d-c} & c \leq x \leq d \\ 0 & d \leq x \end{cases}$$

Alternativamente la función trapezoidal puede ser definida con las operaciones máximo y mínimo como sigue

$$\mu_{Trapezoidal}(x;a,b,c,d) = \max\left(\min\left(\frac{x-a}{b-a}, 1, \frac{d-x}{d-c}\right), 0\right)$$

La figura 2.2, se muestra la función trapezoidal con parámetros (x; 30, 40, 80, 100), donde su descripción es similar a la función triangular, para valores menores a *a* y mayores a d, el valor de pertenencia de los elementos al conjunto es cero, para valores entre *b* y *c* es igual a 1, para valores de x que están entre a y b es descrita por una función lineal creciente, y para x que van de c a d, su valor de pertenencia se describen por la una ecuación lineal decreciente

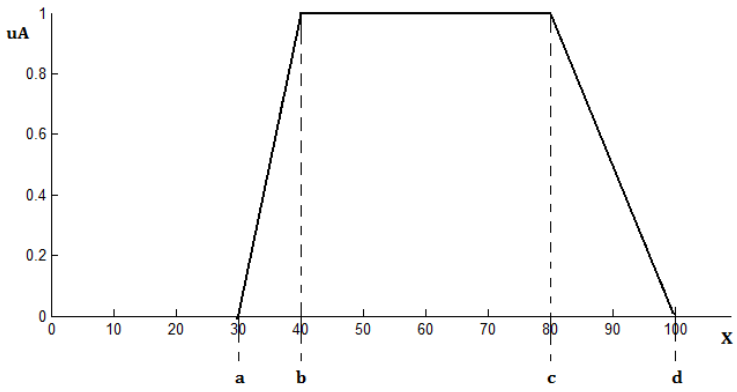


Figura 2.2 Función de membresía trapezaoidal parametrizada. Con parámetros a, b, c, d

2.3.3 Función de membresía Gaussiana

La función gaussiana es definida con dos parámetros {c,σ}

$$\mu_{Gaussiana}(x; c, \sigma) = e^{-\frac{1}{2}(\frac{x-c}{\sigma})^2}$$

Donde c representa el centro y σ determina el ancho de la función. En la figura 2.3 se muestra la morfología de un conjunto difuso con función de pertenencia gausiana. Con parámetros C=30 y $\sigma = 58$, donde el parámetro *c*, determina el valor de pertenencia máximo de la función, que además es el centro de la función y σ determina el ancho de la función.

2.3.4 Función campana generalizada de bell

Es determinada por tres parámetros:

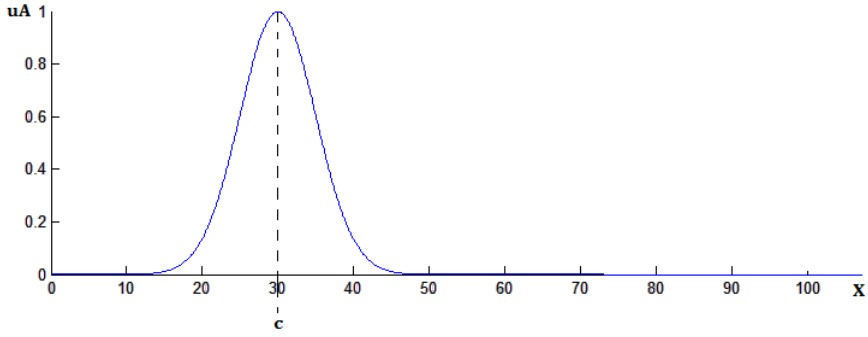


Figura 2.3. Función de membresía Gaussina parametrizada. Con parámetros c, σ

$$\mu_{Bell}(x; a, b, c) = \frac{1}{1 + |\frac{x-c}{a}|^{2b}}$$

Donde el parámetro *b* es usualmente positivo, *c* determina el centro y *a* el ancho, la función campana de Bell es una generalización de la distribución probabilística de Cauchi. En la figura 2.4 se muestra la forma de una función de pertenencia con campana de Bell *a*=10, *b*=5, *c*=70, donde puede observarse que varios valores de x, pueden tener un valor de pertenencia igual a 1

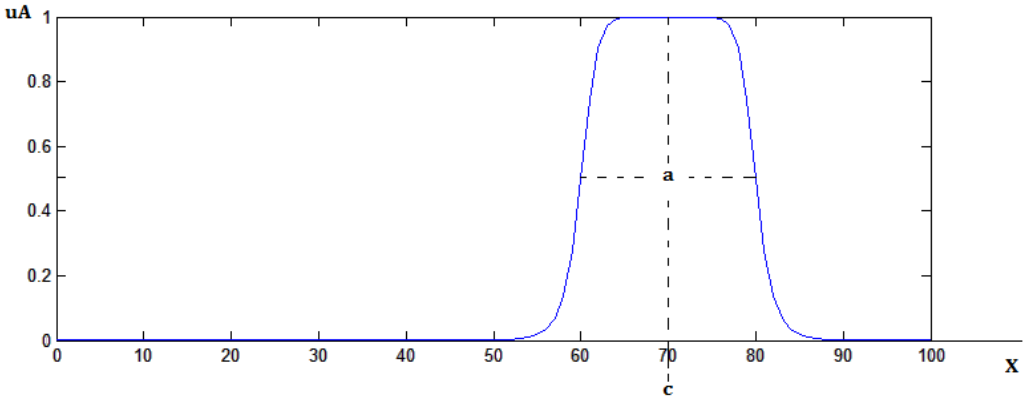


Figura 2.4. Función de membresía Campana de bell. Con parámetros a, b, c

2.4 Operadores difusos

Como en la lógica tradicional, las operaciones lógicas unión, intersección y complemento pueden ser aplicadas con conjuntos difusos.

2.4.1 Unión (máximo)

La operación de unión puede ser definida de muchas diferentes formas. La operación más popular es el máximo.

La unión de dos conjuntos difusos A y B con funciones de membresía $\mu_A(x)$ y $\mu_B(x)$, es un conjunto $C = A \cup B$, con una función de membresía que relaciona al conjunto A con B de la siguiente manera:

$$\mu_C(x) = \max\left(\mu_A(x), \mu_B(x)\right) = \mu_A(x) \vee \mu_B(x)$$

Donde max, calcula el valor de pertenencia máximo de x, en los conjuntos A y B

2.4.2 Intersección (mínimo)

La intersección de dos conjuntos difusos A y B es un conjunto difuso $C = A \cap B$ o C= A and B, la función de membresía que relaciona a A y B es

$$\mu_C(x) = \min\left(\mu_A(x), \mu_B(x)\right) = \mu_A(x) \wedge \mu_B(x)$$

Donde mix, calcula el valor de pertenencia mínimo de x, en los conjuntos A y B

2.4.3 Complemento.

El complemento de un conjunto difuso A se denota como $\overline{A}$ (NOT A), su función de membresía define como:

$$\mu_{\bar{A}}(x) = 1 - \mu_A(x)$$

En la figura 2.5 se muestran un ejemplo de los operadores difusos de conjunción, disyunción y negación. En el inciso a, se muestran dos conjuntos difusos A y B sobre el mismo universo de discurso X. En los incisos b y c se muestran la unión y la intersección respectivamente entre los conjunto del inciso a, y finalmente en el inciso de la figura se muestra la negación del conjunto A.

2.4.4 Subconjunto o conjunto contenido.

Un conjunto difuso A esta contenido en el conjunto difuso B (o A es subconjunto de B), si y sólo si

$$\mu_A(x) \leq \mu_B(x)$$

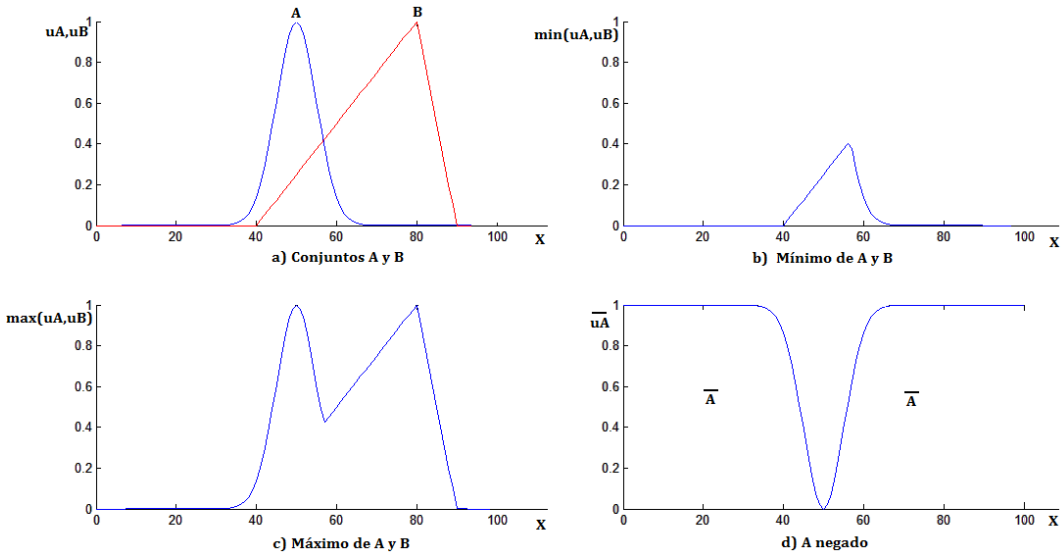


Figura 2.5. Operaciones difusas unión, intersección y complemento.

Otras definiciones que describen operaciones de conjuntos difusos son las siguientes:

2.4.5 Producto Cartesiano.

Si A y B son conjuntos difusos en los universos de discursos X y Y, respectivamente. El producto cartesiano de A y B, denotado por A x B, es un conjunto en el espacio producto $X\,x\,Y$ con función de membresía.

$$\mu_{A\,x\,B}(x,y) = \min\left(\mu_A(x), \mu_B(x)\right)$$

2.4.6 Coproducto Cartesiano

El coproducto cartesiano A+B es un conjunto difuso con función de membresía

$$\mu_{A+B}(x,y) = \max\left(\mu_A(x), \mu_B(x)\right)$$

2.4.7 Operador t-norma

Es el operador de intersección difusa t-norma (norma triangular), es especificada de forma general por una función $T:\ [0,1]\,x\,[0\,,1] \rightarrow [0,1]$ y debe de satisfacer los siguientes requerimientos

Propiedad frontera: $T(0,0) = 0,\ \ T(a,1) = T(1,a) = a$

Propiedad monotonicidad: $T(a,b) \leq\ T(c,d) \quad si\ a \leq c\ \ y\ b\ <\ d$

Propiedad conmutatividad: $T(a,b)\ =\ T(b,a)$

Propiedad asociatividad: $T(a,T(b,c))\ =\ T(T(a,b),c)$

Los operadores t-normas más frecuentes en la literatura de lógica difusa son el mínimo, producto algebraico, producto acotado y producto drástico, a continuación se muestra la representación matemática de las t-normas básicas. En la figura 2.6 se muestra el comportamiento de las t-nomas básicas, donde el valor de los ejes que esta determinado por el rango [0,1] que pertenece a los valores de membresía de los conjuntos difusos A y B, que pueden ser descritos en diferentes universos de discurso.

Mínimo. $T_M(a,b) = \min(a,b) = a \wedge b$

Producto Algebraico. $T_P(a,b) = a.b$

Producto Acotado o Lukasiewicz. $T_L(a,b) = \max(0,(a+b-1)$

Producto Drástico. $T_D(a,b) = \begin{cases} a & if\ \ b = 1 \\ b & if\ \ a = 1 \\ 0 & en\ otro\ caso \end{cases}$

Se puede observar que

$$T_D(a,b) \leq T_L(a,b) \leq T_P(a,b) \leq T_M(a,b)$$

Es decir, cumplen el principio de monotonicidad en este orden lo cual puede ser verificado matemáticamente

2.4.8 Operador s-norma

El operador de unión t-conorma o s-norma es especificado en general por una función $S: [0,1]\ x\ [0,1] \rightarrow [0,1]$ y debe cumplir las siguientes propiedades

Propiedad de frontera: $S(1,1) = 1, S(a,0) = S(0,a) = a$

Propiedad de monotonicidad: $S(a,b) \leq S(c,d) \quad si\ a \geq c\ \ y\ b \geq d$

Propiedad conmutatividad: $S(a,b) = S(b,a)$

Propiedad asociatividad: $S(a,S(b,c)) = S(S(a,b),c)$

Los operadores s-normas más frecuentes en la literatura de lógica difusa son: máximo, suma algebraica, suma acotada y suma drástica. a continuación se muestra su representación matemática y su comportamiento gráfico en la figura 2.7.

Máximo: $S_M(a,b) = \max(a,b) = a \vee b$

Suma Algebraica $S_P(a,b) = a + b - a.b$

Suma Acotada o Lukasiewicz : $S_L(a,b) = \min(1,\ (a+b))$

Suma Drástica: $S_D(a,b) = \begin{cases} a & if\ \ b = 0 \\ b & if\ \ a = 0 \\ 1 & en\ otro\ caso \end{cases}$

Se puede observar que

$$S_M(a,b) \leq S_P(a,b) \leq S_L(a,b) \leq S_D(a,b).$$

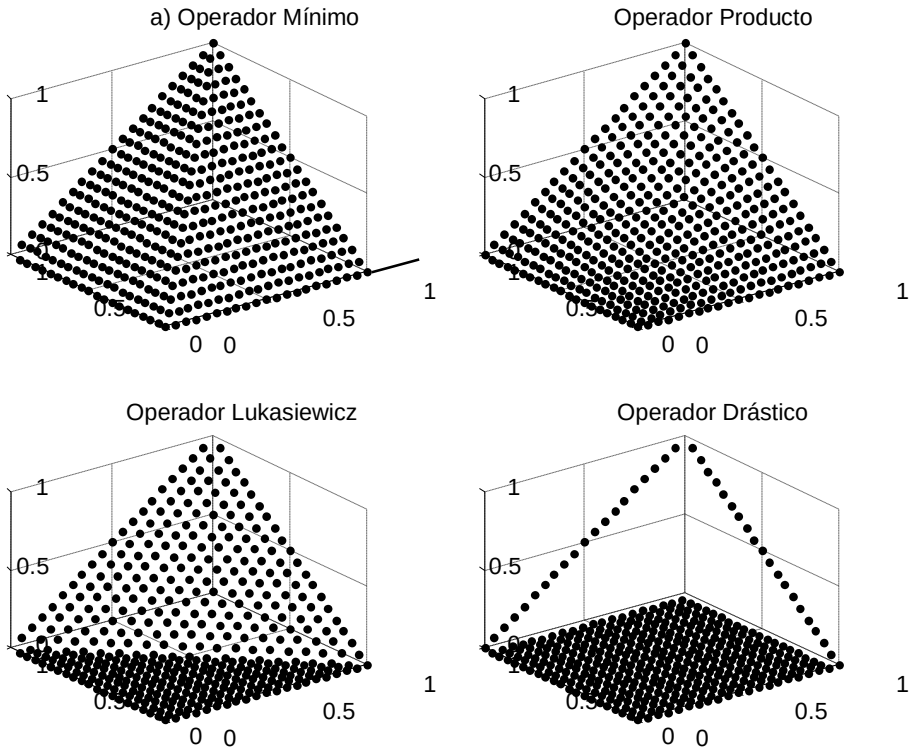


Figura 2.6. Conjunciones t-norma, mínimo, producto, lukasiewicz y drástico.

Es decir los operadores en este orden cumplen con el principio de monotonicidad.

2.4.9 Operadores t-normas y s-normas parametrizados

Las t-normas y s-normas parametrizadas han sido propuestas por varios investigadores tales como: Yager, Schweizer y Sklar, Dubois y Prade, Hamacher, Frank, Sugeno, Dombi sin embargo en esta investigación no se trabajó con estos operadores por que involucran una complejidad matemática que requiere de muchos recursos en su implementación en hardware.

Otras propuestas más recientes sobre operadores paramétricos son las propuestas en Batyrshin (1998,1999,2007), Hernández (2009), que son de implementación sencilla en sistemas digitales, sin embargo estos operadoresse apoyan de funciones generadoras para parametrizar los operadores de conjunción y disyunción básicos.

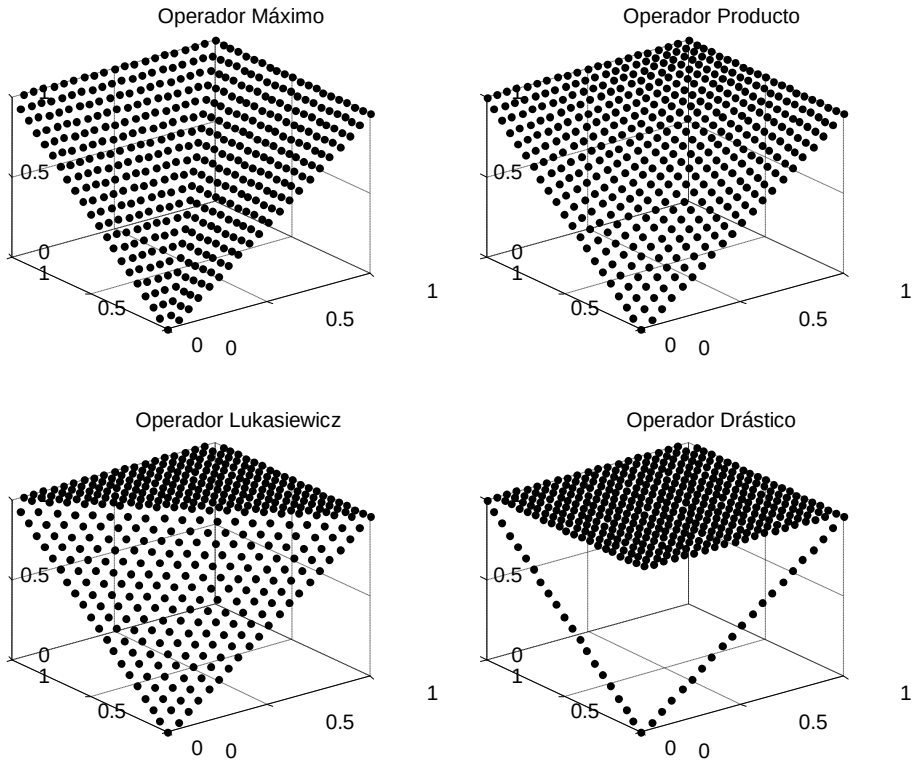


Figura 2.7. Conjunciones s-norma, mínimo, producto, lukasiewicz y drástico.

En este trabajo se presenta una metodología de parametrización de operadores de conjunción, los cuales son llamamos operadores de conjunción paramétricos basado en el método de suma monotónica, qie son generados a partir de las t-normas básicas y un parámetro p (2009).

2.5 Operadores de conjunción paramétrica de suma monotónica.

Este tipo de operadores se desarrollan a partir de los operadores t-norma básicos mostrados anteriormente. En este trabajo, se ha omitido el operador producto, debido a que tienen mayor complejidad en implementación, sin embargo es posible introducir el operador producto.

La idea básica del uso de los operadores paramétricos, es dividir en cuatro cuadrantes el espacio generado por las dos entradas, delimitados por un parámetro p.

2.5.1 Conjunción de suma monotónica con parámetro p.

El método denominado conjunción paramétrica de suma monotónica generaliza el método de la suma ordinal de t-normas y permite la posibilidad de construir una amplia clase de conjunciones paramétricas difusas.

En este trabajo se presenta un método llamado suma monotónica de conjunciones paramétricas difusas, que utiliza los operadores t-norma básicos Drástico, Lukasiewicz y Mínimo y un parámetro p, que puede tomar algún valor del conjunto L =[0,1],

Toda operación de conjunción difusa es una función $T: L\ x\ L\ \rightarrow\ L$ que satisface las siguientes condiciones sobre L.

$T(x, L)\ =\ x,\ T(L, x)\ =\ y$	Condición de Frontera
$T(x, y) \leq\ T(u, v)\ \ Si\ \ x \leq u,\ \ y \leq v$	Condición de monotocidad

Las conjunciones difusas son llamadas t-normas si satisfacen las condiciones conmutativa y asociativa es decir:

$T(x, y)\ =\ T(y, x)$	Propiedad conmutatividad
$T(x, T(y, z))\ =\ T(T(x, y), z)$	Propiedad asociatividad

De la misma manera que la mayoría de las aplicaciones de sistemas difusos, este trabajo contempla solo dos variables de entrada, por tal motivo, la propiedad asociativa para la conjunción difusa, no es requerida en la implementación en hardware de este sistema. Esto es importante, ya que nuestra herramienta de diseño no requerirá satisfacer esta propiedad, lo cual ayuda a la simplicidad del sistema, porque esta propiedad requiere de complejidad de implementación en el sistema.

Método de Conjunción difusa paramétrica p de suma monotónica.

Un conjunto X que contiene un número o una secuencia de números consecutivos de L es llamado un intervalo en L. Si $J = \{1, \ldots, n\},\ 1 < n \leq L - 1$ es un conjunto de

indexes y $(X_j)_{j\epsilon J}$, es una partición de L sobre un intervalo de pares ordenados disjuntos, tal que de $i < j$ es seguido por $x < y$ para todo $x \epsilon X_i$ y $y \epsilon X_j$, denotado por $x \epsilon D_{ij} = X_i \; x \; X_j$.

Suponga que Q es algún conjunto indexado y $(T_{ap} \leq)_{q\epsilon Q}$, es un conjunto parcialmente ordenado de conjunción difusa, Asignando a cada D_{ij} algún $T_{ij} = T_q$ de este conjunto tal que:

$$T_{ij}(x,y) = T_{st}(u,v) \; si \;\; i \leq s, j \leq t \;\; y \;\; x \leq u, y \leq v, \;\; (x,y) \epsilon D_{ij} \,, \;\; (u,v) \epsilon D_{st} \qquad \textbf{(1)}$$

Define a una función T sobre $L \; x \; L$ por $T(x,y) = T_{ij}(x,y) \;\; si \;\; (x,y) \epsilon D_{ij} \,, \;\; i,j \; \epsilon \; J$ Entonces T es llamada suma monotónica de $(D_{ij}, T_{ij})_{i,j \; \epsilon \; J}$ o conjunción difusa de suma monotónica $T_{ij}, \;\; i,j \; \epsilon \; J$

Teorema 1. Una conjunción difusa de suma monotónica, es un conjunción difusa.

Demostración. Si todas las conjunciones difusas usadas en la construcción de T satisfacen la condición de frontera $T(x,L) = x, \; T(L,x) = y$, entonces el resultado de la función resultante T, también satisface esta condición. La monotocidad de T cumple la ecuación (1).

Podemos reemplazar la condición de la ecuación (1), por una condición más simple:

$$T_{ij} = T_{st} \; si \;\; i \leq s, \; j \leq t$$

Entonces, gracias a la monotocidad de las conjunciones difusas de la ecuación 2, esta cumple a la ecuación 1. La función T definida por la ecuación 2 es llamada suma monotónica simple de $(D_{ij}, T_{ij})_{i,j \; \epsilon \; J}$

Corolario. Una conjunción difusa de suma monotónica simple es una conjunción difusa.

Basado en el teorema 1, consideramos dos métodos de conjunción difusas usando un parámetro p, que particiona a L con el intervalo X_j y la partición de $L\,x\,L$ en la correspodiente sección $D_{ij} = X_i\ x\ X_j$.

En el primer método, se define la partición de L como sigue: $X_1 = [0, p]$, $X_2 = (p, L]$, $p \in [0,1)$. El segundo método de partición es definido por: $X_1 = [0, p]$, $X_2 = (p, L-p]$, $X_3 = (L-p, L]$, con $p \in [0,0.5)$, al primer método se le conoce como suma monotónica (p) y al segundo método se le llama suma monotónica (p,L-p)

2.5.1 Conjunción difusa de suma monotónica (p)

La conjunción difusa (p) de suma monotónica, puede ser definida como sigue. Seleccione un conjunto de onjunciones difusas $\{T_{11}, T_{21}, T_{12}, T_{22}\}$ ordenado como sigue: $T_{11} \leq T_{12} \leq T_{22}$ y $T_{11} \leq T_{21} \leq T_{22}$. Se define una conjunción difusa T como sigue

$$T(x, y; p) = \begin{cases} T_{11}(x,y) & si\ \ x \leq p,\ \ y \leq p \\ T_{21}(x,y) & si\ \ x > p,\ \ y \leq p \\ T_{12}(x,y) & si\ \ x \leq p,\ \ y > p \\ T_{22}(x,y) & si\ \ x > p,\ \ y > p \end{cases}$$

La figura 2.8, muestra el espacio de L x L que es particionado en cuatro cuadrantes.

