Design of a Programmable and Modular Analog Fuzzy Controller

By

José Joaquín De Jesús López

Thesis submitted in partial fulfillment of

the requirements for the degree of

MASTER OF SCIENCE

IN

ELECTRICAL ENGINEERING

UNIVERSITY OF PUERTO RICO

MAYAGÜEZ

2004

Approved by:

[redacted]	17/05/04
Rogelio Palomera, Ph.D. President, Graduate Committee	Date
[redacted]	05/17/2004
Manuel Jiménez, Ph.D. Member, Graduate Committee	Date
[redacted]	May 17, 2004
Gerson Beauchamp, Ph.D. Member, Graduate Committee	Date
[redacted]	05/17/2004
Gladys Di Cristina, M.S. Representative of Graduate Studies	Date
[redacted]	May 17, 2004
Jorge Ortiz Alvarez, Ph.D., PE Chairperson of the Deparment	Date

Abstract

Design techniques for analog fuzzy integrated circuits (ICs) and control implementations are used to develop a programmable and modular analog fuzzy controller. It improves flexibility and accuracy of analog fuzzy systems and makes them suitable for diverse applications.

Programmability of the gaussian-type function fuzzifier as well as current mode designs both for an n-input “minimum extracting” circuit with 4n+1 transistors and for an analog multiplier-divider circuit based on operational transconductance amplifiers are presented.

The designed circuitry consists solely of Metal-Oxide-Semiconductor (MOS) transistors compatible with standard Complementary-Metal-Oxide-Semiconductor (CMOS) fabrication processes of the MOSIS company.

Compendio

Las técnicas de diseño para circuitos integrados analógicos difusos y para implementaciones en control son usadas para desarrollar un controlador analógico difuso programable y modular. Este mejora la flexibilidad y precisión de los sistemas analógicos difusos haciéndolos más convenientes para diversas aplicaciones.

Se presenta un fuzificador cuya función de membresía tipo gausiana es programable, un circuito en modo de corriente que calcula el mínimo de n-entradas con 4n+1 transistores y un circuito en modo de corriente multiplicador-divisor analógicos basado en amplificadores de transconductancia.

Los circuitos diseñados están compuestos sólo por semiconductores de óxidos metálicos, los cuales son compatibles con la tecnología estándar de semiconductores complementarios de óxidos metálicos del proceso de fabricación de circuitos integrados de la compaía MOSIS.

Dedication

To God and my family.

Just want to say thanks for always being there for me even though I always thought I didn't need you. I appreciate now more than ever all of the things you have done and still are doing for me. Specially mommy, daddy, Karen and Edgar, thanks for their encouragement and love. And also you, my dear friend, who helped me enduring these years.

Acknowledgment

First I would like to thank University of Puerto Rico for giving me the opportunity to be here. My deepest thanks to my thesis advisor, Dr. Rogelio Palomera; his help and support in the writing of this thesis will always be greatly appreciated and I thanks him for his friendship. Thanks to committee members Dr.Manuel Jiménez and Dr.Gerson Beauchamp for their support. I would also like to express sincere gratitude to my friends, Oscar, Irvin, and Rafael for their friendship. I also want to express my gratitude to Dr. Carlos Cuadros for his invaluable help, support and for the encouraging discussion and guidance.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Control theory deals with the behavior of dynamical systems over time. A system model is a mathematical representation of an engineering phenomenon that involves an input signal, an output signal, and a state function. When one or more output variables of a plant need to show certain behavior over time, a controller may be able to manipulate the plant inputs to realize this behavior at its output, in which case a control system is created.

Figure 1.1 shows a block diagram of an open-loop control system. The system is called open-loop because the output of the plant is not used to modify the controller's input. One of the main disadvantages of this type of control system is its sensitivity to the dynamics of the system under control. To solve this problem, control theory introduces feedback or closed-loop control system (Figure 1.2). For these control system the output of the plant is used by the controller to determine the input to the plant. The control problem is then of synthesizing a controller such that the closed-loop system meets the behavior objectives. General control objectives include: stability of the state function, performance optimization, robustness with respect to signals noise and poorly modeled dynamics, and avoidance of unsafe states.

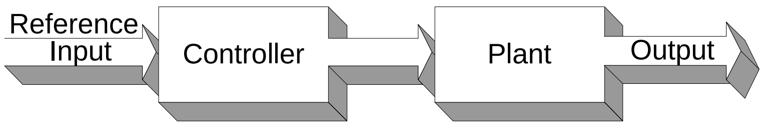

Figure 1.1: Open-loop control system

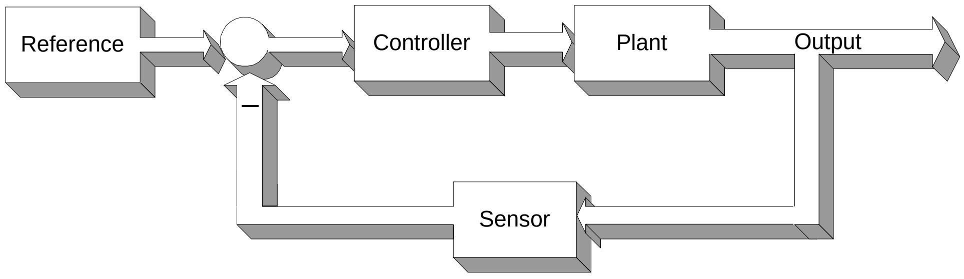

Figure 1.2: Closed-loop control system

Classical control theory has been successfully applied to a large variety of control processes. However, complex, nonlinear, time varying systems or systems suffering from noisy measurements are very difficult to control because of the mathematics involved.

The main idea behind fuzzy control is to build a model capable of controlling the plant without necessarily having a mathematical model as those in modern control theory. Instead, a set of rules is used to control the plant. These rules can be derived in different ways [1].

Fuzzy control is useful when a workable mathematical model for a plant is very difficult or impossible to derive or the mathematical model is very complex and computing it in real time is not an option.

Fuzzy controllers are conceptually simple. They consist of an input stage, a rule processing stage, and an output stage. The input stage maps inputs from sensor or other devices to generate an appropriate membership function or truth value which are fed into the processing stage. The processing stage invokes rules and generates a result calculated from the rules' outputs. Finally, the output stage converts the combined result back into a specific control output value.

Practical Fuzzy Logic Systems (FLS) realizations may be implemented with software and/or hardware systems. Hardware implementation of fuzzy logic have shorter response times, and are preferred in applications demanding real time performance and/or low area occupation and power consumption [2]. The hardware realization may be either digital or analog. In most cases, digital implementations have greater flexibility through their external memory, although they are usually slower than their analog counterparts, for similar logic performance. The main drawback of analog hardware implementations is limited complexity and modifiability of the system. This characteristic makes most realizations limited to specific applications. Another drawback is the higher cost of developing analog circuitry.

On the other hand, analog realizations are preferred for their high speed and low area, and power consumption, when designing fully parallel and multivalued systems. Furthermore, since sensors and actuators employed in many applications are continuous, analog fuzzy controller chips look more attractive than their digital counterpart because the latter requires Analog-to-digital (A/D) and Digital-to-Analog (D/A) converters.

1.1 Objectives

The goal of this work is to design a programmable analog fuzzy controller to improve the flexibility of analog fuzzy systems, and hence to make them adaptable to different applications or plant control.

Programmability is achieved at the input and output stages through voltage or current levels. The circuits that provide these voltages i.e., Low Dropout Regulators (LDO), voltage regulators, and so on are not treated in this work.

In addition, a Sugeno's inference engine [3] and a regular and modular minimum extracting circuit structure are combined to provide the expandability needed to accommodate multiple inputs. Furthermore, the single-stage structure used minimizes accumulated errors and maximizes operation speed.

The methodology used in the design follows a combination of top-down and bottom-up approaches. The design tasks at each level consisted of sequences of synthesis, analysis, and verification steps. Synthesis steps were devoted to finding solutions to the problems that appeared at each level. Analysis tasks measured the system characteristics according to each design level goal, and finally, the verification steps were applied to the set of testing methods (simulations) that evaluated if the solutions proposed at each level met the specifications and whether or not they altered the global system behavior.

The analog design styles adopted for the building blocks and final implementation fall in the following categories:

- Analog design style : *Conventional continuous time schemes*[2]
- Building blocks

1. Fuzzification stage : *Transconductance-mode Membership Function Circuits*
2. Rule processing stage : *Current-mode Minimum Operator*
3. Output : *Sugeno's defuzzifier*

1.2 Thesis overview

A review of the mathematical underpinnings of fuzzy logic and fuzzy sets is given in Chapters 2, to provide a backdrop to present research. This chapter provide details on the areas of fuzzy sets and logic that are relevant to the work presented in later chapters.

Chapter 3 introduces the field of fuzzy Logic Systems. The structure of this system is analyzed in order to identify the various stages and building blocks considered in its hardware implementation. In addition a very brief review of control system theory is provided.

Chapter 4 explains analysis and synthesis steps devoted to the design of each Fuzzy Logic System stage.

Chapter 5 presents circuit synthesis of the system and its PSpice simulation results. Furthermore, an application example illustrates the performance of the proposed fuzzy controller and Matlab simulations validate these results.

Finally, Chapter 6 presents a summary and conclusions from this research and introduces directions for future work.

Chapter 2

Fuzzy Sets and Logic

2.1 Historical overview of fuzzy sets

The precision of mathematics owes its success in large part to the efforts of Aristotle and the philosophers who preceded him. In their efforts to devise a concise theory of logic and mathematics, the so-called "Laws of Thought"were posited [4]. One of these, the " Law of the Excluded Middle,"states that every proposition must either be True or False. When Parminedes proposed the first version of this law around 400 B.C., there were already strong and immediate objections; for example, Heraclitus proposed that things could be simultaneously True and not True.

Plato laid the foundation for what would become fuzzy logic, indicating that there was a third region beyond True and False where these opposites "tumbled about". But it was Lukasiewicz who first proposed a systematic alternative to the bi-valued logic of Aristotle [5].

In the early 1900's, Lukasiewicz described a three-valued logic, along with the mathematics to accompany it. The third value he proposed can best be translated as the term “possible,” and he assigned it a numeric value between True and False [5]. Eventually, he proposed an entire notation and an axiomatic system from which he hoped to derive modern mathematics.

Later, he explored four-valued and five-valued logics, and then declared that in principle there was nothing to prevent the derivation of an infinite-valued logic. Lukasiewicz felt that three- and infinite-valued logics were the most intriguing, but he ultimately settled on a four-valued logic because it seemed to be the most easily adaptable to Aristotelian logic [5].

Knuth proposed a three-valued logic, similar to Lukasiewicz’s, from which he speculated that mathematics would become even more elegant than in traditional two-values logic [6]. His insight, apparently missed by Lukasiewicz, was to use the integral values {-1, 0, +1} rather than {0, 1, 2}. Nonetheless, this alternative failed to gain acceptance and has passed into relative obscurity.

The notion of fuzzy set appears for the first time in a memo from the University of California at Berkeley written by Lotfi A. Zadeh in 1964 . It was later published in The Information and Control Journal [7]. From that time on, it has served as the foundation for numerous papers on fuzzy sets from authors all over the world and set the basis for fuzzy logic. Fuzzy logic is basically a logic with multiple values, that allows values between the conventional evaluations of the precise logic 1 and 0. It also includes operations for ‘and’, ‘or’, ‘not’ and ‘if-then’.

Hence, fuzzy logic extends conventional Boolean logic to handle the concept of the partial truth – the values falling between “ totally true” and “ totally false ”. These values are dealt with using degree of membership of an element to a set. The degree of membership can take any real value in the interval [0,1]. Fuzzy logic makes it possible to imitate the behavior of human logic, which tends to work with “fuzzy” concepts of truth.

Although fuzzy control was not the first engineering application of fuzzy logic, it was the first application that drew huge attention to the practical potential of fuzzy set theory [1]. It uses many elements of fuzzy logic to define a rule-base for the controller .

2.2 Elements of Fuzzy Sets

2.2.1 Basic Concepts

In classical or crisp sets, an element in the universe has a well defined membership or non membership to a given set. Membership to a crisp set F can be defined through a membership function defined for every element x of the universe as

$$\mu_F(x) = \begin{cases} 1 & x \in F \\ 0 & x \notin F \end{cases} \tag{2.1}$$

An example of a graphic for the membership function of a crisp set is illustrated in Figure 2.1. Here, scanning the universe, there is an abrupt and well-defined transition from membership to non membership and vice versa. It is said to be “crisp”.

For an element in a universe with fuzzy sets, the membership transition can be gradual. So the membership function can take any value between 0 and 1. This transition among various degrees of membership can be thought of as conforming to the fact that the boundaries of the fuzzy sets are vague and ambiguous. Fuzzy membership counterpart for Figure 2.1 would be the Figure 2.2. Hence, membership of an element from the universe in this set is measured by a function that attempts to

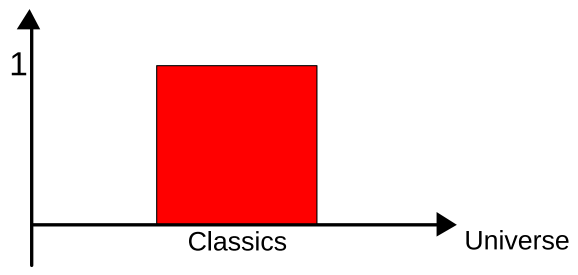

Figure 2.1: Classic sets

describe vagueness and ambiguity. In fuzzy logic, linguistic variables take on linguistic values which are words (linguistic terms) with associated degrees of membership in the set. Thus, instead of a variable height assuming a numerical value of 1.75 meters, it is treated as a linguistic variable that may assume, for example, linguistic values of tall with a degree of membership of 0.92, "very short" with a degree of 0.06, or "very tall" with a degree of 0.7. Each linguistic term is associated with a fuzzy set, each of which has a defined membership function (MF).

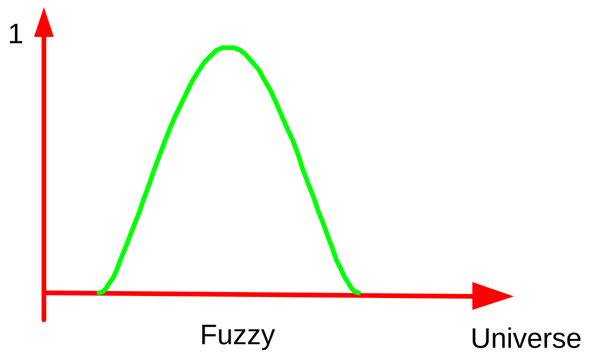

Figure 2.2: Fuzzy sets

Formally, a fuzzy set, is defined as a set of pairs where each element in the universe

U has a degree of membership associated with it:

$$F = \{(x, \mu_F(x)) \mid x \in U,\ \mu_F(x) \in [0,1]\} \tag{2.2}$$

$\mu_F(x)$ is known as the membership function of the set F. Most often, one refers to the fuzzy set just by mentioning the membership function, the universe being implicit.

The value $\mu_F(x)$ is the degree of membership of object x to the fuzzy set F. $\mu_F(x) = 0$ means that x does not belong at all to the set, while $\mu_F(x) = 1$ means that the element is totally within the set [7].

As an example consider the ambient temperature with the concepts of hot and cold. The characteristic functions in the classical logic for this example are given by

$$\mu_{Hot}(x) = \begin{cases} 1 & if\ x \geq 30°C \\ 0 & if\ x < 30°C \end{cases} \tag{2.3}$$

and

$$\mu_{cold}(x) = 1 - \mu_{Hot}(x) = \begin{cases} 1 & if\ x \leq 30°C \\ 0 & if\ x > 30°C \end{cases} \tag{2.4}$$

The 30 °C boundary in this example is arbitrary. Independently of this boundary value, classical logic cannot interpret intermediate values. A graph of the membership function for the classic temperature variable is shown in Figure 2.3.

On the other hand, fuzzy logic solves the crisp problem with membership functions such as given by

$$\mu_{Hot}(x) = \begin{cases} 0 & if\ x \leq 29°C \\ \frac{X-29}{2} & if\ 29°C <\ x\ < 31°C \\ 1 & if\ x \geq 31°C \end{cases} \tag{2.5}$$

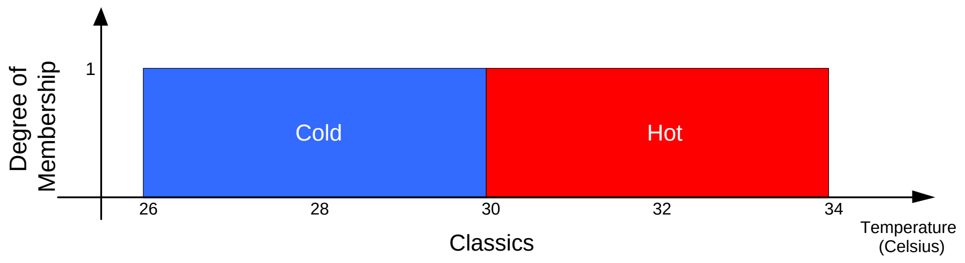

Figure 2.3: Classic temperature set

and

$$\mu_{Cold}(x) = 1 - \mu_{Hot}(x) = \begin{cases} 0 & if\ x \geq 31°C \\ \frac{31-X}{2} & if\ 29°C < \ x \ < 31°C \\ 1 & if\ x \leq 29°C \end{cases} \tag{2.6}$$

Figure 2.4 shows a representation of a fuzzy set using these membership functions. In this set, a 30.5°C temperature belongs 25% to cold and 75% to hot.

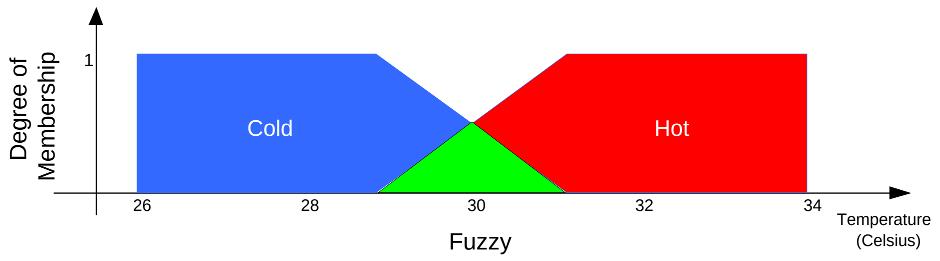

Figure 2.4: Fuzzy temperature set

Further refinements, such as introducing an intermediate warm temperature, can be made. A graph of the membership functions for the classic and fuzzy temperature sets are shown in Figure 2.5. In this Figure, temperature values between 29 and 31°C are represented by other classic and fuzzy sets. Values from 29 to 30°C belong, with different extent, both to cold and warm membership functions. Similarly, values from 30 to 31°C belong, with different extent, both to warm and hot membership functions.

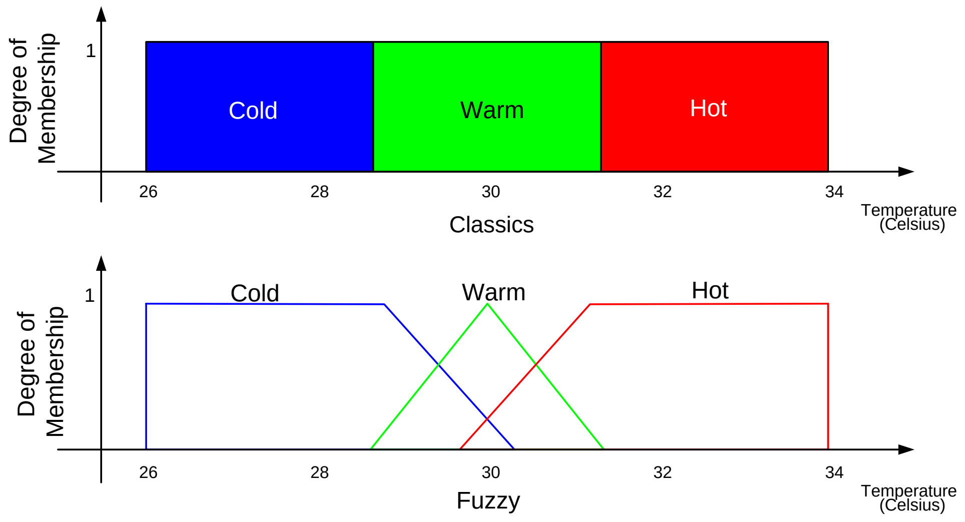

Figure 2.5: Fuzzy temperature set with more precision

In any case, the crisp nature is always present in classical sets, while fuzzy sets allow gradual membership to better adapt to our subjective criteria.

The membership function $\mu_F(u)$ describes, therefore, the degree of membership of the different elements of the speech universe from fuzzy set. The selection of the form of membership function is subjective and depends on the context. However, for practical reasons, triangular, trapezoidal and bell shape functions are the most commonly used in engineering applications. These are shown in Figure 2.6.

2.2.2 Basic operations on Fuzzy Sets

Fuzzy Sets can be operated with each other in the same way as classical sets.

Classical sets will be denoted by simple capital letters, like A, B, C, whereas fuzzy sets by adding a tilde underneath these letters, i.e. $\underset{\sim}{A}$, $\underset{\sim}{B}$, $\underset{\sim}{C}$.

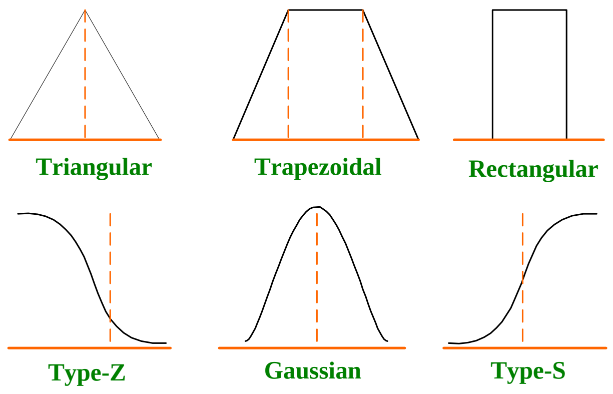

Figure 2.6: Different types of fuzzy set membership functions

Empty and Universal Fuzzy Sets

A fuzzy set is empty iff $\mu_\phi(x) = 0$ and universal iff $\mu_x(x) = 1 \; \forall x \in U$.

Equal Sets

Two fuzzy sets $\underset{\sim}{A}$ and $\underset{\sim}{B}$ are equal iff

$$\mu_{\underset{\sim}{A}}(x) = \mu_{\underset{\sim}{B}}(x) \quad \forall x \in U. \tag{2.7}$$

Absolute and Relative Complements

The absolute complement(NOT) of a fuzzy set $\underset{\sim}{A}$ is denoted by $\overline{\underset{\sim}{A}}$ and is defined by

$$\mu_{\overline{\underset{\sim}{A}}}(x) = 1 - \mu_{\underset{\sim}{A}}(x) \quad \forall x \in U. \tag{2.8}$$

Figure 2.7 shows absolute complements of fuzzy set.

