UNIVERSITY OF ÇUKUROVA
INSTITUTE OF NATURAL AND APPLIED SCIENCE

MSc. THESIS

Abdullah PADAK

DEVELOPING A SOFTWARE TO DETERMINE THE MICROCONTROLLER SPECIFICATIONS FOR FUZZY LOGIC CONTROL APPLICATIONS

DEPARTMENT OF ELECTRICAL AND ELECTRONICS ENGINEERING

ADANA, 2006

ÇUKUROVA ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

DEVELOPING A SOFTWARE TO DETERMINE THE MICROCONTROLLER SPECIFICATIONS FOR FUZZY LOGIC CONTROL APPLICATIONS

Abdullah PADAK

YÜKSEK LİSANS TEZİ

ELEKTRİK ELEKTRONİK MÜHENDİSLİĞİ ANABİLİM DALI

Bu tez 25 / 12 / 2006 Tarihinde Aşağıdaki Jüri Üyeleri Tarafından Oybirliği İle Kabul Edilmiştir.

İmza.....	İmza.....	İmza.....
Yrd.Doç.Dr. Murat AKSOY	Doç.Dr. Turgut İKİZ	Yrd.Doç.Dr Mehmet KARAKILÇIK
DANIŞMAN	ÜYE	ÜYE

Bu tez Enstitümüz Elektrik Elektronik Mühendisliği Anabilim Dalında hazırlanmıştır.

Kod No:

Prof. Dr. Aziz ERTUNÇ
Enstitü Müdürü

ÖZ

YÜKSEK LİSANS TEZİ

BULANIK MANTIK KONTROL UYGULAMALARINDA KULLANILACAK MİKRODENETLEYİCİNİN ÖZELLİKLERİNİN BELİRLENMESİ İÇİN BİR YAZILIM GELİŞTİRİLMESİ

Abdullah PADAK

ELEKTRİK ELEKTRONİK MÜHENDİSLİĞİ ANABİLİM DALI

FEN BİLİMLERİ ENSTİTÜSÜ

ÇUKUROVA ÜNİVERSİTESİ

Danışman: **Yrd. Doç. Dr. Murat AKSOY**
Yıl: **Aralık 2006 Sayfa: 73**
Jüri: **Yrd. Doç. Dr. Murat AKSOY**
Doç. Dr. Turgut İKİZ
Yrd. Doç. Dr. Mehmet KARAKILÇIK

Bulanık Mantık Kontrol Sistemleri (BMKS) kullanımı son yıllarda iyice yaygınlaşmış ve uygulama alanı da genişlemiştir. BMKS tasarım çalışmalarında kullanılmak üzere birçok yazılım geliştirilmiştir. Bu yazılımlardan bazıları önceki çalışmalar kısmında verilmektedir.

Bu çalışmada, BMKS uygulamalarında kullanılacak en uygun özelliklerdeki mikrodenetleyicinin seçilmesini sağlayan, Windows tabanlı, kullanımı kolay bir yazılım geliştirilmiştir. Bu yazılım ile en uygun mikrodenetleyici seçilebilmektedir. Performans ve fiyat açısından da en uygun çözümler elde edilebilmektedir. Ayrıca farklı alanlarda örnek uygulamalar incelenerek mikrodenetleyici seçim kriterleri tanımlanmış ve formüle edilmiştir. Buna bağlı olarak geliştirilen yazılımın altyapısını teşkil eden bir algoritma oluşturulmuştur. "Microsoft Visual Basic" programı ile geliştirilen yazılım, kullanıcının tanımlayacağı sistem parametrelerini kullanarak, gerçekleştirilecek uygulama için yeterli olacak minimum performans ve kapasite değerlerini hesaplayarak en uygun mikrodenetleyici özelliklerini görüntüleyecektir.

Anahtar Kelimeler: Bulanık Mantık, Bulanık Mantık Kontrol Sistemi, Mikrodenetleyici, Mikroişlemci, Microsoft Visual Basic.

ABSTRACT

MSc. THESIS

DEVELOPING A SOFTWARE TO DETERMINE THE MICROCONTROLLER SPECIFICATIONS FOR FUZZY LOGIC CONTROL APPLICATIONS

Abdullah PADAK

DEPARTMENT OF ELECTRICAL AND ELECTRONICS ENGINEERING
INSTITUTE OF NATURAL AND APPLIED SCIENCES
UNIVERSITY OF ÇUKUROVA

Supervisor: Assist. Prof. Dr. Murat AKSOY
Year: December 2006, Pages: 73
Jury: Assist. Prof. Dr. Murat AKSOY
Assoc. Prof. Dr. Turgut İKİZ
Assist. Prof. Dr. Mehmet KARAKILÇIK

In recent years, the Fuzzy Logic Control Systems (FLCSs) have become widespread and their application areas have also extended. There are numerous softwares developed to use in design of Fuzzy Logic Control Systems. Some of them are given in previous works section.

In this study, a Windows operating system based, user-friendly software to determine the optimum microcontroller specifications for Fuzzy Logic Control applications is developed. With this software the optimum microcontroller might be chosen. Regarding to performance and cost effectiveness, the best solutions might also be provided. Besides, the microcontroller selection criteria is defined and formulated after studying various applications in different fields. According to this, an algorithm, which forms the fundamentals of the software, is composed. The software, which is developed by "Microsoft Visual Basic" visual programming language, computes the minimum performance and capacity values for the intended application by using the parameters, which are defined by the user and display the optimum microcontroller specifications.

Keywords: Fuzzy Systems, Fuzzy Logic Controllers, Microcontroller for Fuzzy Logic Control, Software for Fuzzy Logic Control Systems, Microsoft Visual Basic.

ACKNOWLEDGEMENTS

I would like to express my heartfelt thanks to my supervisor Assist. Prof. Dr. Murat AKSOY for his great support and patience throughout my studies. His valuable support endured without decreasing. He always advised me for both scientific and life issues. He has been, and still he is, a great guider in my life. Without his guidance this thesis would have never been completed.

I would like to thank Zehan Kesilmiş for his supports and friendship from beginning to the end.

I would like to express my appreciation to Prof. Dr. Süleyman GÜNGÖR, Head of the Department of Electrical and Electronics Engineering, providing materials and study environment.

My parents, they deserve special thanks. They always provided endless support. Even though this thesis lasted longer then expected, they have always been beside me.

Finally, I would like to express my special thanks to my wife and my little daughter. They've never complained for sharing my time, which should have been assigned to them. Their moral support was invaluable for me and provided me determination to complete this thesis.