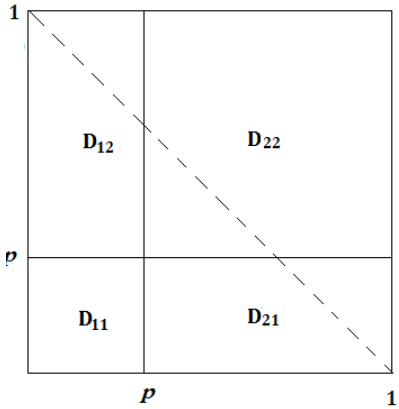


Figura 2. 8. Partición de $L\ x\ L$ en secciones $D_{ij} = X_i\ x\ X_j$ definidos por el parámetro p.

Considerando todas las conjunciones difusas conmutativas no triviales (t-normas básicas), que son obtenidas por este método, a través de las T_D, T_L, T_M, y recordando que estas t-normas pueden ser ordenadas como sigue: $T_D \leq T_L \leq T_M$. Para obtener una conjunción conmutativa, hacemos que $D_{21} = D_{12}$, en la tabla 1, se muestran las siete conjunciones paramétricas conmutativas no triviales obtenidas al considerar este método.

Tabla 1. Operadores de conjunción difusa paramétrica de suma monotónica p.

No.	Conjunción paramétrica p
i	DDDL
ii	DDDM
iii	D LLL
iv	DLLM
v	DMMM
vi	LLLM
vii	LMMM

En la figura 2.9 se muestra la gráfica de superficie de los operadores de conjunción paramétrica de suma monotónica, con parámetro p=80.

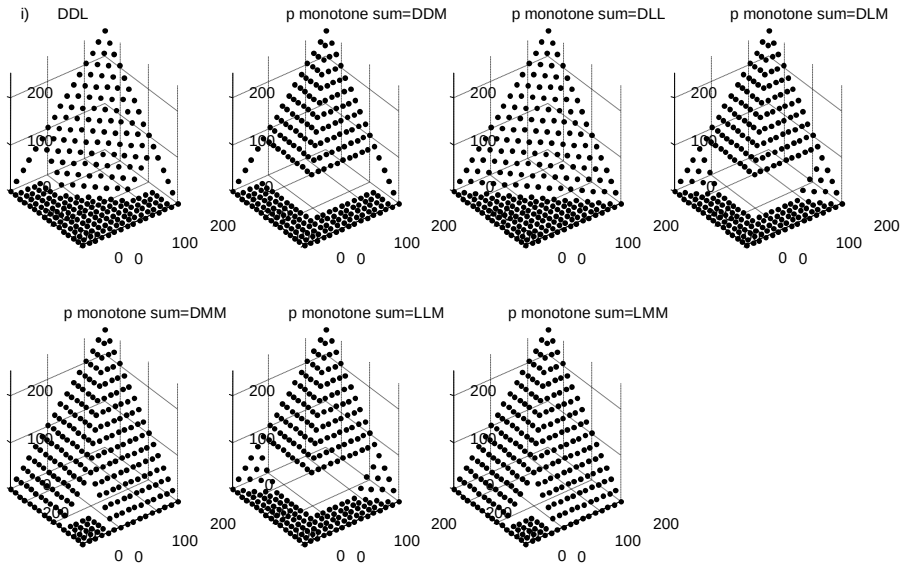


Figura 2.9. Operadores de conjunción difusas de suma monotonica p.

2.5.2 Conjunción difusa paramétrica de suma monotónica (p, 1-p)

La conjunción difusa paramétrica de suma monotónica (p, 1-p) puede ser definida seleccionando un conjunto de conjunciones difusas

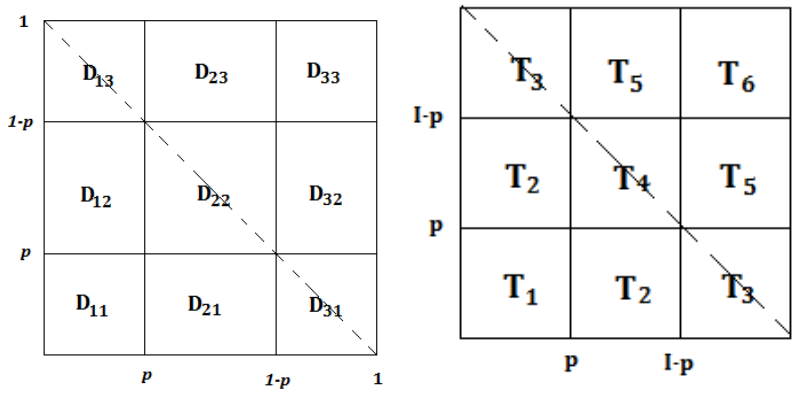
$$\{T_{11}, T_{21}, T_{31}, T_{12}, T_{22}, T_{32}, T_{13}, T_{23}, T_{33}\}$$

Ordenado como sigue: $T_{ij} \leq T_{st} \;\; si \;\; i \leq s \;\; y \;\; j \leq t$

Definimos la conjunción difusa T como :

$$T(x,y;p) = \begin{cases} T_{11}(x,y) & si & x \leq p, & y \leq p \\ T_{21}(x,y) & si & p < x \leq 1-p, & y \leq p \\ T_{31}(x,y) & si & x > 1-p, & y \leq p \\ T_{12}(x,y) & si & x \leq p, & p < y \leq 1-p \\ T_{22}(x,y) & si & p < x \leq 1-p, & p < y \leq 1-p \\ T_{32}(x,y) & si & x > 1-p, & p < y \leq 1-p \\ T_{13}(x,y) & si & x \leq p, & y > 1-p \\ T_{23}(x,y) & si & p < x \leq 1-p, & y > 1-p \\ T_{32}(x,y) & si & x > 1-p, & y > 1-p \end{cases}$$

En la figura 2.10a se muestra el espacio cartesiano para un operador de conjunción divido por nueve t-nomas, sin propiedad de conmutatividad.



a) b)

Figura 2.10. Partición de *L x L* en secciones $D_{ij} = X_i \; x \; X_j$ definidos por el parámetro I-p .

Tabla 2. Combinaciones posibles de conjunción paramétrica I-p.

Núm	Nombre de Conjunción	Operadores Parciales	Núm	Nombre de Conjunción	Operadores Parciales
1	DDDDDL	i	17	DLLLMM	
2	DDDDDM	ii	18	DLLMMM	iv
3	DDDDLL		19	DLMMMM	
4	DDDDLM		20	DMMMMM	v
5	DDDDMM		21	LLLLLM	vi
6	DDDLLL	i	22	LLLLMM	
7	DDDLLM		23	LLLMMM	vi
8	DDDLMM		24	LLMMMM	
9	DDDMMM	ii	25	LMMMMM	vii
10	DDLLLL		26	DDLDLL	iii
11	DDLLLM		27	DDLDLL	iv
12	DDLLMM		28	DDLDMM	
13	DDLMMM		29	DDMDMM	v
14	DDMMMM		30	DDMLMM	
15	DLLLLL	iii	31	DLMLMM	
16	DLLLLM		32	LLMLMM	vii

Para obtener operadores de conjunción conmutativa, hacemos que $T_1 = D_{11}$, $T_2 = D_{21} = D_{12}, T_3 = D_{31} = D_{13}, T_4 = D_{22}\ T_5 = D_{32} = D_{23}, T_6 = D_{33}$. En la figura 2.10b se muestra que el espacio está conformado por 6 t-normas, donde el conjunto de operadores paramétricos I-p deben cumplir el siguiente orden: $T_1 \leq T_2 \leq T_3 \leq T_4 \leq T_5 \leq T_6$ o $T_1 \leq T_2 \leq T_4 \leq T_3 \leq T_5 \leq T_6$. La tabla 2 muestra las 32 posibles combinaciones de los operadores paramétrico de suma monotónica I-p

Si se observa el comportamiento de algunos de los operadores de la tabla 2, son casos parciales de la tabla 1, es decir, del método de suma monotónica p.

La columna 3 de la tabla 2, indica la relación de parcialidad de los operadores, la cual se muestra gráficamente en la figura 2.11. Al eliminar los operadores parciales de la tabla , obtenemos la tabla 3 que lista un total de 16 posibles operadores

Tabla 3. a) Operadores de conjunción sin considerar equivalencia. b) Operadores de conjunción difusa de suma monotonica I-p.

N.	Nombre de Conjunción	Operadores equivalentes
1	DDDDLL	
2	DDDDLM	
3	DDDDMM	
4	DDDLLM	
5	DDDLMM	
6	DDLLLL	Lukasiewicz
7	DDLLLM	a
8	DDLLMM	b
9	DDLMMM	
10	DDMMMM	c
11	DLLLLM	a
12	DLLLMM	b
13	DLMMMM	c
14	LLLLMM	b
15	LLMMMM	c
16	DDLDMM	
17	DDMLMM	d
18	DLMLMM	d

N.	Nombre de Conjunción
1	DDDDLL
2	DDDDLM
3	DDDDMM
4	DDDLLM
5	DDDLMM
6	DDLLLM
7	DDLLMM
8	DDLMMM
9	DDMMMM
10	DDLDMM
11	DDMLMM

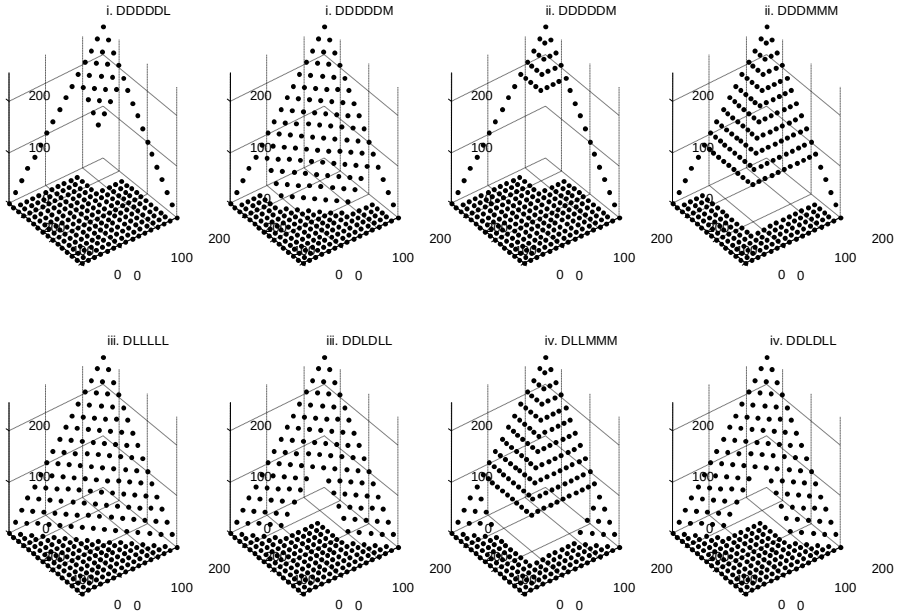


Figura 2.11. Operadores parciales de suma monotonica I-p.

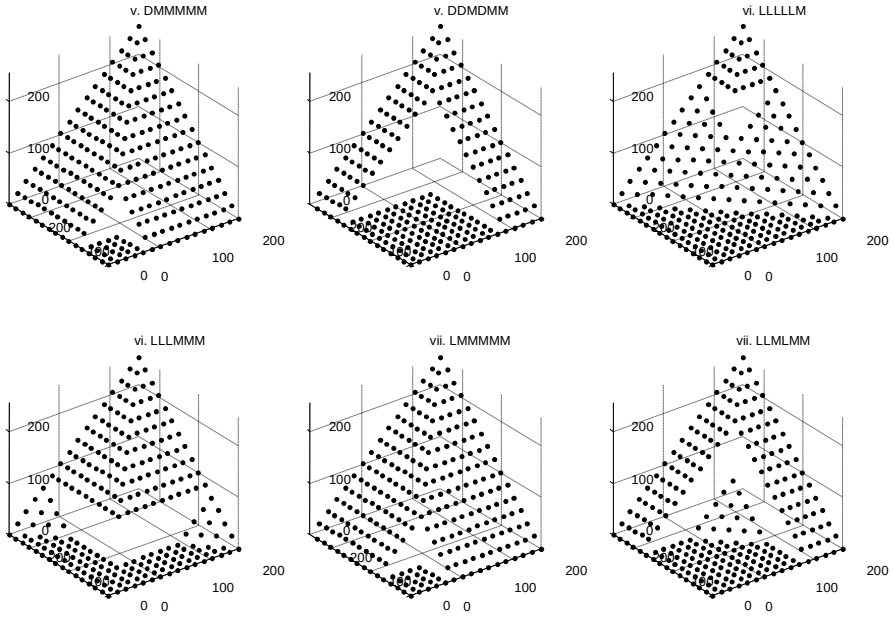


Figura 2. 11. Operadores parciales de suma monotonica I-p.

Ahora solo queda considerar que $D = L = 0$, en T_1 y T_2. La columna 3 de la tabla 3a, indica los operadores equivalentes al consederar esta equivalencia. El operador 6 de la tabla 3a es un caso particular, ya que este operador, es equivalente al operador

Lukasiewicz. Por último la tabla 3.b muestra los operadores paramétricos de suma monotónica I-p, que no tienen información repetida. La figura 2.12 muestra las gráficas de estos operadores

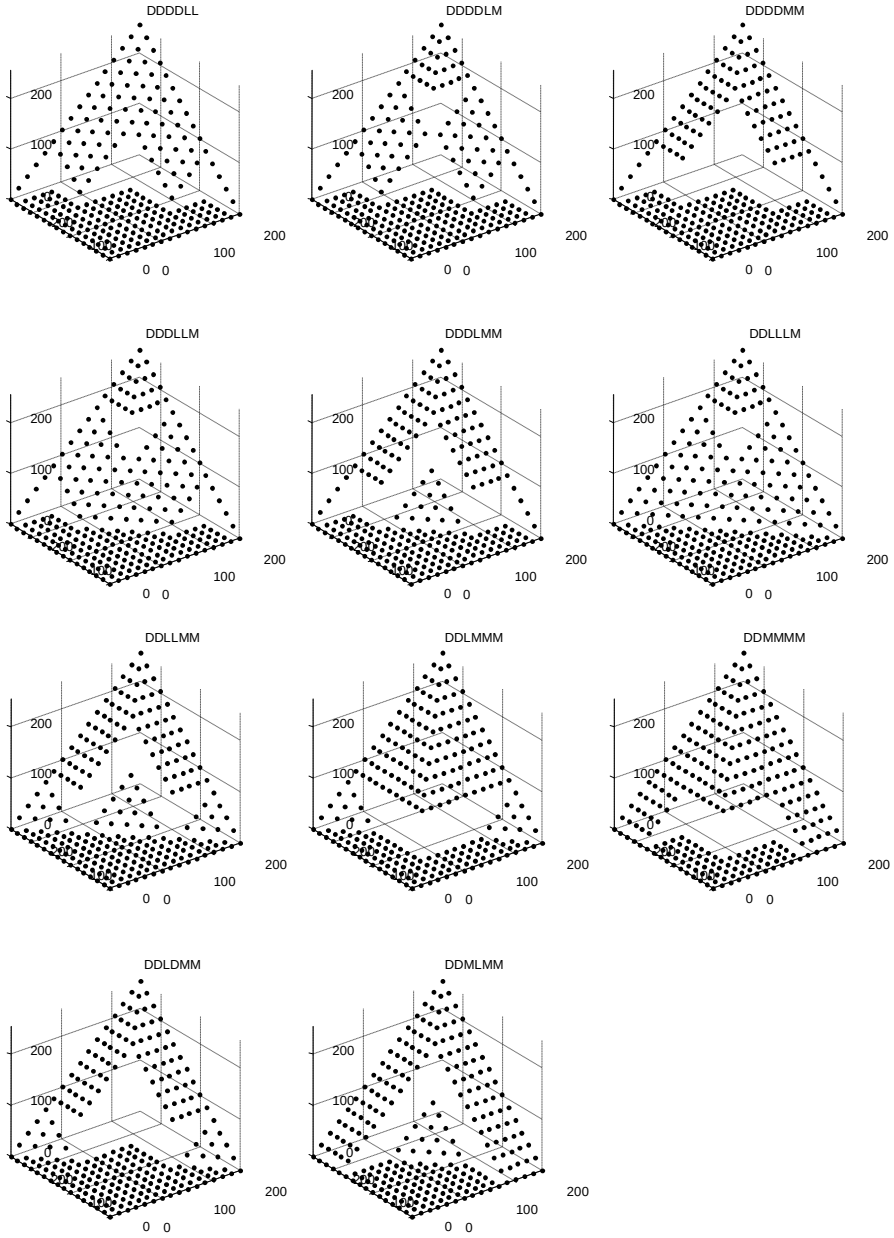


Figura 2.12. Operadores paramétricos de suma monotónica I-p

2.6 Lógica Difusa.

Tal como en la teoría de conjuntos tradicional, en la teoría de conjuntos difusos es necesario encontrar un medio para ampliar el dominio de una función, por ejemplo, dado un conjunto difuso A y una función f(.) , entonces como conocer el valor de la función f(A), a esto se le conoce como principio de extensión.

Principio de extensión

Sea f una función definida como:

$$f : U \rightarrow V$$

Donde $U\ y\ V$ son el dominio y el rango (codominio) respectivamente.

Se define un conjunto difuso $A \subseteq U$ como

$$A = \left\{\frac{\mu_1}{u_1} + \frac{\mu_2}{u_2} + \cdots + \frac{\mu_n}{u_n}\right\}$$

El principio de extensión afirma que la función f(A) es un conjunto difuso definido como sigue:

$$B = f(A) = \left\{\frac{\mu_1}{f(u_1)} + \frac{\mu_2}{f(u_{2)}} + \cdots + \frac{\mu_n}{f(u_n)}\right\}$$

2.6.1 Predicados Lógicos

Si un predicado lógico proposicional P, es un enunciado lingüístico contenido en un universo de proposiciones que pueden ser complemente falsas o verdaderas. El valor de verdad de la proposición P, puede asignársele un valor de verdad binario, llamado T(P), de la misma manera que un elemento en un conjunto .

Una proposición lógica difusa T(P) tiene un valor de verdad sobre el intervalo cerrado [0,1]. El valor de verdad de la proposición P esta dado por:

$$T(P) = \mu_A(x) \quad \text{donde} \quad 0 \leq \mu_A \leq 1$$

Por lo que el grado de verdad de $P:\ x\ \in A$ es el grado de membresía de x en A. Los conectivos lógicos de negación, disyunción, conjunción e implicación son definidos de la misma manera que en los operadores de la teoría de conjuntos

Negación lógica.

Si el predicado T(P)

$$T(P) = \mu_A(x)\ \ donde\ \ 0 \leq \mu_A \leq 1$$

T(P) negado se determina como:

$$T(\bar{P}) = 1 - T(P)$$

Disyunción. Sean P yQ dos proposiciones

$$P \vee Q = x \in A\ o\ B$$

Por lo tanto

$$\mathrm{T}(P \vee Q) = \max\left(T(P), T(Q)\right)$$

Conjunción

$$P \wedge Q = x \in A\ y\ B$$

Por lo tanto

$$\mathrm{T}(P \wedge Q) = \min\left(T(P), T(Q)\right)$$

Implicación

$$P \rightarrow Q = \ x\ esta\ en\ \ A,\ \ entonces\ \ x\ esta\ \ B$$

Por lo tanto

$$T(P \rightarrow Q) = T(\bar{P} \vee Q) = \max\left(T(\bar{P}), T(Q)\right)$$

Una implicación de lógica difusa deberá resultar en una regla difusa

$$T(P \rightarrow Q) = Si\ x\ es\ A, entonces\ y\ es\ B$$

Que equivale a la siguiente relación difusa.

$$R = (A\ x\ B) \cup (\bar{A}\ \ x\ Y)$$

Con un función de membresía

$$\mu_R = \max\{(\mu_A(x) \wedge \mu_B(y)), (1 - \mu_A(x))$$

2.6.2 Relaciones Difusas.

Las relaciones difusas son utilizadas en áreas tales como el control difuso y la toma de decisiones. Las relaciones difusas binarias son conjuntos difusos en el espacio producto X x Y, donde se mapea cada elemento en X x Y un grado de membresía entre 0 y 1. Particularmente las relaciones difusas unarias son conjuntos difusos con una función de membresía de una dimensión, de manera análoga, una relación difusa binaria es un conjunto difuso con función de membresía de dos dimensiones, y así se pueden generalizar las relaciones de orden n.

Relaciones difusas binarias.

Si X y Y son dos universos de discursos, entonces:

$$R = \{ ((x,y), \mu_R(x,y)) \mid (x,y) \in X \, x \, Y \}$$

2.6.3 Composición

Sean $R_1 \, y \, R_2$ dos relaciones difussa definidas sobre X x Y y Y x Z, respetivamente. La composición max_min de $R_1 \, y \, R_2$ es un conjunto difuso definido por:

$$R_1 \circ R_2 = \left\{ \left[(x,z), \max_y \min \left(\mu_{R_1}(x,y), \mu_{R_2}(y,z) \right) \right] \mid x \in X, y \in Y, z \in Z \right\}$$

Equivalentemente

$$\mu_{R_1 \circ R_2}(x,z) = \max_y \min[\mu_{R_1}(x,y), \mu_{R_2}(y,z)]$$

$$\mu_{R_1 \circ R_2}(x,z) = \vee_y [\, \mu_{R_1}(x,y) \wedge \mu_{R_2}(y,z)]$$

2.6.4 Variables Lingüísticas

Una variable lingüística, es una variable difusa que es caracterizada por un quinteta (x,T(x),X,G,M) donde, x es el nombre de la variable, T(x) es el conjunto de términos difusos o valores difusos, X es el universo de discurso, G es la regla sintáctica que

genera los términos en T(x), y M es la regla semántica, la cual se asocia con cada valor lingüístico A, con un conjunto difuso M(A).

2.6.5 Reglas difusas Si-Entonces.

Una regla difusas si-entonces (también conocida como regla difusa, implicación difusa o proposición condicional difusa) asume la forma

$$Si\ x\ es\ A\ entonces\ y\ es\ B$$

donde A y B son valores lingüísticos definidos por conjuntos difusos en los universos de discurso X y Y, respectivamente. A menudo a la parte de la regla "x es A " se llama el antecedente o premisa, mientras que a "y es B" se llama la consecuente o conclusión.

Antes de que podamos emplear reglas difusas *si-entonces* para modelar y analizar un sistema, en primer lugar tenemos que formalizar lo que se entiende por la expresión:

"si x es A entonces y es B",

que es también representado como A → B. La expresión describe una relación entre dos variables x e y, lo que sugiere que una regla difusa si-entonces se define como una relación binaria difusa R en el espacio producto X x Y. En términos generales, hay dos maneras de interpretar una regla difusa A → B.

1. A acoplado con B

   Entonces R se puede expresar como

$$R = A \rightarrow B = A\ x\ B = \int_{X\ x\ Y} \frac{\mu_A(x)\ \tilde{*}\ \mu_B(y)}{(x,y)}$$

   Donde $\tilde{*}$ es un operador T-norma

2. A implica B, donde la regla puede representarse por cuatro formas.
   a) Implicación material

$$R = A \rightarrow B = \bar{A} \cup B$$

b) Calculo proposicional

$$R = A \rightarrow B = \bar{A} \cup (A \cap B)$$

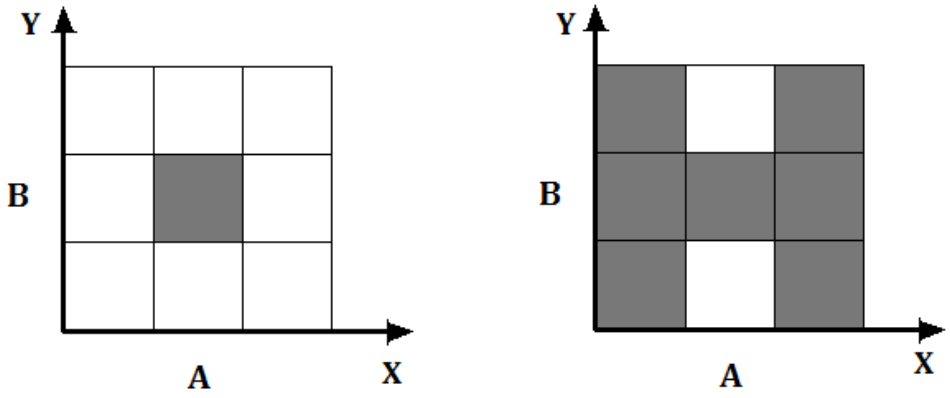
c) Calculo proposicional extendido

$$R = A \rightarrow B = \overline{(A} \cap \bar{B}) \cup B$$

d) Generalización mudus ponens

$$\mu_R(x, y) = sup\,\{(c \mid \mu_A(x) \tilde{*} \mu_B(y) \quad y \quad 0 \leq c \leq 1)$$

Estas interpretaciones de la implicación difusa son mostradas gráficamente en la figura 2.13.



a) **b)**

Figura 2.13. Interpretaciones de la implicación difusa. a) A Acoplado con B. b) A implica B

2.7 Razonamiento difuso

EL razonamiento difuso, también conocido como el razonamiento aproximado, es un procedimiento de inferencia de conclusiones que se deriva de un conjunto reglas difusas si-entonces y los hechos conocidos.