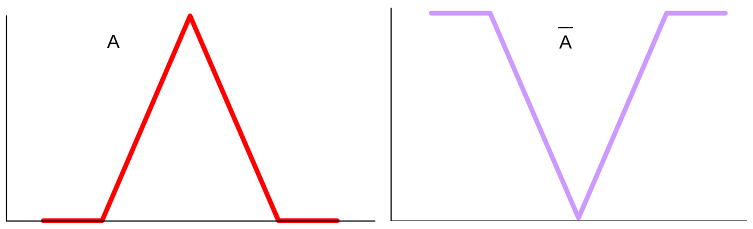

Figure 2.7: The Absolute Complement of a fuzzy set

The relative complement of $\underset{\sim}{A}$ with respect to $\underset{\sim}{B}$, denoted by $\underset{\sim}{B}$ - $\underset{\sim}{A}$, is defined by

$$\mu_{\underset{\sim}{B}-\underset{\sim}{A}}(x) = \mu_{\underset{\sim}{B}}(x) - \mu_{\underset{\sim}{A}}(x) \tag{2.9}$$

provided that

$$\mu_{\underset{\sim}{B}}(x) \geq \mu_{\underset{\sim}{A}}(x) \tag{2.10}$$

Figure 2.8 shows relative complements of fuzzy set.

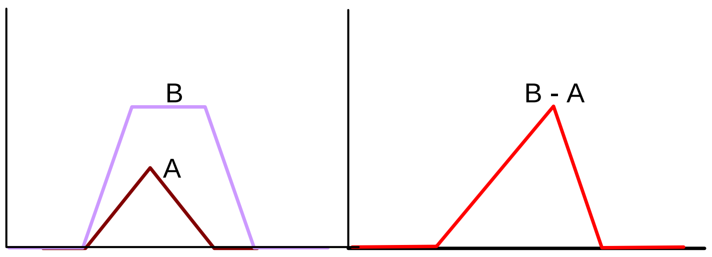

Figure 2.8: Relative complement of a fuzzy set

The Union of fuzzy sets

The Union of Fuzzy Sets $\underset{\sim}{A}$ and $\underset{\sim}{B}$ is a fuzzy set whose membership function for an element of the speech universe is the greatest or maximum of the membership functions of $\underset{\sim}{A}$ and $\underset{\sim}{B}$.

$$\mu_{\underset{\sim}{A} \cup \underset{\sim}{B}}(x) = max(\mu_{\underset{\sim}{A}}(x),\ \mu_{\underset{\sim}{B}}(x)) \tag{2.11}$$

Figure 2.9 shows union of fuzzy sets.

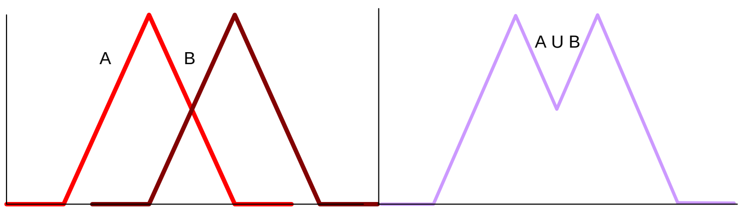

Figure 2.9: The Union on fuzzy sets

The intersection of fuzzy Sets

The intersection of fuzzy sets $\underset{\sim}{A}$ and $\underset{\sim}{B}$ is a fuzzy set whose membership function for an element of the speech universe is the lowest or minimum of the membership functions of $\underset{\sim}{A}$ and $\underset{\sim}{B}$.

$$\mu_{\underset{\sim}{A} \cap \underset{\sim}{B}}(x) = min(\mu_{\underset{\sim}{A}}(x),\ \mu_{\underset{\sim}{B}}(x)) \tag{2.12}$$

Figure 2.10 shows intersection of fuzzy sets.

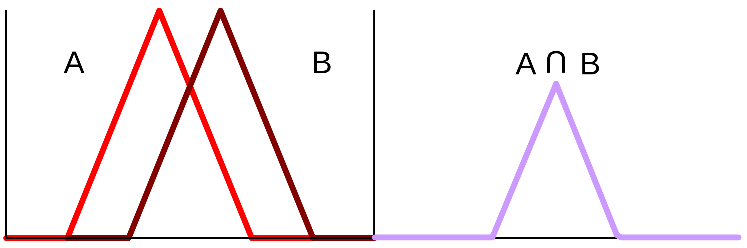

Figure 2.10: The Intersection of fuzzy Sets

Min and max operations for the intersection and union are not the only definitions that have been used. There are other possibilities that can be found in [8].

Cartesian Product

The previous operations assumed that the fuzzy sets are defined in the same universe. Cartesian Product does not require this hypothesis.

The Cartesian product of a fuzzy sets $\underset{\sim}{A}$ on universe X and fuzzy set $\underset{\sim}{B}$ on universe Y is a fuzzy set $\underset{\sim}{C}$, which is contained within the full Cartesian product space of the universe, or

$$\underset{\sim}{A} \times \underset{\sim}{B} = \underset{\sim}{C} \subset X \times Y \tag{2.13}$$

where the fuzzy set $\underset{\sim}{C}$ has a membership function defined as

$$\mu_{\underset{\sim}{C}}(x, y) = \mu_{\underset{\sim}{A} \times \underset{\sim}{B}}(x, y) = min(\mu_{\underset{\sim}{A}}(x), \mu_{\underset{\sim}{B}}(y)) \tag{2.14}$$

2.3 Other algebraic operations

Algebraic Product

The product of two fuzzy sets $\underset{\sim}{A}$ y $\underset{\sim}{B}$ in the same universe of discourse is the new fuzzy set $\underset{\sim}{A} \cdot \underset{\sim}{B}$ with a membership function that equals product of the membership function of $\underset{\sim}{A}$ and the membership function of $\underset{\sim}{B}$.

$$\mu_{\underset{\sim}{A} \cdot \underset{\sim}{B}} = \mu_{\underset{\sim}{A}}(x) \cdot \mu_{\underset{\sim}{B}}(x) \tag{2.15}$$

Figure 2.11 shows algebraic product of fuzzy sets.

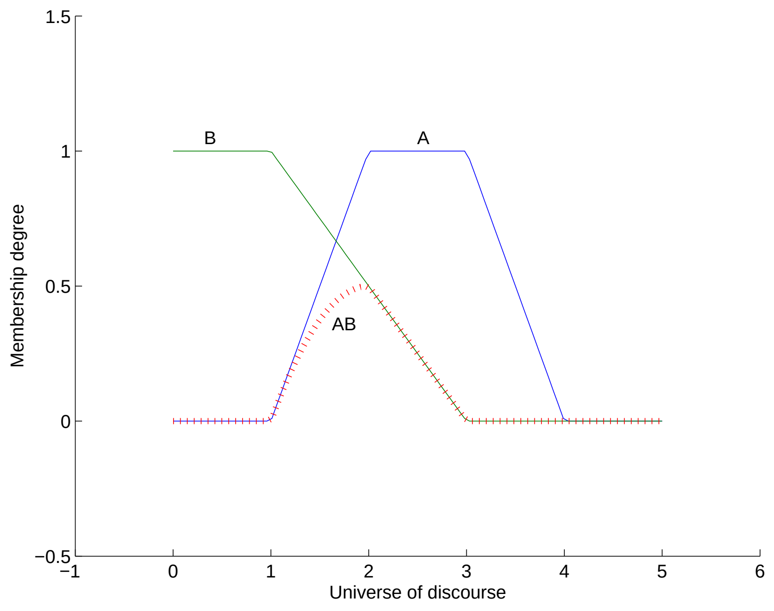

Figure 2.11: The Algebraic Product of Fuzzy Sets

Algebraic Sum

The algebraic sum of fuzzy sets $\underset{\sim}{A}$ and $\underset{\sim}{B}$ is a fuzzy set $\underset{\sim}{C}$,

$$\underset{\sim}{C} = \underset{\sim}{A} + \underset{\sim}{B}, \tag{2.16}$$

where

$$\mu_{\underset{\sim}{C}}(x) = \mu_{\underset{\sim}{A}}(x) + \mu_{\underset{\sim}{B}}(x) - \mu_{\underset{\sim}{A}}(x)\mu_{\underset{\sim}{B}}(x) \tag{2.17}$$

Figure 2.12 shows algebraic sum of fuzzy sets.

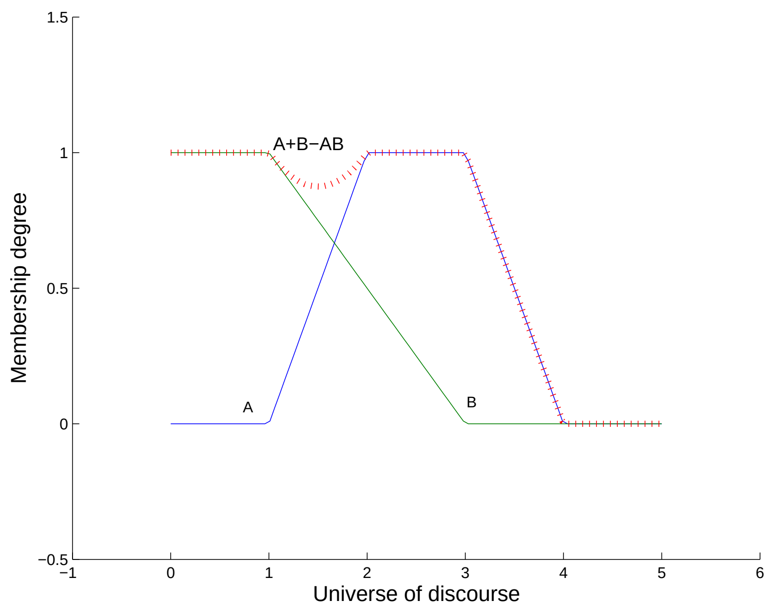

Figure 2.12: The Algebraic Sum of Fuzzy Sets

Bounded Sum

The bounded sum of two fuzzy sets $\underset{\sim}{A}$ and $\underset{\sim}{B}$ is denote by

$$\underset{\sim}{C} = \underset{\sim}{A} \oplus \underset{\sim}{B} \tag{2.18}$$

where

$$\mu_{\underset{\sim}{c}}(x) = min(1, \mu_{\underset{\sim}{A}}(x) + \mu_{\underset{\sim}{B}}(x)) \tag{2.19}$$

Figure 2.13 shows bounded sum of fuzzy set.

Bounded Difference

The bounded difference of two fuzzy sets $\underset{\sim}{A}$ and $\underset{\sim}{B}$ is denote by

$$\underset{\sim}{C} = \underset{\sim}{A} \ominus \underset{\sim}{B} \tag{2.20}$$

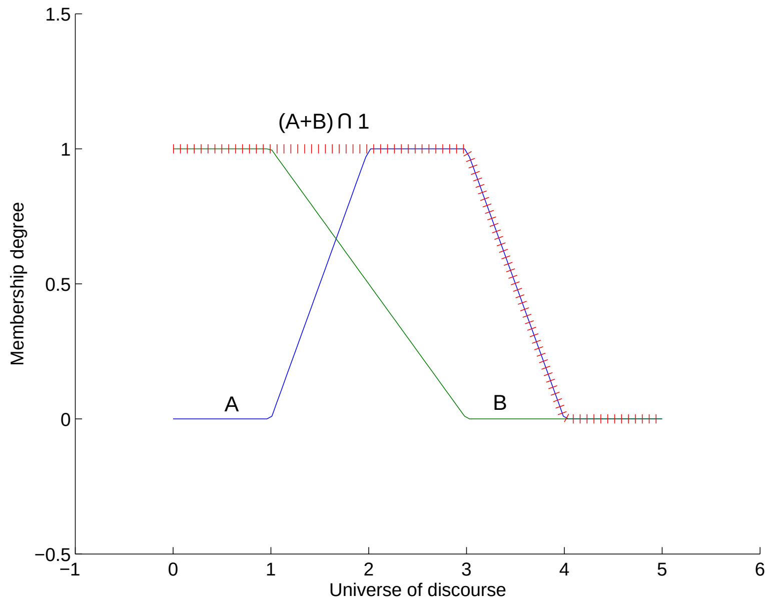

Figure 2.13: The Bounded Sum of Fuzzy Sets

where

$$\mu_{\underset{\sim}{c}}(x) = min(1, \mu_{\underset{\sim}{A}}(x) - \mu_{\underset{\sim}{B}}(x)) \tag{2.21}$$

Figure 2.14 shows bounded difference of fuzzy set. The bounded-difference operation is particularly useful because operations such as intersection, union, complement, absolute difference, implication, and equivalence can be expressed by bounded-difference operations and arithmetic sums [9].

2.3.1 Fuzzy Relations

Fuzzy relations map elements of one universe to those of another universe, through the Cartesian product of the two universes. However, the “strength” of the relation between ordered pairs of the two universes, say X and Y, is not measured with the characteristic function, but rather with a membership function expressing various

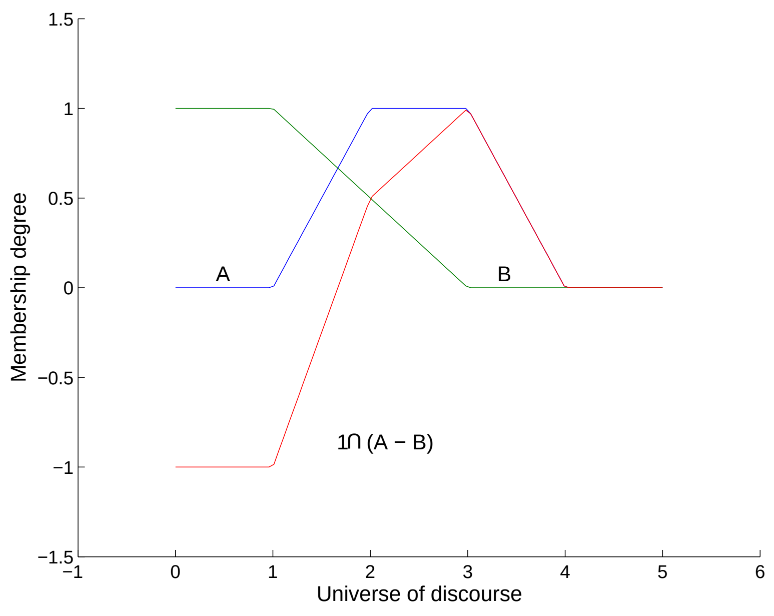

Figure 2.14: The Bounded Difference of Fuzzy Sets

"degrees" of strength of the relation on the unit interval [0,1]. Hence, a fuzzy relation $\underset{\sim}{R}$ is a mapping from the Cartesian space $X \times Y$ to the interval [0,1], where the strength of the mapping is expressed by the membership function of the relation for ordered pairs from the two universes, or $\mu_{\underset{\sim}{R}}(x, y)$.

$$\underset{\sim}{R} = X \times Y = \{(x, y) \mid x \in X, y \in Y\} \tag{2.22}$$

When X = Y, $\underset{\sim}{R}$ is known as a fuzzy relation on X.

For finite universe $X = \{x_1, x_2, ..., x_m\}$ and $Y = \{y_1, y_2, ..., y_n\}$ and the fuzzy sets $\underset{\sim}{A}$ and $\underset{\sim}{B}$, then a fuzzy relation $\underset{\sim}{A} \times \underset{\sim}{B}$ can be expressed by an $m \times n$ matrix:

$$\mathbf{M}_{\underset{\sim}{\mathbf{R}}} = \begin{pmatrix} \mu_{\underset{\sim}{R}}(x_1, y_1) & \mu_{\underset{\sim}{R}}(x_1, y_2) & \dots & \mu_{\underset{\sim}{R}}(x_1, y_n) \\ \mu_{\underset{\sim}{R}}(x_2, y_1) & \mu_{\underset{\sim}{R}}(x_2, y_2) & \dots & \mu_{\underset{\sim}{R}}(x_2, y_n) \\ \vdots & \vdots & \vdots & \vdots \\ \mu_{\underset{\sim}{R}}(x_m, y_1) & \mu_{\underset{\sim}{R}}(x_m, y_2) & \dots & \mu_{\underset{\sim}{R}}(x_m, y_n) \end{pmatrix} \tag{2.23}$$

This matrix is referred to as a fuzzy matrix. The elements of the fuzzy matrix have values within the interval [0,1], since μ_R has values within that range.

Fuzzy relations play an important role in fuzzy modeling, fuzzy diagnosis, and fuzzy control. They also have applications in fields such as psychology, medicine, economics, and sociology [8].

2.3.2 Composition of Fuzzy Relations

Consider two fuzzy relations $\underset{\sim}{R}$ on $\underset{\sim}{A} \times \underset{\sim}{B}$ and $\underset{\sim}{S}$ on $\underset{\sim}{B} \times \underset{\sim}{C}$. Contrary to crisp relations, fuzzy $\underset{\sim}{R}$ and $\underset{\sim}{S}$ may be composed in various ways, since $\mu_{\underset{\sim}{R}}$ and $\mu_{\underset{\sim}{S}}$ can assume values in the range [0,1], not just 0 or 1. Their composition is denoted by $\underset{\sim}{R} \circ \underset{\sim}{S}$ and is also a fuzzy relation on $\underset{\sim}{A} \times \underset{\sim}{C}$ [8]. The best known and the most frequently used is the so-called "max-min" composition. It is defined by

$$\mu_{\underset{\sim}{R} \circ \underset{\sim}{S}}(a, c) = \{\underset{b \in B}{max}\ min\ (\mu_{\underset{\sim}{R}}(a, b), \mu_{\underset{\sim}{S}}(b, c)) \mid (a, c)\} \tag{2.24}$$

Symbolically as

$$\mu_{\underset{\sim}{R} \circ \underset{\sim}{S}}(a, c) = \underset{b}{\vee}\{\ \mu_{\underset{\sim}{R}}(a, b) \wedge \mu_{\underset{\sim}{S}}(b, c)\} \tag{2.25}$$

where, $\wedge$ and $\vee$ are the symbolic representation for minimum and maximum operators respectively.

2.4 Fuzzy Logic

2.4.1 Crisp Logic

Logic can be a means to compel us to infer correct answers, but it cannot by itself be responsible for our creativity or for our ability to remember. In other words, Logic can assist us in organizing words to make clear sentences, but it cannot help us determine what sentences to use in various contexts. The interest in Logic arise from the study of truth in logical propositions; in classical predicate logic this truth is of binary nature: -a proposition is either true or false [8].

Rules are a form of proposition. A proposition is an ordinary statement involving terms which have been defined. In traditional propositional logic, a proposition must be meaningful to call it "true" or "false," whether or not we know which of these terms properly applies [10].

Logical reasoning is the process of combining given propositions into other propositions, and then doing this over and over again. A new proposition can be obtained from a given one by prefixing the clause "it is false that $\cdots$ ". This is the operation of negation (denoted $\sim p$). Propositions can be combined in many ways, all of which are derived from three fundamental operations: *Conjunction* (denoted $p \wedge q$), where we assert the simultaneous truth of two separate propositions p and q; *Disjunction* (denoted $p \vee q$) where we assert the truth of either or both of two separate propositions; and, *Implication* (denoted $p \rightarrow q$) which usually takes the form of an IF-THEN rule. The IF part of an implication is called the *Antecedent*, Whereas the THEN part is called the *Consequent*. Additionally, $p \leftrightarrow q$ is the *equivalence* relation; it means that p and q are both true or false.

A *tautology* is a proposition formed by combining other propositions (p, q, r,...) which is valid regardless of the truth or falsehood of p, q, r,... . Tautologies are important because they represent valued reasonings. In traditional propositional logic there are two very important inference rules, *Modus Ponens* and *Modus Tollens.* These rules are tautologies and works with two premises:

Modus Ponens-Premise 1: "x is A"; Premise 2: "IF x is A THEN y is B"; *Consequence*; "y is B." Modus Ponens is associated with the implication "A implies B" $[A \rightarrow B]$. In terms of propositions p and q, Modus Ponens is expressed as $(p \wedge (p \rightarrow q)) \rightarrow q$ [10]. The rule is summarized as

$$\begin{array}{c} \underline{\textit{Modus Ponens}} \\ if\ x\ is\ A \quad then\ y\ is\ B \\ x\ is\ A \\ \hline \therefore \qquad y\ is\ B \end{array} \tag{2.26}$$

Modus Tollens-Premise 1: "IF x is A THEN y is B"; Premise 2:"y is not B" ; *Consequence*: "x is not A." in terms of propositions p and q, Modus Tollens is expressed as $(\bar{q} \wedge (p \rightarrow q)) \rightarrow \bar{p}$ [10].

$$\begin{array}{c} \underline{\textit{Modus Tollens}} \\ if\ x\ is\ A \quad then\ y\ is\ B \\ y\ is\ not\ B \\ \hline \therefore\ x\ is\ not\ A \end{array} \tag{2.27}$$

Whereas Modus Ponens plays a central role in engineering applications of logic, due in large part to cause and effect, Modus Tollens does not seem to have yet played much of a role. This could be due to the causal nature of the engineering applications.

2.4.2 Fuzzy Logic

Dr Lotfi Zadeh from UC/Berkeley introduced Fuzzy Logic in the 60's as a means to model uncertainty in natural language [7]. Fuzzy logic extends conventional Boolean logic to handle the concept of the partial truth – with values really between " totally true " and " totally false " truths, making it easier to imitate the behavior of the human reasoning.

The extension of crisp logic to fuzzy logic is made by replacing the bivalent membership functions of crisp logic with fuzzy membership functions. Fuzzy values are assigned to evaluate the truth of propositions, and operations with these values are applied to evaluate composite propositions.

The meaning of a fuzzy proposition such as *"x is A"* is defined by the membership function that represents fuzzy set A. Assigning a meaning for compound fuzzy propositions, such as *"x is A and y is B"* or *"(x is A or x is B) and y is C"*, implies the calculation of the membership function that characterizes the fuzzy relation induced by the proposition. In order to do so, it is necessary to define the interpretation for the linguistic connectives "and" and "or" and for the operator "not".

The logical connectives of negation, disjunction, conjunction, and implication are also defined for fuzzy logic. These connectives are given in the followings equations for two simple propositions: proposition $\underset{\sim}{P}$ defined on fuzzy set $\underset{\sim}{A}$ an proposition $\underset{\sim}{Q}$ defined on fuzzy set $\underset{\sim}{B}$. T($\underset{\sim}{P}$) denotes the truth value of proposition $\underset{\sim}{P}$.

Basic connectives operations:

Negation

$$T(\bar{\underset{\sim}{P}}) = 1 - T(\underset{\sim}{P}) \tag{2.28}$$

Disjunction

$$\underset{\sim}{P} \vee \underset{\sim}{Q} : x\ is\ \underset{\sim}{A}\ or\ \underset{\sim}{B} \qquad T(\underset{\sim}{P} \vee \underset{\sim}{Q}) = max\ (T(\underset{\sim}{P}), T(\underset{\sim}{Q})) \tag{2.29}$$

Conjunction

$$\underset{\sim}{P} \wedge \underset{\sim}{Q} : x\ is\ \underset{\sim}{A}\ and\ \underset{\sim}{B} \qquad T(\underset{\sim}{P} \wedge \underset{\sim}{Q}) = min\ (T(\underset{\sim}{P}), T(\underset{\sim}{Q})) \tag{2.30}$$

The main differences among the different inference schemes presented by different authors come from the interpretation of implication, that is, substituting the above mentioned operators by c-norms, s-norms, and t-norms, respectively more about these operators can be found in [2].