CONTENTS **PAGE**

LIST OF TABLES

LIST OF FIGURES **PAGE**

LIST OF SYMBOLS

B_i^j	Input linguistic value
D	Output linguistic values
D_m	Membership value of y with relation to the linguistic value
E^i	Weightings to the weighted average method of defuzzification
E_m	Crisp weighting for the linguistic value
$f(y)$	Crisp output value
F_c	Clock frequency
I_s	Inference speed
k	Total number of output linguistic values defined in the universe of discourse
KB	Number of knowledge base
M	Memory demand
μ	Membership function
N_i	Number of inputs
N_o	Number of outputs
N_p	Number of parallel processor
N_r	Number of rules
R_k	kth rule of the fuzzy system
R_s	Resolution
S	Union of all the linguistic values
S_i	Output linguistic value with clipped membership function
X	Universe of discourse
x	Members of the universe of discourse

LIST OF ABBREVIATIONS

FL	Fuzzy Logic
FLC	Fuzzy Logic Control
FLCS	Fuzzy Logic Control System
µ	Micro
µC	Micro Controller
µP	Micro Processor
OS	Operating System
DOS	Disc Operating System
PLC	Programmable Logic Controllers
GUI	Graphical User Interface
IEEE	Institute of Electrical and Electronics Engineers
et al.	And Others
e.g.	For Example [exempli gratia (Lat.)]
BM	Bulanık Mantık
BMK	Bulanık Mantık Kontrol
BMKS	Bulanık Mantık Kontrol Sistemi
MF	Membership Function
MSc	Master of Science
etc.	And so on, and so forth [et cetera (Lat.)]
KB	Knowledge Base
KBM	Knowledge Base Memory
OMF	Output Membership Function
FLIPS	Fuzzy Logical Inferences per Second
IMFs	Input Membership Functions
OMFs	Output Membership Functions
SRs	Sets of Rules
FAM	Fuzzy Associative Memory
COG	Center of Gravity
FRBS	Fuzzy Rule-Based System

1. INTRODUCTION

Fuzzy Logic Control (FLC) is an up-to-date control method which is mostly utilized in nonlinear applications and for the systems which are not easy to define and model mathematically and suitable to describe linguistically. For a long time it has been preferred because it has severe advantages in some systems. Each input/output variable can be represented linguistically. And all the operations between these variables can be done linguistically. This provides the designers flexibility; everything that can be expressed linguistically can be used in FLC.

Fuzzy Logic is a departure from classical two-valued sets and logic, that uses "soft" linguistic (e.g. large, hot, tall) system variables and a continuous range of truth values in the interval [0,1], rather than strict binary (True or False) decisions and assignments.

Formally, fuzzy logic is a structured, model-free estimator that approximates a function through linguistic input/output associations.

"Fuzzy Logic Control (FLC)" gained an increasing popularity specifically for especially various control applications in recent years. It even started to be used for the control applications of home electronics. The main reason for this is the increasing importance of energy class evaluation for all kinds of electronic goods both industrial and home-use applications. Regarding to increasing complexity and required better control quality required better control characteristics and energy saving, thrifty control systems. And even for some applications especially for the ones which are difficult to model mathematically and for nonlinear systems the FLC is the remedy for implementing the easiest solution.

For some brief general information about FLC several quotation from various articles and books has been done. This will be merely an overview of FLC. The ideas picked from these articles and books will give you a horizon about FL.

FL is a new theory extending our capabilities in modeling uncertainty (Larsen, 1997). Uncertainty is one of the application areas of FL because uncertainty can best be modeled by using linguistic variables. Some adaptive control techniques can be

productive for this subject. The goal of the adaptive controller is to provide stable control of systems with significant uncertainty (Spooner et al., 2002).

Fuzzy set theory provides a major newer paradigm in modeling and reasoning with uncertainty. Though there were several forerunners in science and philosophy, in particular in the areas of multi-valued logics and vague concepts, Lotfi A. Zadeh, a professor at University of California at Berkeley was the first to propose a theory of fuzzy sets and an associated logic, namely fuzzy logic (Larsen, 1997).

Describing the parameters for the system to be analyzed is necessary to consider the system with its all aspects. To clarify the performance and capacity criteria, parameters for the FLCS some proper application instances will be utilized. This will be very useful especially for new designers.

1.2. Application Areas of Fuzzy Logic

Let us see the application areas of FL all over the world and decide the how widespread it is being used. Today as FL is gaining popularity in certain application fields for many applications it is also getting popularity in a variety of machines and processes today. You can see some of the applications in use today are in Table 1.1 (Yan et al, 1994).

Table 1.1. Applications of Fuzzy Logic Controllers (FLCs) and functions performed.

Application	**FLC Function(s)**
Video camcorder	Determine best focusing and lighting when there is movement in the picture
Washing machine	Adjust washing cycle by judging the dirt, size of the load, and type of fabric
Television	Adjust brightness, color, and contrast of picture to please viewers
Motor control	Improve the accuracy and range of motion control under unexpected conditions
Subway train	Increase the stable drive and enhance the stop accuracy by evaluating the passenger traffic conditions. Provide a smooth start and smooth stop.
Vacuum cleaner	Adjust the vacuum cleaner motor power by judging the amount of dust and dirt and the floor characteristics
Hot water heater	Adjust the heating element power according to the temperature and the quantity of water being used
Helicopter control	Determine the best operation actions by judging human instructions and the flying conditions including wind speed and direction

Table 1.2. The historical development of Fuzzy Logic until 1993 (Kandel et al., 1998).

Year	**Event**
1961	Lotfi Zadeh claims in his paper for a new kind of "fuzzy" mathematics.
1965	Lotfi Zadeh introduces the concept of fuzzy sets.
1969	Marinos (Duke University) conducts the first research aiming at hardware implementation of Fuzzy Logic.
1972	M. Sugeno presents the idea of fuzzy measures.
1974	E. Mamdani presents a fuzzy application to control a steam machine in an academic framework.
1980	Yamakawa build the first fuzzy circuit with discrete bipolar components.
1982	The first industrial application in a cement kiln in Denmark
1984	Togai and Watanabe present the first VLSI implementation of Fuzzy Controllers.
1986	Hitachi puts into operation a fuzzy controlled subway system.
1987	Yamakawa presents the first analog Fuzzy Controller.
1988	Togai implements the first digital fuzzy processors.
1989	Laboratory for International Fuzzy Engineering Research starts in Japan.
1990	Yamakawa establishes the Fuzzy Logic Systems Institute (FLSI) in Japan.
1992	The first IEEE International Conference on Fuzzy Systems.
1993	The first issue of the IEEE Transactions on Fuzzy Systems.