Regla de composición de inferencia

La regla de composición de inferencia desempeña un papel fundamental en el razonamiento difuso. La regla composicional se fundamenta sus principios en la composición de funciones. Por lo que se explicará este concepto a continuación.

Suponga una función y=f(x), que regula una relación entre x y y, si x =a , entonces se puede inferir que y = f(a) =b. Una generalización de este proceso determina que si a es un intervalo, entonces el valor de f(x) es un intervalo b = y = f(a), lo cual se demuestra haciendo una extensión cilíndrica de a hasta interceptar con la curva f(x) y posteriormente se realiza una proyección sobre el eje y, que determina el valor del b, como se muestra en la figura 2.14.

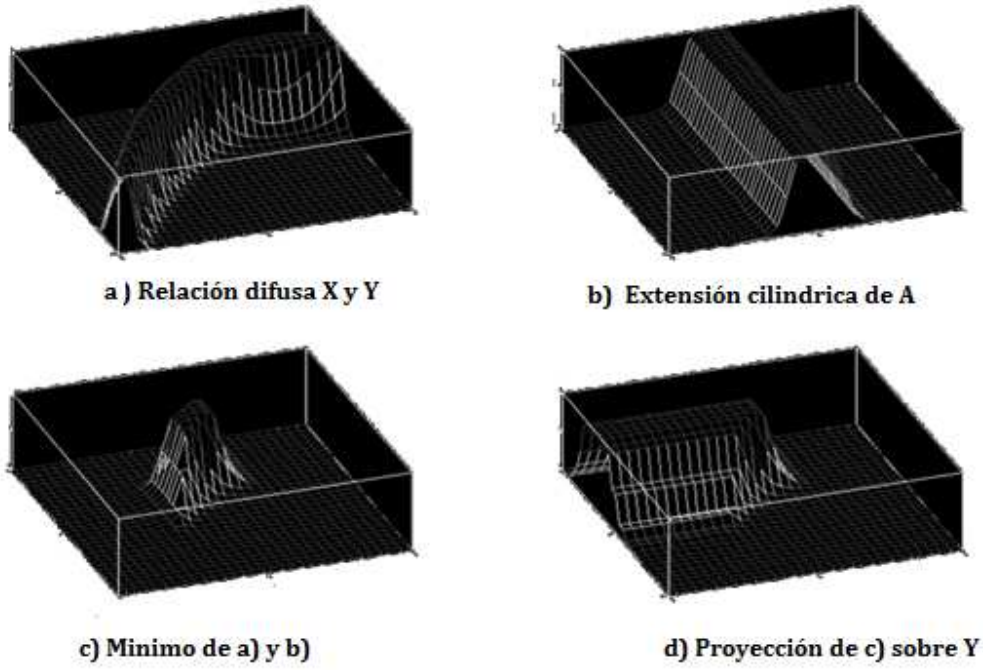


Figura 2. 14. Composición difusa.

De la misma forma se puede realizar este proceso y ser mapeado a lógica difusa, Si F es una relación difusa sobre X x Y y A es un conjunto difuso en X, para encontrar el valor del conjunto difuso generado al evaluar la relación difusa, con A como entrada, nuevamente se construye una extensión cilíndrica c(A) con base A, la intersección de c(A) y F y posteriormente hacemos una proyección de la intersección sobre el eje Y, la cual nos genera el valor del conjunto B

Matemáticamente se puede expresar la regla composicional de inferencia la cual desempeña un papel clave en el razonamiento difuso.

$$\mu_B(y) = \max_x \min[\mu_A(x), \mu_F(x,y)]$$

$$\mu_B(y) = \vee_x[\, \mu_A(x) \wedge \mu_F(x,y)]$$

Por lo que el conjunto B inferido puede ser representado como

$$B = A \circ F$$

Usando la regla composicional de inferencia, se puede formalizar un procedimiento de inferencia a partir de un conjunto de reglas si, entonces, llamado razonamiento aproximado o difuso.

Regla de inferencia difusa "si-entonces"

La regla de inferencia básica modus ponens, se utiliza para inferir el valor de verdad de la proposición B, a partir del valor de verdad de A y la proposición $A \rightarrow B$.

Debido que la mayor parte del razonamiento humano es aproximado, la regla de inferencia modus ponens es empleada con un manejo aproximado. Por ejemplo si se tiene la regla de implicación, *"si la ropa está sucia, entonces agregar jabón"*, y si tenemos el hecho, *"la ropa esta poco sucia"*, entonces se puede inferir la proposición *"agregar poco jabon"*, esto se escribe como

Premisa 1 (hecho):	x es A ',
Premisa 2 (regla):	si x es A entonces y es B,
consecuencia (conclusión):	y es B ',

A el procedimiento de inferencia anterior se le llama razonamiento aproximado o razonamiento difuso, y también puede llámesele modus ponens generalizado (GMP).

Definición. Razonamiento aproximado (razonamiento difuso)

Si A, A' y B son conjuntos difusos en X, X y Y respetivamente. La implicación $A \rightarrow B$ puede ser expresada como una relación difusa R sobre X x Y. Entonces el

conjunto difuso B' inducido por "x es A'" y la regla difusa *"if x es A entonces y es B"* es definido por

$$\mu_{B'}(y) = \max_{x} \min[\mu_{A'}(x), \mu_R(x,y)]$$

$$\mu_{B'}(y) = \vee_x[\, \mu_{A'}(x) \wedge \mu_R(x,y)]$$

O equivalentemente

$$B' = A' \circ R = A' \circ (A \rightarrow B)$$

Inferencia difusa de una regla con antecedente simple

Es el caso donde se tiene una regla difusa de la forma

Si x es A entonces y es B

Como sea mencionado anteriormente la regla de implicación puede ser sustituida por una relación difusa. Y para el hecho *x es A'*, tenemo que la conclusión del conjunto B' se determina como sigue

$$\mu_{B'}(y) = \vee_x[\, \mu_{A'}(x) \wedge \mu_R(x,y)]$$

Que puede ser simplificado de la siguiente manera

$$\mu_{B'}(y) = \vee_x\big[\, \mu_{A'}(x) \wedge \mu_A(x) \wedge \mu_B(y)\big]$$

$$\mu_{B'}(y) = \vee_x\big[\, \mu_{A'}(x) \wedge \mu_A(x)\big] \wedge \mu_B(y)$$

Si $w = \vee_x\big[\, \mu_{A'}(x) \wedge \mu_A(x)\big]$ y donde el valor $\vee_x$ es sustituido por el valor supremo del conjunto difuso resultante, de acuerdo al GMP

$$\mu_{B'}(y) = w \wedge \mu_B(y)$$

En la figura 2.15 se muestra gráficamente el proceso de la inferencia difusa de una regla con un antecedente simple, donde los conjuntos A y B pertenecen a la regla *Si x es A entonces y es B* y el conjunto *A'*, es el hecho de entrada, que genera al conjunto consecuente *B'*, que está determinado por área gris.

Inferencia difusa de una regla con antecedente múltiple

Una regla difusa si_entonces con dos antecedentes es usualmente escrita como *"si x es A y y es B, entonces z es C"*. El razonamiento difuso usando la metodología modus ponens generalizado.

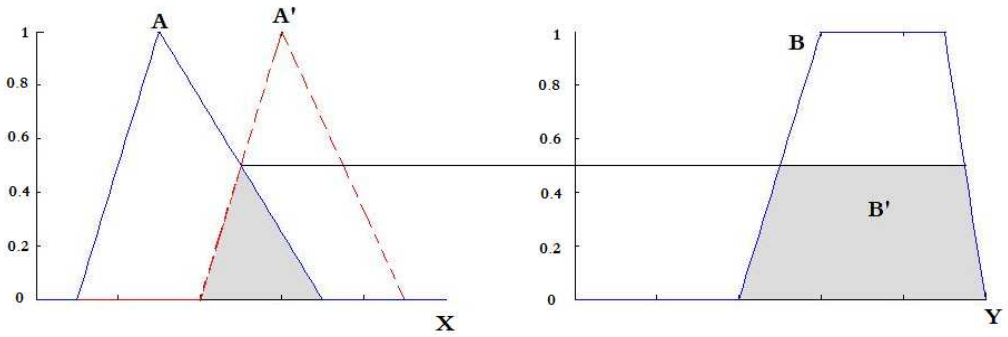


Figura 2.15. Razonamiento difuso de una regla, con antecedente simple.

Premisa 1 (hecho): x es A' y y es B',

Premisa 2 (regla): si x es A y y es B' entonces z es C,

Consecuencia (conclusión): z es C'

El antecedente de la regla difusa, de la premisa 2 puede ser escrito como el conjunto $A \wedge B$, por lo que la regla difusa puede ser expresada de la siguiente manera.

$$A \wedge B \rightarrow C$$

Desarrollando el mismo procedimiento que en el caso de regla difusa con antecedente simple $A \rightarrow B$ tenemos que la regla difusa $A \wedge B \rightarrow C$ y el hecho de $A' \wedge B'$, de acuerdo a GMP generan al conjunto C' que puede ser expresado como:

$$C' = (A' \wedge B') \circ R = (A' \wedge B') \circ (\mathrm{A} \wedge \mathrm{B} \rightarrow C)$$

Donde la función de membresía de la relación $R = \mathrm{A} \wedge \mathrm{B} \rightarrow C$ puede obtenerse de la siguiente manera.

$$R = \mathrm{A} \wedge \mathrm{B} \rightarrow C = \mathrm{A} \wedge \mathrm{B} \wedge C = \int_{X\ x\ Y\ x\ Z} \frac{\mu_A(x) \wedge \mu_B(y) \wedge \mu_C(z)}{(x, y, z)}$$

Por lo que la función de pertenencia del conjunto difuso C' puede ser expresada de la siguiente manera.

$$\mu_{C'}(z) = \vee_{x,y}\left[\ [\mu_{A'}(x) \wedge \mu_{B'}(y)] \wedge [\mu_A(x) \wedge \mu_B(y) \wedge \mu_C(z)]\ \right]$$

$$\mu_{C'}(z) = \vee_{x,y}\left[\ \mu_{A'}(x) \wedge \mu_{B'}(y) \wedge \mu_A(x) \wedge \mu_B(y) \wedge \mu_C(z)\ \right]$$

$$\mu_{C'}(z) = \vee_{x}\left[\ \mu_{A'}(x) \wedge \mu_A(x)\ \right] \wedge \vee_{x}\ [\mu_{B'}(y) \wedge \mu_B(y)] \wedge \mu_C(z)$$

Si decimos que $w_1 = \vee_x \left[\ \mu_{A'}(x) \wedge \mu_A(x)\ \right]$ y $w_2 = \vee_x\ [\mu_{B'}(y) \wedge \mu_B(y)]$ entonces

$$\mu_{C'}(z) = w_1 \wedge w_2 \wedge \mu_C(z)$$

La figura 2.16 muestra el proceso mencionado, donde los conjuntos difusos A y B y C, describen a la regla *si x es A y y es B, entonces z es C,* y con entradas (hechos) A' y B', que generan los pesos W_1 *y* W_2 respectivamente, que finalmente el operador de conjunción (mínimo), determina el valor del conjunto difuso C'.

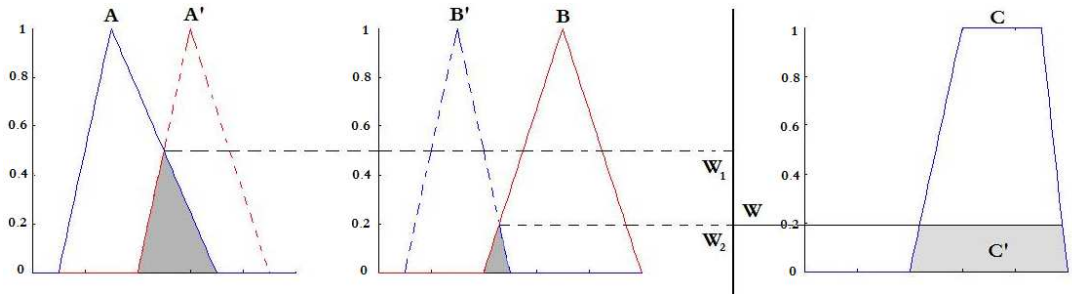


Figura 2. 16. Razonamiento difuso de una regla, con antecedente múltiple.

Inferencia difusa con múltiples reglas y antecedente múltiple

La interpretación de múltiples reglas es usualmente tomada como la unión de las relaciones difusas correspondientes a las reglas difusas. De acuerdo al GMP tenemos

Premisa 1 (hecho): x es A' y y es B',

Premisa 2 (regla 1): $si\ \ x\ es\ A_1\ \ y\ \ y\ es\ B_1\ entonces\ z\ es\ C_1$

Premisa 3 (regla 2): $si\ \ x\ es\ A_2\ \ y\ \ y\ es\ B_2\ entonces\ z\ es\ C_2$

Consecuencia (conclusión): z es C'

Si decimos que las reglas de inferencia difusa, pueden ser expresadas como las siguientes relaciones, $R_1 = (A_1\ x\ B_1 \rightarrow C_1)$ y $R_2 = (A_2\ x\ B_2 \rightarrow C_2)$. Considerando que el operador de composición max-min es distributivo sobre el operador de ∪, tenemos:

$$C' = (A' x\ B') \circ (R_1\ \cup R_2)$$

$$C' = [(A' x\ B') \circ R_1]\ \cup [(A' x\ B') \circ R_2]$$

$$C' = C_1' \cup\ C_2'$$

Donde $C_1'\ y\ C_2'$ son los conjuntos de inferencia para las reglas difusas 1 y 2 respectivamente y el operador ∪ puede ser calculado con cualquier s-norma. En la figura 2.17 se muestra la gráfica de la implicación difusa con múltiples reglas de antecedentes múltiples.

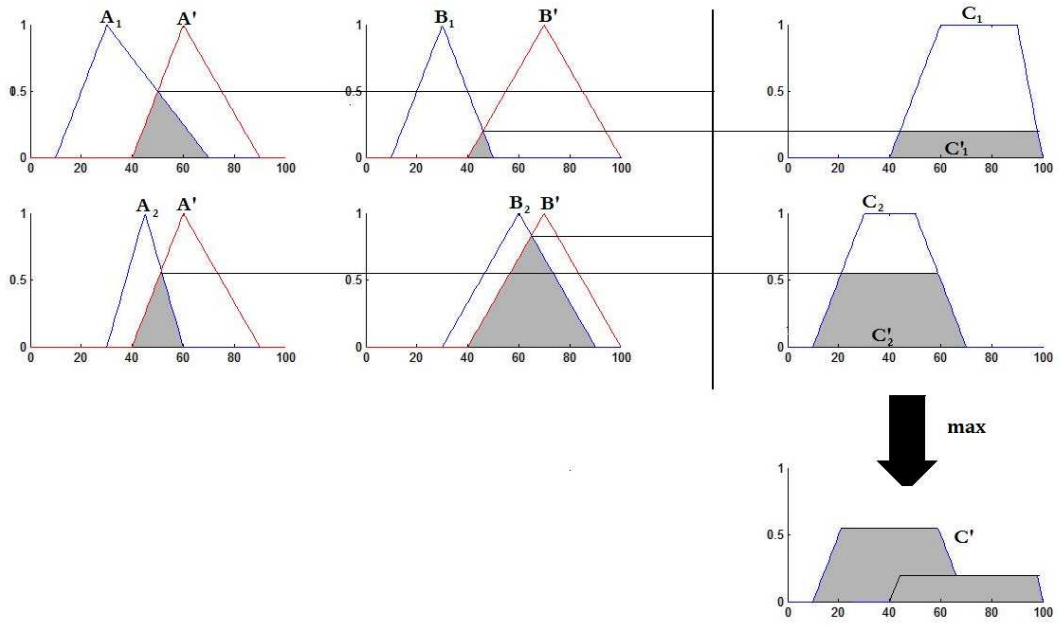


Figura 2. 17. Razonamiento difuso con múltiple reglas de antecedentes múltiple.

2.8 Sistemas de inferencia difusa.

La estructura básica de los sistemas de inferencia difusa consiste en tres componentes: una Base de reglas en donde se definen las reglas difusas. Base de

datos o diccionario, donde se definen los conjuntos difusos y un mecanismo de razonamiento.

Un sistema difuso puede tener como entradas conjuntos difusos o entradas certeras (conjuntos singletons) y como salidas conjuntos difusos o salidas certeras. En la implementación de sistemas digitales es necesario que la salida sea un valor certero, por lo que es necesario implementar un método que extraiga un valor certero de un conjunto difuso, a este método se le conoce como defusificador en el caso de trabajar con un sistema difuso Mamdani

Cuando se tienen entradas y salidas certeras el sistema difuso implementa un mapeo no lineal del espacio de entrada al espacio de salida. Este mapeo es realizado de acuerdo al número de reglas difusas. Cada una de ellas describe un comportamiento local de mapeo. Donde regularmente el antecedente de la regla determina el espacio de la región de entrada y el consecuente determina el espacio de la región de salida

Existen tres tipos de sistemas de inferencia difusa que han sido utilizado ampliamente en muchas aplicaciones, la diferencia entre estos tres tipos de sistemas difusos está en la implementación del consecuente de las reglas difusas , por lo que los procesos de agregación y defusificación difieren uno del otro .

2.8.1 Modelo difuso Mamdani.

Utiliza el sistema de razonamiento visto anteriomente, en la figura 2.18 se muestra un sistema de inferencia difuso tipo Mamdani de dos reglas.

Como se muestra en la figura, el sistema difuso tiene como entradas conjuntos de entrada singletons (valores certeros) y como salida un conjunto difuso, para aplicaciones de control, es necesario, extraer un valor certero del conjunto.

Defusificación.

Este proceso se refiere a la forma de cómo extraer un valor certero de un conjunto difuso. En general hay cinco métodos para defuzificar un sistema difuso.

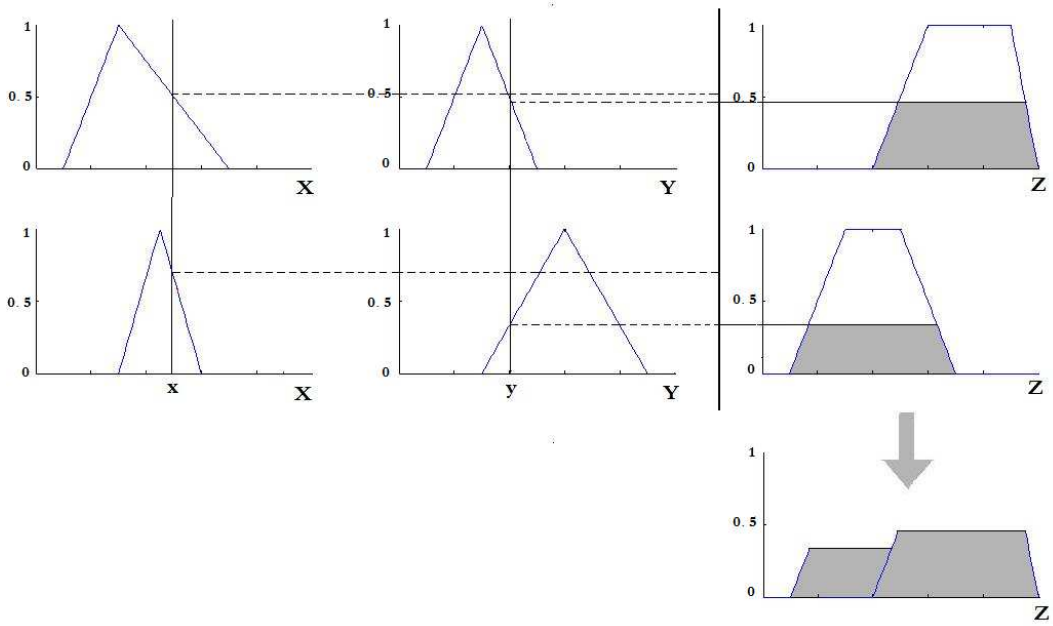


Figura 2.18. Inferencia difusa, método Mamdani

Centro del área.

$$\boldsymbol{z_{COA}} = \frac{\int_{\boldsymbol{Z}} \mu_{\mathrm{A}}(z) z dz}{\int_{\boldsymbol{Z}} \mu_{\mathrm{A}}(z) dz}$$

Donde $\mu_{\mathrm{C}}(z)$ es la función de membresía de la salida agregada, este método se basa en el cálculo del valor esperado de la distribución probabilística.

Bisección del área: $\boldsymbol{Z_{BOA}}$

El método de la bisección del área satisface:

$$\int_{\boldsymbol{a}}^{\boldsymbol{Z_{BOA}}} \mu_{\mathrm{A}}(z) dz = \int_{\boldsymbol{Z_{BOA}}}^{\boldsymbol{b}} \mu_{\mathrm{A}}(z) dz$$

Donde a = min{ z | z ∈ Z } y b = max {z | z ∈ Z}. E sto es, una línea vertical $z = z_{BOA}$ que divide en dos regiones de misma área al conjunto difuso.

Promedio de los máximos: Z_{MOM}

Obtiene el valor promedio de z de los valores de z donde $\mu_A(z)$ alcanza el valor máximo.

$$z_{MOM} = \frac{\int_{Z'} z dz}{\int_{Z'} dz}$$

Donde $Z' = \{z \mid \mu_A(z) = \mu^*\}$

Primer valor máximo: Z_{SOM}

Es el valor mínimo de z, donde $\mu_A(z) = \mu^*$

El último valor máximo: Z_{SOM}

Es el valor máximo de z, donde $\mu_A(z) = \mu^*$

La figura 2.19 muestra los 5 tipos de defusificación descritos anteriormente

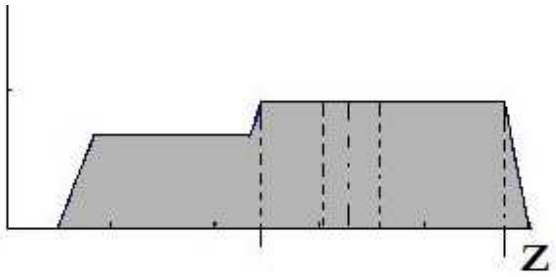


Figura 2.19. Métodos de defusificación.

El cálculo necesario para llevar acabo cualquiera de estas cinco operaciones de defuzzificación consume mucho tiempo. Por otra parte, las operaciones de defuzzificación no son fácilmente analizables matemáticamente, así que la mayoría de los estudios se basan en resultados experimentales.

Otras variantes

El operador de composición difusa max-min es solo una de las variantes del razonamiento difuso tipo mamdani, Sin embargo, teniendo en cuenta la eficiencia de cálculo o manipulación matemática, un sistema difuso en la práctica puede tener un mecanismo de razonamiento que utilice cualquier operador t-norma o s-norma por lo que en un sistema de inferencia difuso Mamdani, podemos asignar una función para cada una de las siguientes operadores en la regla difusa:

- **Operador AND** (por lo general de t-norma) para calcular la fuerza de disparo del una regla con antecedente AND.
- **Operador OR** (por lo general de s-norma) para calcular la fuerza de disparo de una regla con antecedente OR.
- **Operador de implicación** (por lo general de t-norma) para cálculo de la MF del consecuente de una regla basado en la regla de disparo.
- **Operador de agregación** (por lo general de s-norma) para calcular la agregación de la MF Del conjunto de salida
- **Operador de defusificación** para extraer un valor certero a partir de la función de membresía del conjunto de salida difuso.

Calculo computacional de defusificación para sistemas difusos tipo mamdani, De acuerdo a la composición de suma de productos, la salida de un sistema de inferencia difuso Mamdani con defuzzificación centroide es igual al promedio de los pesos de los centroides de las MF de los consecuentes, donde cada uno de los factores de peso es igual al producto de la fuerza de disparo y el área del consecuente.

Por lo que z puede determinarse como:

$$z = \frac{\sum_{i=1}^{n} w_i a_i z_i}{\sum_{i=1}^{n} w_i a_i}$$

Donde

$$a_i = \int_{\boldsymbol{Z}} \mu_{\mathrm{C_i}}(z) z dz \quad \text{y} \quad z_i = \int_{\boldsymbol{Z}} \mu_{\mathrm{C_i}}(z) dz$$

2.8.2 Modelo Sugeno.

El modelo difuso Sugeno (también conocido como modelo difuso TSK) fue propuesto por Takagi, Sugeno y Kang, en un esfuerzo para desarrollar un enfoque sistemático para la generación de reglas difusas a partir de un conjunto de datos de entrada y salida.