Implication

$$\underset{\sim}{P} \rightarrow \underset{\sim}{Q} : if\ x\ is\ \underset{\sim}{A},\ then\ x\ is\ \underset{\sim}{B} \tag{2.31}$$

The most commonly interpretations that have been adopted are:

- Zadeh

$$T(\underset{\sim}{P} \rightarrow \underset{\sim}{Q}) = T(\underset{\sim}{\bar{P}}) \vee \underset{\sim}{Q}) = max\ (T(\underset{\sim}{\bar{P}}), T(\underset{\sim}{Q})) \tag{2.32}$$

- Mamdani

$$T(\underset{\sim}{P} \rightarrow \underset{\sim}{Q}) = min\ (T(\underset{\sim}{P}), T(\underset{\sim}{Q})) \tag{2.33}$$

- Larsen

$$T(\underset{\sim}{P} \rightarrow \underset{\sim}{Q}) = T(\underset{\sim}{P}) \cdot T(\underset{\sim}{Q}) \tag{2.34}$$

Engineers usually prefer Mamdani's or Larsen's formulas, while Zadeh's interpretation, derived directly from classical logic, is used in social sciences.

The implication connective is more useful when modeled in rule-based form: In this interpretation $\underset{\sim}{P} \rightarrow \underset{\sim}{Q}$, "IF x is $\underset{\sim}{A}$, THEN y is $\underset{\sim}{B}$" is equivalent to the fuzzy relation, $\underset{\sim}{R} = (\underset{\sim}{A} \times \underset{\sim}{B}) \cup (\bar{\underset{\sim}{A}} \times Y)$, where Y is the universe of discourse of $\underset{\sim}{B}$. The membership function of $\underset{\sim}{R}$ is expressed by the following formula:

$$\mu_{\underset{\sim}{R}}(x,y) = \vee[(\mu_{\underset{\sim}{A}}(x) \wedge \mu_{\underset{\sim}{B}}(y)), (1 - \mu_{\underset{\sim}{A}}(x))] \tag{2.35}$$

Modus Ponens is a rule of inference pertaining to the *if / then* operator. Modus Ponens states that if the antecedent of a conditional is true, then the consequent must also be true. Modus Tollens also is a rule of inference pertaining to the *if / then* operator. Modus Tollens states that if the consequent of a conditional is false, then the antecedent must also be false.

Fuzzy inference refers to computational procedures used for evaluating fuzzy linguistic descriptions. There are two important inferencing procedures: generalized Modus Ponens (GMP) and generalized Modus Tollens (GMT).

In fuzzy logic, Modus Ponens is extended to *generalized Modus Ponens*- Premise 1:"IF u is A THEN v is B"; Premise 2: "u is $A^{'}$"; *Consequence*: "v is $B^{'}$".

Compare Modus Ponens and Generalized Modus Ponens to see their subtle differences. Namely, in the latter, fuzzy set $A^{'}$ is not necessarily the same as rule antecedent fuzzy set A, and fuzzy set $B^{'}$ is not necessarily the same as rule consequent B.

GMP allows us to compute the consequent $B^{'}$. It is formally stated as

$$\begin{array}{cc} \underline{\textit{Modus Ponens}} & \underline{\textit{Generalized Modus Ponens}} \\ if\ x\ is\ A \quad then\ y\ is\ B & if\ x\ is\ A \quad then\ y\ is\ B \\ x\ is\ A & x\ is\ A^{'} \\ \hline \therefore \qquad y\ is\ B & \therefore \qquad y\ is\ B^{'} \end{array} \tag{2.36}$$

Let's consider a linguistic description involving only a simple *if / then* rule with known implication relation $\underset{\sim}{R}$(x, y) and a fuzzy value A' approximately matching the antecedent of the rule. Consequent can be obtained by taking the composition of fuzzy set A' and fuzzy relation $\underset{\sim}{R}$(x, y), and in general is symbolically given as follows:

$$\mu_{B'}(y) = \bigvee_{x} \{ \mu_{A'}(x) \wedge \mu_{\underset{\sim}{R}}(x, y)\} \tag{2.37}$$

In GMT a rule and a fuzzy value approximately matching its consequent are given and it is desired to infer antecedent that is

$$\begin{array}{ll} \underline{\textit{Modus Tollens}} & \underline{\textit{Generalized Modus Tollens}} \\ if\ x\ is\ A \quad then\ y\ is\ B & if\ x\ is\ A \quad then\ y\ is\ B \\ \qquad\qquad y\ is\ not\ B & \qquad\qquad y\ is\ B' \\ \hline \therefore\ x\ is\ not\ A & \therefore\ x\ is\ A' \end{array} \tag{2.38}$$

Since Modus Tollens is seldom used, no further insight is needed for the purpose of this work.

2.4.3 Hedges

Another important feature of fuzzy systems is the ability to define "hedges," or modifiers of fuzzy values. These operations are provided in an effort to maintain close ties to natural language, and to allow for the generation of fuzzy statements through mathematical calculations. As such, the initial definition of hedges and operations upon them will be quite a subjective process and may vary from one project to another. Nonetheless, the system ultimately derived operates with the same formality as classic logic.

The simplest example is in which one transforms the statement "Jane is old" to "Jane is very old." The hedge "very" is usually defined as follows:

$$m\text{``}very\text{''}A(x) = mA(x)^2 \tag{2.39}$$

Thus, if mOLD(Jane) = 0.8, then mVERYOLD(Jane) = 0.64. Other common hedges are "more or less" typically $mA(x)^{0.5}$, "somewhat," "rather," "sort of," and so on. Again, their definition is entirely subjective, but their operation is consistent: they serve to transform membership/truth values in a systematic manner according to standard mathematical functions.

Each of these functions is frequently used in fuzzy set theory, in order to represent a wide range of linguistic hedges of type I. For example, Zimmermann follows Zadeh when he proposes [11]

$$veryA = Con(A) = A^2, \tag{2.40}$$

$$moreorlessA = Dil(A) = A^{0.5}, \tag{2.41}$$

$$plusA = A^{1.25}, \tag{2.42}$$

Chapter 3

Control and Fuzzy Logic System

3.1 Basic Control Systems

The purpose of control is to influence the behavior of a system by changing an input or inputs to that system according to a rule or set of rules that model how the system operates. The system being controlled may be mechanical, electrical, chemical or any combination of these.

Basic control theory uses a mathematical model to define a relationship that transforms the desired state (requested) and observed state (measured) of the system into an input or inputs that will alter the future state of that system. The Figure 3.1 shows a basic closed-loop control system.

Once the mathematical model is known (which is the hardest part of it), one can determine if the system is stable and characterize its dynamical behavior. Systems are usually classified according to the variables involved and to the equations that specify their dynamics. Thus there are linear, bilinear, polynomial, and stochastic

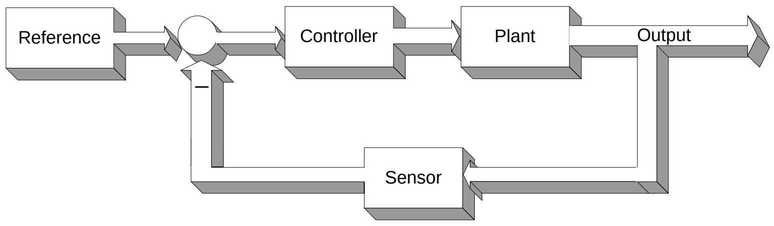

Figure 3.1: Basic Closed-Loop Control System

systems depending on the character of the transition map.

The control problem for a system is then to synthesize a controller such that the closed-loop system meets the control objectives. General control objectives include: stability of the state function, optimization of performance, robustness with respect to noise signals and poorly modeled dynamics, and avoidance of unsafe states [8].

As the complexity of the system increases, as when multi-input/multi-output is required, it becomes more difficult to formulate the mathematical model needed for classical control. Fuzzy technology can alleviate these problems for the following reasons.

Nonlinear characteristics are realized in fuzzy logic by partitioning the rule space, but weighting the rules, and by the nonlinear membership function. Rule-based systems compute their output by combining results from different parts of the partition, each part being governed by separate rules. In fuzzy reasoning, the boundaries of these parts overlap, and the local results are combined by weighting them appropriately. That is why the output in a fuzzy system is a smooth, nonlinear function [2].

3.2 Fuzzy Logic Systems

A FLS receives a crisp input and may deliver either a fuzzy set or a crisp value. The basic FLS contains four components: a rule set, a fuzzifier, an inference engine, and a defuzzifier. Rules may be provided by experts or can be extracted from numerical data. In either case, the engineering rules are expressed as a collection of IF-THEN statements. These statements are related to fuzzy sets associated with linguistic variables [10].

The fuzzifier maps the input crisp numbers into the fuzzy sets to obtain degrees of membership. It is needed in order to activate rules, which are in terms of the linguistic variables. The inference engine of the FLS maps the antecedent fuzzy (IF part) sets into consequent fuzzy sets (THEN part). This engine handles the way in which the rules are combined. In practice, only a very small number of rules are actually used in engineering applications of Fuzzy Logic (FL) [12].

In most applications, crisp numbers must be obtained at the output of an FLS. The defuzzifier maps output fuzzy sets into a crisp number, which becomes the output of the FLS. In a control application, for example, such a number corresponds to a control action.

3.2.1 Fuzzification

The first step in fuzzy logic processing involves a domain transformation called fuzzification (Figure 3.2). Crisp inputs are transformed into fuzzy inputs. To transform crisp input into fuzzy input, membership functions must first be defined for each input.

Once membership functions are defined, fuzzification takes a real time input value, such as temperature, and compares it with the stored membership function information to produce fuzzy input values.

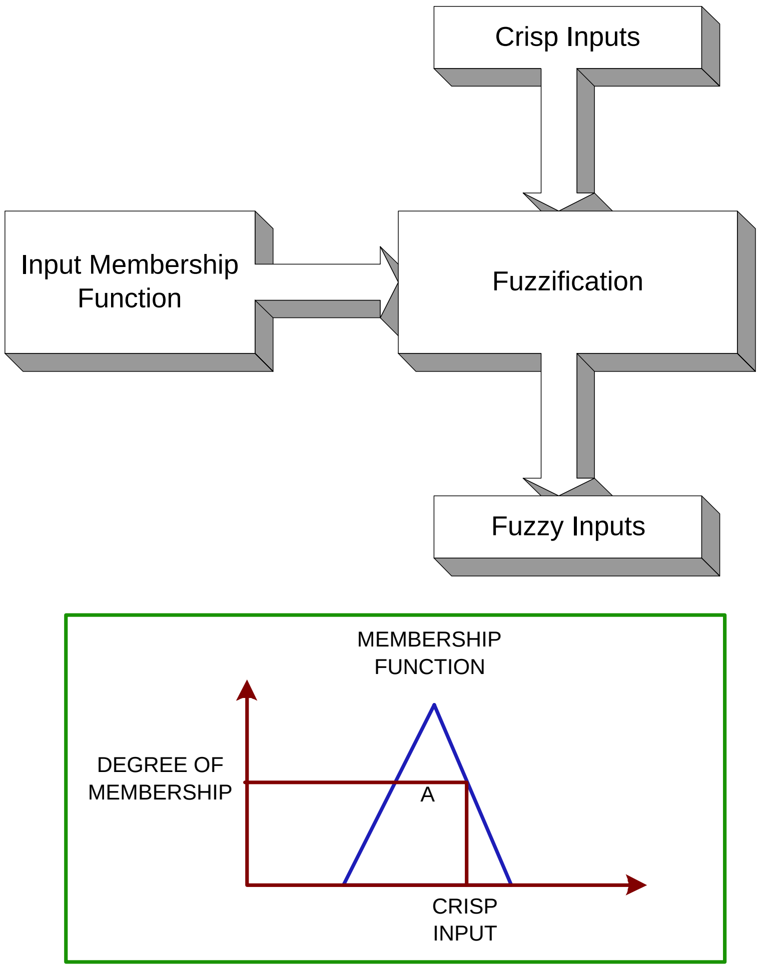

Figure 3.2: Fuzzification

The first step in fuzzification is to assign fuzzy labels in the Universe of discourse of each of the crisp inputs. So for temperature, we might assign a range of labels like those shown in Figure 3.3.

Each crisp input into a fuzzy system can have multiple labels assigned to it. In

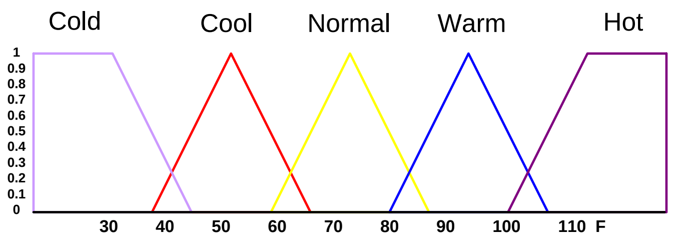

Figure 3.3: Temperature Membership Functions

general, the greater the number of labels assigned to describe an input variable, the higher the resolution of the resultant fuzzy control system, resulting in a smoother control response.

However, a large number of labels requires added computation time. Moreover, an excessive number of labels can lead to an unstable fuzzy system [1]. As a result, the most common number of labels for each variable in a fuzzy system fall between 3 and 9 [1]. The number is usually Taken to be an odd number 3, 5, 7, 9, though this is not a requirement, but something that has been seen in applications [1].

The control surface fuzzy sets on each side of the zero (or normal) action set should be balanced and symmetric. Thus if you have a variable, Temperature, a fuzzy region LOW should also have a corresponding region HIGH as well as a normal temperature set of NORMAL.

Next, membership functions are defined to give numerical meaning to each label. Each membership function identifies the range of inputs values that corresponds to a label.

Unlike Boolean logic, the membership function of a label does not define boundaries where the label applies fully on one side of a cutoff and not at all on the other side of the cutoff. Instead there is a region where input values gradually change from being fully applicable to completely inapplicable.

Membership functions can have several different shapes, like those shown to the Figure 2.6. Trapezoidal, bell, and triangular are the most frequently used. Although other shapes may be more representative of natural occurring phenomena, they require more complicated equations or large look-up tables to be represented accurately. A Fuzzy singleton is a fuzzy set whose support is a single point in U with a membership function of one.

Singletons are easily represented in a computer and allow for simpler defuzzification algorithms. They are, therefore, frequently used to describe fuzzy outputs.

An input membership function is created by specifying a number, that is, the degree of membership, for each possible input value for a given label. The y-axis $\{\mu\}$ values refer to the degree to which the crisp input value (temperature) applies to each of the membership function labels (cool, warm, etc.). Input values can belong to more than one fuzzy set. Describing crisp inputs in fuzzy terms allows the system to gracefully respond to gradual change in input temperature.

3.2.2 Fuzzy Inference

Fuzzy logic based systems use RULES to represent the relationship between observations and actions. These rules consist of a precondition (IF-part) and a consequence (THEN-part). The precondition can consist of multiple conditions linked together with AND or OR conjunctions. Conditions may be negated with a NOT. The computation of fuzzy rules is called *Fuzzy Inference.*

Fuzzy rule inference consists of two steps:

- Inferencing, which determines the fuzzy subset of each output variable for each rule. Usually only MIN or PRODUCT are used as inference rules. In MIN inferencing, the output membership function is clipped off at a height corresponding to the rule premises's computed degree of truth (fuzzy logic AND). In PRODUCT inferencing, the output membership function is scaled by the rule premises's computed degree of truth.

- Composition, which combines the fuzzy subsets for each output variable into a single fuzzy subset. Usually MAX or SUM are used. In MAX composition, the combined output fuzzy subset is constructed by taking the point wise maximum over all of the fuzzy subsets assigned to variable by the inference rule (fuzzy logic OR). In SUM composition, the combined output fuzzy subset is constructed by taking the point wise sum over all of the fuzzy subsets assigned to the output variable by the inference rule.

IF-THEN rules are a common way of representing and communicating knowledge in everyday conversation. Anyone who has written a program or machine code knows how complicated (and difficult to debug, read, and maintain) the if-then lines can get. Fuzzy rules offer a way of getting around that trading the precise representation of the values that variables must assume with much more intuitive fuzzy representations.

In binary logic the consequent is either true or false. In fuzzy logic partial truths are allowed so the consequent is as partially true as the antecedent allows it to be.

In general a rule by itself doesnt do much. Whats needed are a set of rules that can play off one another. The fuzzy inference methodology allows “fair” competition between these rules to produce sophisticated answers using seemingly simple premises.

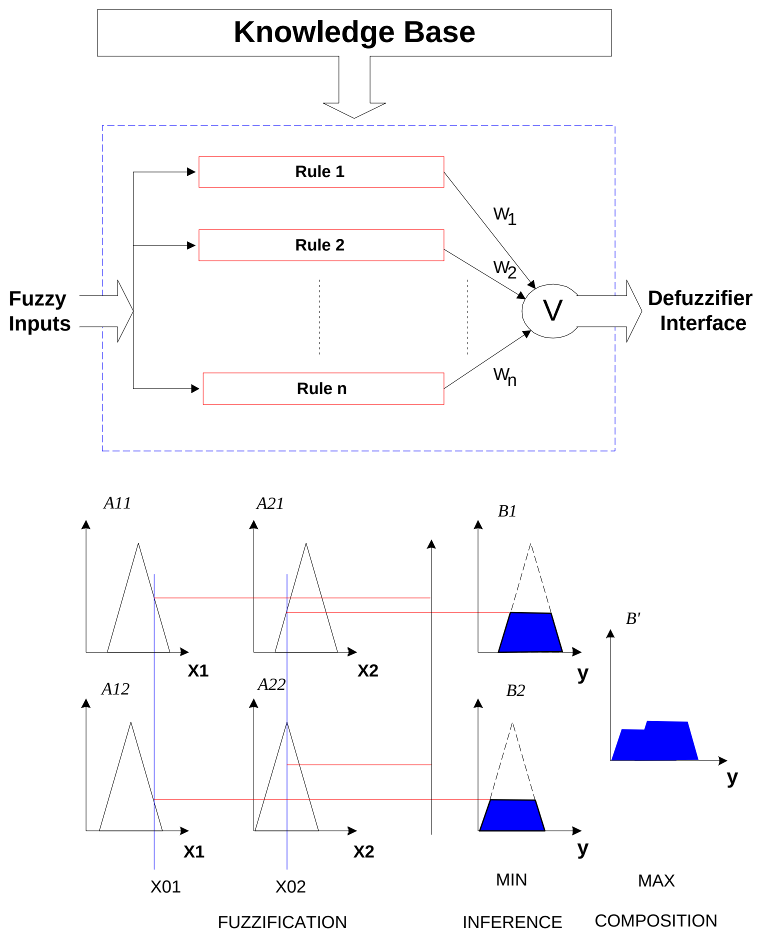

Figure 3.4: Inference process

3.2.3 Defuzzification

This stage is used to convert the fuzzy output set to a crisp number. Two of the more common techniques are the Centroid and Maximum methods. In the Centroid method, the crisp value of the output variable is computed by finding the value of the center of gravity of the membership function. In the Maximum method, one of the variable values at which the fuzzy subset has its maximum truth-value is chosen as the crisp value for the output variable. Figure 3.5 illustrates the complete process with an example.

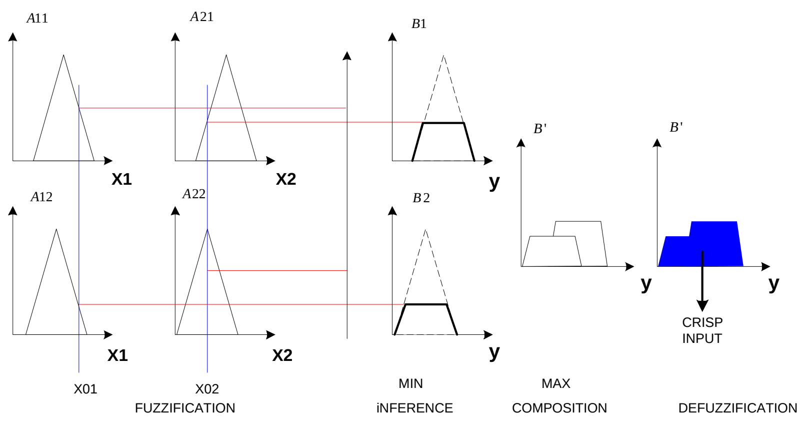

Figure 3.5: Fuzzy Logic System

3.3 Fuzzy Logic System Implementation

A fuzzy logic controller can be implemented using software, on microprocessors, DSP board computers, or using special purpose hardware. Each implementation has its advantage and drawbacks.

Practical FLS realizations may be implemented with software and/or hardware systems. Hardware implementations of fuzzy logic are attracting more and more the attention of system designers for cases where the controller requires very short response times [12]. Also, hardware realizations of fuzzy controllers are demanded in those applications demanding real time performance and/or low area occupation and power consumption. The hardware realization may be either digital or analog. In most cases, the digital implementations have greater flexibility through their external memory although they are usually slower than their analog counterpart, for similar logic performance. Besides speed, other advantages of analog realizations relate to the limitations of sampling theorem (not applicable to analog circuits) and to the interface limitations. For example, analog-to-digital conversion may limit speed.

The main drawback of analog hardware implementations is the limited complexity and modifiability of the system [13]. This problem has been partially dealt with by the implementation of reconfigurable and adaptive controllers [12][14][15]. Yet there are still many problems to be solved in this direction.

Another approach contemplates using mixed signal circuitry, realizing the fuzzy processing itself in the analog domain employing digital circuitry for programmability and reconfigurability. Generally speaking, this approach is expected to feature higher operation speed, lower power consumption, and smaller area occupation than the traditional digital realization with digital FLS requiring A/D, D/A converter. These techniques fully exploit the functional capabilities of the MOS transistor (MOST) to realize the fuzzy operators with very simple circuitry. Now, one advantage of a fuzzy system is that its precision has usually more tolerance when compared with non-fuzzy systems, and therefore the design phase is not so restrictive in what concerns to precision. Yet, this tolerance has a limit. Their output signal may hence become largely erroneous. Post-fabrication tuning of some critical parameters, guided by learning processes, can attenuate these errors; but they must still remain bounded for convergence [15].

The design of *dedicated fuzzy integrated circuits* is of great interest, because of the increasing number of fuzzy applications requiring highly parallel and high-speed fuzzy processing. They attempt to give a concrete expression to the idea of "fuzzy computers" (sometimes called computers of the sixth generation), which deal with analog values or some digital representation of them. Fuzzy processors are designed to optimize fuzzy logic functions in terms of their implementation size and execution speed. Practical systems often require a large number of rule evaluations per second. This requires in turn, a large amount of real- time data processing, making the speed of fuzzy circuits of prime importance. The architectures of fuzzy processors are generally suited to the structure of approximate reasoning and decision- making algorithms and typically include three distinct parts: proceeding fuzzification, inference and defuzzification. Fuzzy controllers are made on one hand of a knowledge base, which contains rules and membership functions outlines, as well as other configuration parameters of the system. An inference mechanism based on interpolative reasoning interfaces the controller to a real process in a feedback loop configuration.