Since the first reported application of Fuzzy Logic, the number of industrial and commercial developments, covering a wide range of technological domains, has grown incessantly. Nowadays, countless researchers from different areas are hardly working on the subject while contributing with smart and interesting solutions for engineering. Table 1.1, which summarizes the historical development of Fuzzy Logic, highlights some of its most significant milestones as reported by Kandel et al (Dualibe et al., 2003). After effectiveness in some certain areas of FLC was realized in 1980's R&D activations have increased rapidly since then. In the following table some of the R&D studies of various areas have been listed.

Table 1.3 Main R&D areas in Fuzzy Logic and their major topics (Dualibe et al., 2003)

R&D Area	**Main Topics**
Fuzzy Mathematics:	Foundations of Fuzzy Logic. approximate reasoning, evolutionary computation, identification and learning algorithms, rule base optimization.
Control Systems:	Fuzzy Control theory and applications, process and environmental control stability criterions issues, multi-level supervisory control.
Pattern Recognition and Image Processing:	Supervised and unsupervised learning, classifiers design and integration. Signal/image processing and analysis, computer vision, multimedia applications.
Soft Computing and Hybrid Systems:	Intelligent information systems, data base systems, data mining, intelligent agents, reliability engineering, Neuro-Fuzzy Systems, Internet computing, networks traffic modeling and control.
Electronic Systems:	Fuzzy hardware implementation and embedded applications.
Robotics and Automation:	Fuzzy Logic in robotics, industrial automation and other industrial applications.

At the beginning, the most popular applications of Fuzzy Logic were found in the domain of Control System. Nowadays, the application of this soft-computing technique has been extended to other fields such as Signal Processing, Image Processing and Switching Power Control, for instance. As real-time applications need ever faster, more autonomous and less power-consuming circuits the choice of on-chip controllers becomes an interesting option. The attractiveness of analog circuits for implementing Fuzzy hardware relies on its natural compatibility with most used Fuzzy algorithms and the needlessness of A/D and D/A converters for interfacing sensors and actuators (Dualibe et al., 2003). In table 1.4 you can see some different domains where FL is used.

Table 1.4. Some reported applications of Fuzzy Logic in different domains (Dualibe et al., 2003).

Application	**Application**
Process Control	Car Parking Modeling
5MW Nuclear Reactor Control	Automotive Engine Idle Control
Cement Kiln Control	**Image Processing**
Sake Brewing Manufacturing	Real-Time Motion Estimation
Activated Sludge Wastewater Plant	Image Color Correction Systems
Water Purification Plant	Facsimile Image Fuzzy Processor
Multilayer Waste Incinerator	Image Processing in IQTV
Environmental Emissions Control	Fuzzy Stack Filters for Image Processing
Aero-Space Applications	**Sign-al Processing**
Aircraft Carrier Landing System	Non-linear Channel Equalization
Attitude Control of a Flexible Satellite	AGC Control for Radio Communications
Flexible Wing Roll Control	Decision Feedback Equalizer
Pitch Control of an Interceptor Missile	Magnetic Recording Equalization
Robotics and Automation	Analog Filter Tuning
Mobile Robot Control	Adaptive Noise Cancellation
Robot Arm Control	Real-Time Source Separation
Container Crane Control	Fuzzy Digital PLL Filter
Industrial Automotive	Earthquake Precursors Detection
Car Modeling in Extreme Situations	**Power Electronics**
Automotive Automatic Transmission	Control of DC/DC Converters
Automotive Anti-Skid Braking	Control of PWM DC Converters
Autonomous Vehicle Controller	Embedded Low Cost Control for DC Motors
Automatic Train Operation	Fuzzy Control of induction AC Motors

1.3. Hardware for Fuzzy Logic Control Systems

Fuzzy developers can have difficulties to decide the hardware to use in their FLCS applications because of lack of experience for FLCS or the hardware to use for the system at issue. It is natural that it can take time for a fuzzy developer to gain capability to build up comprehensive solutions for such complex systems.

I hope this study will be capable of fulfilling the gap in this field. By means of the software Fuzzy designers will be capable of selecting the optimum μC or μP and other related hardware which can be designated as the "rest of the controller hardware" for also putting forward optimum, cost-effective solution. We can say unused capacity or unused performance is the money wasted. The objective for any designer should be the best performance for the best price.

Initially, brief information about basics of fuzzy logic, microcontrollers and fuzzy logic control applications has been given. Then the required parameters were defined by investigating a few sample applications of FLC.

Then, some useful FLC software tools was introduced and explained as previous works. Fuzzy developers may utilize these software tools during their works.

After giving general information about FLCS, the system was illustrated by block diagram. Then each block was explained individually and the interaction between them took shape as we went on.

Next stage should be a detailed explanation of the parameters which are effective for determining the specifications of the μC for a predefined FLCS. And then a FLCS application will be discussed and by defining all the parameters of this selected application the developed algorithm has been tested. The result was also discussed and if required, a calibration for the μC selection algorithm will be carried out. After tuning up the algorithm it will be retested and result will be discussed again.

Lastly, in result and conclusion section all the results was discussed and overall thesis has been summarized. Achievement of the thesis was discussed. Some recommendation for future works is given.

2. SOFTWARE TOOLS FOR FUZZY LOGIC

2.1. FuzNum Calc

The fuzzy number calculator (Figure 2.1) has lots in common with the crisp calculator you probably have nearby. It has two Setup keys, Setup A and Setup B, that let you enter two numbers from the keypad. The minus (-) key allows negative numbers. Use the operation keys to perform addition *(C=A+B)*, subtraction *(C=A-B)*, multiplication *(C=AxB)*, and division *(C=A/B)*. It also has a Clear Entry (CE) key. The numbers you enter, ranging from –9 to +9, appear on the calculator's screen. After you click the operation button, the screen displays the results on a scale from –100 to +100. The scale shifts automatically to display the numbers you enter and the results calculated. You can perform calculations on fuzzy numbers exclusively, crisp numbers exclusively, or combine fuzzy and crisp numbers in one operation (McNeill et al., 1994).