Una regla difusa típica en un modelo difuso Sugeno tiene la forma

si x es A y y es B entonces z = f (x, y)

Donde A y B son conjuntos difusos antecedentes, mientras que z = f (x, y) es una función en los reales en el consecuente. Por lo general, f (x, y) es un polinomio que está en función de las variables de entrada x e y. Si f(x,y) es igual a una constante el sistema difuso se conoce como modelo difuso Sugeno de orden cero y cuando el polinomio f(x,y) describe una ecuación de primer orden al sistema se le conoce como sistema difuso Sugeno de primer orden.

El valor de salida z se calcula con la suma de productos de las fuerzas de disparo por el valor z del la función consecuente, entre la sumaria de las fuerza de disparos de la base de reglas. En la siguiente ecuación se muestra el cálculo de z.

$$z = \frac{\sum_{i=1}^{n} w_i z_i}{\sum_{i=1}^{n} w_i}$$

A diferencia del modelo difuso Mamdani, el modelo difuso Sugeno no puede seguir estrictamente la regla de composición de inferencia en su mecanismo de razonamiento difuso. Esto plantea algunas dificultades cuando las entradas a un modelo difuso Sugeno son difusas, sin embargo este caso nunca se presenta en un sistema digital, ver figura 2.20.

Sin el tiempo de consumo de el proceso de defusificacion los modelos tipo sugeno son los candidatos para modelar sistemas difusos basados en datos.

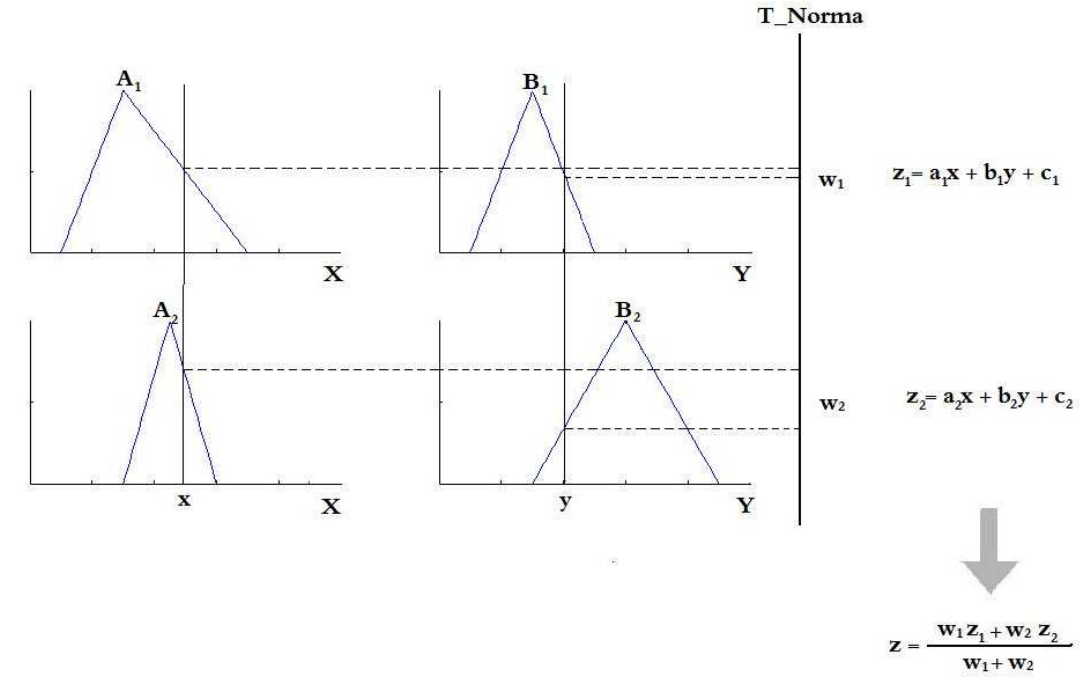


Figura 2.20. Inferencia difusa, método Sugeno.

2.8.3 Modelo Tsukamoto.

En los modelos de inferencia difusos tipo Tsukamoto, el consecuente de cada regla difusa *si-entonces* está representado por un conjunto difuso con una función de membresía monotónica, como se muestra en la Figura 2.21. Como consecuencia, la salida inferida de cada regla se define como un valor certero inducido por el valor de la fuerza de disparo de la regla, como se muestra en la figura. El valor de salida certero es calculado de la misma manera que en los sistemas difusos sugeno. El modelo Tsukamoto no sigue estrictamente el proceso de inferencia composicional difusa.

2.9 Reconfiguración Parcial de FPGAs

La reconfiguración parcial (PR) es una característica que permite diseñar múltiples módulos de un sistema utilizando los mismos recursos o fuentes físicas de un dispositivo en tiempo compartido. En un diseño PR los módulos se

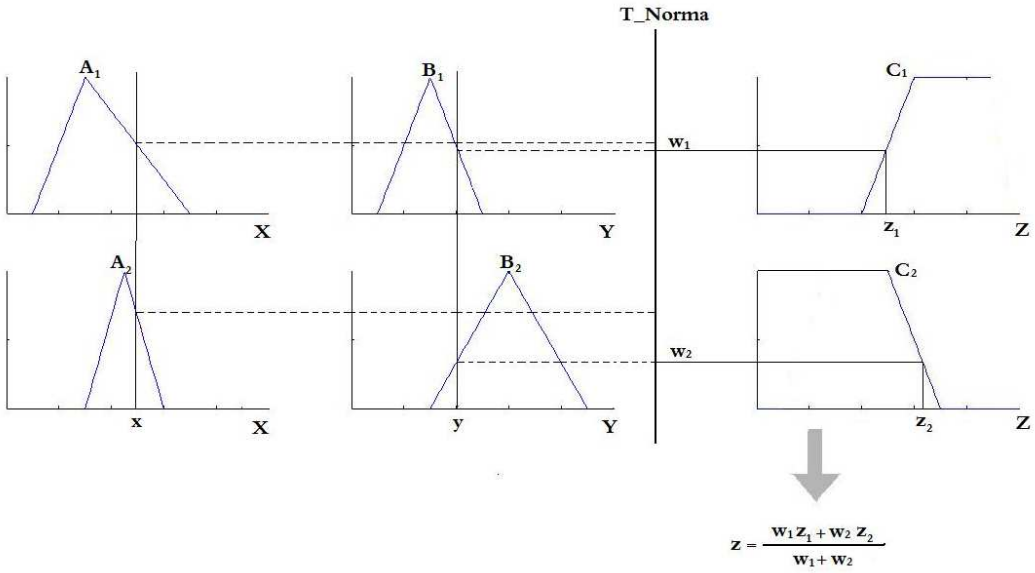


Figura 2.21. Inferencia difusa, método Tsukamoto

pueden intercambiar en tiempo de ejecución, incluso mientras el diseño base continua operando.

Mediante el uso de la reconfiguración parcial, los diseñadores pueden incrementar la funcionalidad de un FPGA, lo que permite que un sistema sea implementado con menos recursos o diseñar sistemas más compactos. En la figura 2.22 se muestra un sistema que tiene un módulo de comunicación de radio enlace, un módulo de enlace de video, otro módulo que comunica al sistema con un dispositivo periférico y tres regiones parcialmente reconfigurables PRRA, PRRB y PRRC, en donde cada región puede tener dos o más módulos con diferente función, que alternaran su implementación en el dispositivo.

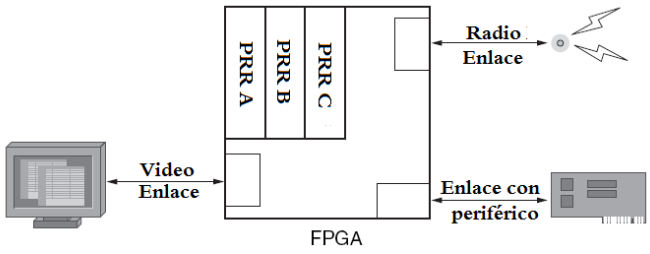


Figura 2.22. Reconfiguración parcial, con enlaces dinámicos.

La reconfiguración parcial es útil para sistemas con múltiples funciones que pueden compartir los mismos recursos del dispositivo FPGA. En tales sistemas, una parte del FPGA funciona continuamente mientras que otros sectores del FPGA se desactivan y son parcialmente reconfigurados para proporcionar nuevas funcionalidades.

Esta técnica es análoga a la administración de un microprocesador con el cambio de contexto entre procesos de software, excepto que en el caso de la reconfiguración parcial en FPGA se administra la funcionalidad de hardware. Permitiendo la configuración de un cierto número de *frames (área mínimas programables)* de la *FPGA* sin activar *PROG* (que borra la memoria de configuración), ni apagar y encender la *FPGA*.

Por lo tanto un dispositivo puede dividirse en dos regiones. Región Estática o no parcialmente reconfigurable: zona de la *FPGA* que no se modificar al reconfigurar el FPGA en tiempo de ejecución, y la región parcialmente reconfigurable que es la zona de la *FPGA* que va a ser reconfigurada. Existen diferentes formas de realizar una reconfiguración parcial en la tecnología de Xilinx por lo que se mencionan los tipos de reconfiguración parcial.

2.9.1 Reconfiguración parcial estática: Es un tipo de reconfiguración parcial del *FPGA* que detiene el funcionamiento u operación del FPGA, cuando se hace una modificación de una parte de la implementación del dispositivo.

2.9.2 Reconfiguración parcial dinámica (o activa): En este tipo de reconfiguración es posible reconfigurar determinado número de *frames* del *FPGA* sin parar el funcionamiento del resto de ella. Este es el método que se va a trabajar en esta tesis.

2.9.3 Auto-Reconfiguración parcial: Este tipo de reconfiguración se lleva a cabo con el microprocesador embebido del FPGA que se comunica con la memoria de

programación del *FPGA* y lo reconfigura. El microprocesador debe formar parte del área estática (aunque sus periféricos no tienen por qué).

La forma y tamaño de cada región parcialmente reconfigurable (PRR) es definida por el usuario a través de la restricción de área o rango (AREA_GROUP), cada PRR tiene al menos un módulo reconfigurable y usualmente múltiples módulos reconfigurables (PRM) que pueden ser cargados dentro de PRR. Cada PRM es diseñado e implementado individualmente usando las herramientas de diseño del E*arly Access Partial Reconfiguration.*

La figura 2.23 se muestra una región parcialmente reconfigurable, PRRA, en donde se le pueden cargar tres módulos parcialmente reconfigurables (PRM), A1, A2 y A3. Cada uno de estos PRM contiene diferente funcionamiento lógico, además se tiene una región estática

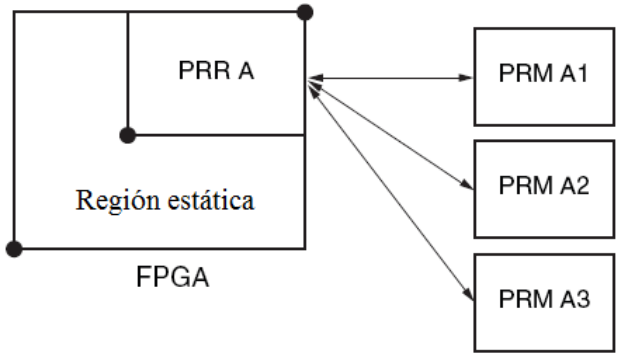


Figura 2.23. Región de reconfiguración parcial con tres módulos reconfigurables

2.10. Flujo de Reconfiguración

Xilinx ha desarrollado hasta la fecha cuatro flujos de diseño de reconfiguración parcial, sin embargo hasta la fecha sólo se han distribuido parches para este tipo de investigaciones. El presente trabajo utiliza la tercera versión del flujo de diseño, el cual es llamado *Early Access Partial Reconfiguration (EA_PR),* en Julio de 2010

salió la última versión de la reconfiguración parcial, sin embargo, en este trabajo se le da continuidad a un trabajo de reconfiguración de la institución donde se trabajó con el flujo EA_PR, además que es la versión disponible en el instituto.

A continuación se describen los aspectos más importantes a considerar en el flujo de diseño EA_PR.

El flujo de diseño EA_PR requiere de varios pasos adicionales que no son requeridos en el diseño tradicional de sistemas en FPGA. El flujo EA_PR involucra la implementación de diseño estático y PRR separadamente. Se consideran 7 pasos en el flujo de diseño EA PR que se enuncian a continuación

1. Descripción del diseño; código y síntesis.
2. Establecimiento de restricciones de tiempo, definición de pines de E/S
3. Implementación del diseño no reconfigurable.
4. Realizar análisis de tiempo y localización (ubicación de recursos)
5. Implementación de los módulos *top-level* y la parte estática
6. Implementación de los módulos reconfigurables
7. Unión y obtención de *bitstreams*

Pasos de la reconfiguración parcial

Diseño de la descripción y síntesis HDL

El flujo de diseño EA_PR, soporta descripción de diseño para lenguajes VHDL y Verilog, y la síntesis puede realizarse con cualquier herramienta actual de síntesis del ISE software.

La reconfiguración parcial requiere un enfoque de diseño jerárquico que debe realizarse estrictamente según los pasos que se están describiendo, durante el proceso de codificación de HDL (Figura 2.24). Es recomendable que toda la lógica global, tal como bloques E/S, los relojes globales, y DCM, se ubiquen en el módulo

top_level. En casos como una instanciación de un bloque IP, los puertos de E/S, relojes globales y DCM son embebidos dentro del IP. Es posible que un diseño parcialmente reconfigurable contenga E/S en un PRM, sin embargo medidas adicionales son necesarias para garantizar que el módulo estático de nivel superior no cree también controladores para la E/S integrados en el PRM. Otros tipos de recursos, como BUFGs y DCM no se permiten dentro del PRM.

Los múltiples módulos estáticos y reconfigurables puede ser instanciados como caja negra, dentro del módulo de diseño top_level, esto debe ser realizado con una directiva KEEP_HIERARCHY. La síntesis debe ser hecha usando una metodología bottom_up para mantener las fronteras de los módulos. Por otra parte, la inserción de E/S deben ser desactivado para evitar buffer de E/S sean insertadas dentro de los módulos de bajo nivel.

Recomendaciones de diseño para el diseño del Top_level

Sólo debe de contener instanciaciones de caja negra de los módulos inferiores. La descripción del top_level sólo debe contener:

a) Instanciaciones de puertos de E/S
b) Instanciaciones de módulos estáticos
c) Instanciaciones de PRM
d) Declaración de señales
e) Lógica global tal como BUFG y DCM

La comunicación de los módulos estáticos con los reconfigurables debe realizarse a través de módulos bus macros (todas las señales de reloj global que utiliza BUFG), en caso de señales no globales.

Reglas de diseño HDL para la implementación de módulos estáticos

Se recomienda que las instanciaciones de las primitivas del reloj (DCM y BUFG) y los bus macros sean ubicados en el top_level. En la herramienta de diseño PlanAhead

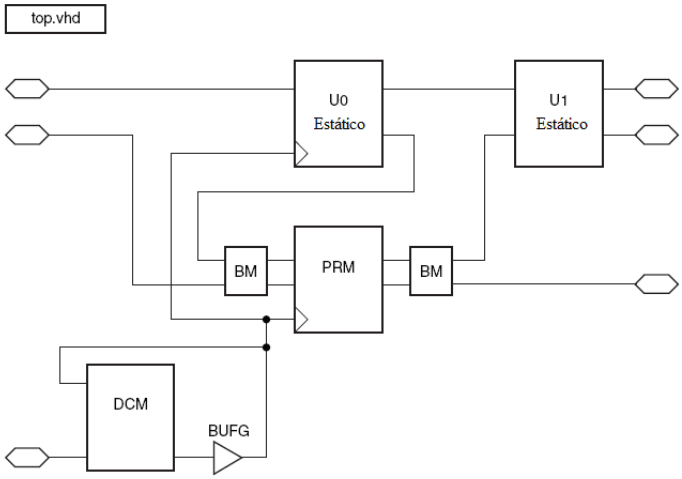


Figura 2.24. Flujo de diseño de la reconfiguración parcial

está disponible la ubicación de los bus macros a diferentes niveles de diseño estático, sin embargo, esta característica no está disponible para DCM y BUFG.

Reglas de diseño HDL para módulos PR

Primitivas de reloj tales como DCM y BUFG no pueden ubicadas en el PRM. Una excepción es la primitiva de reloj regional BUFR, sin embargo debe de seguir una metodología estricta de diseño que se describe en [ug208].

Todos los PRM que vallan a ser ubicados sobre un PRR deberán tener el mismo nombre de módulo y archivo y mismo número y tipo tipos puertos de entrada salida, para permitir que estos sean enlazados a la instanciación top_level. Ver figura 2.25

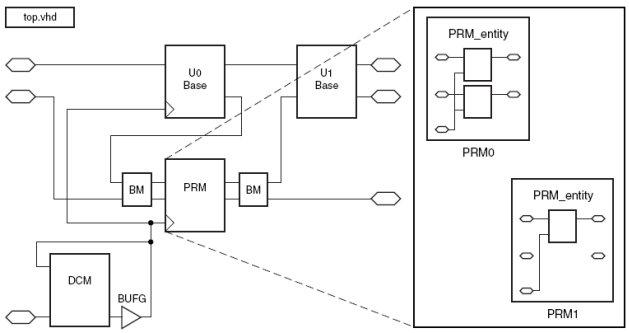


Figura 2.25. Restricción de módulos reconfigurables.

2 . Establecimiento de las restricciones de diseño.

En este paso se diseña la ubicación física de la implementación los módulos que conforma el diseño del sistema, estableciendo las restricciones de ubicación para el place and route, además de establecer las restricciones habituales de toda implementación en FPGA, tales como restricciones de tiempo (periodo) y de los puertos de entrada-salida, esto se realiza en el archivo ***.ucf.***

Los diseño de los PRR deben de tener restricciones del tipo AREA_GROUP, AREA_GROUP RANGE, MODE and LOC

Restricción AREA_GROUP.

Es una restricción de agrupamiento, que asocia elementos de diseño lógico con una etiqueta o grupo en particular. Además esta restricción permite delimitar las áreas reconfigurables de las estáticas. Una restricción AREA_GROUP debe ser definida para cada una de los PRR y no son necesarias para definir regiones estáticas. Un ejemplo es:

```
INST "prm_a" AREA_GROUP = "AG_PRegion1";
```

Restricciones AREA_GROUP RANGE

Establece la ubicación y la forma de la región reconfigurable. La declaración básica de esta restricción es definir un conjunto de slices que serán parte de la región PR. Los slices contienen elementos lógicos como LUT y FF. Si los PRM también utilizan BRAM, I/O y otros tipos de elementos lógicos, entonces se deben de declarar una restricción que defina el conjunto de elementos que se desean incrustar en la PRR. Los requerimientos al definir una restricción AREA_GROUP son las siguientes:

Una restricción AREA_GROUP RANGE es requerida por cada región PR. Las restricciones AREA_GROUP RANGE deben de definir regiones PR rectangulares y no debe haber traslape entre regiones PR. Todos los recursos del dispositivo tales

como I/O, BRAM, y multiplicadores que son parte de un PRM deben tener una restricción AREA_GROUP RANGE.

Las restricciones AREA_GROUP RANGE deben de definir como sigue:

AREA_GROUP "reconfig" RANGE = SLICE_(minX)(minY):SLICE_(maxX)(maxY);
AREA_GROUP "reconfig" RANGE = RAMB16_(minX)(minY):RAMB16(maxX)(maxY);

Donde *(minX,minY)* son las coordenadas del *slice* inferior izquierdo, y *(maxX,maxY)* son las coordenadas del *slice* superior derecho. El rango de *slices* nunca puede caer entre los dos *slices* de un *CLB*, por lo tanto: *(minX,minY)* son siempre pares y *(maxX,maxY)* son siempre impares.

Nota: Una vez definida el área de una región PR, no puede ser modificarse la dimensión en tiempo de diseño, luego entonces, el tamaño de la región PR, está determinada por el PRM más grande que se implemente para dicha región.

La ubicación y forma óptima para una región PR depende del diseño a implementar. La creación de PRR más grandes que el contenido de los recursos lógicos, facilita al software la implementación del *place* and *route*, sin embargo, puede generar un error cuando se intenta cumplir con las restricciones de tiempo.

Herramientas de *Xilinx*, tales como *Floorplanning*, *PACE* de *ISE o PlanAhead*, facilitan bastante esta labor y se explica su uso el anexo 1.

Para FPGA ***Virtex-4*** es recomendable que las regiones reconfigurables no atraviesen la columna central. Si esto ocurriese, todos los elementos (recursos) deben ser instanciados en la parte estática y situados en el ***.ucf.*** El atributo *MODE=RECONFIG* debe ser definido en todas las áreas reconfigurables.

Los *frames* de configuración para FPGA Virtex-4, dividen al dispositivos en cuadrantes. El *frame* de configuración mínimo tiene una dimensión de 16 CLB de altura por un CLB de ancho. Por lo que la dimensión de un PRR, está determinada por el número de *frames* que la integren, esto permite que se puedan generar *bitstreams* pequeños, que conllevan a reducir el tiempo de configuración del dispositivo.

Restricción MODE

Debe ser definida para todas las regiones reconfigurables, esta restricción impide la desestructuración del módulo (NGDBUILD) que generen errores de bloque inesperados durante la implementación de los módulos estáticos y dinámicos.

AREA_GROUP "AG_PRregionA" MODE=RECONFIG;

Restricciones LOC (ubicación).

Deben declararse para activar a puertos de entrada y salida, primitivas de reloj(DCM BUFG,etc), y para la declaración de los bus macros

NOTA: PlanAhead simplifica la ubicación de los busmacros

Asignación de pines:

```
NET "sys_rst_pin" LOC = "AH5";
NET "sys_clk_pin" LOC = "AJ15";
```

Asignación DCM y BUFG:

```
INST "DCM_0" LOC = "DCM_X0Y1";
INST "BUFG_0" LOC = "BUFGMUX2P";
```

Ubicación de Bus macro:

```
INST "bm_l2r_leds" LOC = SLICE X42Y88;
```

Paso 3. Diseño de la implementación NO_PR

Antes de iniciar la reconfiguración parcial se recomienda realizar este paso (aunque no es necesario). Realizar un análisis de ubicación y tiempo, para ayudar a determinar el mejor rango de regiones reconfigurables y la ubicación de los bus macros

Cuando se implementa un diseño no PR, la restricción MODE=RECONFIG debe ser comentada o borrada del el archivo UCF. Posteriormente la siguiente restricción debe ser agregada en cada región reconfigurable

AREA_GROUP “AG_PRregionA” GROUP = CLOSE

Esta restricción evita que elementos (recursos) que no son del area_group sean ubicados dentro de la región del area group, esto sólo es necesario para la implementación del diseño no_PR. Una versión de diseño deberá ser construida por cada PRM o combinación de PRM Los archivos .nmc delos bus macros deberán ser ubicados dentro del directorio del diseño no_PR.

Paso 4. Análisis de Tiempo y Localización

Se trata de otro paso no necesario, pero que permite mejorar la eficiencia del diseño reconfigurable. Observando los resultados del análisis temporal y de localización se puede modificar la localización de los *buses macro* o de las PRR, y con ello mejorar los tiempos.

Paso 5. Implementación del Top y la Parte Estática

Para la implementación del *top* y de la parte estática es importante recordar volver a incluir la directiva *MODE=RECONFIG* en el fichero ***.ucf***. Los archivos ***.nmc*** que definen los *buses macro* deben estar situados en cada uno de los directorios de implementación.

Paso 6. Implementación de los Módulos Reconfigurables

Una vez que se ha realizado la implementación del *top* y de la parte estática se inicia la implementación de cada uno de los módulos reconfigurables por separado. Se debe copiar el fichero *static.used* de la carpeta de la implementación de la parte estática y renombrarlo a *arcs.exclude*. También deben de incluirse las definiciones de los *buses macro*.

Paso 7. Unión y Obtención de los Bitstreams

El último paso del flujo *EA_PR* consiste en la unión del *top*, de la parte estática y de los módulos reconfigurables. Durante este paso se crea un *bitstream* completo y uno parcial con cada uno de los módulos reconfigurables, así como un *bitstream* con el área reconfigurable en blanco y *bitstreams* en diferencias. Para ejecutar esta parte existen unos *scripts* especiales del *EA_PR* llamados *PR_verifydesign* y *PR_assemble*.

Buses Macro

Los b*uses macro* proporcionan un modo de fijar la comunicación entre la parte reconfigurable y la parte estática, haciendo que los puertos del módulo reconfigurable sean compatibles con el diseño de la parte estática**.**

Por tanto todas las conexiones entre la parte estática y los módulos reconfigurables tienen que pasar a través de *bus macros*, exceptuando la señal de reloj (ya que es manejada automáticamente por las herramientas de implementación de forma transparente al usuario). Las señales de *reset* también deben de atravesar *bus macros*.