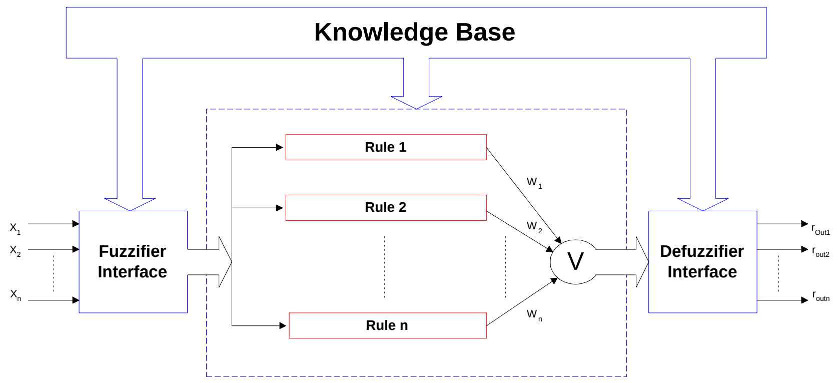

Figure 3.6: Fuzzy Logic Based Processing Unit

The structure of dedicated fuzzy circuits is characterized by the number and shape range of input and output variables, the number of simultaneous rules they can evaluate, the type(s) of inference(s) (size of premises, operators, consequences,...) and the type(s) of defuzzification method(s). Their performance is evaluated according to their processing speed, that is the number of fuzzy logic inferences per second (FLIPS). Their precision is evaluated by the error and noise generation in analog circuits and number of bit representing fuzzy values in digital ones). Fast response is required for non-linear functions such as MIN and MAX where output signals can be subject to sudden discontinuities.

Fuzzy chips can be efficiently used in expert systems and in control and command field applications to achieve real-time performance. They are, however, less efficient for applications that deal with a large amount of data because of the limited I/O. A solution has been suggested, consisting in the interconnection of compatible circuits, and splitting the large number of input variable [16].

Fuzzy software is useful when an application can be modeled to simulate and calculate in advance a multidimensional response characteristic. They give the different parameters relative to an optimal characteristic in order to design or program a fuzzy dedicated circuit. Some of these parameters (fuzzy sets and inference rules) have to be adjusted in the real implementation. In some cases, the numeric response characteristic can be stored as a look-up table, providing then response values for real-time processing without any more inference calculations. The stored values are provided by fuzzy software, when a model exists, by the measure of an operator's action or by an adaptive system with a retroactive learning scheme (which comes close to the principle of fuzzy dedicated neural networks)[17].

Software implementations [18] of fuzzy systems provide higher flexibility. Furthermore, they usually support fuzzy systems with an arbitrary number of rules, without any limitations concerning the number and type of membership functions, and with a wide range of inference mechanisms.

On the other hand, since these systems are usually on a computer platform not working in real time, software approaches are not adequate for applications demanding fuzzy systems with a small size, low power consumption, and high inference speed [2]. For these applications, hardware approaches are more appropriate.

Among the benefits of using hardware instead of software are the speed and compactness that the system can achieve. These limitations preclude its use in important applications areas such as robotics, aeronautics, voice and image processing, automotive electronic and so on. The main drawback of hardware implementation is the complexity and the limited modifiability of the system [13].

Design and implementation of fuzzy system obviously depends on the requirements of the application towards which it is addressed. Three requirements are particularly relevant to decide how to implement the fuzzy system:

1. The time available for rule processing.
2. The size it can occupy and
3. The power it can consume.

There are two basic procedures for developing fuzzy hardware. One employs general-purpose microprocessors, occasionally expanded with new instructions or new circuitry. The other approach is based on application-specific integrated circuits (ASICs), optimized for fuzzy logic-based inference systems.

General-purpose microelectronic systems are not efficient in terms of cost, size, and inference speed, even considering the modifications mentioned above, as has been shown by several authors [2][19].

Implementations of fuzzy systems with dedicated hardware are highly influenced by several facts.

- Choice of a realization strategy conditions the storage and computing resources that will be required.
- Implementation technique and the design methodology employed to realize the integrated circuits.
- Realization of the basic building block of the fuzzy system depends greatly on the analog or digital design style employed.

There are three different strategies that can be followed when considering the microelectronic implementation of a fuzzy system.

1. Off-line strategy.

   Consist of precomputing all the output values that should be provided by the system for every possible combination of the input signals. The microelectronic circuits suitable to this strategy are standard digital memories and programmable logic devices (PLDs). The response time in this approach is very short because only one data has to be retrieved from a memory and this can be performed in a few nanoseconds.

2. Semi-off-line strategy.

The use of this strategy also eliminated the problems of parallelism and non-standard operations of the fuzzy systems. Besides, it reduces the memory size (the number of data to store) if compared with the previous strategy. The cost of this reduction is an increase in hardware complexity because additional blocks have to be included not only to perform the operations required but also to identify the grid cell activated by the inputs.

3. On-line strategy.

   This strategy is usually employed to implement generic and complex fuzzy systems, in which the inference process is concurrently performed as the input changes. The input stage should contain operators to fuzzify the inputs. The rule processing stage should contain operators that implement antecedent connectives so as to obtain the activation degrees of the rules. The membership function or the singleton value of the rule consequents has to be generated. The output stage should contain operators that implement the rule aggregation and the defuzzification operator.

The analog domain is characterized by a high number of possible design styles for the same functionality. In continuous-time analog design styles, circuits work with continuous voltages, transconductance or currents in the time domain and whose amplitudes can take on continuous value within a defined range. In the discrete time analog design style, signals are discrete in time and their amplitudes maintain a continuous range of values. Continuous or discrete-time techniques can be employed depending on the nature of the signal that holds information [20].

Conventional continuous time schemes can be classified according to their operation mode:

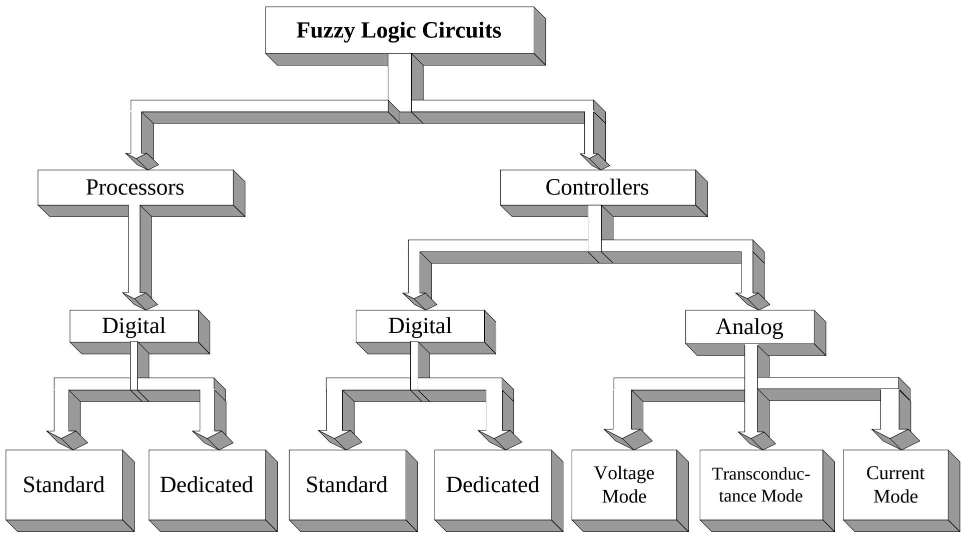

Figure 3.7: Fuzzy Logic Circuits

A *Voltage mode*

- RC active: based on resistors (R), capacitors (C) and singled-ended amplifiers.
- MOSFET-C: employs MOS transistors, capacitors (C) and fully differential amplifiers as basic elements.

B *Current mode*

- The basic elements in the current mode design are the MOS transistor mainly grouped into current mirrors.

C *Transconductance mode*

- GM-C: relies on capacitors (C) and Operational Transconductance Amplifiers (OTAs).

Current-mode circuits are especially suitable for realizing fuzzy systems because the basic fuzzy logic operators can be implemented with very few transistors.

The two discrete-time techniques most commonly employed are switched capacitor circuits in voltage mode techniques, and switched current limits in current mode techniques.

Continuous-time Analog Techniques offer a very good relation between area occupation and inference speed. This is achieved by eliminating the need for controlling the signals and by fully exploiting the potentiality of the basic element of a MOS integrated circuit, which is the MOS transistor [13][20]. MOS transistors technology offers in addition better results in terms of space and power [21].

The first microelectronic realizations of fuzzy logic-based systems with analog techniques were carried out by Yamakawa [22][23]. These ICs implement Min-Max Mamdani's method with the Center-of-Area defuzzification method. Hence, they manage fuzzy sets to represent both the antecendets and consequents of the rules. The response time of the rule and defuzzification chip is slow. The silicon area occupied by both is very large due to the lines that handle the discrete output universe of discourse [2]. This is why the solutions to solve control problems reported in [24] rely on the connection of several rule chip.

To avoid the needed of sweeping the points of the output universe of discourse most analog fuzzy ICs implements a Zero-Order Takagi-Sugeno or singleton fuzzy system [3].

From an architectural point of view, a relevant feature of fuzzy systems is their inherent parallelism. That is, several rules are activated for each input combination so data can be processed in parallel within each rule. Besides parallelism, another important issue at the system level is the chip programmability. The parameters that can be programmed in a fuzzy IC include:

- The number of fuzzy sets to cover the rule antecedent;
- The parameters that define their membership functions;
- The number of fuzzy sets employed for the output variable;
- The parameters that outline the rule base.

In a custom IC implementation all these parameters are fixed during the fabrication process. General purpose fuzzy ICs require programmable circuitry whose complexity increases with the number of programmable parameters [19][25].

The main architectures most commonly considered at present are: Rule-by-Rule, Circuitry Sharing and Active Rule-driven [2]. The characteristics of each approach are summarized in Table 3.1. The first analog fuzzy ICs employ Rule-by-Rule architecture and also later realizations, such as [26]. A Circuitry Sharing architecture was proposed in [27] to implement current-mode fuzzy ICs that work with the Fuzzy Mean as defuzzification method and the minimum operator as antecedent connective. This circuit again occupied a large area. A fully parallel architecture to implement programmable fuzzy ICs following the idea of active rules was proposed in [28]. This circuits shows be most efficient of the others, but require on line normalization circuits, thereby adding complexity and space in the system.

All three architectures process the rules in parallel and their processing core can be realized with analog or digital techniques.

The main advantage of using continuous-time analog techniques is that the operations to be performed within each rule can be realized in parallel in a very efficient

Architecture	Action Characteristics	Advantages	Drawbacks
Rule-by-Rule	Contains a data path for each rule.	High flexibility to define the rule base. Very modular	Managing redundant information. The programming circuitry have a very large area.
Circuitry Sharing	Solves the problem of area and power consumption for non programmable ICs sharing circuitry among different rules.	Reduces the number of MFC and Membership Function Generators	Programmable fuzzy ICs increase complexity.
Active Rule-driven	Contains the circuitry to identify only the active sets and the computing blocks need for processing only the active rules.	The number of MFCs is reduced. The number connective circuits and MFGs required are also reduced.	This is more efficient in order to implement a complex fuzzy system with an IC.

Table 3.1: Characteristics, advantages, and drawbacks of different architectures

way thanks to the use of multiple input circuits that offer low area and power consumption.

Chapter 4

Programmable and Modular Analog Fuzzy Controller

4.1 Hardware Implementation

Basically, a fuzzy logic controller (FLC) can be divided into two main blocks:

1. Fuzzification
2. Fuzzy inference

In fuzzification, the input data are crisp values obtained from various sensors. Therefore, a fuzzification interface is needed to calculate the membership of the observed inputs to the defined linguistic terms in the preconditions of the fuzzy rules [29]. This functional correspondence can be realized by an analog fuzzification circuit or so-called membership function circuit (MFC). The output of a MFC gives the grade of membership of the input to a related linguistic term in current or voltage mode.

Fuzzy inference is the process of formulating the mapping from a given input to an output using fuzzy logic. The mapping provides a basis from which decisions can be made. There are two main types of fuzzy inference systems (FIS) here: Mamdani-type and Sugeno-type [2]. These two types of inference systems vary somewhat in the way outputs are determined.

1. Mamdani's FIS

   Mamdani-inference systems employ consequents described by fuzzy sets in each rule, i.e, it expects the output inference to be also a fuzzy set. The resulting consequent fuzzy sets are agregatted to form the output fuzzy set of the system. If a crisp output is required, defuzzification is done with arithmetic operations such as centroid. An example of a Mamdani inference system is shown in Figure 4.1. To compute the systems output, given a set of inputs, one must go through six steps:

   (a) Determining a set of fuzzy rules

   (b) Fuzzifying the inputs using the input membership functions,

   (c) Combining the fuzzified inputs according to the fuzzy rules to establish a rule strength,

   (d) Finding the consequence of each rule by combining the rule strength and the output membership function,

   (e) Combining the consequences to get an output distribution, and

   (f) Defuzzifying the output distribution if a crisp output is needed.

   A block diagram illustrating the circuits necessary to realize these steps is shown in Figure 4.2.

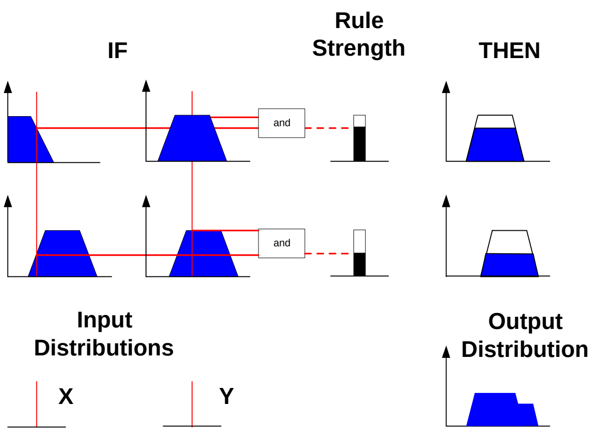

Figure 4.1: A two input, two rule Mamdani's FIS

2. Sugeno's FIS

   Sugeno's FIS is quite similar to Mamdani's FIS [3]. The primary difference is that the crisp output consequence is not computed by clipping an output membership function at the rule strength. In fact, in Sugeno's method there is no output membership function at all. Instead, the output is a crisp number called singleton, computed by multiplying the output of each rule by a constant and then adding up the results. This is illustrated in Figure 4.3. "Rule strength" in this example is referred to as "degree of applicability" and the output is referred to as the "action". Also notice that there is no output distribution, only a "resulting action" which is the mathematical combination of the rule strengths (degree of applicability) and the outputs (actions).

   One of the large problems with the Sugeno's FIS is that there is no good intuitive method for determining the coefficients, λ.

   Figure 4.4 shows a block diagram for Sugeno's FIS. The functionality and architecture of these three blocks are presented below.

   In this work, Sugeno's FIS was chosen, because it requires less circuits and is

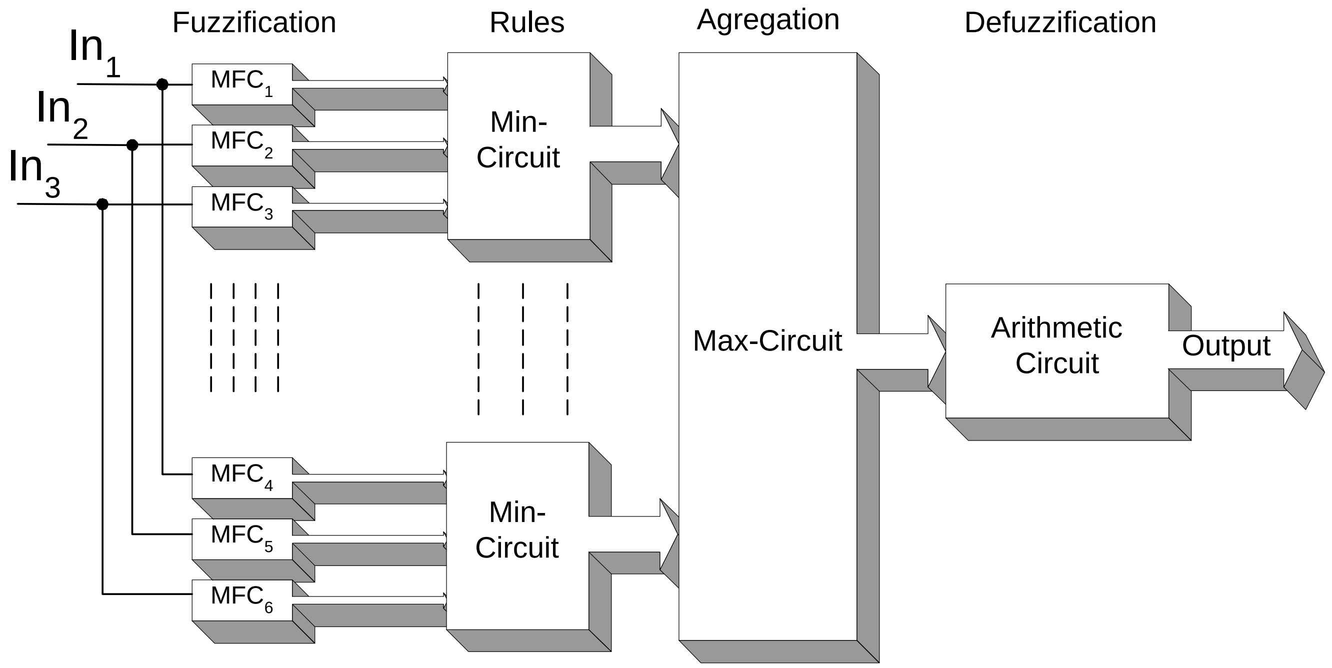

Figure 4.2: Block diagram of Mamdani's FIS

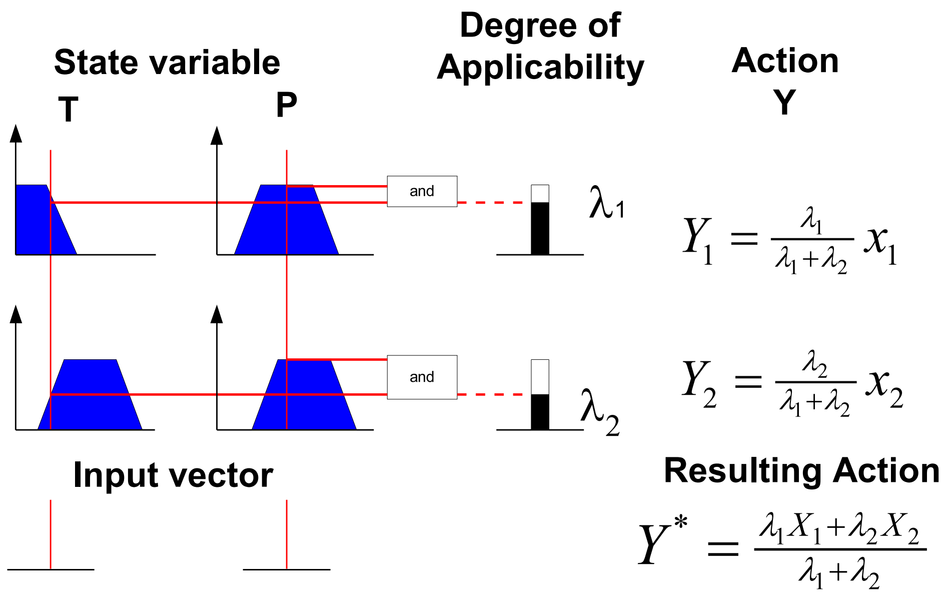
State variable
T
P
Degree of Applicability
Action
Y
and
and
λ_1
λ_2
$Y_1 = \frac{\lambda_1}{\lambda_1+\lambda_2} x_1$
$Y_2 = \frac{\lambda_2}{\lambda_1+\lambda_2} x_2$
Input vector
Resulting Action
$Y^* = \frac{\lambda_1 X_1 + \lambda_2 X_2}{\lambda_1+\lambda_2}$

Figure 4.3: A two input, two rule Sugeno's FIS

faster for a crisp output.

To design fuzzy analog circuits, the following factorsmust be considered.

1. Since shallow stages of fuzzy logic are processed in parallel, the accumulation of errors in the fuzzy circuits does not become a serious problem. This is a merit of the fuzzy process in terms of its hardware implementation.

2. Since the fuzzy process enjoys high speed, thanks to its shallowness in logic stages, it is very important to design the fuzzy analog circuits for high speed operation.

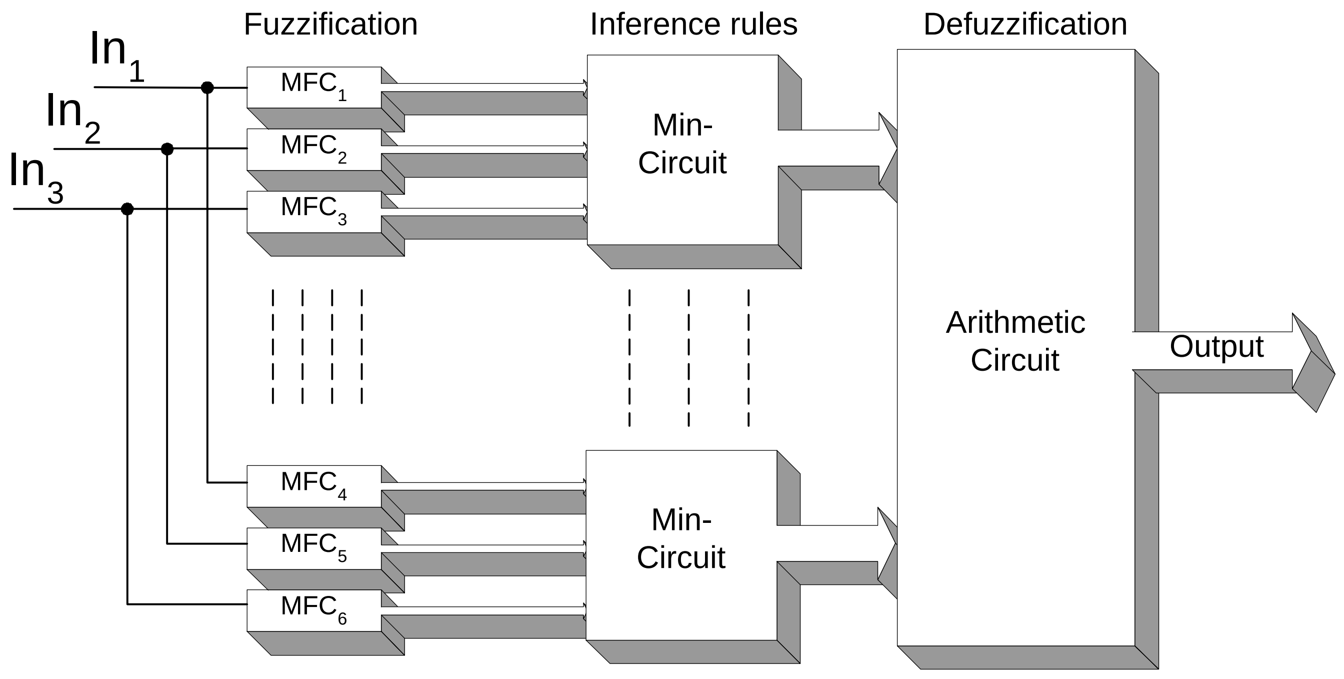

Figure 4.4: Block diagram of Sugeno's FIS

4.2 Proposed Fuzzy Logic Controller

This thesis proposes a programmable and modular fuzzy logic controller based on the basic Sugeno structure. the novelties of this controller are mainly in the fuzzification and defuzzification stages. Immediate benefits introduced by the approach adopted here are the reduction in circuit complexity by the elimination of the normalization circuit, as well as the reduction in the number of arithmetic circuits required in the defuzzification stage. Moreover, this controller has full programmability in both the fuzzification and the defuzzification stages with voltage or bias currents. Previous approaches only emphasized one of them.