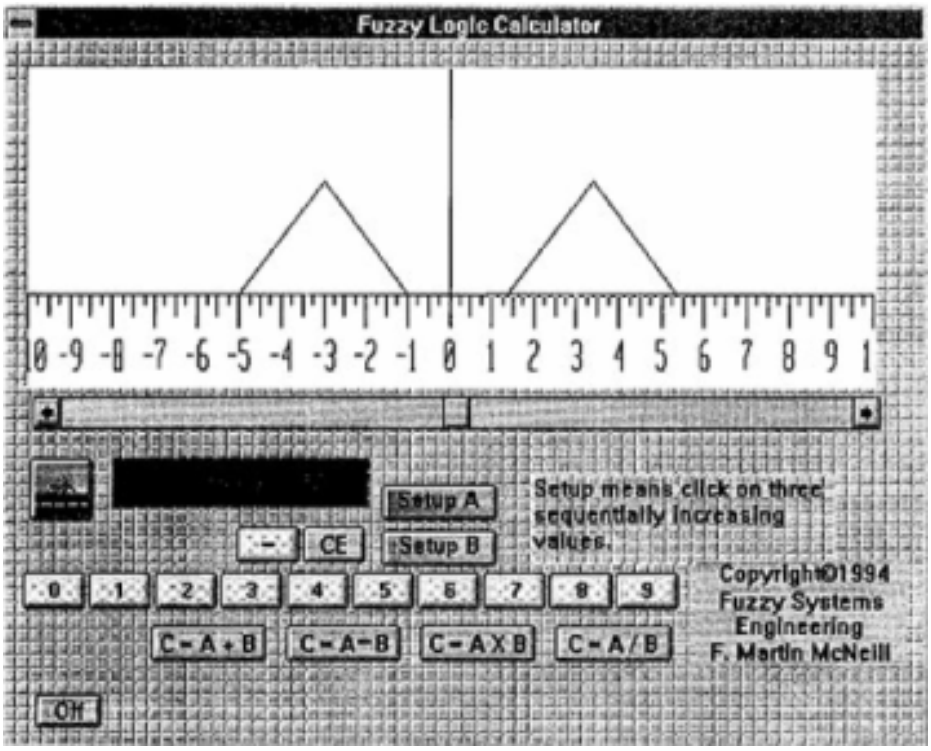

Figure 2.1. Opening screen of the FuzNum Calc.

2.2. MATLAB Fuzzy Logic Toolbox

One of the most useful software tool for Fuzzy Logic is the Fuzzy Logic Toolbox of Matlab. With this toolbox as explained below you can you can create, edit fuzzy inference systems with an easy graphical user interface (GUI).

2.2.1. What is the Fuzzy Logic Toolbox?

The Fuzzy Logic Toolbox is a collection of functions built on the MATLAB numeric computing environment. It provides tools for you to create and edit fuzzy inference systems within the framework of MATLAB, or if you prefer you can integrate your fuzzy systems into simulations with Simulink, or you can even build stand-alone C programs that call on fuzzy systems you build with MATLAB. This toolbox relies heavily on GUI tools to help you accomplish your work, although you can work entirely from the command line if you prefer.

The toolbox provides three categories of tools:

- Command line functions
- Graphical, interactive tools
- Simulink blocks and examples

The first category of tools is made up of functions that you can call from the command line or from your own applications. Many of these functions are MATLAB M-files, series of MATLAB statements that implement specialized fuzzy logic algorithms. You can view the MATLAB code for these functions using the statement.

2.2.2. What Can the Fuzzy Logic Toolbox Do?

The Fuzzy Logic Toolbox allows you to do several things, but the most important thing it lets you do is create and edit fuzzy inference systems. You can create these systems by hand, using graphical tools or command-line functions, or

you can generate them automatically using either clustering or adaptive neuro-fuzzy techniques.

If you have access to Simulink, the simulation tool that runs alongside MATLAB, you can easily test your fuzzy system in a block diagram simulation environment. If you have Real-Time Workshop capabilities available, you can generate realtime or non-realtime code from the Simulink environment.

The toolbox also lets you run your own stand-alone C programs directly, without the need for Simulink. This is made possible by a stand-alone Fuzzy Inference Engine that reads the fuzzy systems saved from a MATLAB session (the stand-alone code, unlike that generated by the Real-Time Workshop, does not run in real time). You can customize the stand-alone engine to build fuzzy inference into your own code. All provided code is ANSI compliant.

Because of the integrated nature of MATLAB's environment, you can create your own tools to customize the Fuzzy Logic Toolbox or harness it with another toolbox, such as the Control System, Neural Network, or Optimization Toolbox, to mention only a few of the possibilities.

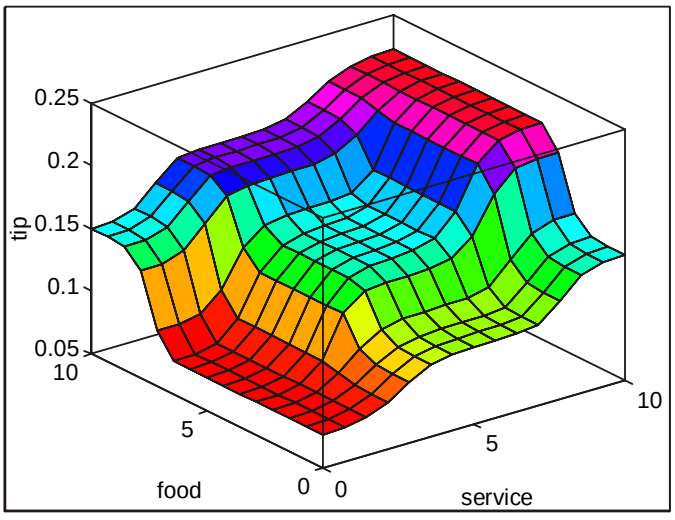

Figure 2.2. Fuzzy Control Surface.

The picture above was generated by the three if-then rules by using the graphical tools in the Fuzzy Logic Toolbox (Jang et al., 1997).

2.2.3. Constructing a FIS File in MATLAB

The rule structure of FIS makes it easy to incorporate human expertise with the target system directly into the modeling process. When the input/output data of a target system is available, conventional system identification techniques can be used. A process for constructing a FIS can be summarized as follows:

- Choose a specific type of FIS (Mamdani or Sugeno).
- Select relevant input-output variables.
- Determine the number of linguistic terms associated with each input-output variables (determine the membership function for each linguistic term).
- Design a collection of fuzzy if-then rules.
- Choose the defuzzification method to extract a crisp value that best represents a fuzzy set.
- Run through test suite to validate system, adjust details as required.

In this study, the fuzzy controllers for the single fuzzy controller scheme are characterized as follows:

- Seven fuzzy sets for each input.
- Seven fuzzy sets for the output.
- Triangular and trapezoidal membership functions for each input.
- Constant membership function for the output.
- Fuzzification using continuous universe of discourse.
- Implication using the "min" operator.
- Sugeno inference mechanism based on fuzzy implication.
- Defuzzification using the "wtaver" method.