Hay distintos tipos de *bus macro*, dependiendo de la dirección de la señal; de izquierda a derecha, de derecha a izquierda, de arriba a abajo (sólo ***Virtex-4***), de abajo a arriba (sólo ***Virtex-4***) y dependiendo de tamaño del busmacro; ancho (4 CLBs) y estrecho (2 CLBs). Los busmacros macros pueden ser síncronos o asíncronos.

Las definiciones de los *buses macro* se encuentran en unos archivos de extensión ***.nmc***. Su título en inglés sigue los siguientes criterios:

busmacro_device_direction_synchronicity_[enable]_width.nmc

Las definiciones de los *buses macro* que son utilizados en un proyecto, deben siempre copiarse a la carpeta principal del proyecto *ISE* que se realice. La localización y elección de los *buses macro* debe hacerse en función de si estarán situados en la frontera derecha o izquierda del área reconfigurable y de si las señales entran o salen de dicha área. Para entradas a un área reconfigurable, si se entra por la frontera izquierda deberá elegirse un *l2r*, y por la frontera derecha un *r2l* y para salidas de un área reconfigurable, si se sale por la frontera izquierda deberá elegirse un *r2l*, y por la frontera derecha un *l2r*:

Los *buses macro* además de ser declarados e instanciados en el código *vhdl* deben obviamente localizarse en la *FPGA* mediante el fichero de restricciones ***.ucf***. Por ejemplo un *bus macro r2l* localizado en la fila *Y38*, si la frontera derecha del área reconfigurable se extiende hasta la columna *X41* deberá localizarse de la siguiente forma:

INST "busmacro_l2r_async_narrow_0" LOC = "SLICE_X40Y38";

En ocasiones las declaraciones de los *buses macro* que se consiguen tienen delimitadores de bus distintos a los que soporta el sintetizador, o los nombres de sus entradas y salidas no corresponden con los que nos interesan. En el Anexo 4 se especifica un método para poder editar los archivos ***.nmc*** en modo texto.

Herramientas de implementación de sistemas de Lógica difusa

Con éxito que se han tenido los sistemas digitales de lógica difusa, se han desarrollado varias herramientas de implementación en FPGA. A continuación se enuncias algunos de los trabajos con mayor trascendencia en la ciencias de la investigación

A Dynamically-Recongurable FPGA Platform for Evolving Fuzzy Systems

Es Plataforma de hardware (FPGA) para sistemas difusos evolutivos utiliza reconfiguración dinámica. Realiza cambios estructurares de la arquitectura y el ajuste de los parámetros que describen a los conjuntos difusos, la reglas de inferencia, agregación. Se basa en una metodología cooperativa y coevolutiva usando algoritmos genéticos.

La plataforma de desarrollo se implementada en un FPGA, virtex II pro, mediante flujo de diseño de diferencias. Sus características principales son: Características: 4 variables de entrada de 3 conjuntos difusos cada uno, 1 variable de salida (singletos). 20 reglas difusas, Operador mínimo y máximo. Simulación en Matlab, Migración a hardware

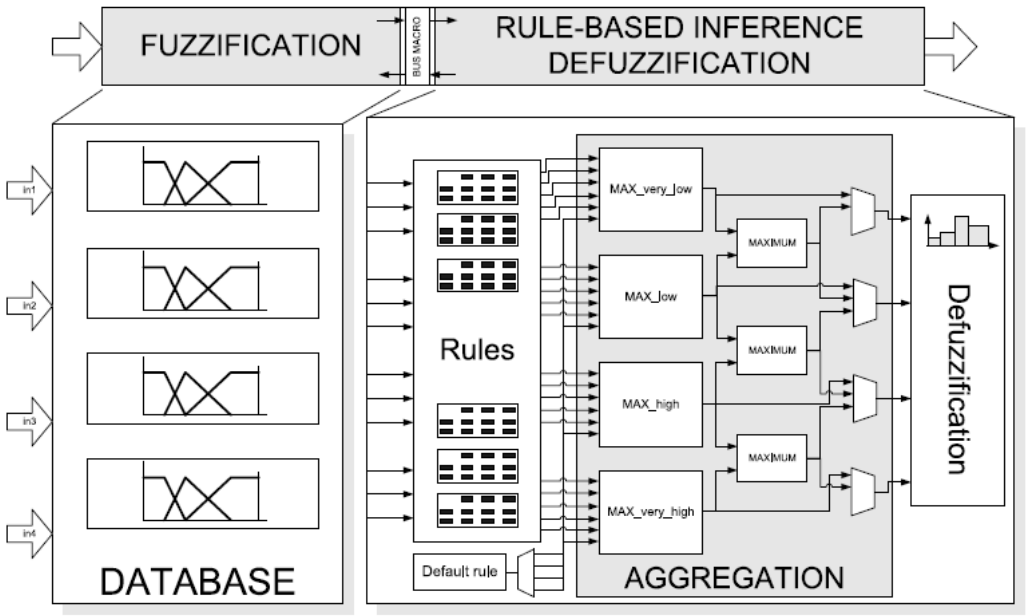


Figura 2.25. Restricción de módulos reconfigurables.

Automatic Synthesis of Fuzzy Logic Controllers.

Es una herramienta con un conjunto de herramientas de CAD, dispone de un entorno Gráfico, una biblioteca de funciones y una metodología de diseño para la manipulación de la biblioteca de funciones.

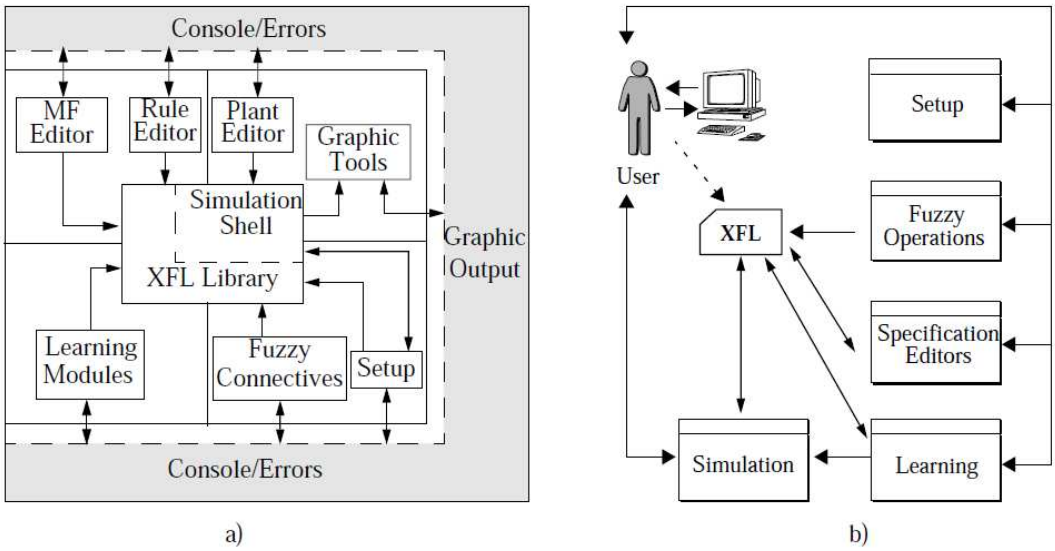


Figura 2.25. Restricción de módulos reconfigurables.

AFAN, a tool for the automatic design of fuzzy and neural controllers

Herramienta de software para generar controladores neuronales y difusos. AFAN Obtiene la Resolución, la velocidad y el consumo de la aplicación a fin de seleccionar la arquitectura del controlador que mejor se adapte a un conjunto de especificaciones de usuario. Genera como resultado un archivo que contiene un código VHDL sintetizable para un controlador neural o difuso.

An FPGA-Based Adaptive Fuzzy Coprocessor

Diseño de una arquitectura de un coprocesador de propósito general de lógica difusa (FPGA), ejecuta diferentes algoritmos de lógica difusa en un tiempo de ejecución.

Obtiene un alto rendimiento, gracias a que el procesador ejecuta instrucciones segmentadas, la organización eficaz de los datos permite aumento significativo computacional

CAPÍTULO 3. METODOLOGÍA Y DESARROLLO

En este capítulo se describe, las características , la metodología y desarrollo realizado en el proceso de diseño a nivel de Hardware del sistemas difusos utililizando los modelos Mamdani, Sugeno y Tsukamoto.

Se describe la funcionalidad y comportamiento de los módulos específicos, es decir, los módulos que describen las funciones de membresía, los operadores renfigurables, las reglas difusas, la obtención del valor certero, y después de esto, se realiza la integración módulos que describen los módulos Mamdani, Sugeno, Tsukamoto.

Por último, se hace un análisis sobre las similitudes y diferencias entre los módulos, para modelar las posibles regiones con reconfiguración parcial.

En este se trabajo se han considerado, las características de implementación en base a la estadística, donde la mayor parte de las investigaciones. Las características del sistema difuso de esta tesis son:

3.1 Características de la herramienta de diseño

- Sistema de 8 bits
- 2 variables de entrada
- 1 variable de salida
- Funciones de membresía parametrizables
- Operadores T_Norma parametrizables
- Operadores S_Norma
- 9 Reglas difusas parametrizable
- Modelo difuso Mamdani
- Modelo difuso Sugeno

- Modelo difuso Tsutkamoto

3.2 Restricciones del sistema

La implementación del diseño es realizada con los FPGA de la CYCLONE II de Altera y VIRTEX IV de Xilinx. Sin embargo unos de los objetivos de este proyecto es utilizar lógica reconfigurable, con la finalidad de acelerar el proceso de diseño de un sistema difuso, además de integrar los modelos difusos Mandani, Sugeno, Tsukamo en un mismo sistema..

Por tal motivo, la herramienta de diseño, trabaja con el manejo de archivos *bitstream*, por lo que la herramienta sólo funciona con el FPGA virtex 4 de Xilinx. Además es necesario contar con el software de diseño *Early Access parcial reconfiguration*, el ISE 9.2i.

3.3.1 Variables de entrada y funciones de membresía.

Como se mencionó en las especificaciones del sistema difuso, se cuenta con dos variables de entrada, que son traducidas a dos variables lingüísticas conformadas por tres conjuntos difusos cada uno.

Como la implementación en hardware es de 8 bits, entonces, El universo de discurso de las variables de entrada se está acotado a los valorares de [0, 255]

La implementación de las funciones de membresía que describen a los conjuntos difusos, se determinan con un solo parámetro de entrada. Para lograr esto, se define el conjunto difuso $A_{1,}$ con una función de membresía línea de pendiente negativa cuando la entrada es menor o igual a P_x, y un valor de membresía cero, en otro caso.

El conjunto A_2 es descrito por una función de membresía triangular, con una pendiente positiva para valores de x menores o iguale a P_x y una pendiente negativa

para x mayores a P_x. Por último el conjunto de A_3, es descrito por una función de membresía con valores diferentes a ceros, para x mayores de P_x.

En la figura 3.4 se muestra la gráfica que describe a una variable lingüística, determinada por tres conjuntos difusos, y un parámetro de ajuste.

Un método para la implementación de las funciones de membrecía, es describir a los conjuntos independientemente, es decir hacer la implementación de cuatro pendientes, los cual conlleva al uso de cuatro divisores, sin embargo, la implementación en hardware de los divisores consume una gran cantidad de recursos. Por lo tanto el diseño debe de realizarse con el criterio de utilizar el número mínimo de divisores.

El criterio es dividir el dominio de discurso en dos espacios delimitados por *Px*, para, entonces, como es visible en la figura 3.4, cada segmento existen dos pendientes, en la cuales la pendiente negativa es igual a *I* menos la pendiente positiva. Por lo tanto, el problema se va a reducir a sólo el cálculo de las pendientes positivas de cada segmento.

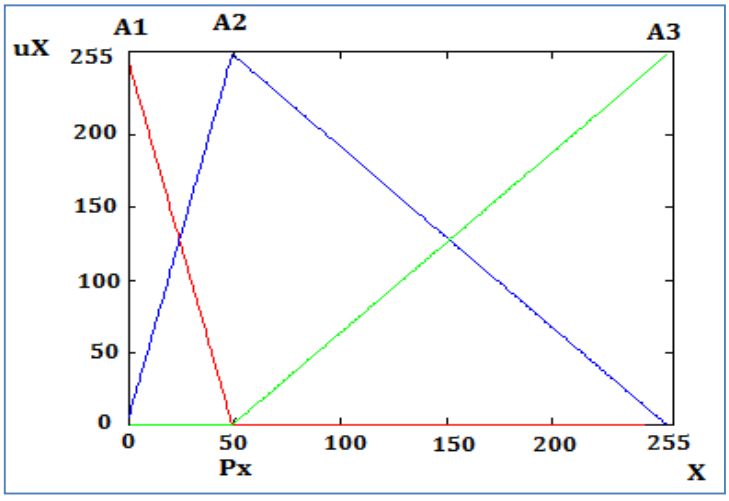


Figura 3.4. Función de membresía con un solo parámetro

Se define una ecuación μ_+ que describe las rectas con pendiente positiva de la figura 3.14. como se muestra a continuación.

$$\mu_+(x,p) = \begin{cases} \dfrac{x * I}{Px} & x \leq Px \\ \dfrac{(x - Px) * I}{I - Px} & x > Px \end{cases}$$

Entonces para determinar el valor de membresía de cada conjunto de entrada, se definen las siguientes ecuaciones.

$$\mu_{A_1}(x,p) = \begin{cases} I - \mu_+(x,p) & x \leq Px \\ 0 & x > Px \end{cases}$$

$$\mu_{A_2}(x,p) = \begin{cases} \mu_+(x,p) & x \leq Px \\ I - \mu_+(x,p) & x > Px \end{cases}$$

$$\mu_{A_3}(x,p) = \begin{cases} 0 & x \leq Px \\ \mu_+(x,p) & x > Px \end{cases}$$

En la figura 3.5, se muestra el diagrama lógico del módulo que implementa las funciones de membresía, de los conjuntos difusos de entrada, donde la ecuación μ_+ es implementada mediante un comparador que compara a x con Px, para determinar la salida de los multiplexores MUXN y MUXD, que seleccionan los posibles valores del numerador y denominador de la ecuación. A la salida del divisor se obtiene el valor de μ_+ para cualquier valor de x. posteriormente los multiplexores MUX1, MUX2 y MUX3 describen las ecuaciones μ_{A_1}, μ_{A_2} y μ_{A_3} respectivamente. Por último se han implementa el caso especial donde x = Px.

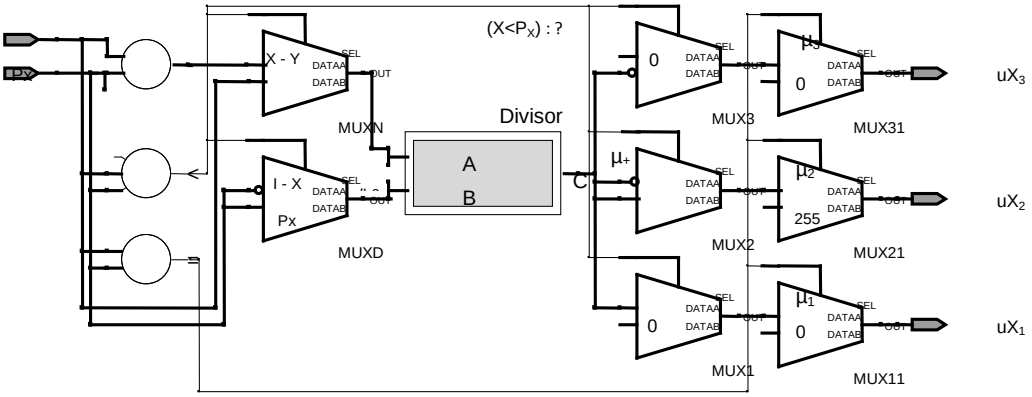


Figura 3.5. Diagrama a bloques de la función de membresía

Implementación de los operadores t-normas.

La implementación de los operadores de conjunción paramétricos se realiza a partir de las ecuaciones que describen a las t-normas básicas citadas en el capítulo 2, a continuación se presenta los diagramas lógicos que describen a las t-normas Mínimo, Lukasiewicz y Drástico y posteriormente, se describe el diseño de los operadores parametrizables.

3.3.2 T-norma mínimo.

El módulo t-norma mínimo $T_M(x,y)$, tienen dos entradas, que provienen de la salida de las funciones de membresía. La implementación usa un comparador para controlar la salida de un multiplexor que determina el mínimo de las entradas.

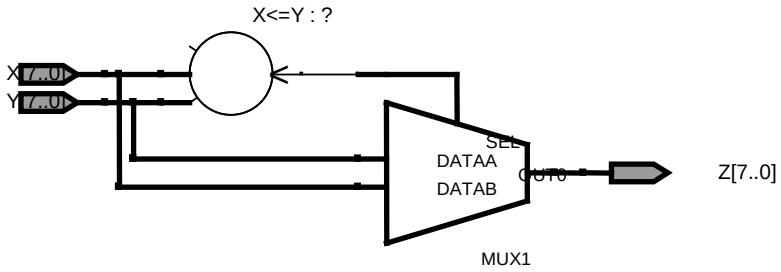
3.3.3 T-norma Lukasiewicz.

La implementación del operador t-norma Lukasiewicz, se suman las entradas $x + y$ y, si la suma es mayor a I, entonces la salida del operador es $255 - x + y$, en otro caso la salida es cero. La implementación se realiza mediante dos sumadores y un multiplexor.

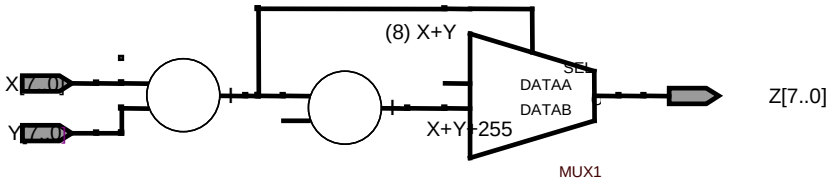
3.3.4 T-norma drástico.

El operador t-norma drástico, tiene una salida diferente a cero, cuando alguna de las entradas es igual a I, teniendo como salida la entrada que es diferente a *I*. La implementación se realiza con dos comparadores y dos multiplexores.

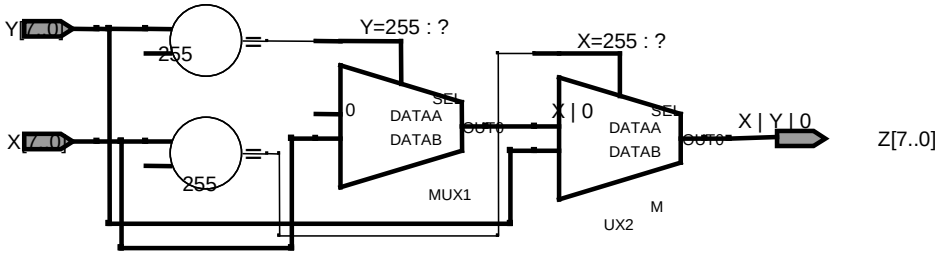
Los diagramas RTL que implementan a los operadores t-norma básicos mínimo, Lukasiewicz y drástico se muestran en la figura 3,9



a) Operador mínimo



b) Operador Lukasiewizt



c) Operador drástico

Figura 3.9. Diagrama a bloques de los operadores: a) Mínimo, b) Lukasiewicz y c) Drástico

3.3.5 Operadores de conjunción parametricos

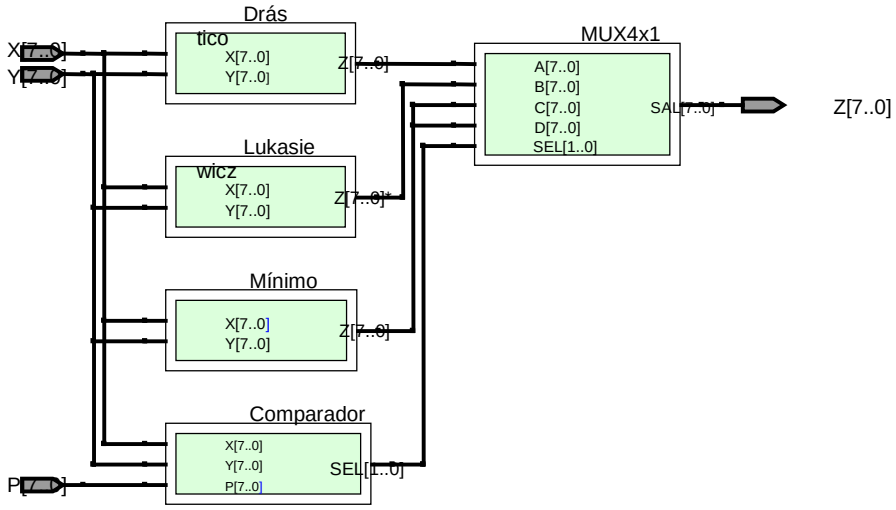
Como se mencionó en el capítulo anterior, en esta tesis se trabaja con operadores de conjunción paramétricos, formados con los operadores t-norma mínimo, producto y drástico.

La tabla 2.1 mostrada en el capitulo anterior, contiene las siete operadores de conjunción generados por el método de suma monotonica p y la tabla 2.3b, lista los 11 operadores generados por el método de suma monotónica (p, I-p). Estos operadores, más los operadores de las t-normas básicas deben ser implementados en sistema difuso. En la tabla 3.1 se muestran los 21 operadores de conjunción que son implementados en la herramienta de software.

Tabla 3.1 Operadores de Conjunción paramétricos implementados en FPGA

1	Mínimo	11	DDDDLL
2	Lukasiewica	12	DDDDLM
3	Drástico	13	DDDDMM
4	DDL	14	DDDLLM
5	DDM	15	DDDLMM
6	D LL	16	DDLLLM
7	DLM	17	DDLLMM
8	DMM	18	DDLMMM
9	LLM	19	DDMMMM
10	LMM	20	DDLDMM
		31	DDMLMM

La metodología que se emplea para implementar los operadores paramétricos, es utilizando el principio de reconfiguración parcial par tal motivo, por lo que la implementación de los operadores se hará de acuerdo al diagrama de bloques que se muestra en la figura, donde un operador paramétrico es formado por cinco módulos básicos: los módulos Drástico, Lukasiewicz, mínimo, son conectados a un multiplexor, que controla la salida, por medio del módulo comparador, el cual se encarga de comparar los valores de los pesos que generaron las funciones de membresía, con el parámetro p, y determinar la conjunción empreada para obtener el resultado.

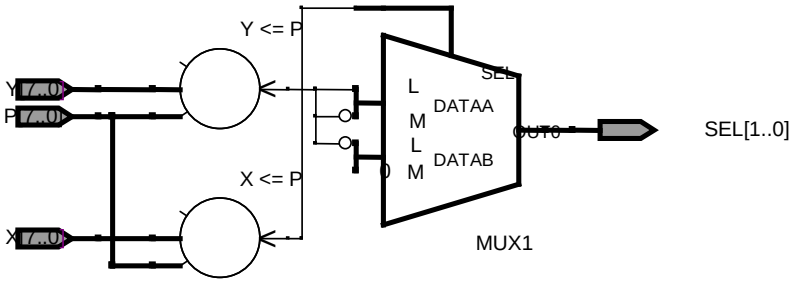


En los casos triviales, Mínimo, Lukasiewicz y Drástico, el módulo comparador, sólo configura su salida, con “10”, “01” y “00” respectivamente, sin importar el valor de las entradas.

Para la implementación de un operador de conjunción usando el método de suma monotónica p, se realizan prácticamente 2 comparaciones, y un multiplexor. A continuación se muestra el código que implementa al operador DLM (operador 7 de la tabla 3.1).

```
if X<=P then
   if Y<= P then
        SEL<= D
   else
        SEL<= L
   end_if
else
   if Y<= P then
        SEL<= L
   else
        SEL<= M
   end_if
end_if
```

El diagrama lógico que implementa al módulo comparador, para la implementación del operador DLM se muestra en la figura, consta de dos comparadores y un multiplexor. La implementación de los demás operadores generados por el método de suma monotónica p, se realizan prácticamente de la misma manera, variando las asignación de la salida, en los comparadores del código mostrado.



En la implementación de los operadores generados por el método de suma monotónica (p,I-p), es prácticamente una expansión de método anterior, ya que estos operadores agregan la comparación con I-p (o p negada). A continuación se muestra el pseudocódigo para implementar el operador DDLLLM

```
if  X<=P  then
   if  Y<= P) then
       SEL<= D
   else_if Y<= Pn then
          SEL<= D
       else
          SEL<= L
       end_if
   end_if

else if  X<= Pn then
   if  Y<= P then
       SEL<= D
   else_if  Y<= Pn then
          SEL<= L
       else
         SEL<= L
          end_if
end_if

 else
   if Y<= P then
       SEL<= L
   else_if  Y<= Pn then
          SEL<= L
       else
          SEL<= M
          end_if;
   end_if;
end_if
```

La diagrama lógico del modulo comparador, para implementar el operador DDLLLM, se muestra en la figura, formado por cuatro comparadores y tres

multiplexores que determinan la configuración de los pines de salida. Los demás operadores generados por este método, se implementan de manera similar.En el anexo1, se muestra la metodología paso a paso, para la implementar los operadores de conjunción usando reconfiguración parcial.