The circuit used in the fuzzification stage for this work stems from the Ota-Wilamowski circuit [30] [31]. The original circuit operates fully in voltage mode, but has been modified here to obtain full "voltage input - current output" mode operation. This eliminates the need of feedback from the defuzzification stage as proposed in the original work. Moreover, the theoretical discrepancies that appeared in Ota and Wilamowski work have been corrected in this thesis. This circuit allows full programmability in both shape and position of the membership function.

For the minimum inference stage, Huang's circuit [32] was adopted. This circuit offers great modularity as per the number of rules, and offers the best performance when compared with other circuits, both in space and in power.

The defuzzifciation stage utilizes a multiplier-divider circuit based on Kaewdang et al.'s work [33], originally proposed for bipolar technology. This technology allows a linear dependence on bias currents. The assumption in [33] that the results are directly extendable to CMOS technology are shown to be false, since the dependence is now on the square root values. Yet, the circuit proposed here with CMOS technology has been adapted to operate as a programmable multiplier-divider circuit to conform to the needs of the weighted sum required in Sugeno's algorithm. This approach has provided the added benefit of eliminating the divider required in previous controllers, as well as the need for a normalization circuit between the inference and the defuzzification stages, since the weight for each rule is tuned independently of the other rule outputs.

In conclusion, the proposed fuzzy controller offers programmability at both the fuzzification and defuzzification stages by modification of previous circuits, and modularity at the rules stage by the selection of Huang's circuit.

The following sections describe the circuits and performance.

4.3 Fuzzification circuit

The analog membership function circuit (MFC) generate a membership function with their input/output transfer characteristic, so that they are arithmetic MFCs. A transconductance-mode MFCs work with a voltage as the input signal and provide current as the output signal. In a current-mode MFCs the input and output are currents. An input signal represented by a voltage can be drawn to several transconductance-mode MFCs with the same wire, without need of replication. On the contrary, an input signal represented by a current only can drive a single current-mode MFC, that is, the fan-out is 1. In order to connect a current-mode input signal to L MFCs the current should be replicated L times.

Various approaches have been proposed to generate analog nonlinear membership functions in either current, voltage or transconductance mode. As a general case in the solutions reported in [24] [34] [35] [36] [37]. J. Choi et al. [34] introduced a voltage-input/current-output programmable Gaussian function network with capacitors for the programmability. Therefore, periodic refreshing is necessary to maintain an accurate programmed value on the capacitors. Also, a membership function circuit which can realize several types of membership functions using bipolar transistors was proposed in [24]. However, a disadvantage of this design is that it needs emitter-follower arrays for impedance transformation and level/temperature compensation.

An approach to realize MFC consist of two stages: the first for S- and Z- subfunctions generation, and the second for the combination of these sub-functions [38]. The positive and negative slopes of the generated membership function are regulated within the S- and Z- sub-function circuits, while the position and the type of the function are assigned with reference voltages or currents in the combination part[29]. S-shape function, Z-shape function, triangular-shaped function can be considered as special shapes of the trapezoidal-shaped function.

The approach taken for the design of the circuit used in this work biases the MOS transistor in the strong-inversion region, where its current has a power-law dependence on the gate bias voltages [30]. The strong-inversion region of the MOS circuits provides the features of high current driving, large dynamic range, and high noise immunity. The transistors with large channel lengths are used to avoid the channel-length modulation effect. A block diagram of the implementation of the Gaussian-type membership function circuit is shown in Figure 4.5.

A circuit schematic of the membership block is shown on Figure 4.6. Reference voltages are applied.

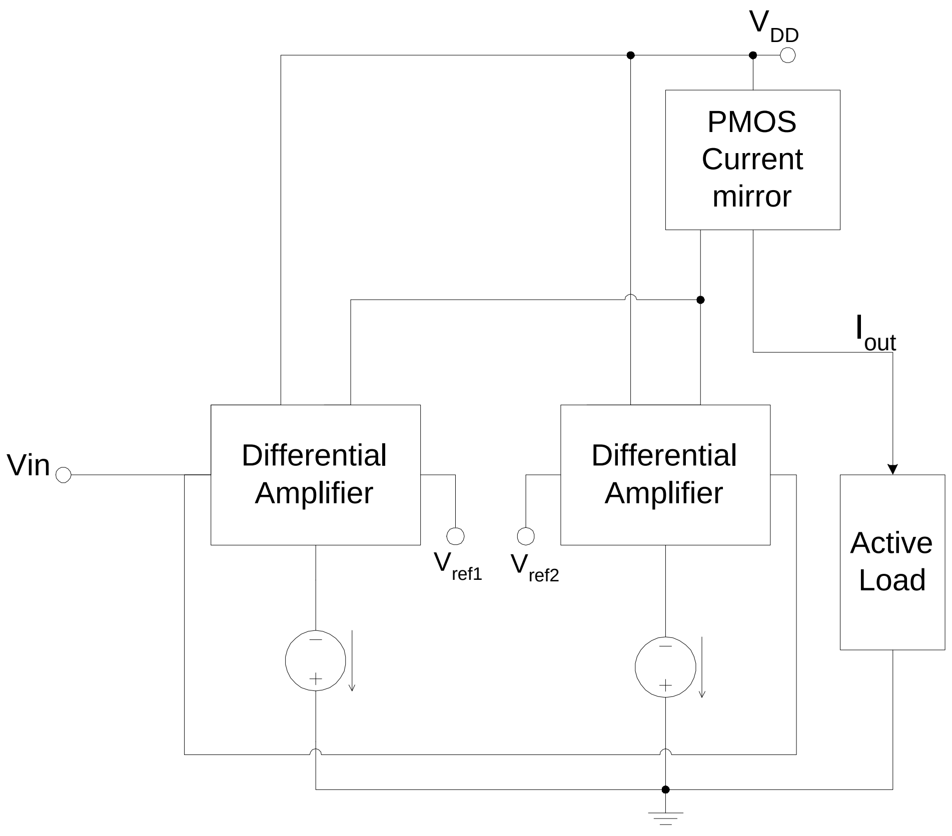

Figure 4.5: Block diagram of a Gaussian Membership function circuit

4.3.1 Theory of operation

Two reference voltages are applied to one of the inputs of each differential amplifier. The average of these two reference voltages determines the mean of a Gaussian-type curve, and the input voltage is applied to the other input of each differential amplifier. The active load was designed using two MOS transistors [39].

A circuit realization for the block diagram of Figure 4.5 is given in Figure 4.6

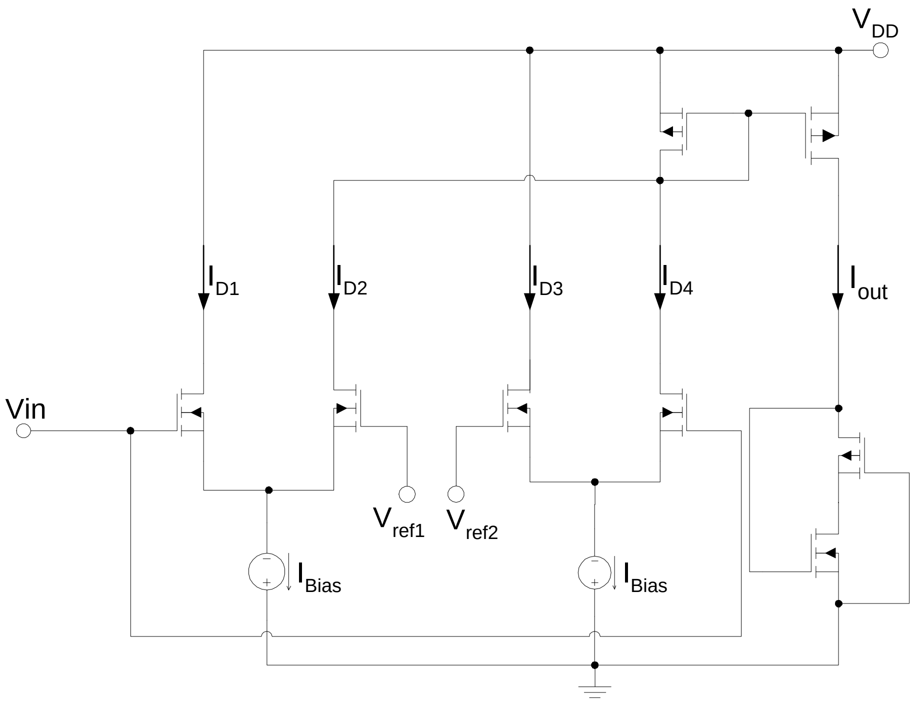

Figure 4.6: Circuit schematic of the membership function circuit

For a transistor operating in the saturation region, its drain currents are given by a quadratic approximation, i.e.

$$I_D = \frac{K'W}{2L}(V_{GS} - V_{th})^2 \tag{4.1}$$

where K' is the transconductance parameter. From this equation the drain currents for the differential amplifiers shown in Figure 4.6 can be derived as

$$I_{D1} = \frac{I}{2} + V_{ID1}\frac{k'W_1}{4L_1}\sqrt{\frac{4I}{\frac{K'W_1}{L_1}} - V_{ID1}^2} \tag{4.2}$$

$$I_{D1} = \frac{I}{2} + V_{ID1}\frac{\beta_1}{2}\sqrt{\frac{2I}{\beta_1} - V_{ID1}^2} \tag{4.3}$$

where

$$V_{IDi} = V_{IN} - V_{REFi}, \qquad \beta_i = \frac{K'W_i}{2L_i}, \qquad (i = 1, 2) \tag{4.4}$$

and

$$I_{D2} = \frac{I}{2} - V_{ID1}\frac{\beta_2}{2}\sqrt{\frac{2I}{\beta_2} - V_{ID1}^2} \tag{4.5}$$

and similarly,

$$I_{D3} = \frac{I}{2} + V_{ID2}\frac{\beta_3}{2}\sqrt{\frac{2I}{\beta_3} - V_{ID2}^2} \tag{4.6}$$

$$I_{D4} = \frac{I}{2} - V_{ID2}\frac{\beta_4}{2}\sqrt{\frac{2I}{\beta_4} - V_{ID2}^2} \tag{4.7}$$

The parameters used by N-MOS and P-MOS transistors have typical values used in a standard MOSIS technology. V_{REF1} must be greater than V_{REF2} in order to generate the Gaussian-type curve. Drain current equations are valid when voltage V_{GS} is higher than V_{th}. Hence, the following condition must be met.

$$-\sqrt{\frac{2I}{\beta_i}} < V_{IDi} < \sqrt{\frac{2I}{\beta_i}}; \qquad i = 1, 2 \tag{4.8}$$

If this is true, the output current I_{OUT} is then given by

$$I_{OUT} = I_{D2} + I_{D4} \tag{4.9}$$

$$I_{OUT} = I - V_{ID1}\frac{\beta_4}{2}\sqrt{\frac{2I}{\beta_4} - V_{ID1}^2} - V_{ID2}\frac{\beta_2}{2}\sqrt{\frac{2I}{\beta_2} - V_{ID2}^2} \tag{4.10}$$

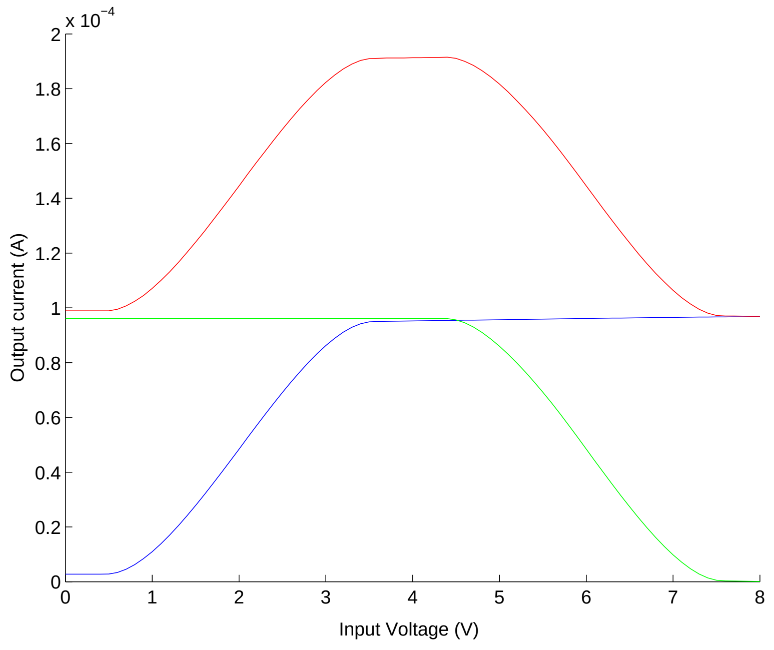

Figure 4.7: Simulation result of the output current I_{OUT}

4.3.2 Simulation results

Simulated characteristics of the output current I_{OUT} are shown in Figure 4.7. The output current is shifted up because two currents I_{D2} and I_{D4} are added.

The values of the control parameters α_i and β_i are chosen to obtain the desired shape of Gaussian-type curves. As the ratio of W_i/L_i increases the value of β_i increases and so does the slope of the edges in the Gaussian-type curve. Figure 4.8 shows the output characteristic of the Gaussian-type curves for different W/L ratios. When symmetrical curves are desired, transistors sizes on both differential amplifiers must match.

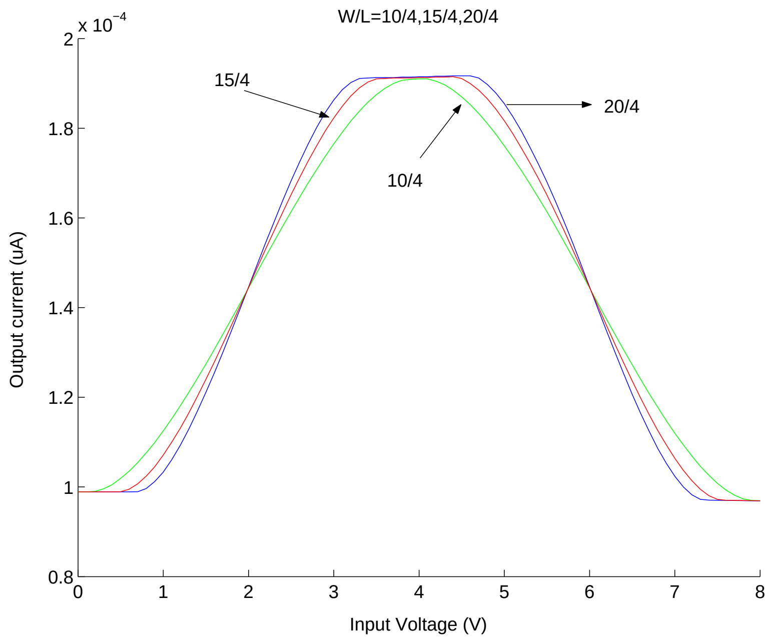

Figure 4.8: Output current simulation results for different W/L ratios

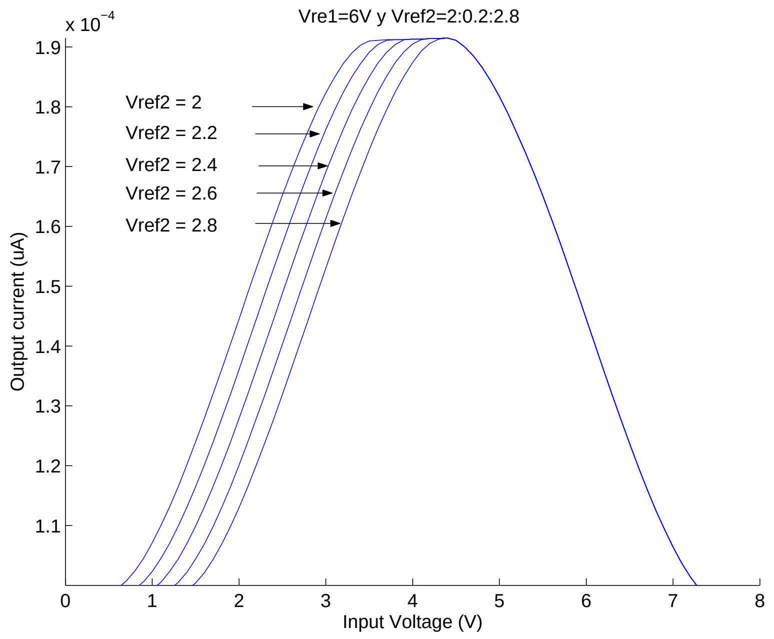

Figure 4.9: Output current simulation results for different reference voltages.

Programmability of the output current shape is also obtained by varying the difference between the reference voltages. Simulation results in Figure 4.9 show that the output current curve approaches a trapezoidal shape as the difference between the reference voltages becomes large.

4.4 Fuzzy Inference circuits

Much effort has been devoted by many researchers to hardware implementation of fuzzy logic, and specially to the application of fuzzy logic inference method to real time systems [25][31]. Among the fundamental fuzzy logic functions the most popular ones to implement logical "AND" and logical "OR" are minimum of (MIN) and maximum of (MAX), respectively. To design a fuzzy logic controller that uses MIN and MAX operations, these must be performed over multiple inputs. The current mode 2-input MIN and MAX circuits for fuzzy logic controllers developed by Yamakawa [38] are shown in Figures 4.10 and 4.11. Here, it should be emphasized that both circuits can handle only two inputs. Theoretically, by taking advantage of the 2-input MIN/MAX circuit, the binary tree structure can be used to implement the multiple input MIN/MAX circuits. In fact, when the number of inputs is large, accumulation of errors of multistage circuits cannot be neglected and overall operation speeds can become low [40].

To reduce the accumulated errors caused by the tree structure of the multi-input MAX/MIN circuit, Sasaki proposed a one-stage current mode multi-input MAX circuit. Based on De Morgan's laws, Sasaki used two's complement circuits and a multi-input MAX circuit to implement each multi-input MIN cell [40].

A block diagram of the MAX circuit is shown in Figure 4.12. This circuit consists

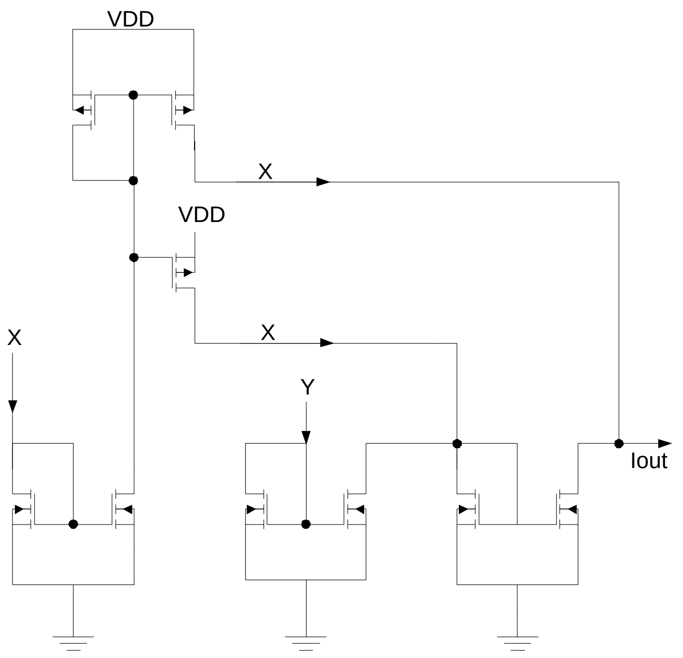

Figure 4.10: Yamakawa's MIN circuit

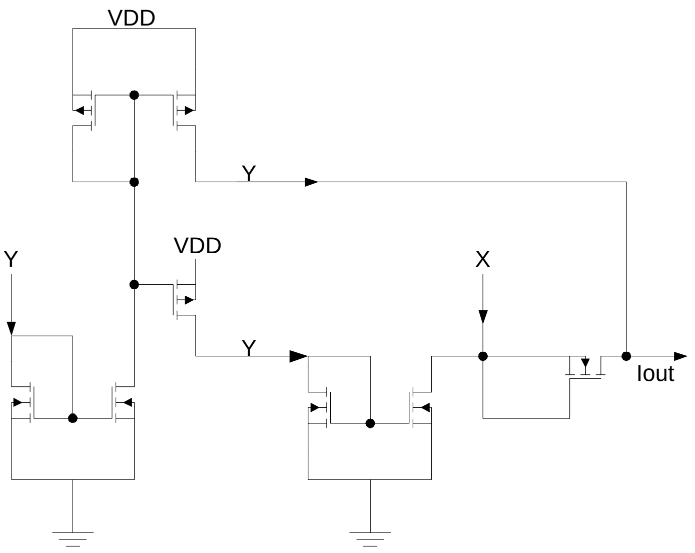

Figure 4.11: Yamakawa's MAX circuit

of NMOS current mirrors connected to form positive-feedback loops. When the current mirrors are assumed ideal, the gain of each feedback loop becomes one. Detailed circuit operation is described in [40].

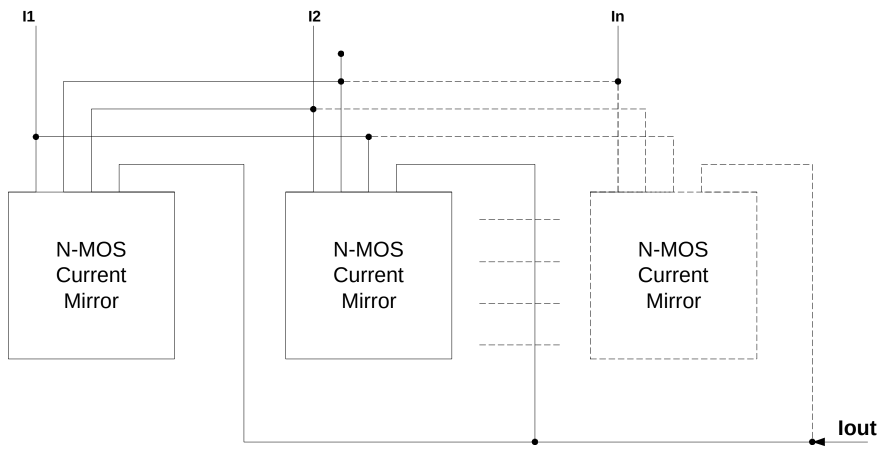

Figure 4.12: Block diagram of Sasaki's n-input MAX circuit

Many dual logic-function pairs have been introduced in fuzzy logic and it has been proved that De Morgan law's hold for several such pairs, including MIN and MAX functions [40]. As a result a multi-input MIN circuit can be synthesized by applying those rules. An n-input MIN circuit can be implemented with n-input MAX circuit and n+1 complement circuits. The complement function can be defined by

$$\bar{I} = MV \ominus I \tag{4.11}$$

where MV is the maximum logical value.