2.3. FCM, Fuzzy Control Manager

This is a Windows program that can be used intuitively to define fuzzy systems of any complexity graphically. All relevant data can be displayed while developing, debugging, or optimizing the system. In constructing the system, there is no limit on the number of variables or rules to be defined. The membership functions can be

defined by an unlimited number of inflection points (or Singletons for outputs, if desired). Codes for implementing the constructed fuzzy controller can be generated in C-language, assembler, or binary, depending on the FCM version used. FCM versions include (Web1):

- FCM-3000 which operates on an FP-3000 adapter board and generates
- Special C code which performs an appropriate access to the FP-3000 processor.
- FCM-SOFT which supports the online operation of a VME bus board containing an FP-3000 processor.
- FCM-TEAM which supports a microcontroller board with many analog and digital I/O lines.
- FCM-SPS which is used for direct programming of a fuzzy module of a programmable controller containing an FP-3000.
- The FCM-Modicon which supports fuzzy programming of an AEG-Modicon FCM-I3HD-CINT which generates codes for the NEC 78K0,78K3, V series microcontrollers.
- FCM-I3HD-75X which generates code for microcontrollers of the NEC 75X family.
- FCM-I3HD-17K which generates code for microcontrollers of the NEC 17K family.
- FCM-PX-1500 which generates code for the Panasonic MN1500 microcontroller family.
- FCM-HC11 which generates code for 68HC11 microcontrollers.
- FCM-8051 which generates code for microcontrollers of the 8051 family.

2.4. FuzzGen

This is a freeware program, and it is probably the earliest fuzzy logic code generator. It allows the user to put together full fuzzy decision-making algorithms graphically, then allows the user to generate code in Pascal, Basic, or C/ C++ that

will implement the decision process. The software could be used intuitively, but it also has a help file. Figures 2.3 to 2.4 give its capabilities and ease of use (Web2).

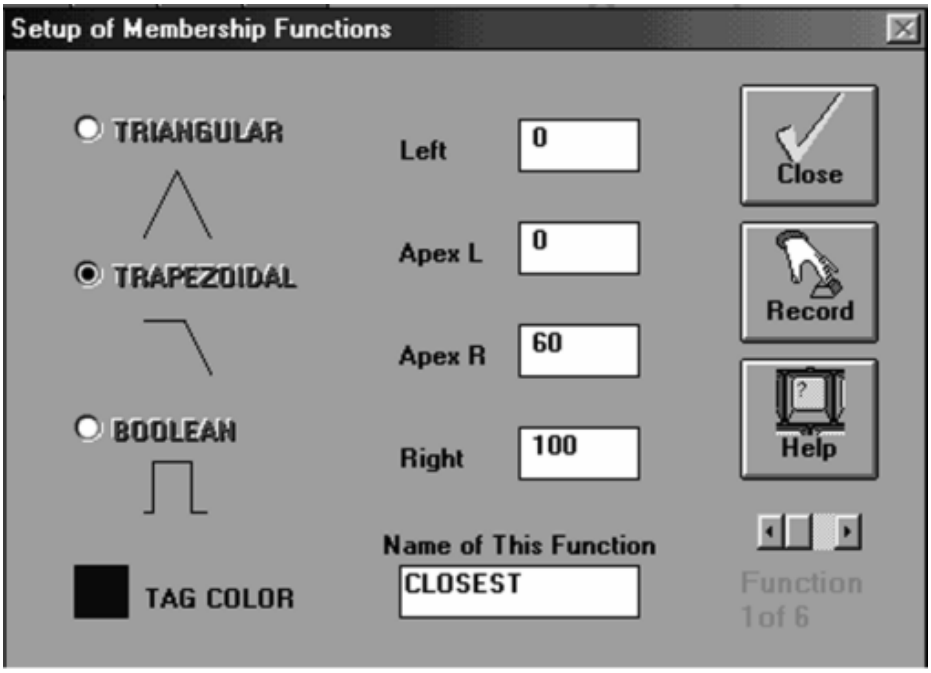

Figure 2.3. Membership selection in FuzzGen.

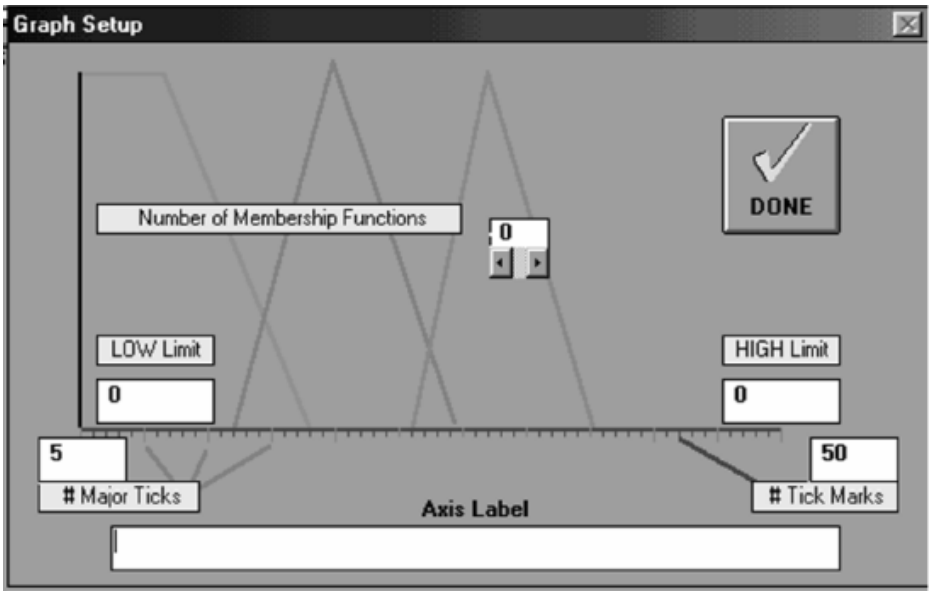

Figure 2.4. Graphical definition of membership functions in FuzzGen.