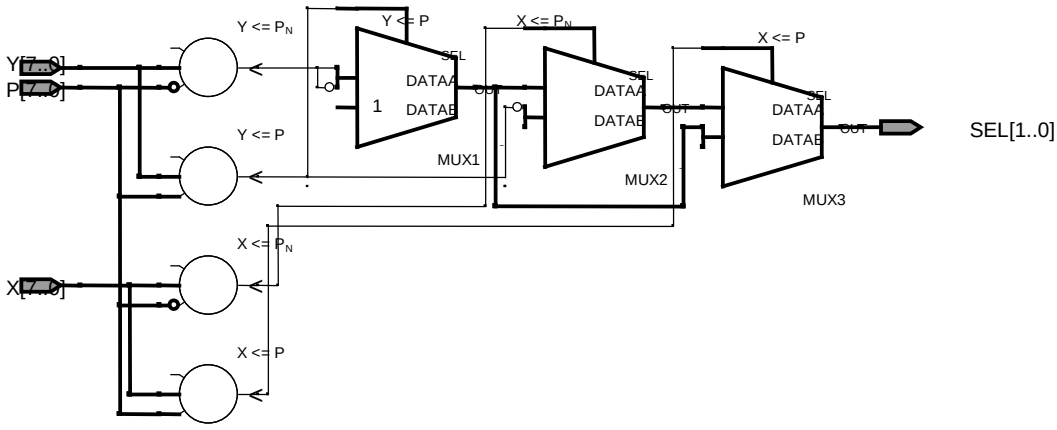


Figura 3.8. Diagrama a bloques de los operadores paramétricos p con suma monotónica

3.3.6 Implementación las reglas difusas.

El sistema difuso consta de 9 reglas difusas a lo más, en las cuales cada conjunto de la entrada x se relaciona con una conjunción con cada conjunto difuso de la entrada y. en la figura 3.6 se muestra las combinaciones de los antecedentes del conjunto de reglas del sistema difuso.

$$R_0 : si \quad x\ es\ A_1 \quad y \quad y\ es\ B_1$$
$$R_1 : si \quad x\ es\ A_1 \quad y \quad y\ es\ B_2$$
$$R_2 : si \quad x\ es\ A_1 \quad y \quad y\ es\ B_3$$
$$R_3 : si \quad x\ es\ A_2 \quad y \quad y\ es\ B_1$$
$$R_4 : si \quad x\ es\ A_2 \quad y \quad y\ es\ B_2$$
$$R_5 : si \quad x\ es\ A_2 \quad y \quad y\ es\ B_3$$
$$R_6 : si \quad x\ es\ A_3 \quad y \quad y\ es\ B_1$$
$$R_7 : si \quad x\ es\ A_3 \quad y \quad y\ es\ B_2$$
$$R_8 : si \quad x\ es\ A_3 \quad y \quad y\ es\ B_3$$

Figura 3.6. Conjunto de reglas difusas

Para la implementación, se desarrollan 9 módulos, nombrados t-norma como se muestra en la figura 3.7, donde se hace la conexión de las salidas correspondientes de los módulos *fuzzy_set*. Además se observa que los módulos t_norma, tiene la entrada p y op_TNorma que corresponde a la parametrización de la conjunción difusa que se presenta a continuación.

3.4.7 Función de Salida en Sistema Sugeno.

La función de salida de salida para el cálculo de los consecuentes, corresponde a un polinomio de la forma siguiente

$$z_i = a_i * x + b_i * y + c_i,$$

donde *i*, va de 0 a 8, y corresponde a la regla difusa *i*.

La suma de los términos de la ecuación anterior pueden sobrepasar, el valor máximo I del sistema digital, por lo tanto, se hace la siguiente modificación para la obtención de la salida digital.

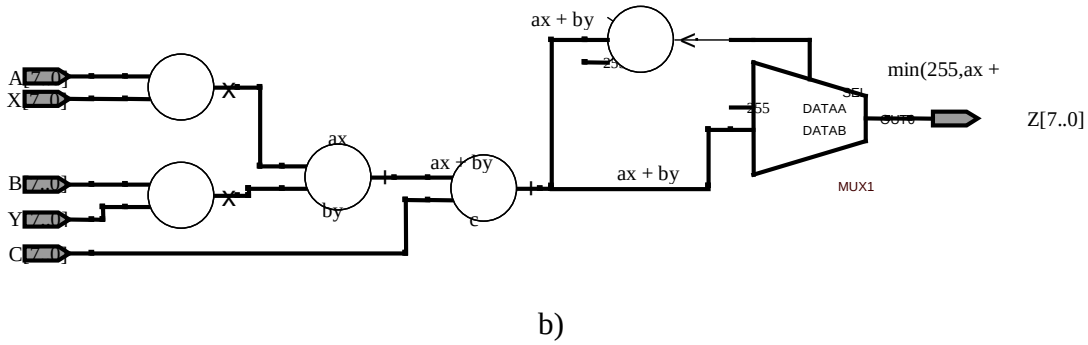
$$z_i = \min\left(I, a_i * x + b_i * y + c_i\right)$$

Con esto se asegura que los valores de la función de salida, no contengan sobreflujo. Además, podemos observar que si los coeficientes a_i y b_i son igual a cero, entonces la función de salida es modelada por una función singleton, que es la forma en que la mayor partes de las investigaciones modelan los sistemas difusos Mamdani.

Si todos los coeficientes de la ecuación a_i, b_i y c_i son igual a cero, entonces podemos interpretar que la regla difusa R_i, no se agrego en la base de reglas del modelos a diseñar.

El diagrama lógico para implementar el módulo de la ecuación $\min\left(I, a_i * x + b_i * y + c_i\right)$ se muestra en la figura a, compuesto por dos multiplicadores, dos sumadores, y para acotar la salida, se usa un comparador y un multiplexor, en la parte b de la

figura se muestra el diagrama de bloques que muestra la implementación de los nueve conjuntos de salida.



b)

Figura 3.10. Diagrama a bloques de la función de salida para sistema Sugeno. a) Diagrama lógico del valor de certero del conjunto de salida sugeno, b) diagrama de boque de las 9 funciones de salida.

3.4.8 Función monotónica de Salida en Sistema Tsukamoto.

El cálculo del consecuente de cada regla difusa si_entonces, se realiza a través de una función de membrecía monotónica, como las que se muestran en la figura Xi. Donde la ecuación esta en función de los parámetros Ai y Bi, que determinan los limetes inferior y superior de la función monotónica y del valor de salida del cálculo del antecedente, es decir, el peso Wi. Como salida la función monotónica calcula el valor certero Zi, de cada regla, como se mostro en la figura x, del capítulo 2.

La ecuación que describe a la función monotonica creciente de la figura x a, se calcula mediante la ecuación de la recta, y es descrita como sigue

$$w_i = \frac{255(z_i - a_i)}{b_i - a_i}$$

Como la salida a calcula es z_i entonces, se despeja esta variable, por lo que la expresión que calcula z_i esta dado por:

$$z_{i+} = \frac{w_i(b_i - a_i)}{255} + a_i$$

El cálculo de la ecuación que describe a la función monotónica decreciente se describe como sigue:

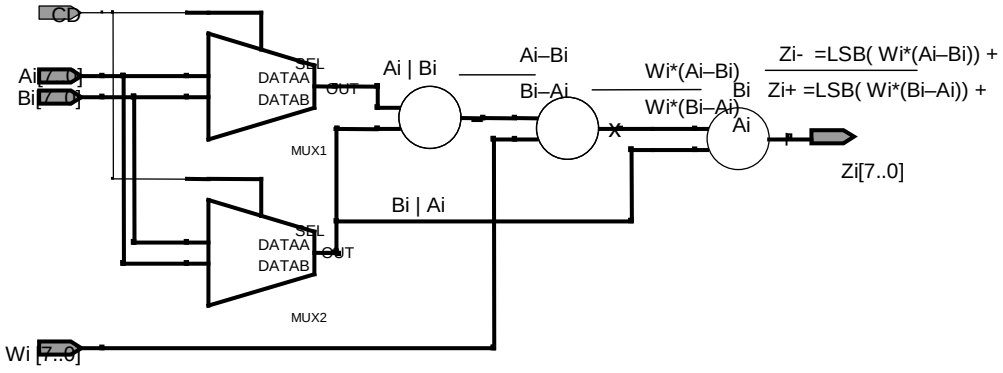
$$z_{i-} = \frac{w_i(a_i - b_i)}{255} + b_i$$

Generalizando la ecuación que describe al consecuente de la regla difusa del modelo Tsukamoto, se agrega el parámetro de entrada CD para seleccionar, si la función es creciente o decreciente, se tienen la siguiente ecuación.

$$Z_i(x, y; p) = \begin{cases} z_{i+} & si\ \ DC = 1 \\ z_{i-} & si\ \ DC = 0 \end{cases}$$

En la figura x, se muestra el diagrama lógico, que describe a la ecuciación anterior, donde para unificar la ecuación, se utilizan dos multiplexores, la primera parte de las salidas de cada componente lógico corresponde a la entrada CD=0, cuando CD=1, le las salida, de los compontes lógicos será la otra expresión. Hay que notar que la salida del miltiplicador, se tiene un número binario de 16 bits, por lo tanto, en lugar de dividir esta salida entre 255, como dice la ecuación anterior, se toma la parte menos significativa del número binario, que corresponde a una división entre 256.

Lo anterior implica, que la implementación tiene una perdida de información, sin embargo, esta no es de consideración, ya que los resultados fueron verificados en sorfware, y no tenían diferencia.



3.4.9 Obtención de valor certero de salida de los sistemas difusos.

El valor certero del sistema difuso tanto para el modelo Sugeno como el Tsukamoto y Mamdani (con funciones de salida singletons), se calcula el factor de la suma de productos de $w_i * z_i$ entre la suma de los pesos calculados en el módulo regla_antesedente, como lo índica la siguiente ecuación

$$z_C = \frac{\sum_{i=1}^{n} w_i z_i}{\sum_{i=1}^{n} w_i}$$

La implementación del numerador multiplica las nueve entradas Wi con las correspondientes Zi. Basta con desarrollar 9 módulos donde se multiplican las entradas $w_i \, y \, z_i$. Y posteriormente hacer un módulo, donde se sumen las multiplicaciones y se sumen los pesos w_i, y por último hacer la división correspondiente Ver figura

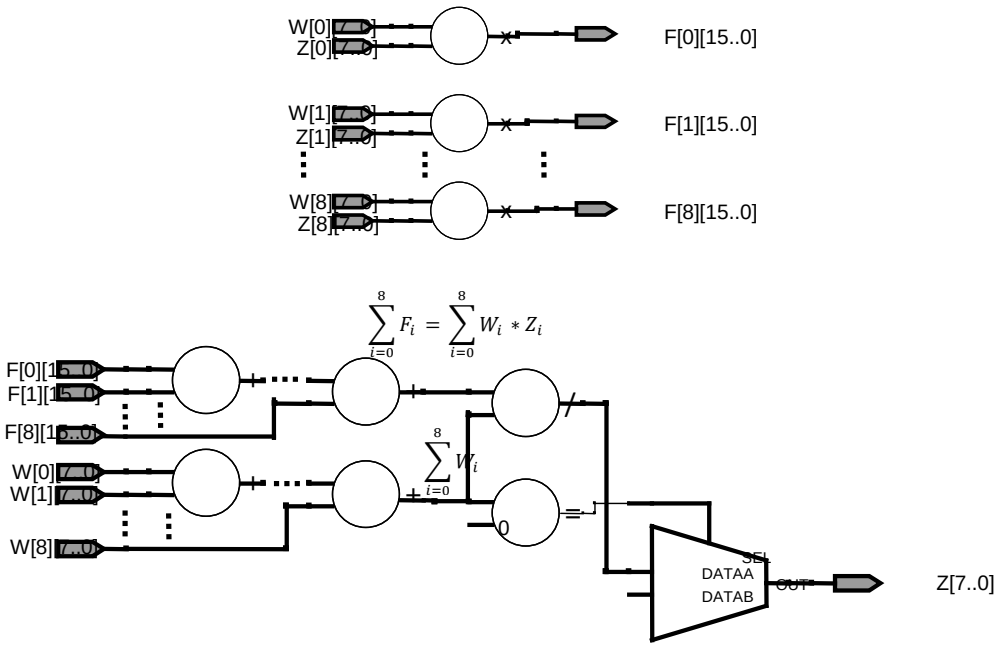


Figura 3.11. Diagrama a bloques de la obtención del valor certero para sistema Sugeno y Tsukamoto

3.3.10 Modelado de sistemas difusos.

A continuación se realiza la integración de los módulos anteriormente descritos, para formar los modelos Mamdani, Sugeno y Tsukamoto.

3.3.11 Implementación del sistema Mandani.

La implementación del sistema Mamdani, con nueve reglas se muestra en la figura x, la cual integra los módulos, Fuzzy_x, Fuzzy_y que implementan las funciónes de membresía de conjuntos de las entradas, cuyas seis salidas, alimentan a los módulos drásticos, Lukasiewicz, minimo y comparados, que forman a los operadores de conjunción paramétricos, estos bloques son compuestos por 9 mudulos en paralelo, de los mostrados en los diagramas correspondientes mostrados en las secciones anteriores, por lo que la salida de estos módulos son un vector de nueve localidades, que es multiplexada generar los pesos del antecendente de la regla difusa si-entonces wi. Con i=0…8. Los conjuntos de salida para el modelo mamdani son singletons, por lo que las entradas Ai no necesitan procesamiento, por último se realiza el cálculo de valor de salida z.

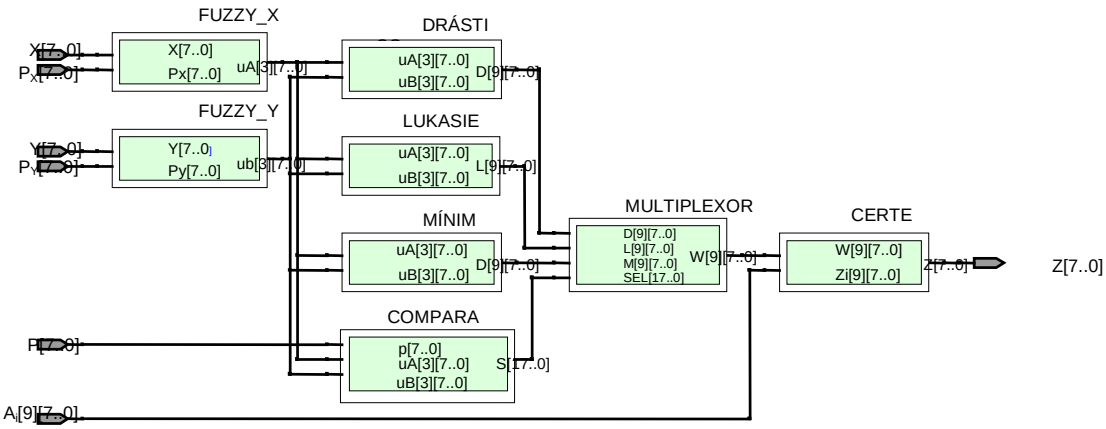


Figura 3.1. Diagrama a bloques de la implementación del modelo Mamdani

3.3.11 Implementación del sistema Sugeno.

La implementación del modelo sugeno, difiere del módelo Mamdani únicamente en la implementación de las funciones de membresía de los conjuntos consecuetes de las reglas difusas. En la figura x se muestra la implementación del modelo Sugeno.

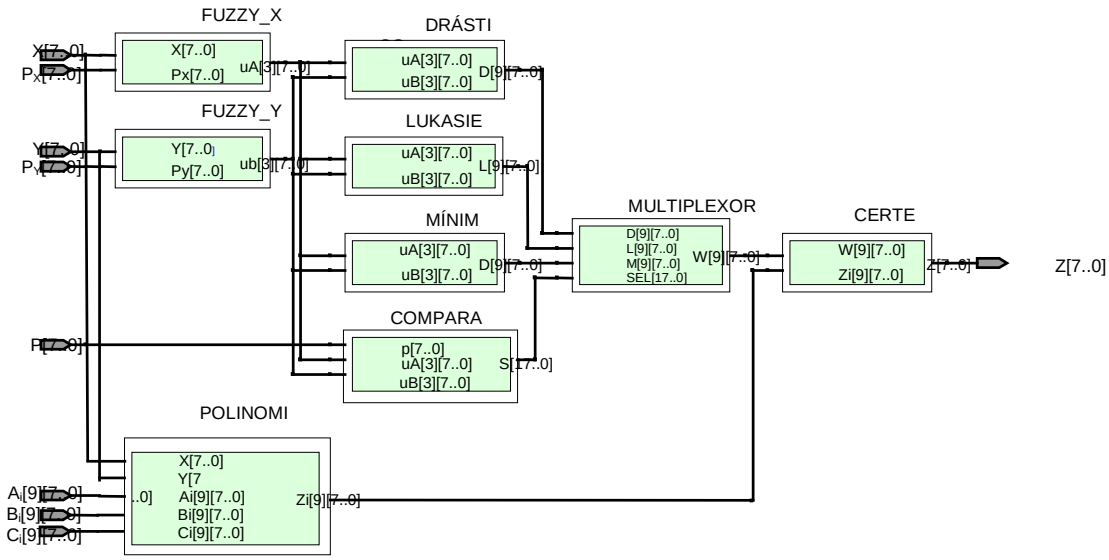


Figura 3.2. Diagrama de bloques de la implementación del modelo Sugeno

3.3..13 Implementación del sistema Tsukamoto.

La implementación del modelo Tsukamoto, solo se tiene que cambiar la función de de membresía, por una función monotonica que genera los valores Zi. Este modulo depende de los valores de salida de multiplexor, es decir, Wi, y los parámetros de entrada CD, Ai y Bi que determinan el comportamiento de la función de membresía Zi.

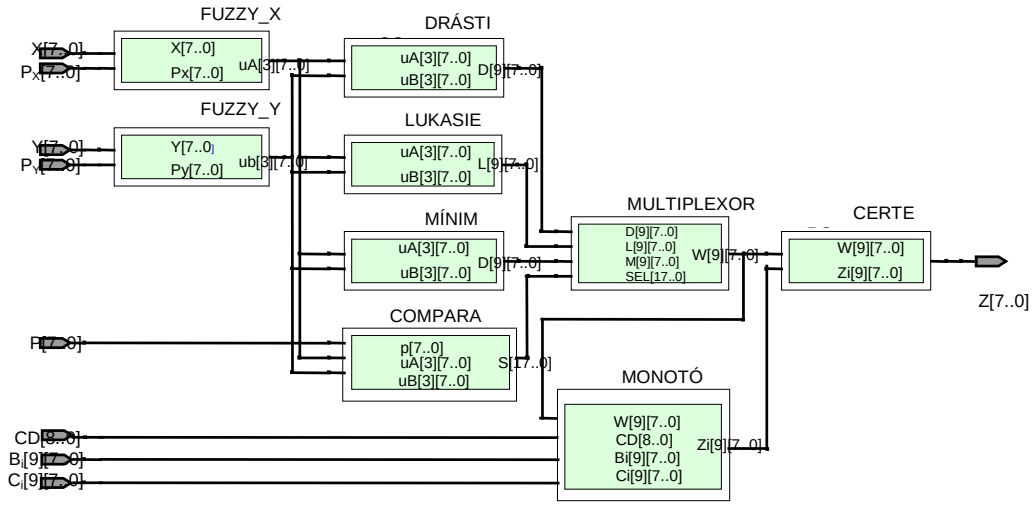


Figura 3.3. Diagrama de bloques de la implementación del modelo Tsukamoto

5. CONCLUSIONES, TRABAJO FUTURO Y PRODUCTOS DE LA INVESTIGACIÓN

5.1 CONCLUSIONES

En este trabajo se ha desarrollado una herramienta de diseño de sistemas difusos, muy flexible y totalmente combinacional. Lo que le da portabilidad a casi cualquier tipo de aplicación de dos entrada y una salida.

Con la parametrización de los conjuntos difusos, se logra una nueva metodología de ajuste de sistemas difusos de acuerdo a la relación entre las variables de entradas, incrementado la flexibilidad de modelado del sistema, ya que la metodología se suma monotónica p y 1-p, generan 16 operadores de conjunción paramétrica.

Al utilizar el enfoque de reconfiguración dinámica en el diseño de la herramienta, se logra decrementar el tiempo de diseño de sistemas difusos al utilizar *bitstream parciales*, en lugar de código vhdl o bitstreams completos en el mejor de los casos, teniendo tiempos de reconfiguración a .25us y sin necesidad de que el sistema se detenga.

5.2 TRABAJO FUTURO.

- Desarrollar metodología implementación etapa de optimización utilizando reconfiguración dinámica.
- Desarrollar módulos de comunicación reconfigurables, para las entradas y la salida.
- Desarrollar metodología e implementación para que los operadores sean asociativos.

5.2 PRODUCTOS DE LA INVESTIGACIÓN

- *"FPGA Implementation of (p)-Monotone Sum of Basic t-norms",* en el *ieee world congress on computational intelligence 2010 in Barcelona España.*
- *"FPGA implementation of fuzzy system with parametric membership functions and parametric conjunctions", en 9th Mexican International Conference on Artificial Intelligence, 2010 en Pachuca Hidalgo, México .*
- *FPGA implementation of parametric Fuzzy Mandani Systems en Ninth International Conference on Appplication of Fuzzy Systems and Soft Computing. Agosto de 2010. En Praga, Republica Checa.*
- *"FPGA implementation of (p, I-p) – Monotone Sum of Basic t-norm" en el World Conference on Soft Computiong in San Francisco California. 2011*

Anexo 1. Implementación de operadores parametrizables utilizando reconfiguración parcial.

Introducción

En este anexo se explica paso a paso, el diseño de los operadores paramétricos utilizando reconfiguración parcial, mediante las herramientas de diseño ISE, PlanAhead de Xilinx y usando la tarjeta evaluación ML403 verificar el diseño en hardware mediante software iMPACT de Xilinx descargando los archivos de configuración (bitstreams) completo y parciales

En el diseño se definen una región parcialmente reconfigurable (PRR), con 21 módulos reconfigurables (RM), que implementan los tres operadores básicos Drástico, Lukasiewicz y Mínimo, los 7 operadores parametrizables de suma monotónica (p) y los 11 operadores parametrizables de suma monotónica (p, I-p).

Procedimiento

El propósito de este anexo, es mostrar con un caso de uso, la elaboración de un diseño dinámicamente reconfigurable, con el objetivo que el lector, pueda expandir o modificar el proyecto elaborado en esta tesis.

El flujo de diseño de implementación parcial de un sistema, utiliza una metodología button-down. El archivo top-level (mayor jerarquía), se muestra en la Figura A.1, y consta de una PRR, con 21 RM. El diseño está integrado con un módulo UART para comunicar el Harware con la computadora, un módulo de control para la manipulación de los operandos y tres módulos que describen a los operadores Drástico, Lukasiewicz y Mínimo conectados a un multiplexor, que es controlado por

un módulo comparador que determina la salida del sistemas, de acuerdo al valor UAi, UBi y de p. El usuario interactúa con el diseño a través de la HyperTerminal, Los módulos dinámicos se descargan con el software Impact, o con la herramienta de diseño, que se explica en el capítulo 4.

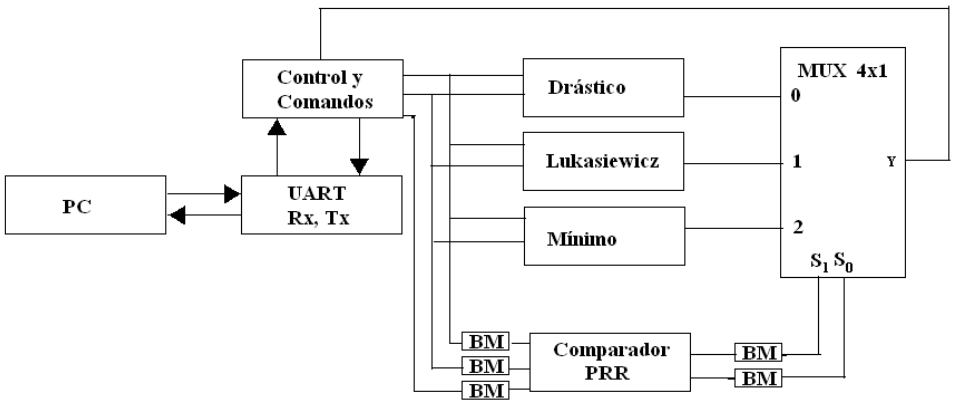


Figure A.1. Diseño de los operadores parametrizables con RM.