The MIN circuit realization is shown in Figure 4.13. This circuit realizes the logic operation

$$Out = (1 - min(1 - I_1, 1 - I_2, ..., 1 - I_n)) = max(I_1, I_2, ..., I_n) \tag{4.12}$$

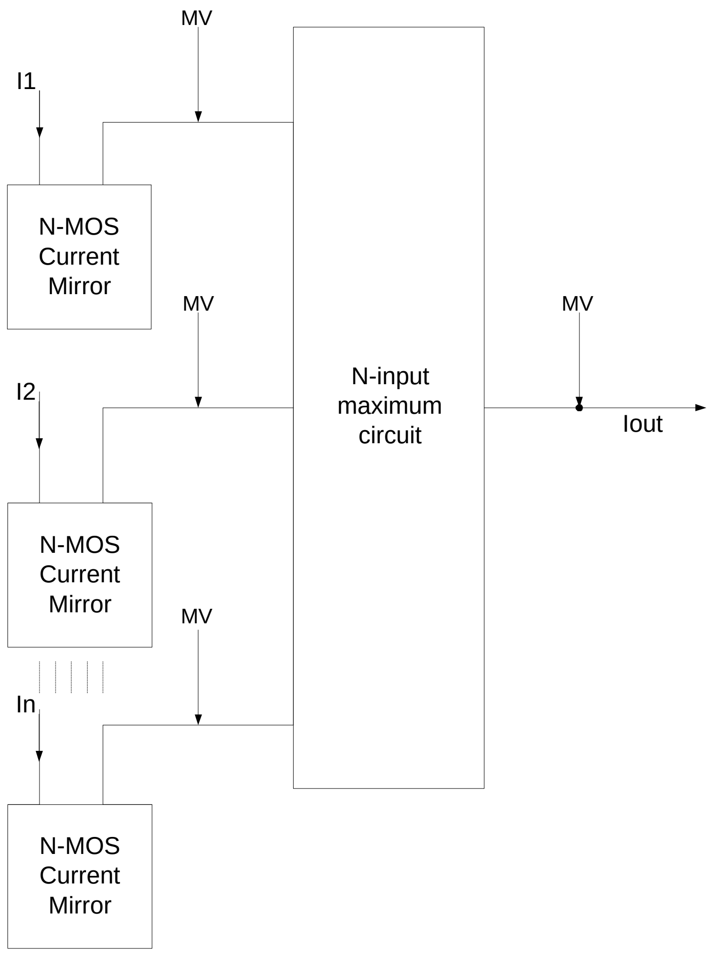

Figure 4.13: Block diagram of Sasaki's n-input MIN circuit

Operating speed and accuracy of these circuits are higher than those from binary tree realizations that use two-input MAX/MIN circuits [38]. However, since this kind of MIN circuit is still a three stage structure. Accumulated errors and slow operating speed induced problems have not been solved yet [32].

According to the concept of the loser-take-all circuit Huang et al. [41], proposed a one stage architecture for the current mode multi-input MIN circuit. The circuit shown in Figure 4.14 is designed with MOS transistors that operate in the saturation region. Each single input MIN cell is made of four transistors. Two of them constitute a basic current mirror to transmit the input current to the MIN cell, another one acts as a current limiter, and the last one forms the multi-input source-couple circuit. The circuit is shown in Figure 4.14. The detail circuit's theory of operation is described in [32].

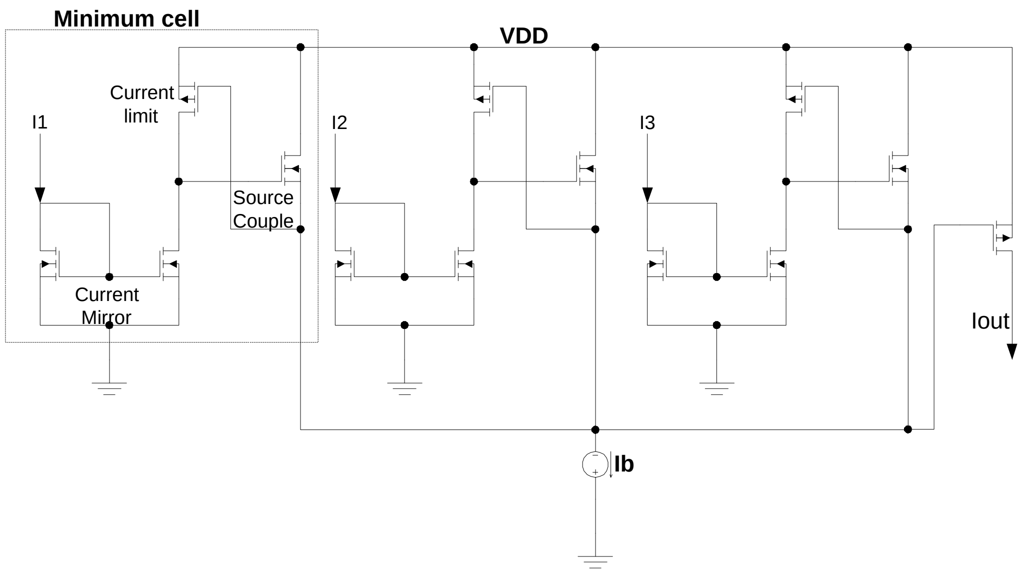

Figure 4.14: Circuit schematic of Huang's Minimum circuit

Table 4.1 shows that the proposed circuit exhibits the advantages of low power dissipation [1]. Since, this structure is regular and modular, it can be easily expanded to accommodate any number of inputs. Furthermore, its one-stage architecture avoids the problem of accumulated errors and allows high operating speed.

Author	Number of transistors for n-input minimum circuit	Power Dissipation
Yamakawa [38]	$7(n-1)$	1.34mW (2 inputs)
Sasaki [40]	$(n+1)^2+2n$	0.2mW (3 inputs)
Huang [32]	$4n+1$	0.125mW (3 inputs)

Table 4.1: Comparison of various multi-input MIN circuits

4.4.1 Simulation results

PSpice simulations have been performed in order to verify the performance of each architecture. A two-input MIN circuit (Figure 4.10), and three-input MIN circuits (Figures 4.13 and 4.14) were design with MOS transistor arrays. The DC output current characteristics are shown in Figures 4.15, 4.16 and 4.17. They agree very well with the theoretical prediction and they indicate that Huang's three-input MIN circuit is the most accurate.

In order to estimate the speed of response of these circuits, PSpice transient simulations of the each architecture were conducted. The results for the output current (I_{OUT}) are given in Figures 4.18, 4.19 and 4.20. They show that Huang's MIN circuit has the fastest response, and that it tracks very more closely changes in the minimum of the inputs.

[1]Power dissipation for 3-input, except 2-input Yamakawa's MIN circuit

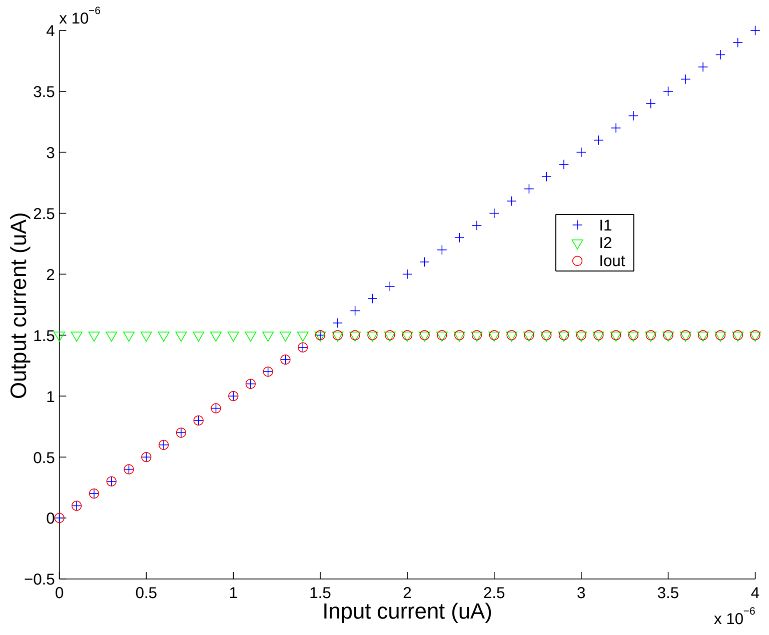

Figure 4.15: DC Characteristic of Yamakawa's two-input Minimum circuit

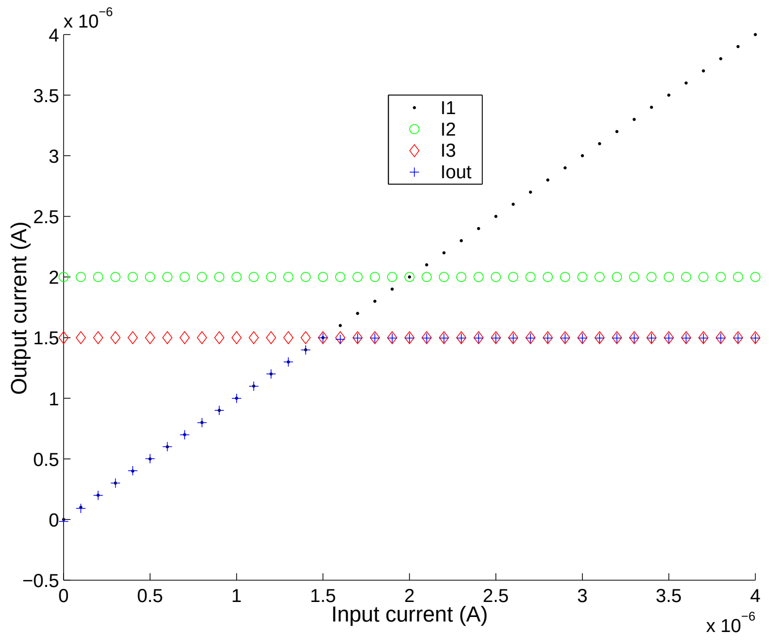

Figure 4.16: DC Characteristic of Sasaki's three-input Minimum circuit

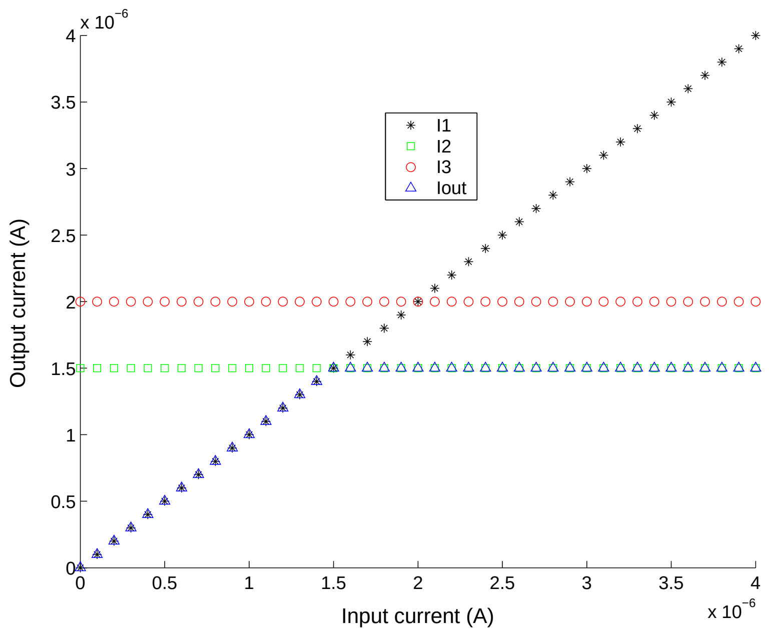

Figure 4.17: DC Characteristic of Huang's three-input Minimum circuit

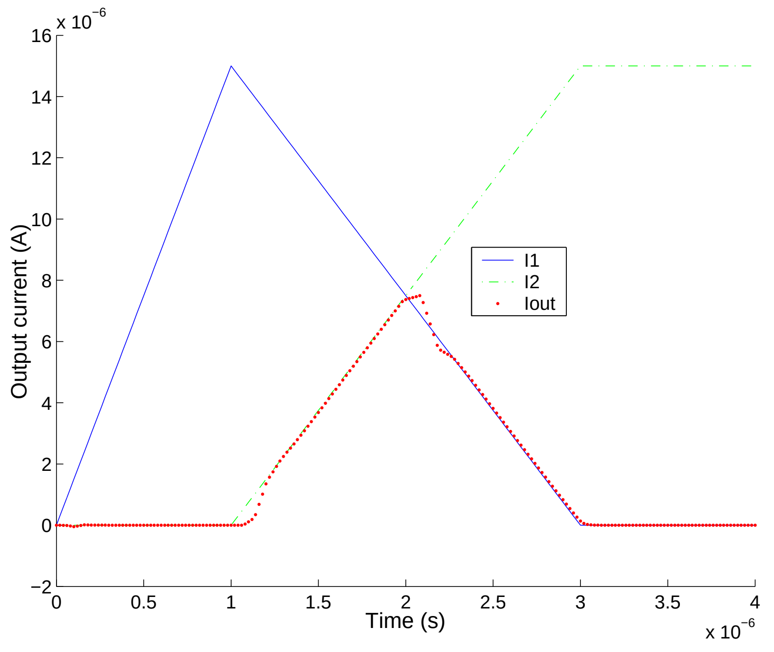

Figure 4.18: Transient response of Yamakawa's two-input Minimum circuit

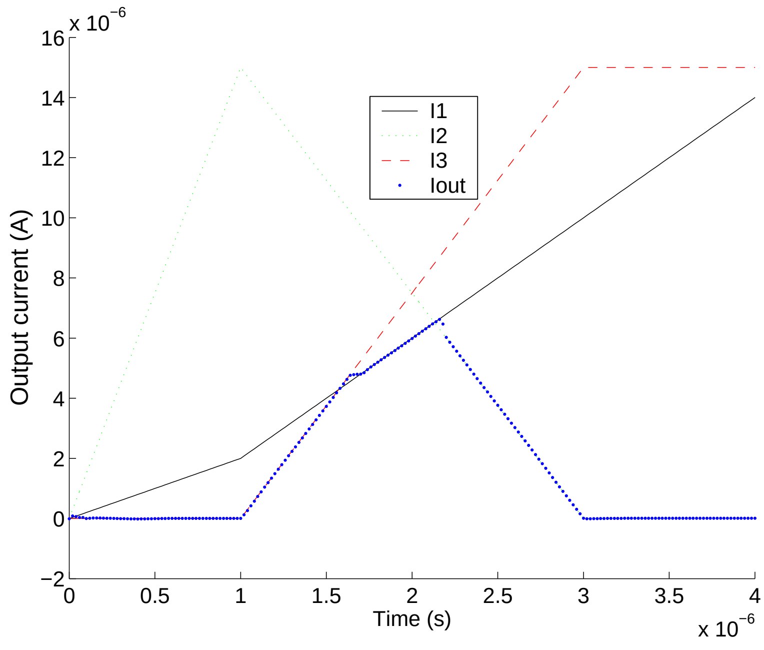

Figure 4.19: Transient response of Sasaki's three-input Minimum circuit

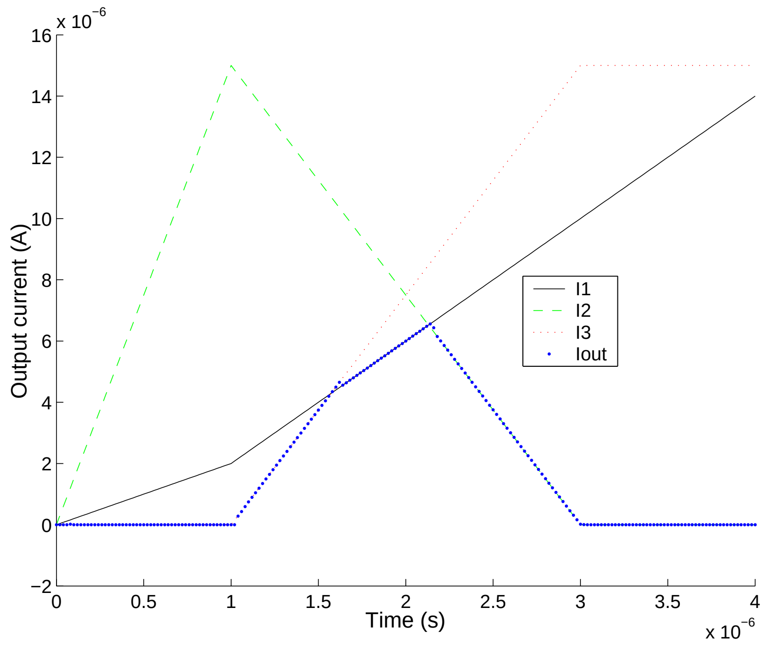

Figure 4.20: Transient response of Huang's three-input Minimum circuit

4.5 Defuzzification circuit

Defuzzification produces a *crisp* output for our FLS from the fuzzy set that is the output of the inference block. The immediate payoff is simplified defuzzification process. A Takagi-Sugeno type system was chosen to generate the *crisp* output. The global output is a crisp number computed by multiplying each input by a constant and then adding up the results, i.e,

$$y = \frac{\sum_{i=1}^{I} \lambda_i x_i}{\sum_{i=1}^{I} \lambda_i} \tag{4.13}$$

Which can be implemented with wide band current mode Operational Transconductance Amplifier (OTA) based analog multiplier-dividers. Some characteristics of a realization that solely used OTA are: simplicity, low cost, single high impedance input node, wide range current controlled transconductance and passive component independent integrability and programmability [33].

Figure 4.21 shows a circuit schematic of the CMOS basic OTA used here. When the OTA operate in the linear range, its transconductance gain is given by

$$gm = \sqrt{2K' \frac{W}{L} I_{Bias}} \tag{4.14}$$

which can be tuned by adjusting the DC bias current I_{Bias}.

4.5.1 Theory of operation

Circuit schematic of the proposed current-mode analog multiplier-divider is shown in Figure 4.22. The input signal current I_{in1} is applied to OTA1, which behaves as a current-controlled grounded resistor due to the unity feedback loop around itself.

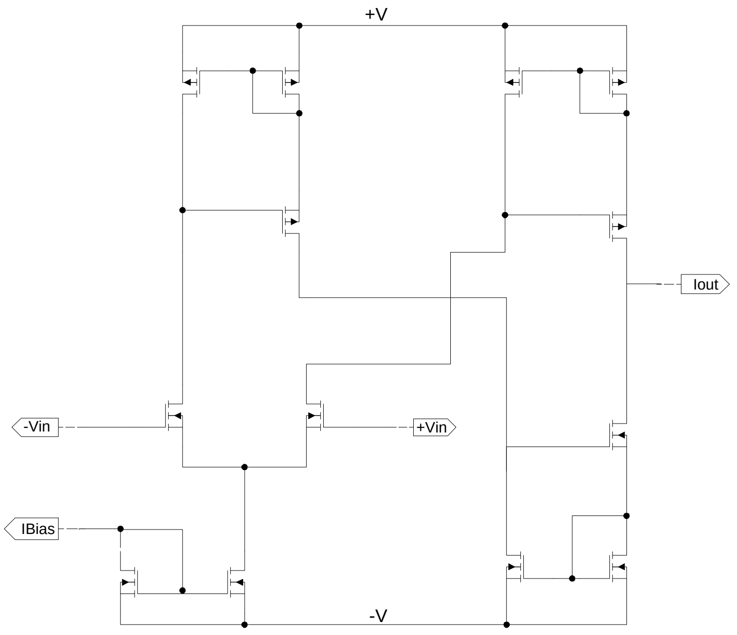

Figure 4.21: Circuit schematic of the OTA

This loop in turn forces I_{OUT} to be equal to I_{IN1}. The voltage produced by OTA1 is then applied both to OTA2 and OTA3. An input signal current I_{in2} is added with I_{Bias2} to form the bias current I_{Bias} for OTA2. For reasons to be seen later. Output currents I_{O2} and I_{O3} from OTA2 and OTA3, respectively, can be derived as follows

$$V_{O1} = \frac{I_{in1}}{g_{m1}}, \tag{4.15}$$

$$I_{O2} = g_{m2}V_{O1} = \frac{g_{m2}}{g_{m1}}I_{in1}, \tag{4.16}$$

$$I_{O2} = \frac{\sqrt{(I_{Bias2} + I_{in2})}}{\sqrt{I_{Bias1}}}I_{in1}, \tag{4.17}$$

and

$$I_{O3} = -g_{m3}V_{O1} = -\frac{g_{m3}}{g_{m1}}I_{in1}, \tag{4.18}$$

$$I_{O3} = -\frac{\sqrt{I_{Bias3}}}{\sqrt{I_{Bias1}}}I_{in1}, \tag{4.19}$$

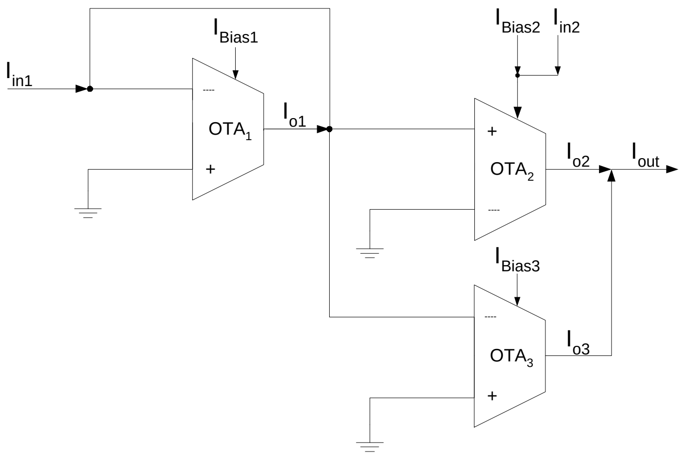

Figure 4.22: Current mode analog multiplier-divider circuit

where $g_{m1} = \sqrt{2K'\frac{W}{L}I_{Bias1}}$, $g_{m2} = \sqrt{2K'\frac{W}{L}(I_{Bias2} + I_{in2})}$ and $g_{m3} = \sqrt{2K'\frac{W}{L}I_{Bias3}}$ are the transconductance gains for OTA_1, OTA_2 and OTA_3, respectively. The multiplier-divider output current I_{OUT} given by the sum of I_{O2} and I_{O3}, and can be expressed as

$$I_{OUT} = I_{O2} + I_{O3} = \frac{\sqrt{(I_{Bias2} + I_{in2})}}{\sqrt{I_{Bias1}}}I_{in1} - \frac{\sqrt{I_{Bias3}}}{\sqrt{I_{Bias1}}}I_{in1} \tag{4.20}$$

$$I_{OUT} = \frac{I_{in1}}{\sqrt{I_{Bias1}}}(\sqrt{I_{Bias2} + I_{in2}} - \sqrt{I_{Bias3}}), \tag{4.21}$$

which is in the form of a current mode analog multiplication-division function. The circuit can work as a four-quadrant multiplier if I_{in1} and $(\sqrt{I_{Bias2} + I_{in2}} - \sqrt{I_{Bias3}})$ are the input signals. On the other hand, the circuit can work as a two-quadrant divider if I_{in1} (or $\sqrt{I_{Bias2} + I_{in2}} - \sqrt{I_{Bias3}}$) and I_{Bias1} are the input signals.