2.5. fuzzyTECH

fuzzyTECH is a leading family of software development tools for fuzzy logic and neural-fuzzy systems. The software supports both English and German languages. fuzzyTECH families of particular interest here include:

- fuzzyTECH Editions for General Target Hardware including fuzzyTECH Professional and fuzzyTECH online editions.
- fuzzyTECH MCU Editions for Embedded Control including:
- fuzzyTECH MCU-HC05/08 Edition which supports all Motorola 68HC05xx and 68HC08xx families of microcontrollers.
- fuzzyTECH MCU-ST6 Edition which supports all STMicroelectronics ST6 family microcontrollers.
- fuzzyTECH MCU-MP Edition which supports all Microchip's microcontrollers.
- fuzzyTECH MCU-51 Edition which supports all 8051 and 80251 microcontrollers.
- fuzzyTECH MCU-374xx Edition which supports all 8-bit Mitsubishi microcontrollers.
- fuzzyTECH MCU-HC11/12 Edition which supports all Motorola 68HC11xx and 68HC12xx families of microcontrollers.
- fuzzyTECH MCU-96 Edition which supports all Intel MCS-96 microcontrollers.
- fuzzyTECH MCU-320 Edition which supports Texas Instruments digital Signal Processors.
- fuzzyTECH IA Editions for Industrial Automation including fuzzyTECH (Web3).

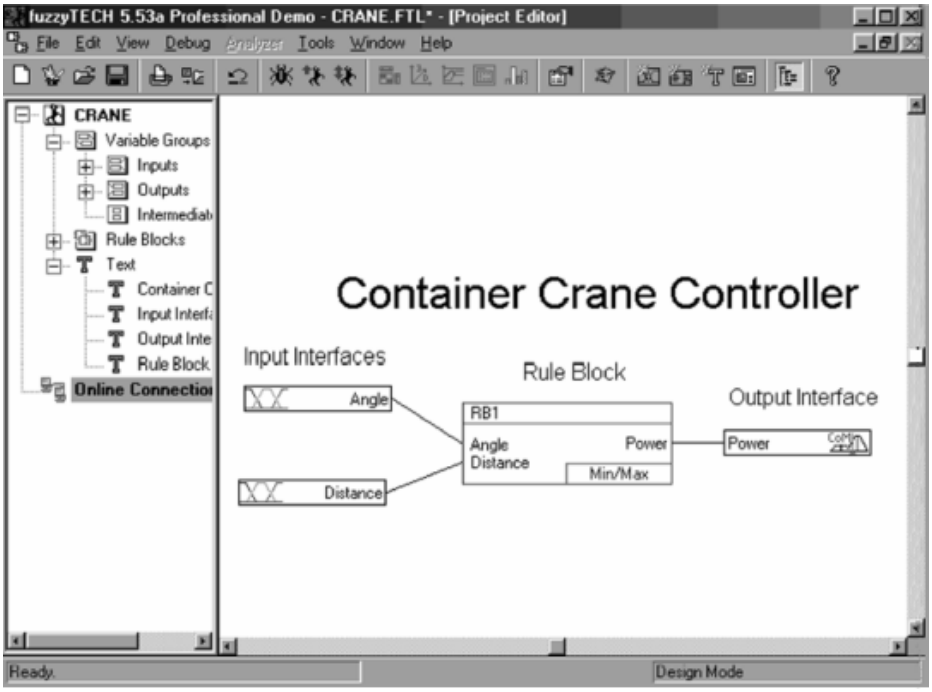

Figure 2.5. Opening one of the fuzzyTECH example files.

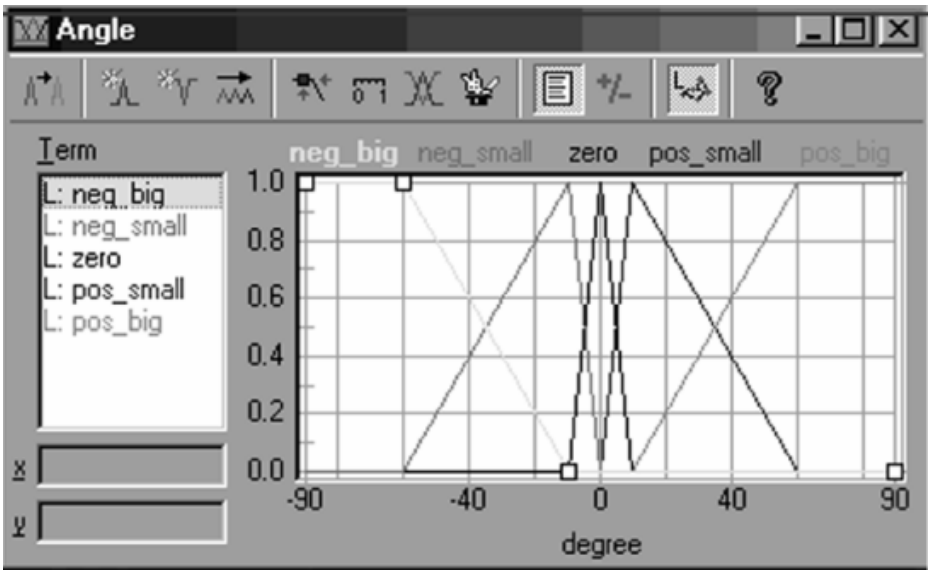

Figure 2.6.a. Defining inputs for the Container Crane Controller Example (Angle).

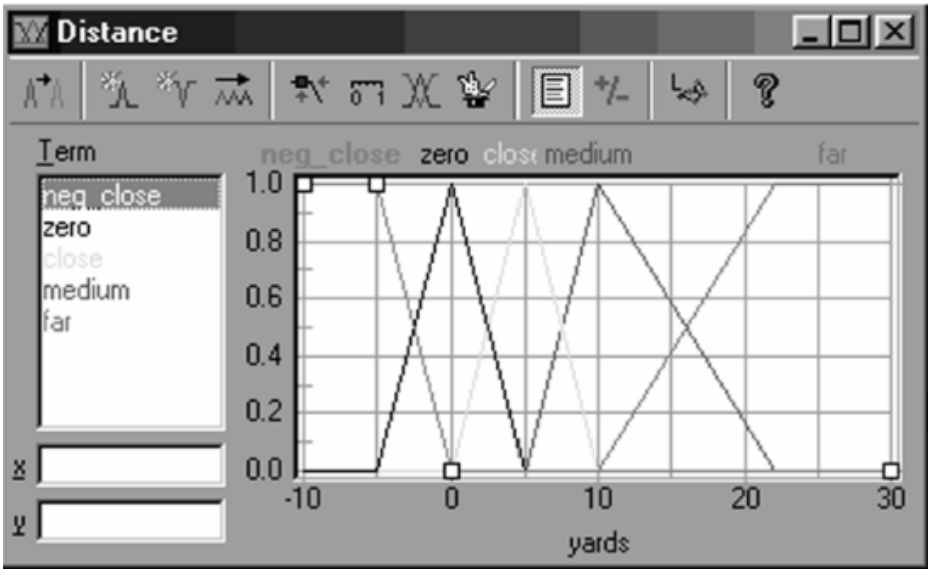

Figure 2.6.b. Defining inputs for the Container Crane Controller Example (Distance).