En el disco que se anexa a este documento se encuentra el directorio, anexo1 con la siguiente ruta E:/anexo1/, el cual contiene los código fuente (archivo .vhd y .v), archivos ucf, edn necesarios para implementar esté diseño. La implementación de los módulos multiplexor, comparador y de los operadores Drástico, Lukasiewicz y Mínimo han explicados en el capítulo 3.

Para comenzar la implementación de un diseño, se recomienda hacer la siguiente estructura de directorios para llevar un orden, de los archivos generados en el proceso de diseño.

Copiar el directorio /anexo1 en tu computadora, que contiene los subdirectorios: data, image, netlists, planahead, source, synth y flat como se muestra acontinuación

Tu_Ruta/
Anexo1/

/Data/
/Image/
/Netlists/
/planahead/
/source/
/ synth/
/flat/

El directorio Data contiene los archivos nmc que describen a los bus macros, y el archivo ucf usado en el planAhead para definir las restricciones de la implementación del diseño

El directorio image contendrá todos los archivos bitstream que serán creados en la herramienta PlanAhead durante el flujo de diseño, y serán utilizados en la creación del proyecto en la herramienta IMPACT.

El directorio *netlists* contendrá todos los archivos NGC que serán generados en la herramienta de síntesis, dentro del programa ISE.

El directorio */planahead* contendrá los archivos generados en el proceso de diseño con la herramienta PlanAhead.

El directorio */source* contiene los archivos fuentes .vhd y .v necesarios para el diseño propuesto.

Directorio *synth* . En este directorio se crearán prácticamente un directorio por cada archivo fuente (.vhd o .v), dentro directorio se creará un proyecto en el ISE, y generará entre otros archivos, el archivo ngc que será copiado en la carpeta netlist, como se indica posteriormente.

El directorio /*flat*/ contiene el diseño mostrado en la figura A1.

El flujo de diseño de un sistema dinámicamente reconfigurable puede dividirse en tres secciones: Síntesis de los módulos estáticos y reconfigurables en el ISE, la implementación con planAhead y la configuración con la herramienta Impact.

Síntesis del Diseño.

Lo primero que se debe realizar es la síntesis del diseño será el proyecto por defecto o inicial, es decir, con el módulo reconfigurables inicial del proyecto (min). Esta síntesis se realiza de la misma manera que se hace en los diseño sin reconfiguración parcial (el módulo parcialmente reconfigurable se tratan como estático). Copiar el contenido de anexo1/source a el directorio /synth . Una vez que se ha verificado el proyecto inicial *flat*. Se realiza la síntesis de los módulos individualmente.

Para sintetizar un proyecto en el ISE, primero se debe crear y configurar un proyecto, llenando formularios. La figura A2, muestra la ventana inicial del Wizard de creación de proyecto en el ISE. Realiza los siguientes pasos para crear un proyecto.

1. Abrir ISE: **File->New Project…**
2. En la ventana **New Project Wizard,** seleccionar **Create new Project**
3. En el campo **Project Location** direccionar **/anexo1/synth**
4. Nombra "*top*" como nombre del proyecto (se crea el directorio top)
5. En el campo de texto Top-Level Source Type : selecciona HDL.
6. Click Next

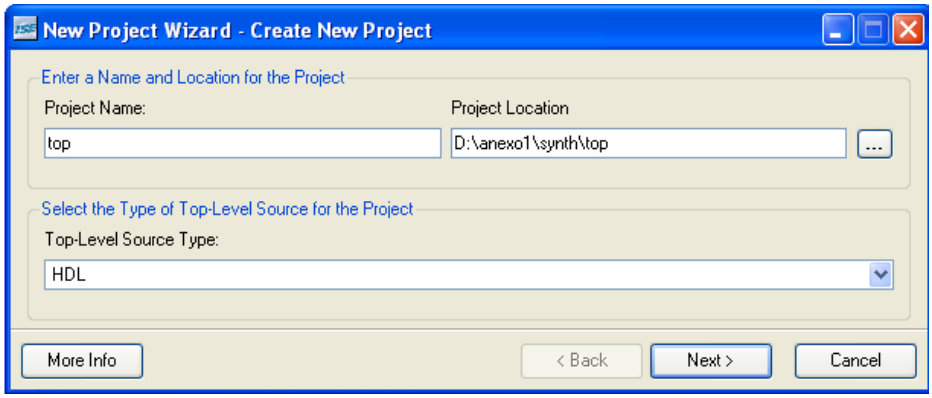


Figure A.2. Configuración de proyecto; ventana Create New Projecto en el ISE

En la siguiente plantilla, se configura el dispositivo a utilizar, en esta tesis se trabajó con la tarjeta ML403, los siguientes pasos de configuración se muestran en la figura A3. En la ventana **New Project Wizard - Device Properties,** seleccionar en

1. **Family** *Virtex4*
2. **Device** *XC4VFX12*
3. **Package** *SF363*
4. **Speed** *-10*
5. Click **next** dos veces

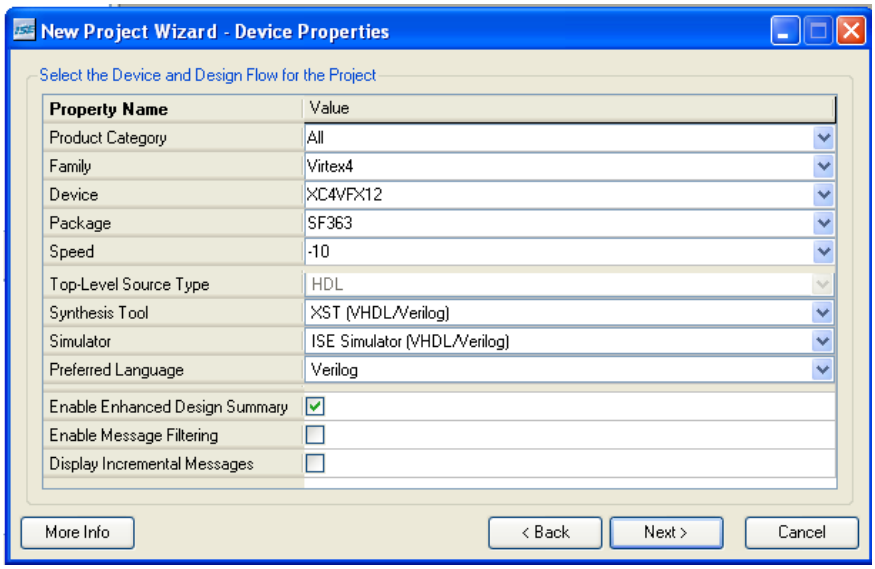


Figure A.3. Configuración de proyecto; ventana Device Properties en el ISE

Posteriormente a esto, se abre la ventana **Add Existing Sources** se configurara lo siguiente:

1. Click el boton **Add Source**
2. Localizar el archivo *top.vhd* en el directorio **/anexo1/synth**
3. *Clic* en el botón *next.*, ver figura A4

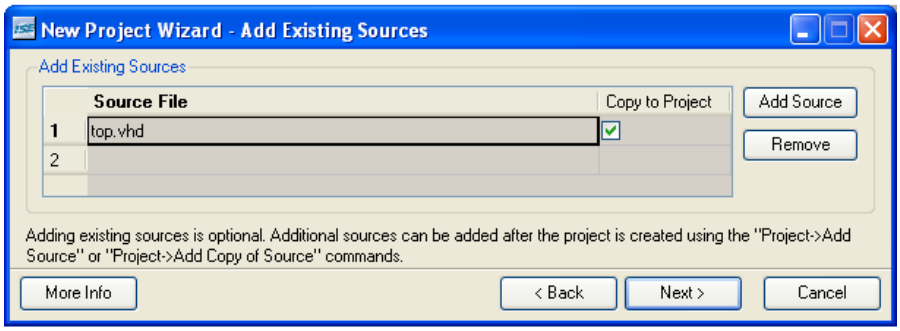


Figure A.4. Configuración de proyecto; ventana Add Existing Sources en el ISE.

A continuación se despliega la ventana **Project summary** que muestra la información de nuestro proyecto, dar nuevamente click en next. Posteriormente aparecerá la ventana **Adding Source Files**; seleccionar la asociación de diseño **Synthesis/Implementation Only** ver figura A5 y dar click **OK** para terminar el Wizard**.**

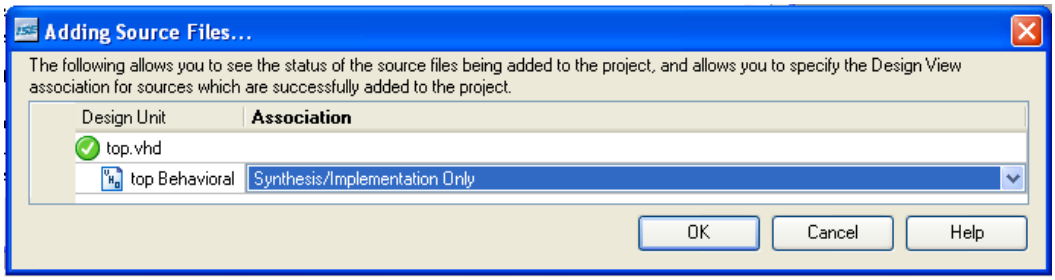


Figure **A.5. Configuración de proyecto; ventana Add Souce Files… en el ISE.**

En el panel principal del ISE, dar doble-click sobre **Synthesize – XST** en la ventana **Process** para realizar la síntesis del módulo top.

Cerrar el proyecto, y crear los demás proyectos prácticamente de la misma manera a los módulos Drástico, Lukasiewicz y Mínimo, Mux, Comparador, Registro, Demux_Reg, Control y BaudClk. Nota que antes de realizar la síntesis (doble-click sobre **Synthesize – XST**) se deshabilitan las IO, para permitir que estas sean un submódulo del top. Esto es definitivo en el flujo de diseño de reconfiguración parcial. Para realizar este proceso **En la ventana Processes** ver **Mínimo**, desplaza el puntero del mouse sobre **Synthesize-XST y da** click –derecho, y aparece la ventana Proccess Properties como en la figura A6 seleccionar **Properties…**

1. La ventana **Process Properties** se abrirá; selecciona la opción **Xilinx Specific Options** que se muestra en el panel Category ubicado a la izquierda de la ventana
2. Deshábilita **Add I/O Buffers**
3. Click en **Apply** y luego Click en **OK**
4. Doble-Click sobre **Synthesize – XST** en la ventana **Process** para sintetizar el módulo

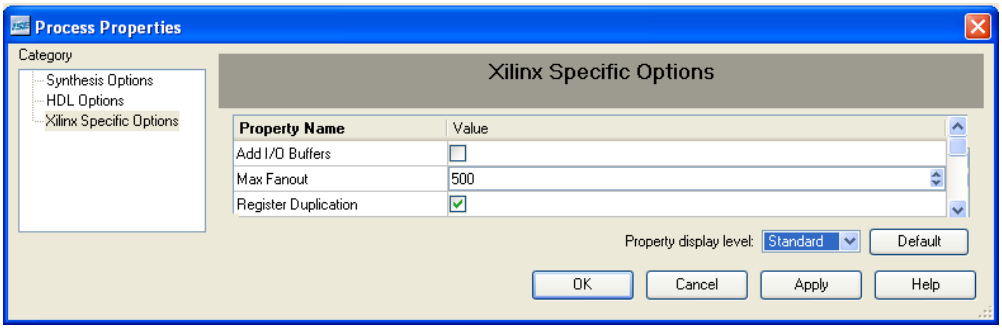


Figure A.6. Deshabilitación de Add I/O buffer, que determinan que un módulo es submódulo de otro.

Repetir este proceso para los demás módulos. Asegurarse que los **Add I/O Buffer** estén deshabilitados antes de realizar la síntesis, ya que esto indica que son submódulos del top.

Tener claro que deben realizar las 21 implementaciones del módulo reconfigurable **Compara**. Para identificar una implementación de las demás crear una carpeta lamada rModule, y crear un directorio por operador por ejemplo: MIN, DDL, DDM, DDDDDL, etc, Agregar el módulo compara al cada proyecto

(ej, *Anexo1/source/MIN*). Al finalizar la síntesis de este proyecto, observa en el

Explorador de Windows se crea entre otros el archivo Compara.ngc (y no min o ddl o ddm etc).

Crear un proyecto con PlanAhead

Abrir PlanAhead Seleccionando **Start** → **Programs** → **PlanAhead10.1** → **PlanAhead 10.1,** Click en **Create A New Project y** posteriormente ir al directorio /anexo1/PlanAhead. Nota que el nombre del proyecto seleccionado es **project_1** (Si tu ya tienes un proyecto en este directorio, entonces la herramienta lo nombrará con el numero de proyecto siguiente). Nombrar el proyecto como **conjunction**, como se muestra en la figura

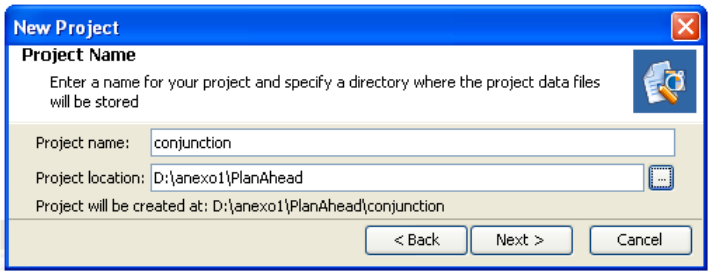


Figure A.7. Creación de proyecto en PlanAhead.

Click **Next** dos veces y dar click sobre el botón Browse y busca el directorio **anexo1/Synth/top** y selecciona el archivo **top.ngc** que implementa al módulo top-level ngc. Click sobre el botón **Add** y selecciona el directorio **anexo1/Synth/static** y agrega los archivos ngc. Similarmente, click nuevamente sobre el botón **Add**, selecciona el directorio **anexo1/Synth/rModule_compara/minimo** agregar el archivo ngc. El archivo **compara** implementa a el módulo que configura las entradas de selección del multiplexor, de acuerdo a la metodología explicada en el capítulo 4, finalmente ir al directorio **anexo1/Data** que agrega los archivos end que implementan la uart.

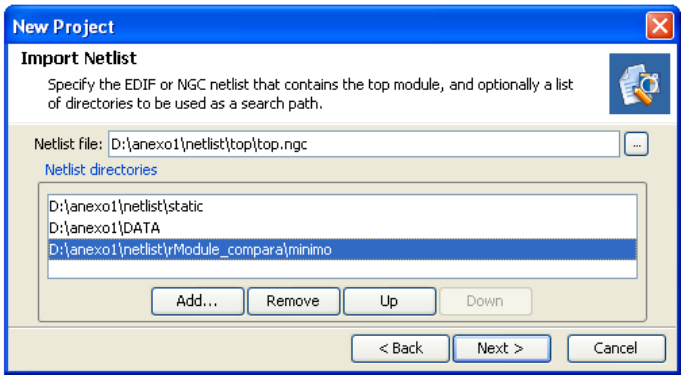


Figure A.8. Creación de proyecto en PlanAhead, ventana Import Netlist.

Click **Next** y se abre la forma **Select part**, seleccionar el dispositivo **xc4vfx12ff668-10**, y dar click en **OK** presionar **Next** e ir a la forma **Import Constraints**

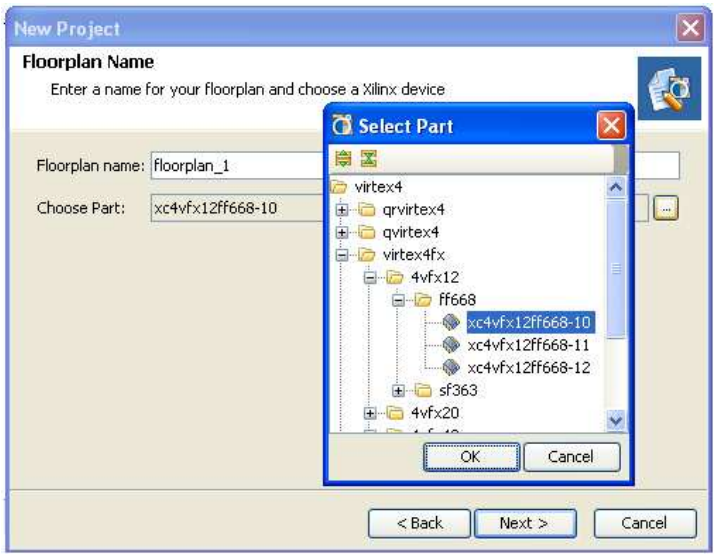


Figure A.9. Creación de proyecto en PlanAhead, ventana FloorPlan.

En la ventana **Import Constraints**, click sobre el botón **Add** y busca el directorio **OperadoresParametricos/Data**, seleccionar el archivo **top.ucf**. Click **OK** para agregar este, click **Next** y **Finish**.

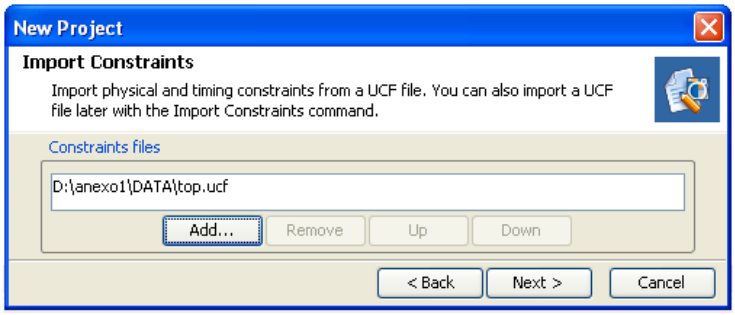


Figure A.10. Creación de proyecto en PlanAhead, ventana para importar restricciones.

Con esto se han importado los archivos necesarios para abrir el floorplan Usando el Explorador de Windows, se puede verificar que el directorio project_1 se ha creado dentro del directorio PlanAhead. Dentro del directorio project_1, el directorio project_1.data contiene los directorios floorplan y netlist recientemente creados. Por último seleccionar **File → Set PR Project,** que configura al actual proyecto (Project_1.ppr) como un proyecto PR

Creación de los bloques estáticos y reconfigurables

En PlanAhead seleccionar todos los módulos excepto el módulo reconfig_compara en el panel Netlist, dar click-derecho y seleccionar **New Pblock** para crear un Pblock que contiene estos módulos . Nombra este bloque como **static** y teclea ok. El Pblock **Static** es creado y es mostrado en la ventana Physical Hierarchy

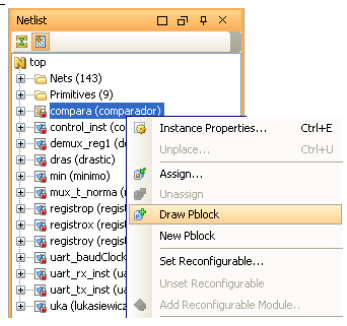
a) Selección de un Nuevo PBlock

b) Nombrar el Pblock

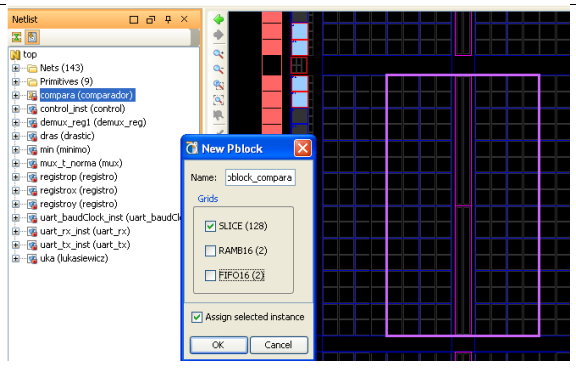
c) Verificación de la creación del Proyecto *static*

Figure A.11. Creación del bloque estático.

Seleccionar el módulo **reconfig_compara** en el panel Netlist, dar click-derecho y entonces seleccionar la opción **Draw Pblock** ( )



a) Configuración del módulo reconfigurable



b) Diseño de módulo reconfigurable.

Figure A.12. Creación del bloque parcialmente reconfigurable

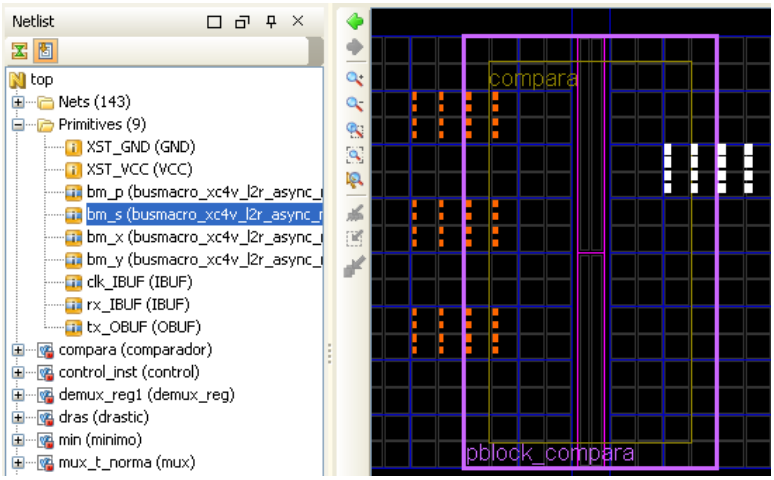
Dibujar un rectángulo en la ventana floorplan arrastrando el mouse del **Slice_X4Y80** al **Slice_X9Y95** (se puede ver el número de slice dentro de la barra de estado de la ventada de floorplan moviendo el mouse) y click **OK,** nombra este como **pblock_reconfig_compara.**

Colocación de busmacros.

En la barra barra de herramientas, seleccionar **Create Site Constraint Mode** dando click en el icono que se muestra en la figura A12 (Esto permitirá ubicar los componentes dentro de la ventana floorplan)

Expandir la carpeta de las primitivas instanciadas en el módulo top-level, que se muestra en el panel Netlist, y seleccionar los bus macro instanciados y associardos al bloque compara y ubicarlos al lado derecho del rectángulo que implementa al Pblock

a) Icono para crear restricciones ubicado en la barra de herramientas



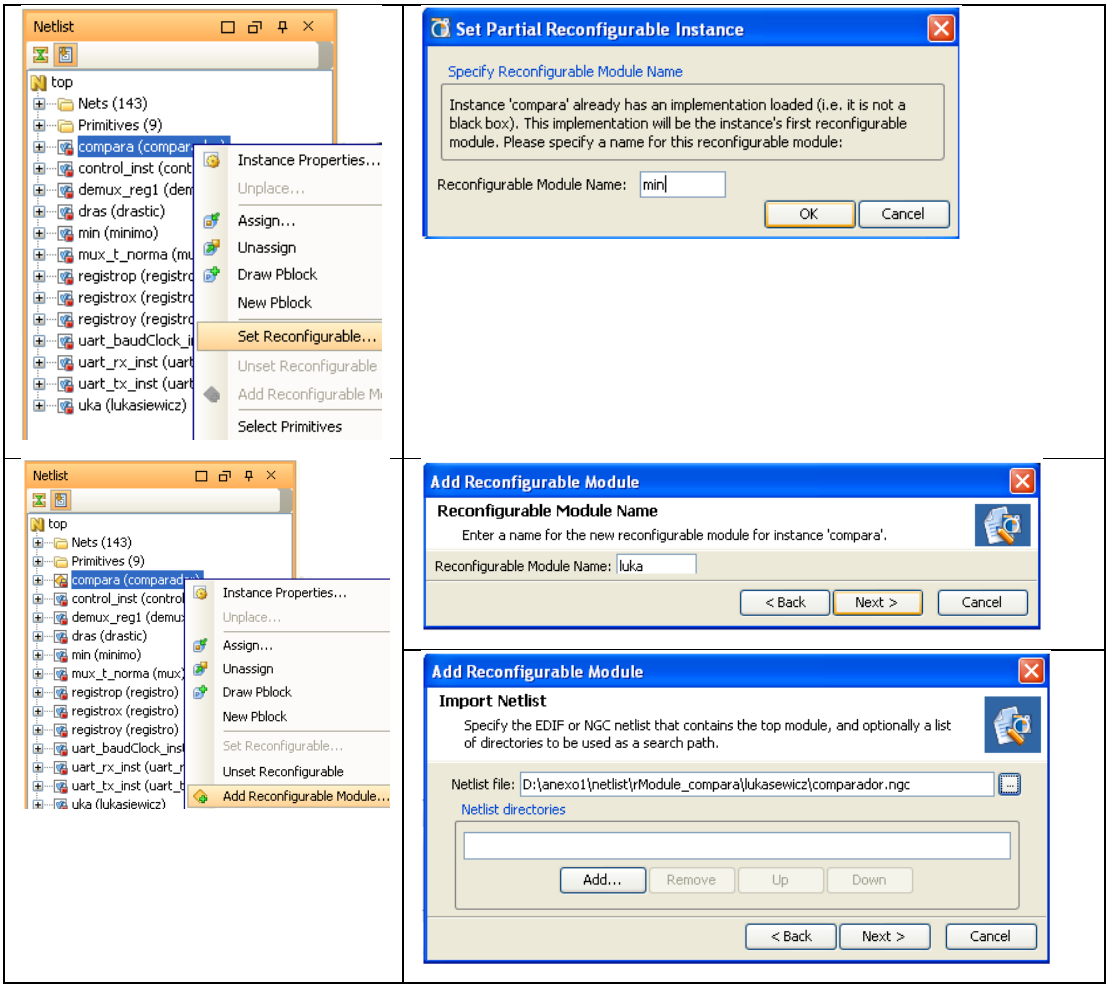
b) Colocación de busmacros que se relacionan con el módulo compara.