If $I_{Bias1} = I_{Bias2} = I_{Bias3} = I_{Bias}$, then I_{OUT} can be expressed as

$$I_{OUT} = I_{in1}\left\{\sqrt{\frac{I_{Bias} + I_{in2}}{I_{Bias}}} - 1\right\} \tag{4.22}$$

4.5.2 Simulation results

PSpice simulations have been conducted to characterize the performance of the multiplier-divider, which was designed with MOS transistors arrays. The DC transfer characteristics of the presented multiplier-divider shown in Figure 4.23 indicate high linearity within the following ranges:

$$-10\mu A \leq I_{in} \leq 10\mu A \tag{4.23}$$

$$25\mu A \leq I_{Bias} \leq 355\mu A \tag{4.24}$$

$$-18\mu A \leq I_{out} \leq 18\mu A \tag{4.25}$$

Simulation results and theoretical predictions agree with 3% error.

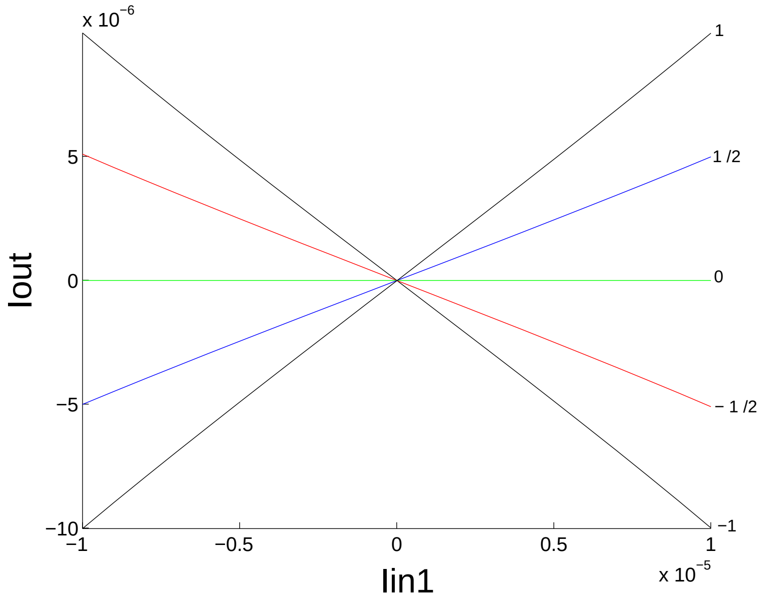

Figure 4.23: DC Characteristic of the multiplier-divider

Figure 4.24 shows the input and output wave-forms when I_{IN1} is a 1MHz $20\mu A_{p-p}$ sine wave, and I_{in1}, I_{Bias1}, $I_{Bias2} + I_{in2}$ are $25\mu A$, $81\mu A$ and $100\mu A$ dc values respectively.

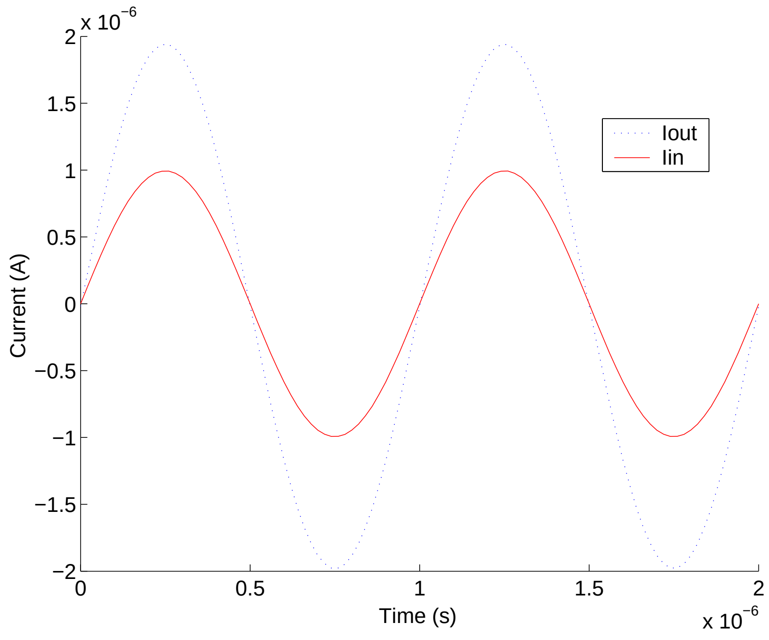

Figure 4.24: Input-Output wave-forms of the multiplier-divider

Separate analog implementation of multiplication and division in equation (4.13) requires a bigger area than that used by our multiplier-divider design.

In this work the following approach is used:

If I_{Bias} in equation (4.22) is fixed, then from equation (4.13) results

$$\frac{\lambda_i}{\sum_{i=1}^{I} \lambda_i} x_i = \frac{\sqrt{I_{Bias} + I_{in2}} - \sqrt{I_{Bias}}}{\sqrt{I_{Bias}}} I_{in1} \tag{4.26}$$

where λ can be tuned by the input current I_{in2}, under the constraint

$$\sum_{i=1}^{I} \lambda_i = \sqrt{I_{Bias}} \tag{4.27}$$

Chapter 5

Implementation and simulation results

Performance of the proposed approach is assessed here by implementing a three-input single-output Analog Fuzzy Controller with two rules. It has been simulated and compared with the mathematical simulation provided by Matlab Fuzzy Tool.

Antecedent part consists of three linguistic terms where each is characterized by a membership function. The shapes of these membership functions were programmed through control voltages (two for each membership function).

The structure of the inference engine can be expanded by joining n basic cells in order to implement an n-input MIN circuit. The weight for each rule can be changed by modifying the input bias current of the respective multiplier according to equations (4.26) and (4.27).

Block diagram of the complete fuzzy controller is presented in Figure 5.1, where MFC and MIN blocks represent the membership function and minimum extraction function respectively. Since this outputs of the multiplier-divider circuits are currents, they are added by hard-wiring them together.

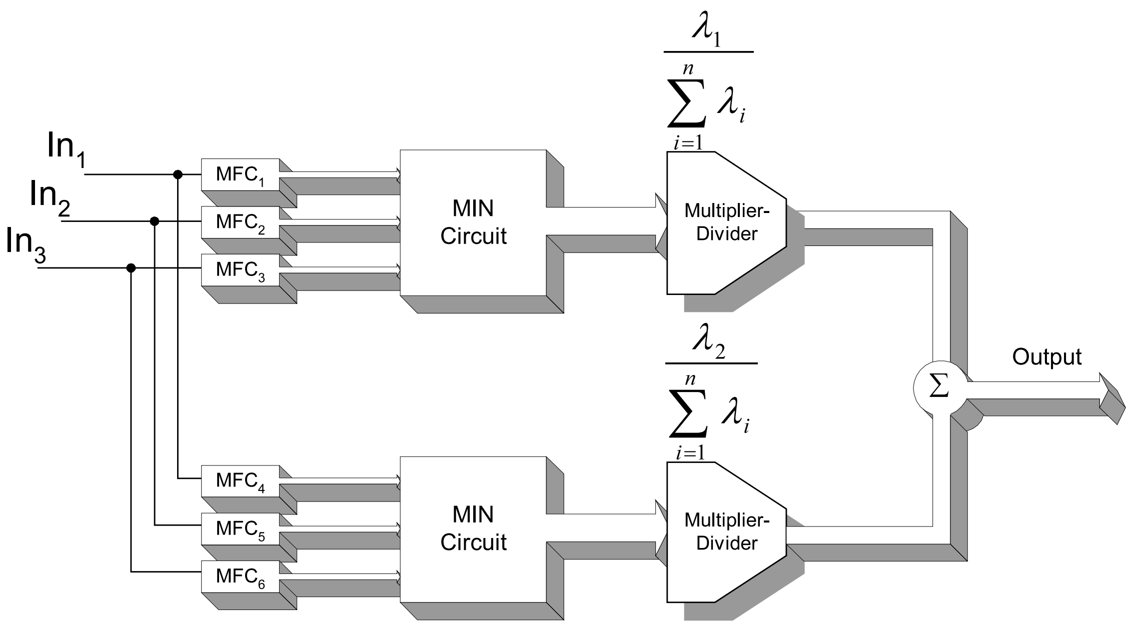

Figure 5.1: Block diagram of the Analog Fuzzy Controller

5.1 Application

To make a controller for the water level in a tank work at least the current water level must be known and the valve opening must be adjusted (to regulate the amount of water coming), Figure 5.2. The controller's inputs are water level error (desired water level minus actual water level) and water level rate of change. The controller decides the valve's rate of movement from its current position. The commands for the valve's movement can be: "no changes","open fast" to allow more water flow or "close fast" to reduce water flow. Similarly, commands for the level can be "low", "okay" or "high" and the level rate can be "negative", "zero or none" or "positive".

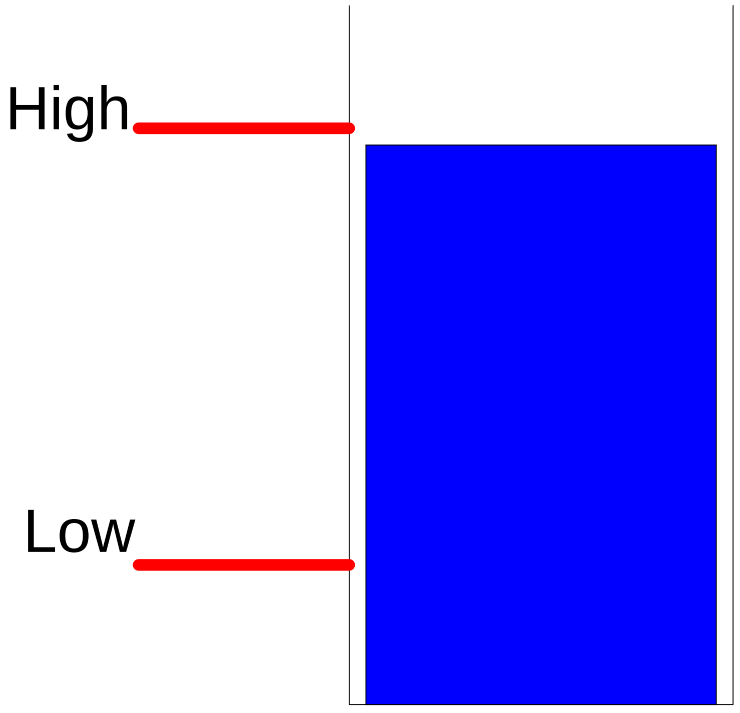

Figure 5.2: Water Level Control

With these terms a first set of rules might be:

1. If (level is okay) then (valve position unchanged)
2. If (level is low) then (valve opens fast)
3. If (level is high) then (valve closes fast)
4. If (level is okay) and (level rate negative), then (valve closes slowly)
5. If (level is okay) and (rate is positive), then (valve open slowly)

5.1.1 Matlab fuzzy controller

The block diagram shown in Figure 5.3 corresponds to a water level control system application in the demostration example section of Matlab [42]. The example is called sltank [42]. Figure 5.4 shows the internal structure of the fuzzy logic controller included in the block diagram of Figure 5.3. The controller's membership functions are depicted in Figures 5.5 and 5.6, whereas its transfer characteristics are displayed as a surface plot, valve command versus water level and its rate of change, in Figure 5.7.

Table 5.1 shows normalized input value range for each linguistic variable used in this matlab example.

Linguistic Variable	Range
Level	$\pm$ 1
Rate	$\pm$ 0.1
Valve	$\pm$ 1

Table 5.1: Matlab input values

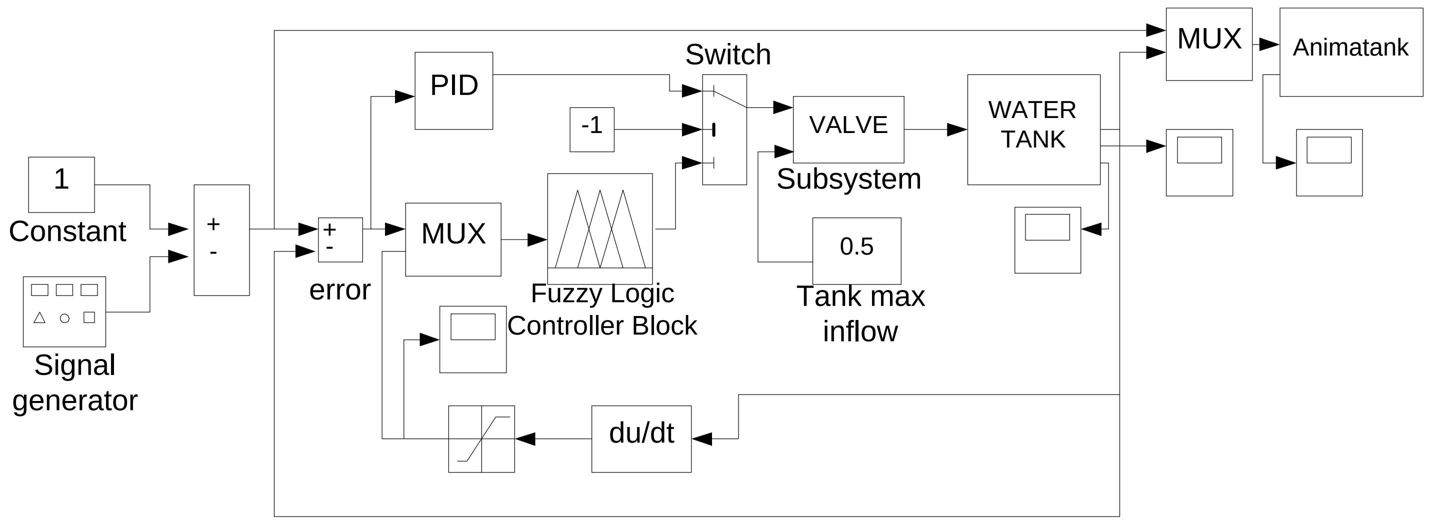

Figure 5.3: Simulink block diagram

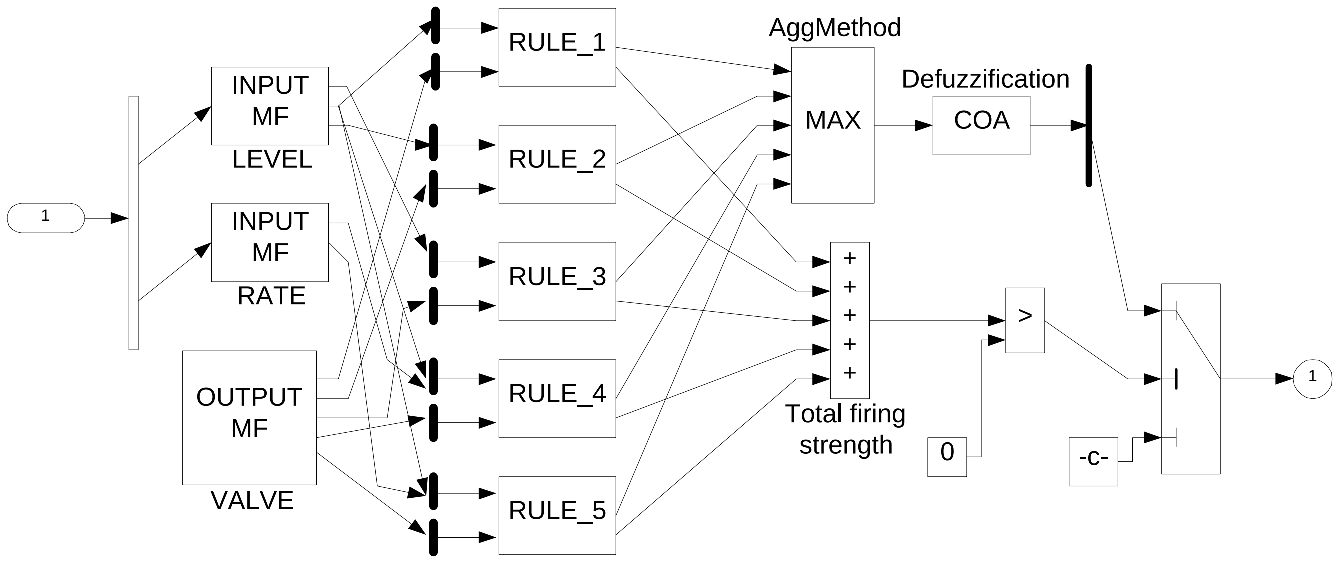

Figure 5.4: Circuit schematic under Fuzzy Logic Controller block

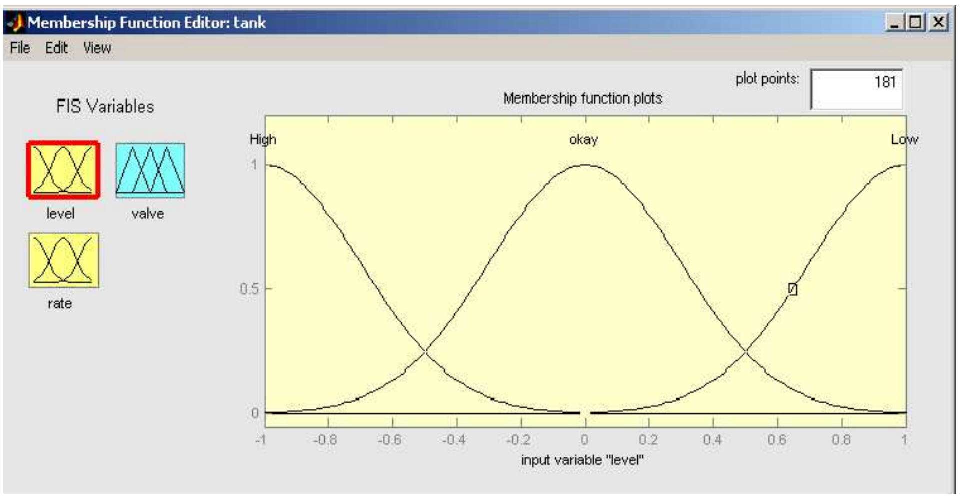

Figure 5.5: Matlab level's fuzzy set

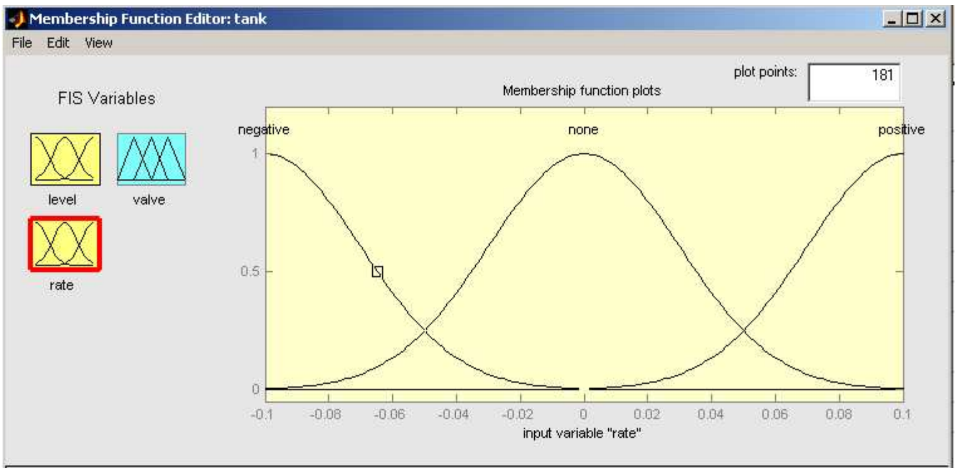

Figure 5.6: Matlab Rate's fuzzy set

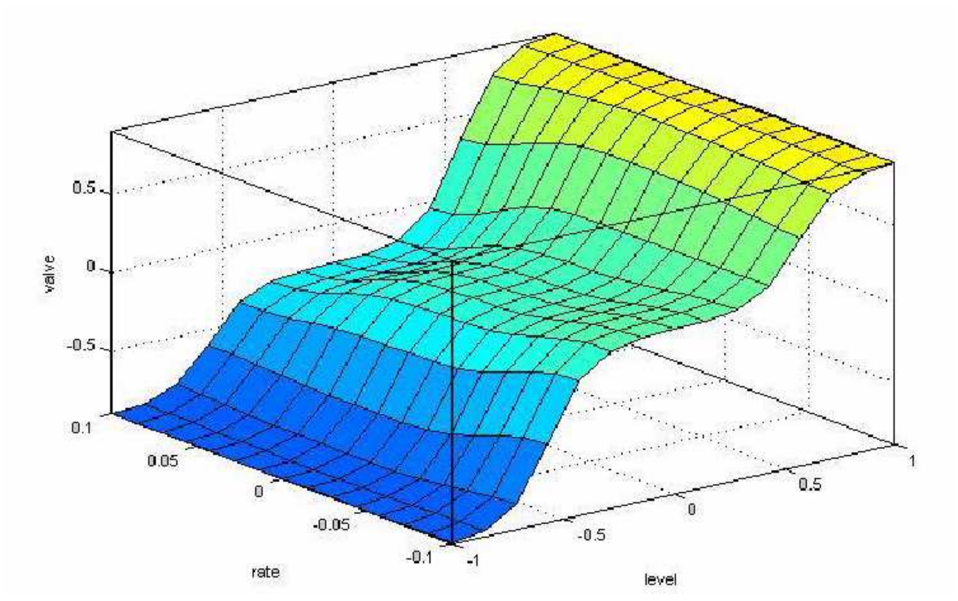

Figure 5.7: Output of Matlab Fuzzy Logic Controller

5.1.2 Proposed Analog Fuzzy Controller

The proposed Analog Fuzzy Controller consists solely of MOS transistors that are compatible with standard CMOS fabrication processes. Structure diagram of the fuzzy controller to regulate the water level in a tank is shown in Figure 5.8. Simulations for this controller were conducted in PSpice at level-3, where AMI06N and AMI06P were used for the CMOS transistor models.

Due to circuit constraints the linear range, for the linguistic variables, in our analog fuzzy controller got limited to those values shows in Table 5.2.