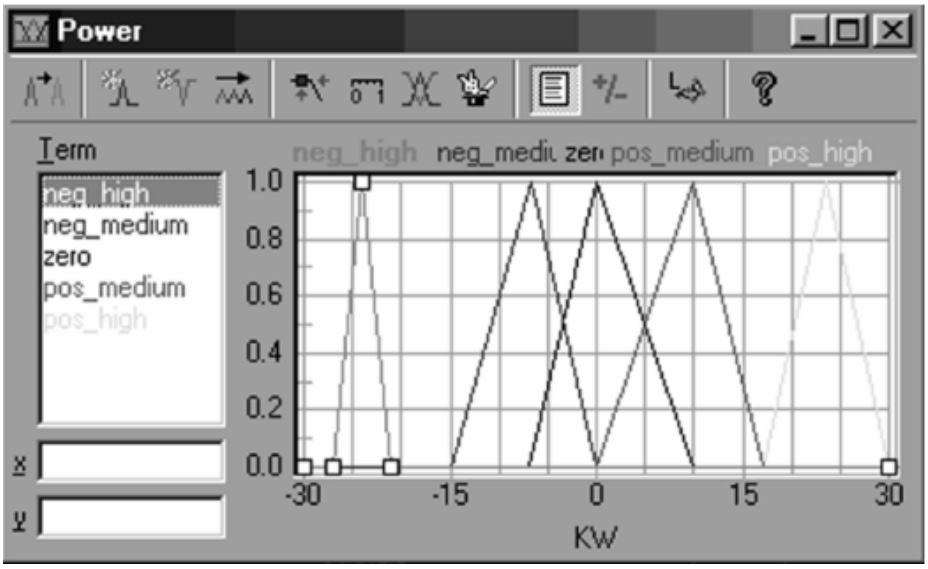

Figure 2.7. Defining the output of the Container Crane Controller example.

#	IF		THEN	
	Angle	Distance	DoS	Power
1	pos_small	zero	1.00	neg_medium
2	zero	zero	1.00	zero
3	pos_small	close	1.00	neg_medium
4	zero	close	1.00	zero
5	neg_small	close	1.00	pos_medium
6	neg_small	medium	1.00	pos_high
7	neg_big	medium	1.00	pos_medium
8	zero	far	1.00	pos_medium
9	neg_small	far	1.00	pos_high
10				
11				
12				
13				

Figure 2.8. fuzzyTECH Fuzzy Rules Editor.

3. FUNDAMENTALS OF FUZZY LOGIC CONTROL

3.1. Overview

Before we start studying the design parameters of Fuzzy Systems and their relation with μC specifications, we would better study all the concepts and related other subjects regarding FL with its all aspects. To understand the subject which we are going to study on, we need to have a background which will form the fundamentals. We must be qualified to give answers to some questions such as: What is “Fuzzy Logic” and “fuzzy control system”, what is a “fuzzy set” and “fuzzy subset”, “operations of fuzzy logic”, what is “membership function”, what is the meaning of “rule base” for a FLCS, and what are their functions, What are “advantages and disadvantages of “Fuzzy Logic Control (FLC)”, information about “Fuzzy if-then rules”, Fuzzy numbers and Fuzzy arithmetic, Linguistic variables, Probability theory and FL… etc. When we go on to the next section this basic information will assist “Fuzzy Logic Control Systems (FLCS)” and their parameters to be comprehended easily.

3.2. Fundamentals of FL

As explained above, fundamentals are indispensable to understand rest of the thesis. So we will go on by explaining the concepts that you can meet while working on FL. This will be useful to comprehend this thesis easily. This will be very useful for engineers without FL background.

3.2.1. Fuzzy Logic

Fuzzy logic is a superset of conventional (Boolean) logic that has been extended to handle the concept of partial truth-truth values between "completely true" and "completely false". It was introduced by Dr. Lotfi Zadeh of UC/Berkeley in the 1960's as a means to model the uncertainty of natural language (Web1). The truth

of a logical expression in fuzzy logic is a number in the interval [0,1]. Fuzzy logic has emerged as a profitable tool for the control of complex industrial processes and systems. It is used for processes that have no simple mathematical model, for highly non-linear processes, or where the processing of linguistically formulated knowledge is to be performed. The controllers based on this mathematical approach are known as fuzzy controllers (Cirstea et al., 2002).

Fuzzy logic has the advantage of modeling complex, nonlinear problems linguistically rather than mathematically and using natural language processing (computing with words). The use of fuzzy logic requires, however, the knowledge of a human expert to create an algorithm that mimics his/her expertise and thinking. Also, studying the stability of a fuzzy system is a demanding task (Ibrahim, 2004).

3.2.2. Fuzzy Control Systems

Fuzzy Logic has been applied to problems that are either difficult to face mathematically or applications where the use of Fuzzy Logic provides improved performance and/or simpler implementations. One of its main advantages lies in the fact that it offers methods to control non-linear plants, known difficult to model (Dualibe et al., 2003).

The use of fuzzy logic is relatively simple as compared to conventional system even for the engineers who don't have a control theory background. It was developed by Lotfi Zadeh. Fuzzy Logic tries to emulate some of the properties that human beings use in their control and decision processes (Jackson, 1994).

The most widespread use of fuzzy logic today is in fuzzy control applications. You can use fuzzy logic to make your air conditioner cool your room. Or you can design a subway system to use fuzzy logic to control the braking system for smooth and accurate stops. A control system is a closed-loop system that typically controls a machine to achieve a particular desired response, given a number of environmental inputs. A fuzzy control system is a closed-loop system that uses the process of fuzzification to generate fuzzy inputs to an inference engine, which is a knowledge base of actions to take. The inverse process, called defuzzification, is also used in a

fuzzy control system to create crisp, real values to apply to the machine or process under control. Today fuzzy controllers have been used to control many machines, including washing machines and camcorders (Rao, 1995).

In a Fuzzy Controller, human experience is codified by means of linguistic if-then rules that build up a so-called Fuzzy Inference System, which computes control actions upon given conditions.