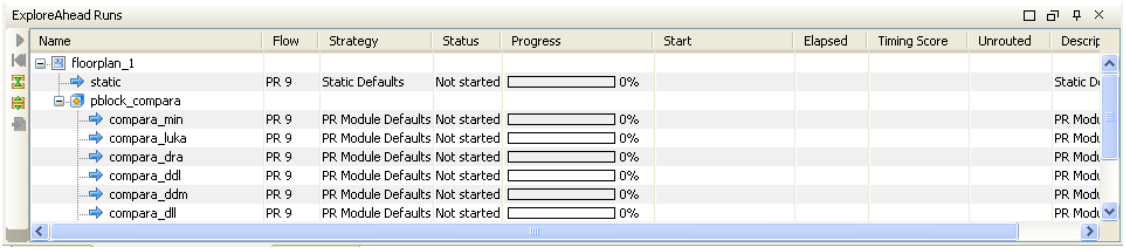
Figure A.12. Ubicación de las busmacros relacionados con el módulo reconfiurable

Nota que una de los CLB que implementa a los bus macros se ubica dentro del Pblock y el botro CLB se encuentra fuera del Pbloque. Ellos abarcan la frontera entre PRR y las regiones estáticas.

Agregar y configurar los RM´s

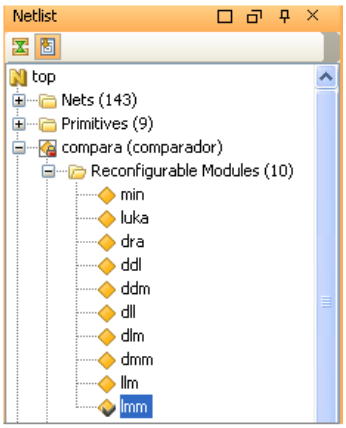
Ahora que hemos definido y dibujado los Pblocks, el siguiente paso es configurar las propiedades de los RM, y definir los nombres de los módulos. Seleccionar el Pblock **reconfig_compara**, dar click-derecho, y seleccionar **Set Reconfigurable…** Teclear **MIN** en el campo Reconfigurable Module Name y dar click en el botón **OK** (el archivo netlist de este módulo fue elegido al crear el proyecto). Nota que en la pestaña **ExploreAhead Runs** se muestran la región reconfigurable con un RM

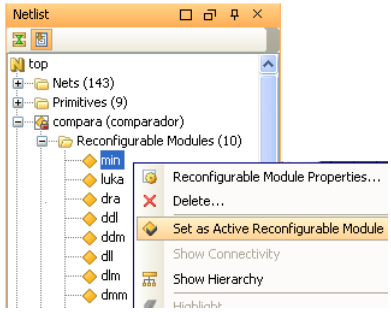




En nuestro diseño, se estipulo la PRR contendría 21 RMs. Seleccionar el Pblock **reconfig_compara**, dar click-derecho, y seleccionar **Add Reconfigurable Module…** Click **Next**, y teclear **LUK** en el campo Reconfigurable Module Name, y dar click en **Next** Click el botón Browse, y buscar en el directorio **OperadoresParametricos\Synth\rModule_compara\LUK**, seleccionar el archivo **rModule_compara.ngc**, y dar click en **Next.** Click sobre **Next** y **Finish** para agregar el RM.

Expandir la carpeta **reconfig_compara** en el panel Netlist y observa que se ha agregado el RM en la PRR. Repetir este proceso para los otros 19 operadores. Ver que el último operador implementado se ha seleccionado como inicial, para ser usado en el static_full.bit.

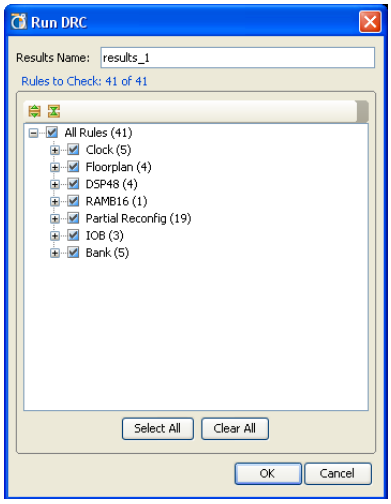




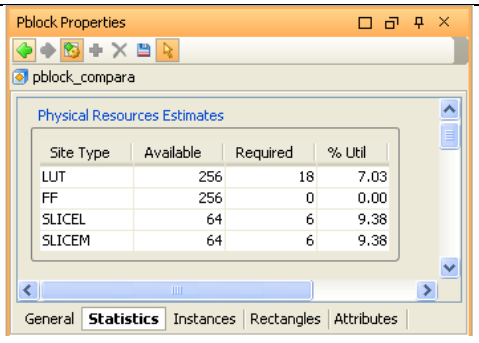
Dar click-derecho, y seleccionar **Set as Active Reconfigurable Module**, sobre el módulo **MIN,** para configurarlos RM activo. Verifica en la pestaña **ExploreAhead Runs** ahora muestra todos RM que se han agregado en la PRR

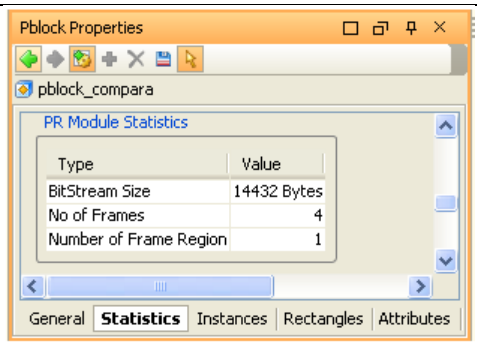
Verificar las reglas de diseño (DRC) y ver información

Antes de correr el flujo de implementación PR, es necesario correr la verificación de las reglas de diseño DRC y corregir los errores de las reglas si es necesario. En PlanAhead, seleccionar **Tools** → **Run DRC …** click **OK** y correr los DRC, Debe de reportar no violaciones



Puede observarse la cantidad de recursos disponibles para las PRR y los utilizados por cada RM. Para hacer esto, seleccionar el pblock reconfig_compara, y seleccionar la pestaña **Statistics** del panel **Pblock Properties**. El **% de utilización** debe ser menor del 100%, en otro caso, el Pblock se debe expandir para suministrar suficientes recursos

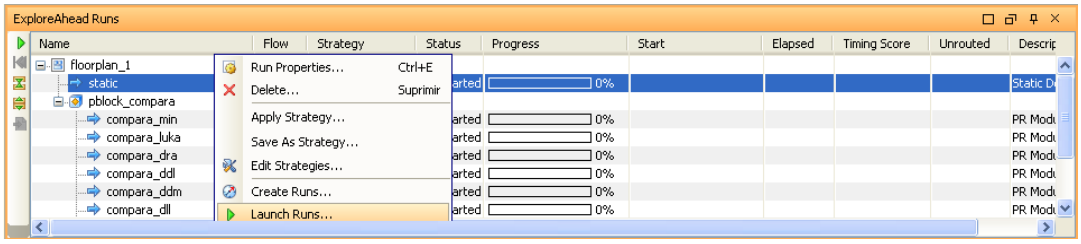




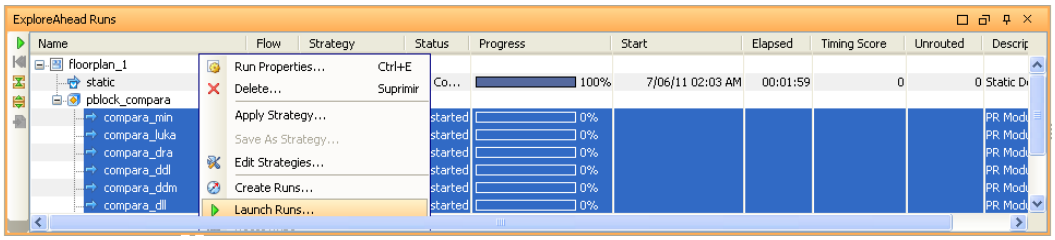
Puede verse también el tamaño del archivo bitstreams de la PRR usando la pestaña PR Módule statistics y despalzando el scrolling en la región correcta

Correr el Flujo de implementación PR

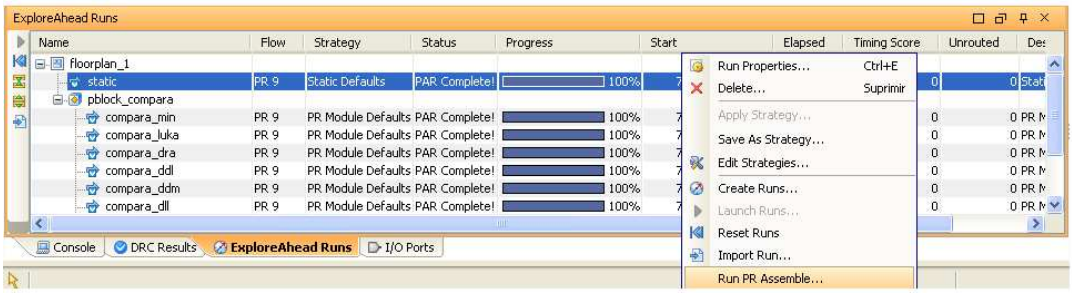
Después de haber creado los RM de la PRR, y ejecutado exitosamente los DRC, se puede correr el flujo de implementación PR. El primer paso en el flujo de implementación PR es correr la configuración inicial es decir la implementación de los módulos estático. Seleccionar **static** en la pestaña **ExploreAhead Runs**, dar click-derecho, y seleccionar **Launch Run … Click OK para completar la implementación**



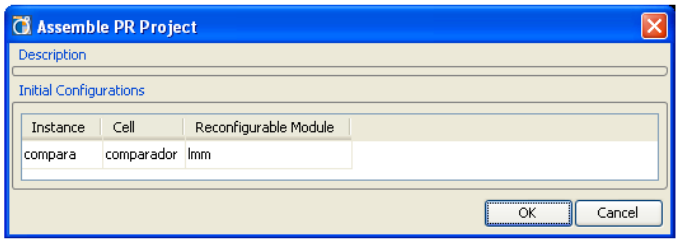
Una vez que se corrió y se completó exitosamente la implementación del pblock **static** , el siguiente paso es correr la implementación de cada uno de los RM de la PRR. Seleccionar todos los RM y dar click-derecho en **Launch Runs…** para iniciar la implementación (Si tu computadora tiene más de un procesador, puedes usar todos, seleccionando el número de procesadores disponibles, esto aumentará la velocidad del proceso de implementación)



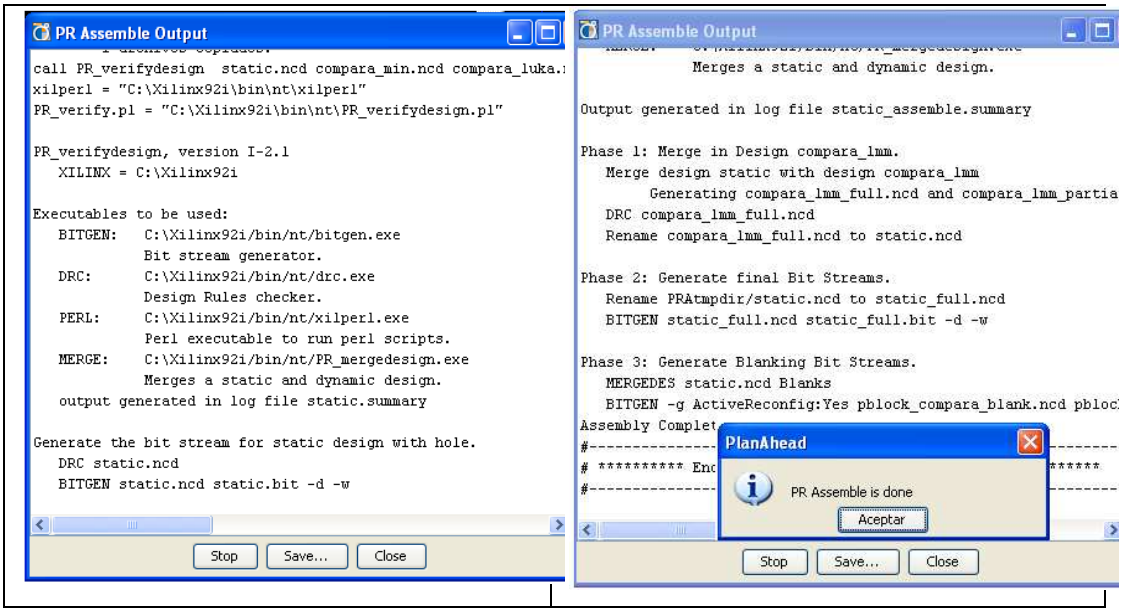
El último paso en el flujo de implementación es correr, ensamblar y verificar el diseño PR, que puede ser lanzado simultáneamente dando click-derecho en uno de los RM y seleccionando **Run PR Assemble …**



Una ventana se mostrará, seleccionando el RM inicial incluido en el bitstream static_full. Seleccionar **MIN** como el RM inicial y dar click en **OK** para iniciar este paso.



Cuando este paso es completado, click en los botones **OK** y **Close.** Usando el Explorador de Windows, buscar el directorio **lab2\project_1\project_1.runs\floorplan_1\merge** y observa los archivos bitstream generados.



Creación de Proyecto con IMPACT

La última fase del diseño de un sistema dinámicamente reconfigurable es realizar pruebas usando los bitstreams static_full.bit and los bitstreams partiales generados. Iniciar el programa iMPACT usando **Start** → **Programs** → **Xilinx ISE 9.1i** → **Accessories** → **iMPACT.** Seleccionar **create a new project and scan the JTAG chain.** Cuando el programa iMPACT detecta el dispositivo en el modo JTAG, click en el botón **Cancel All.** Click-derecho sobre el dispositivo FPGA en el modo JTAG, y seleccionar y seleccionar **Assign New Configuration File …** Buscar el directorio **OperadoresParametricos\merges** y seleccionar el archivo **static_full.bit.** Click-derecho sobre el dispositivo FPGA y seleccionar la configuración del dispositivo. Se puede poner marcha lel software que se diseño en este trabajo, cuando se hacer un cambio del operador, se realiza de manera dinámica. Tambien el sistema puede ser verificado con la hiperterminar

Cerrar el software iMPACT sin hacer guardar. Cerrar el PlanAhead, guardando el proyecto trabajado.

6. Referencias.

Chenn-Jung Huang, Wei-Kuang Lai, Sheng-Yu Hsiao, Hao-Yu Liu, and Rui-Lin Luo. A Bluetooth Routing Protocol Using Evolving Fuzzy Neural Networks. International Journal of Wireless Information Networks, Vol. 11, No. 3 (2004). Pp.307-317.

Chowdhury, S.R.; Saha, H.: A High-Performance FPGA-Based Fuzzy Processor Architecture for Medical Diagnosis. Micro, IEEE (2008) , pp. 38

Christos Georgoulas, Ioannis Andreadis. A real-time fuzzy hardware structure for disparity map computation. Journal of Real-Time Image Processing, Online First™, 2010.

Nambakhsh M, Tavakoli V, Sahba N. FPGA-Core Defibrillator using Wavelet-Fuzzy ECG Arrhythmia Classification. Engineering in Medicine and Biology Society, 2008. EMBS 2008. 30th Annual International Conference of the IEEE(2008). Pp 2673

Haider, T. ; Yusuf, M. ; FPGA Based Fuzzy Link Cost Processor for Energy-Aware Routing in Wireless Sensor Networks - Design and Implementation. 9th International Multitopic Conference, IEEE INMIC (2006). Pp. 1

Kawai, H. ; Yamaguchi, Y. ; Yasunaga, M. ; Glette, K. ; Torresen, J. ; An adaptive pattern recognition hardware with on-chip shift register-based partial reconfiguration. ***International Conference on Electrical and Computer Engineering (*** 2009), pp.169

Lund, T. ; Aguirre, M. ; Torrala, A. ; Fuzzy logic control via an FPGA: a design using techniques from digital signal processing. 2004 IEEE International Symposium on Industrial Electronics (2004), Vol.1 pp. 555

Tipsuwanpom, Runghimmawan., T ; Runghimmawan, T. ; Intajag, S. ; Krongratana, V. ; Fuzzy logic PID controller based on FPGA for process control. 2004 IEEE International Symposium on Industrial Electronics (2004), Vol.2 pp. 1495

Zadeh, L. A. Fuzzy set. Information and Control,8 333353 (1965)

Ross J. Timothy. **FUZZY LOGIC WITH ENGINEERING APPLICATIONS**. Second Edition, John Wiley & Sons Ltd, 2004

Terano, T., Asai, K., Sugeno, M.: Applied Fuzzy Systems. Academic Press Professional, San Diego (1994)

Marco Platzner, Jürgen Teich, Norbert When. **Dynamically Reconfigurable Systems: Architectures, Design Methods and Applications.** Springer, 2010.

Jang, J.-S.R., Sun, C.T., Mizutani, E.: Neuro-Fuzzy and Soft Computing. A Computational approach to Learning and Machine Intelligence. Prentice-Hall International (1997)

Yen, J., Langari, R., Zadeh, L.A.: **Industrial Applications of Fuzzy Logic and Intelligent** Systems.IEEE Press, NJ (1995)

Cervinka, O.: Automatic Tuning of Parametric T-norms and T-conorms in Fuzzy Modeling. In: 7th IFSA World Congress, pp. 416--421. Prague (1997)

Batyrshin, I., Kaynak, O., Rudas, I.: Generalized Conjunction and Disjunction Operations for Fuzzy Control. In: 6th European Congress on Intelligent Techniques & Soft Computing, EUFIT'98, vol. 1, pp. 52--57. Aachen, Germany (1998)

Batyrshin, I., Kaynak, O.: Parametric Classes of Generalized Conjunction and Disjunction Operations for Fuzzy Modeling. IEEE Transactions Fuzzy Systems, 7, 586--596 (1999)

Bikbulatov, A., Batyrshin, I.: Tuning of Operations in Fuzzy Models by Neural Nets. In: 7th Zittau Fuzzy Colloquium, pp. 142—147. Zittau, Germany (1999)

Eskil, M.T., Efe, M.O., Kaynak, O.: T-norm Adaptation in Fuzzy Logic Systems Using Genetic Algorithms. In: ISIE'99, IEEE Intern. Symposium on Industrial Electronics, vol. 1, pp. 398--402. Bled, Slovenia (1999)

Batyrshin, I., Kaynak, O., Rudas, I.: Fuzzy Modeling Based on Generalized Conjunction Operations. IEEE Transactions Fuzzy Systems, 10, pp. 678—683 (2002)

Koprinkova-Hristova, P.D.: Fuzzy Operations' Parameters Versus Membership Functions' Parameters Influence on Fuzzy Control Systems Properties. In: 2nd IEEE Int. Conf. on Intelligent Systems, pp. 219--224 (2004)

Alcalá-Fdez, J., Herrera, F., Márquez, F., Peregrín, A.: Increasing Fuzzy Rules Cooperation Based on Evolutionary Adaptive Inference Systems. Intern. J. Intelligent Systems, 22, pp. 1035-- 1064 (2007)

Aras, A.C., Kaynak, O., Batyrshin, I.Z.: A Comparison of Fuzzy Methods for Modeling. In: IECON 2008, 34th Annual Conf. of the IEEE Industrial Electronics Society, pp. 43—48. Orlando (2008)

McKenna, M., Wilamowski, B.M.: Implementing a Fuzzy System on a Field Programmable Gate Array. In: IJCNN'01, Intern. Joint Conf. Neural Networks, vol.1, pp. 189--194.Washington, DC (2001)

Sánchez-Solano, S., Senhadji, R., Cabrera, A., Baturone, I., Jiménez, C.J., Barriga, A.: Prototyping of Fuzzy Logic–Based Controllers Using Standard FPGA Development Boards. In: RSP'2002, 13th IEEE International Workshop on Rapid System Prototyping, pp. 25—32. Darmstadt (2002)

Mermoud, G., Upegui, A., Peña, C., Sanchez, E.: A Dynamically-Reconfigurable FPGA Platform for Evolving Fuzzy Systems. In: LNCS, vol. 3512, pp. 572--581. Springer, Heidelberg (2005)

Di Stefano, A., Giaconia, C.: An FPGA-Based Adaptive Fuzzy Coprocessor. In: LNCS, vol. 3512, pp. 590--597. Springer, Heidelberg (2005)

Deliparaschos, K.M., Nenedakis, F.I., Tzafestas, S.G.: Design and Implementation of a Fast Digital Fuzzy Logic Controller Using FPGA Technology. J. Intelligent Robotic Systems, 45, pp. 77—96 (2006)

Sanchez-Solano, S., Cabrera, A.J., Baturone, I., Moreno-Velo, F.J., Brox, M.: FPGA Implementation of Embedded Fuzzy Controllers for Robotic Applications. IEEE Trans. Indust. Electronics, 54, pp.
1937--1945 (2007)

Govindasamy, K., Neeli, S., Wilamowski, B.M.: Fuzzy System with Increased Accuracy Suitable for FPGA Implementation. In: INES 2008, Intelligent Engineering Systems Conf., pp. 133--138 (2008)

Montiel, O, Olivas, J., Sepúlveda, R., Castillo, O.: Development of an Embedded Simple Tuned Fuzzy Controller. In: WCCI 2008, IEEE World Congress Computational Intelligence, FUZZ-IEEE, pp. 555--561. Hong Kong (2008)

Montiel, O., Maldonado, Y., Sepúlveda, R., Castillo, O.: Simple Tuned Fuzzy Controller Embedded into an FPGA. In: NAFIPS 2008 Conference Proceedings. New York (2008)

Hemandez Zavala, A., Batyrshin, I.Z., Rudas, I.J., Villa Vargas L., Camacho Nieto, O.: Parametric Operations for Digital Hardware Implementation of Fuzzy Systems. In: LNAI, vol. 5845, pp. 432---443. Springer, Heidelberg (2009)

Huang, Y. J. , Wu, B.C., Chen, C.Y., Chang, C.H., Kuo, T.C.: Solar Tracking Fuzzy Control System Design Using FPGA. In: World Congress on Engineering, vol I, pp. 340--344. London (2009)

Obaid, Z.A., Sulaiman, N., Hamidon, M.N.: FPGA-Based Implementation of Digital Logic Design Using Altera DE2 Board. Int. J. Computer Science and Network Security, 9, pp. 186—193 (2009)

Rudas, I.J., Batyrshin, I.Z., Hernández Zavala, A., Camacho Nieto, O., Villa Vargas, L.: Digital Fuzzy Parametric Conjunctions for Hardware Implementation of Fuzzy Systems. In: ICCC2009, IEEE 7th Int. Conf. Computational Cybernetics, pp. 157--166, Palma de Mallorca, Spain (2009)

Klement, E.P. , Mesiar, R., Pap, E. Triangular Norms. Kluwer, Dordrecht (2000)

Batyrshin, I.Z., Rudas, I.J., Panova, A.: On Generation of Digital Fuzzy Parametric Conjunctions. In: Studies in Computational Intelligence, vol. 243, pp. 79--89. Springer, Heidelberg (2009)

Cortés Antonio, P., Batyrshin, I., Rudas, I., Panova, A., Villa Vargas, L.: FPGA Implementation of (p)-Monotone Sum of Basic t-norms. In: WCCI 2010, IEEE World Congress on Computational Intelligence, FUZZ-IEEE, pp. 1491--1497, Barcelona, Spain, (2010)

Kilts, S.: Advanced FPGA Design. Architecture, Implementation, and Optimization. John Wiley & Sons, New Jersey (2007) 27. Brown, S., Vranesic, Z.: Fundamentals of Digital Logic with VHDL Design. 2nd ed. Mc Graw Hill (2005)

Palnitkar, S.: Verilog HDL: A Guide to Digital Design and Synthesis. 2nd ed. (2008)

Cyclone II Device Handbook, Vol. 1, Altera, 2008, http://www.altera.com/literature/hb/cyc2/cyc2_cii5v1.pdf

DE2 Development and Education Board User Manual. User Manual. Versión 1.4. Altera, 2006,
ftp://ftp.altera.com/up/pub/Webdocs/DE2_UserManual.pdf

Xilinx Inc., System Generator User Guide, http://www.xilinx.com