To convert these ranges to the normalized ones used in the Matlab simulation the followings transformations are applied:

$$Normalized\, level = level - 1.5V \tag{5.1}$$

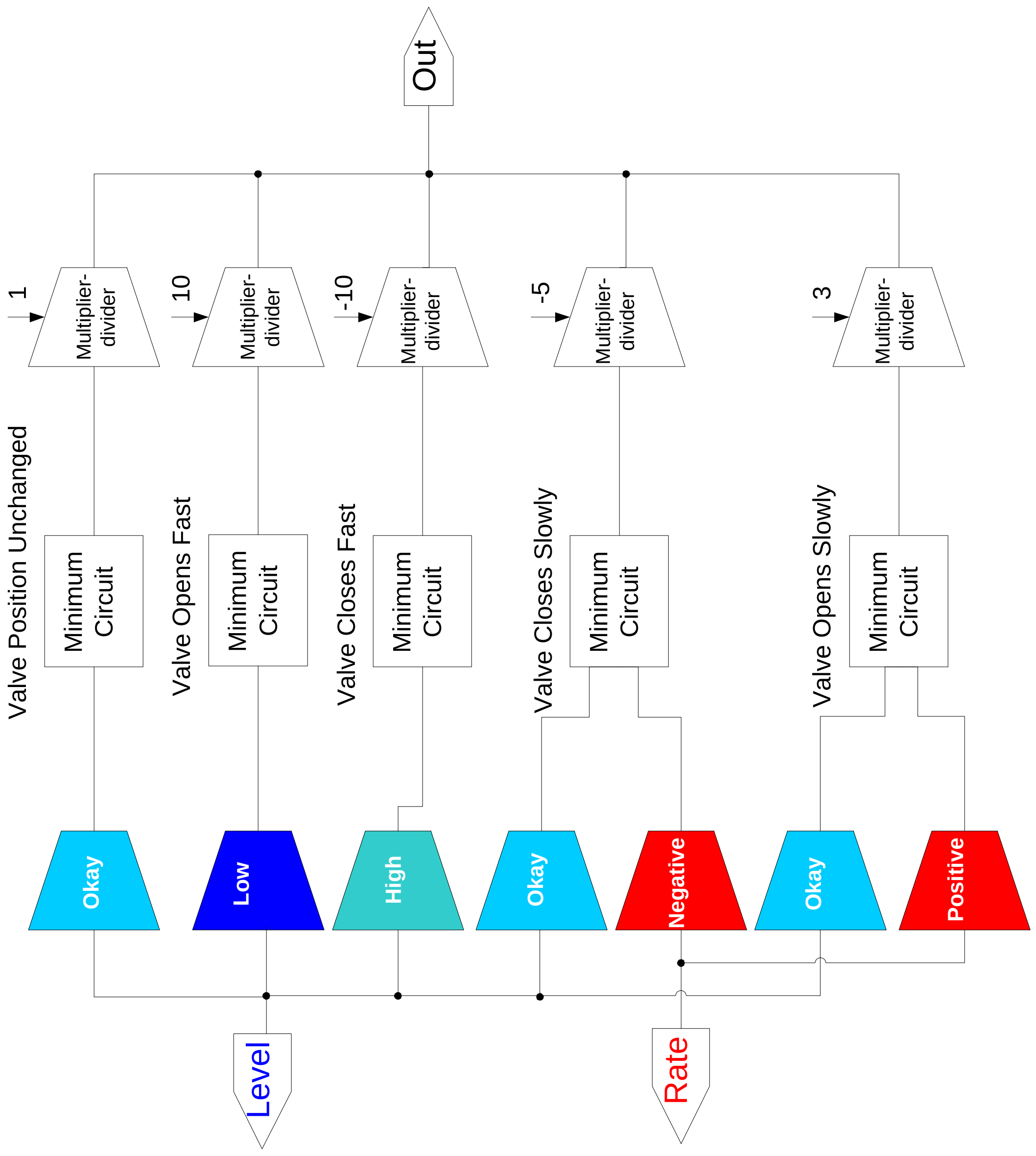

Figure 5.8: Fuzzy controller to regulate the water level in a tank

Linguistic Variable	Range
Level	0.5V to 2.5V
Rate	0.5V to 2.5V
Valve	-2.8μA to 4μA

Table 5.2: Analog fuzzy Controller input voltages and output current

$$Normalized\ rate = (rate - 1.5V)/10 \tag{5.2}$$

$$Normalized\ valve = 0.2941(valve) - 0.1765\mu A \tag{5.3}$$

Figure 5.9 and 5.10 show the membership functions used in this example. Table 5.3 shows reference voltage values for each membership function. Shapes of the curves used in the fuzzification stage were heuristically matched to those in the Matlab model by varying the ratio W/L mentioned above.

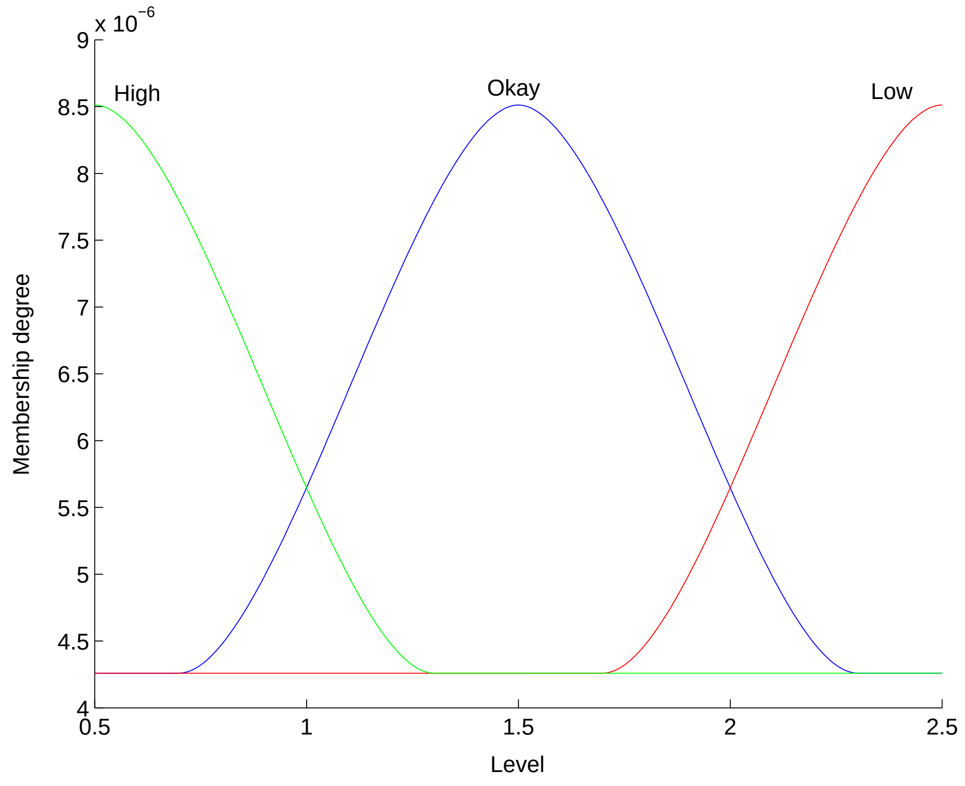

Figure 5.9: Fuzzy sets assigned to the variable Level

Figures 5.11 show contours controller's output from 0.5V to 2.5V in Level for different rates. Comparing these contours with those sketched from the matlab surface of Figure 5.7 we can appreciate a good agreement in the expected results.

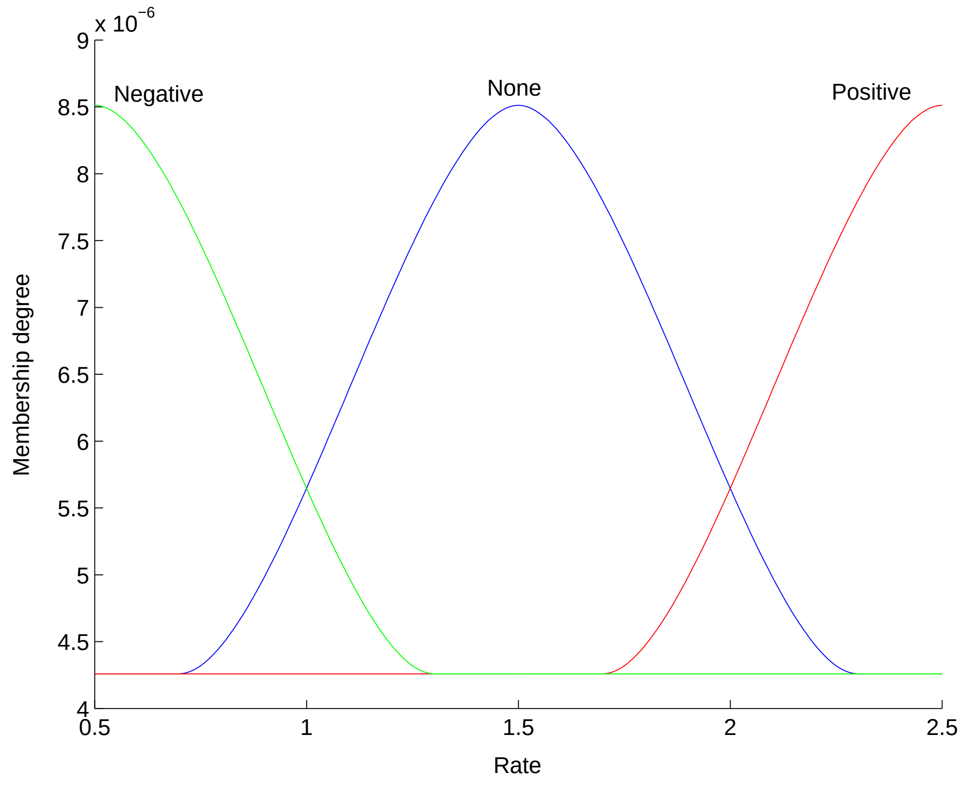

Figure 5.10: Fuzzy sets assigned to the variable Rate

Fuzzy set	Reference voltage 1	Reference voltage 2
High/Negative	0.9V	0V
Okay/None	1.9V	1.1V
Low/Positive	3V	2.1V

Table 5.3: Reference voltages for each membership function

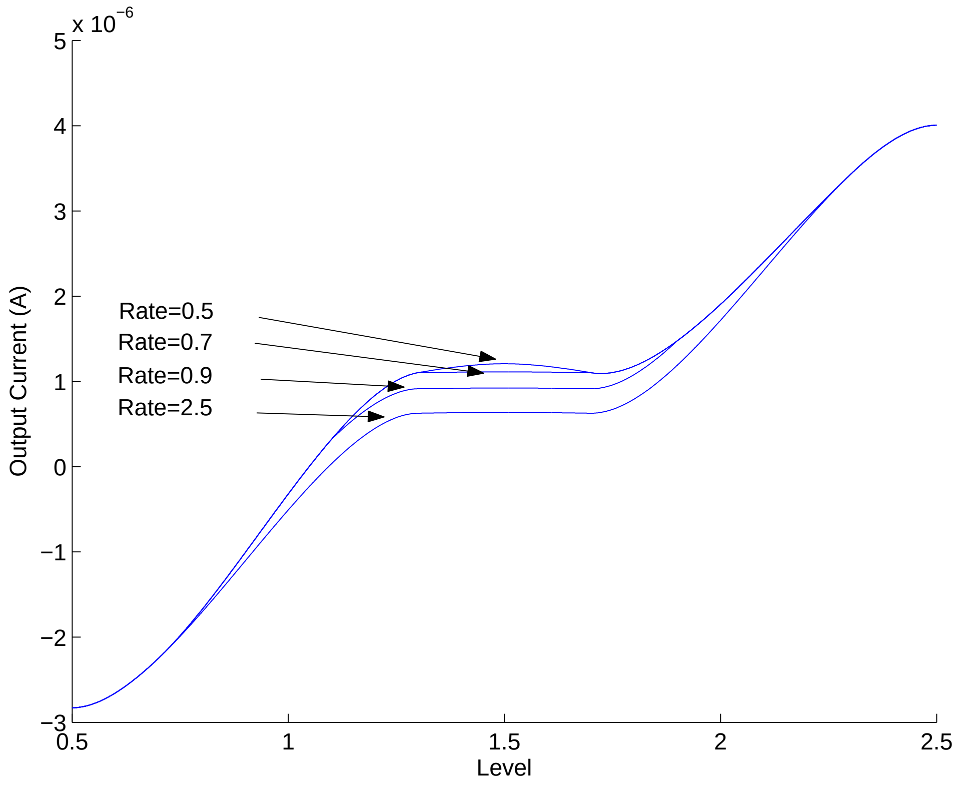

Figure 5.11: Contours controller's output from 0.5V to 2.5V in Level for different rates

Table 5.4 shows weights used with each of these rule, where each weights represent the rule activation degrees.

Rule	Weight
1	0
2	10
3	-10
4	-5
5	3

Table 5.4: Weights used with each rule

Simulations results in Figures 5.7 and 5.11 prove that the analog fuzzy controller performs the same tasks as the fuzzy controller in Matlab; and hence that the former can replace the latter in simple control system applications such as the one discussed.

Tables 5.5, 5.7 and 5.6 show the currents and voltages values used in Analog Fuzzy Controller.

Membership Function Circuit			
Fuzzy	**References Voltages**		
Sets	V_{REF1}	V_{REF2}	I_{BIAS}
Okay	1.9V	1.1V	4.5μA
Low	3V	2.1V	4.5μA
High	0.9V	0V	4.5μA
Negative	1.9V	1.1V	4.5μA
None	3V	2.1V	4.5μA
Positive	0.9V	0V	4.5μA

Table 5.5: Currents and voltages values used in Membership Function Circuit

3-input Minimum Circuit
I_{BIAS}=10μA

Table 5.6: Currents and voltages values used in 3-input Minimum Circuit

Multiplier-Divider Circuit				
Rule	I_{BIAS}			
No.:	1	2	3	I_{IN2}
1	100μA	100μA	100μA	0
2	100μA	100μA	100μA	224μA
3	100μA	100μA	100μA	-96μA
4	100μA	100μA	100μA	-75μA
5	100μA	100μA	4μA	189μA

Table 5.7: Currents and voltages values used in Multiplier-Divider circuit

Chapter 6

Conclusions and future work

An analog CMOS based fuzzy controller whose membership function can be programmed via reference dc voltages has been developed. The transfer characteristics represent the controller's fuzzy set. Each input generates, according to its degree of membership within the set, its respective output and the slopes of the edges in the membership curves are determined by the CMOS transistor's ratio W/L.

The inference stage uses Sugeno's FIS and it is implemented in CMOS technology with an n-input minimum (MIN) extracting circuit. The n-input minimum (MIN) extracting circuit can handle as many fuzzy sets or variables as needed, (hot, warm, cold, chilly, etc.) and it only requires 4n+1 transistors.

The defuzzification stage utilizes an efficient CMOS based multiplier-divider that also exhibits small size (area) and programmability. The programming can be done through dc (bias) currents that set the weighing factors for the respective rules. Originally, this circuit was design with bipolar-based OTA as the active circuit element [33] and the assumed that a CMOS-based OTA which can be linearly tuned by the external DC current bias current can also be used. According with simulations and derived equations, it depends on the square root of the bias currents.

Simulations for the fuzzy controller were conducted both in Matlab with its fuzzy toolbox and in PSpice at level-3. In PSpice AMI06N and AMI06P were used for the CMOS transistors models. Results from the circuit simulator of the analog fuzzy controller agree very well with those from Matlab.

Some directions for future work are:

- Controller's circuit layout and thorough behavior analysis through CAD tools such as Cadence.
- Physical implementation and testing.
- Since each stage topology is fixed, possible development of an analog HDL program to systematically improve programmability and modularity.
- To incorporate the elements of programming control, i.e., the circuits that provide the reference voltages, bias current, and so on.

Bibliography

[1] Riza C. Berkan and Sheldon L. Trubatch. *Fuzzy Systems Design Principles Building Fuzzy IF-THEN Rule Bases.* IEEE PRESS, 1997.

[2] I. Baturone and S. Sánchez-Solano. Microelectronic design of fuzzy logic-based systems. *The CRC Press International Series on Computational Intelligence.*, page 336, 2000.

[3] T. Takagi and M. Sugeno. Derivation of fuzzy control rules for human operator's control actions. *In Proc. IFAC Symposium on Fuzzy Information, Knowledge Representation and Decision Analysis*, pages 55–60, Marseille, 1983.

[4] S. Korner. Laws of thought. *Encyclopedia of Philosophy*, Vol. 4:pp. 414–417, 1967.

[5] C. Lejewski. Jan lukasiewicz. *Encyclopedia of Philosophy*, Vol. 5:pp. 104–107, NY 1967.

[6] Pragya. Lotfi Zadeh Agarwal. Fuzzy logic-incorporating real world vagueness.

[7] L. A. Zadeh. Fuzzy set. *Information and Control*, Vol. 8(3):338–353, June 1965.

[8] Timothy J. Ross. *Fuzzy Logic With Engineering Applications.* International edition, 1995.

[9] S.K. Pal and S. Mitra. Multilayer perceptron, fuzzy sets, and classification. *IEEE Trans. Neural Networks*, 3:683–697, 1992.

[10] J.M. Mendel. Fuzzy logic systems for engineering: A tutorial. *Proceedings of the IEEE*, Vol. 83(3):pp. 345–377, March 1995.

[11] H.-J. Zimmermann. *Fuzzy Set Theory and its Applications.* Kluwer Academy, 1985.

[12] S. Guo and L. Peter. A reconfigurable analog fuzzy logic controller. *IEEE World congress on computational intelligence. Proceeding of the Third IEEE conference on*, Vol. 1:pp. 124–128, June 1994.

[13] I. Baturone, S. Sánchez-Solano, A. Barriga, and J. L. Huerta. Design issues for VLSI implementation of universal aproximator fuzzy system. *Proc. world multiconference on circuits, systems, communications and computers. Athens*, pages pp. 6471–6476, 1999.

[14] I. Baturone, S. Sánchez-Solano, A. Barriga, and J. L. Huerta. Towards the IC implementation of adaptive fuzzy systems. *IEICE Transactions on fundamentals.*, Vol. E81-A(9):pp. 1877–1885, 1998.

[15] A. Rodríguez-Vásquez, R. Navas, M. Delgado-Restituto, and F. Vidal-Verdú. A modular programmable cmos analog fuzzy controller chip. *Circuits and Systems II: Analog and Digital Signal processing, IEEE Transactions*, Vol. 46(3):pp. 251–265, March 1999.

[16] K. Hirota. *Industrial Applications of Fuzzy Technology.* Tokyo, 1993.

[17] C.-H. Chang and J.-Y. Cheung. The dimensions effect of fam rule table in fuzzy pid logic control systems. *Second IEEE int. Conf. on Fuzzy Systems*, 1:441–446, 1993.

[18] Earl Cox. *The Fuzzy Systems Handbook*. McGraw-Hill, Inc, July 1994.

[19] H. Watanabe, W. Dettroll, and K. Yount. A VLSI fuzzy logic controller with reconfigurable, cascadable architecture. *IEEE Solid State Circuits.*, Vol. 25(2):pp. 376–382, April 1990.

[20] I. Baturone, S. Sánchez-Solano, and J. L. Huerta. Self-checking current mode analog memory. *Electronic Letters*, Vol. 33(16):pp. 1349–1350, 1997.

[21] Behzad Razavi. Design of analog cmos integrated circuits. 2000.

[22] T Yamakawa. A simple fuzzy computer hardware system employment min-max operations-a challenge to 6th generation computer. *in Proc. 2nd IFSA World Congress*, pages Pp. 827–830, Tokyo, 1987.

[23] T. Yamakawa. High speed fuzzy controller hardware system: The mega flips machine. *Elsevier Science Pub. Comp. Inc., Information Sciencies*, Vol. 45:113–128, 1988.

[24] T. Yamakawa. A fuzzy inference engine in nonlinear analog mode and its applications to a fuzzy logic controller. *IEEE Transactions Neural Networks*, May 1993.

[25] S. Guo, L. Peters, and H. Surmann. Design and application of an analog fuzzy logic controller. *Fuzzy system, IEEE transaction*, Vol. 4(4):pp. 429–438, November 1996.

[26] M. Sasaki, N. Ishikawa, F. Ueno, and T. Inoue. Current-mode analog fuzzy hardware with voltage input interface and normalization locked loop. *IEICE Transactions on Fundamentals*, Vol. E75-A, N. 6,:650–654, 1992.

[27] I. Baturone, S. Sanchez-Solano, A. Barriga, and J.L.; Huertas. Flexible fuzzy controllers using mixed-signal current-mode techniques. *Fuzzy Systems, 1997., Proceedings of the Sixth IEEE International Conference on*, Vol. 2:pp. 875–880, 1-5 July 1997.

[28] I. Baturone, S. Sanchez-Solano, A. Barriga, and J.L.; Huertas. Mixed-signal design of a fully parallel fuzzy processor. *Electronic letters*, Vol. 34, N. 5,:437–438, 1998.

[29] Florin-Adrian Popescu and Nicolae Varachiu. Using an analog fuzzification circuit for real world applications. *Semiconductor Conference, 2000. CAS 2000 Proceedings.*, Vol. 1:pp. 281–284, October 2000.

[30] Yasuhiro Ota and B. M. Wilamowski. Current mode CMOS implementation of a fuzzy min-max network. *Department of Electrical Engineering University of Wyoming*, pages pp. 1–4, 1995.

[31] Y. Ota and B.M. Wilamowski. Cmos implementation of a voltage-mode fuzzy min-max controller. *Circuits, Systems and Computers*, Vol. 6(2):171–184, April 1996.

[32] C. Y. Huang, C. J. Wang, and B. D. Liu. Modular current mode multiple minimum circuit for fuzzy logic controllers. *Electronic Letters*, Vol. 32(12):pp. 1067–1069, June 1996.

[33] K. Kaewdang, C. Fongsamut, and W.; Surakampontorn. A wide-band current-mode ota-based analog multiplier-divider. *Circuits and Systems, 2003. ISCAS '03. Proceedings of the 2003 International Symposium on ,*, Vol. : 1:349 –352, May 25-28 2003.

[34] J. Choi, B. J. Sheu, and J. C.-F Chang. A gaussian synapse circuit for analog vlsi neural networks. *IEEE Trans. on VLSI Systems*, Vol. 2, No. 1,:129–133, March 1994.

[35] T. Kettner, H. Cshumacher, and K. Goser. Realization of a monolitic analog fuzzy logic controller. *Proceeding of 20th European Solid State Conference*, September 1993.

[36] C.C. Lee. Fuzzy logic in control system. *IEEE transaction Syst., man, Cybern*, Vol. 20(No. 2), 1990.

[37] A. K. Taha, M. M. El-Khatib, and A. S. Badawi. Design of a fuzzy logic programmable membership function circuit. *Seventeenth National Radio Science Conference*, pages pp. C20–1 to C20–6, February 2000.

[38] T. Yamakawa, T. Miki, and F. Ueno. The design and fabrication of the current mode fuzzy logic semi-custom ic in the standard cmos ic technology. *Proc. 15th IEEE Int. Symp. Multiple-Value Logic*, pages 76–82, May 1985.

[39] Y. Ota and B.M. Wilamowski. Vlsi implementation of a programmable current-mode neural network. *Proc. Intelligent Engineering System Trough Artificial Neural Networks, ANNIE'94 conf.*, Vol. 4:71–76, November 1994.

[40] M. Sasaki, T. Inoue, Y. Shirai, and F. Ueno. Fuzzy multiple-input maximum and minimum circuits in current mode and their analyses using bounded-difference equations. *IEEE Transaction Computer.*, Vol. 39(6):pp. 768–778, June 1990.

[41] G. N. Patel and S. P. DeWeerth. An analog vlsi loser-take-all circuit. *Proc. IEEE Int. Symp. Circuits and Systems*, pages 850–853, April 1995.

[42] The Mathworks. Tank level control. Technical report, http://www.mathworks.com/access/helpdesk/help/toolbox/fuzzy/fuzzyt25.html.