Fuzzy process control was first successfully achieved by Mamdani (1976) with a fuzzy system for controlling a cement plant. Since then, fuzzy control has been widely accepted, first in Japan and then throughout the world. A basic simple fuzzy control system is simply characterized. It accepts numbers as input, then translates the input numbers into linguistic terms such as Slow, Medium, and Fast (fuzzification). Rules then map the input linguistic terms onto similar linguistic terms describing the output. Finally, the output linguistic terms are translated into an output number (defuzzification). The syntax of the rules is convenient for control purposes, but much too restrictive for fuzzy reasoning; defuzzification and defuzzification are automatic and inescapable. There are several development environments available for constructing fuzzy control systems. A typical fuzzy control rule might be:

If input1 is High and input2 is Low **then** output is Zero.

Rules for fuzzy reasoning cannot be described so compactly. The application domain of fuzzy control systems is well defined; they work very satisfactorily with input and output restricted to numbers. But the domain of fuzzy reasoning systems is not well defined; by definition, fuzzy reasoning systems attempt to emulate human thought, with no a priori restrictions on that thought. Fuzzy control systems deal with numbers; fuzzy reasoning systems can deal with both numeric and non-numeric data. Inputs might be temperature and pulse, where temperature might 38.58C and pulse might be 110 and “thready”, where “thready” is clearly non-numeric. Output might be “CBC” and “Admit” and “transfer MICU” (not very realistic, but illustrates non-numeric data input and output). Accordingly, rules for fuzzy reasoning do not make fuzzification and defuzzification automatic and inescapable; they may be broken out

as separate operations that may or may not be performed as the problem requires (Siler et al., 2005).

As mentioned before, Fuzzy Logic Controllers has some advantages and disadvantages too. Below you can see the advantages and disadvantages to FLCs for control systems as compared to more traditional control systems.

3.2.2.1. Advantages of Fuzzy Logic Controllers

- Relates input to output in linguistic terms, easily understood
- Allows for rapid prototyping because the system designer doesn't need to know everything about the system before starting
- Cheaper because they are easier to design
- Increased robustness
- Simplify knowledge acquisition and representation
- A few rules encompass great complexity
- Can achieve less overshoot and oscillation
- Can achieve steady state in a shorter time interval (Rao, 1995)

3.2.2.2. Disadvantages of Fuzzy Logic Controllers

- Hard to develop a model from a fuzzy system
- Require more fine tuning and simulation before operational
- Have a stigma associated with the word fuzzy (at least in the Western world); engineers and most other people are used to crispness and shy away from fuzzy control and fuzzy decision making
- Hard to develop a model from a fuzzy system
- Require more fine tuning and simulation before operational
- Have a stigma associated with the word fuzzy (at least in the Western world); engineers and most other people are used to crispness and shy away from fuzzy control and fuzzy decision making (Rao, 1995).

3.2.3. Fuzzy Subsets

Just as there is a strong relationship between Boolean logic and the concept of a subset, there is a similar strong relationship between fuzzy logic and fuzzy subset theory.

In classical set theory, a subset U of a set S can be defined as a mapping from the elements of S to the elements of the set

$$\{0, 1\}, U: S \rightarrow \{0, 1\} \tag{3.1}$$

This mapping may be represented as a set of ordered pairs, with exactly one ordered pair present for each element of *S*. The first element of the ordered pair is an element of the set *S*, and the second element is an element of the set {0, 1}. The value zero is used to represent non-membership, and the value one is used to represent membership. The truth or falsity of the statement $x \text{ is in } U$ is determined by finding the ordered pair whose first element is *x*. The statement is true if the second element of the ordered pair is 1, and the statement is false if it is 0.

Similarly, a fuzzy subset *F* of a set *S* can be defined as a set of ordered pairs, each with the first element from *S*, and the second element from the interval [0,1], with exactly one ordered pair present for each element of *S*. This defines a mapping between elements of the set *S* and values in the interval [0,1]. The value zero is used to represent complete non-membership, the value one is used to represent complete membership, and values in between are used to represent intermediate degrees of membership. The set S is referred to as the universe of discourse for the fuzzy subset F. Frequently, the mapping is described as a function, the membership function of *F*. The degree to which the statement x is in F is true is determined by finding the ordered pair whose first element is *x*. The degree of truth of the statement is the second element of the ordered pair.

In practice, the terms "membership function" and fuzzy subset get used interchangeably.

That's a lot of mathematical baggage, so here is an example. Let's talk about people and "tallness". In this case the set S (the universe of discourse) is the set of people. Let's define a fuzzy subset “tall”, which will answer the question "to what degree is person x tall?" Zadeh describes “tall” as a linguistic variable, which represents our cognitive category of "tallness". To each person in the universe of discourse, we have to assign a degree of membership in the fuzzy subset “tall”. The easiest way to do this is with a membership function based on the person's height (Web1).

$$\begin{cases} \text{tall}(x) = 0, & \text{if height}(x) < 1.5\text{m.}, \\ (\text{height}(x) - 1.5\text{m.}) / 0.75\text{m.} & \text{if } 1.5\text{m.} \le \text{height}(x) \le 2.0\text{m.}, \\ 1, & \text{if height}(x) > 2.0\text{m.} \end{cases} \tag{3.2}$$

A graph of this looks like:

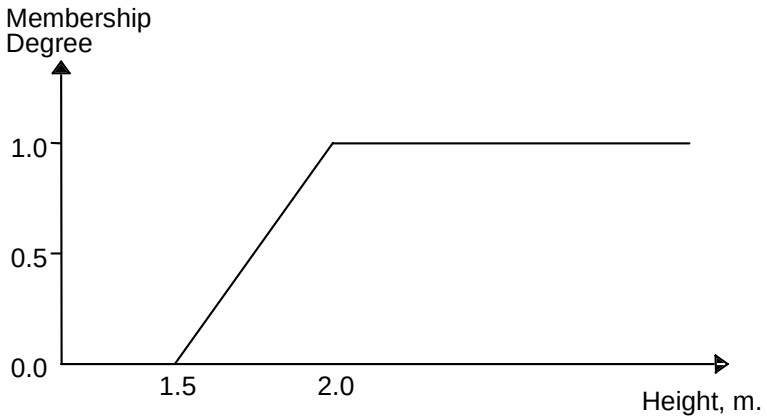

Figure 3.1. Membership function of height.

Given this definition, here are some example values:

Table 3.1. Degree of membership function for height (Web1).

Person	**Height**	**Degree of tallness**
Billy	0.9652	0.00 (I think)
Yoke	1.6510	0.21
Drew	1.7526	0.38
Erik	1.7780	0.42
Mark	1.8542	0.42
Kareem	2.1844	1.00 (depends on who you ask)

Expressions like "*A* is *X*" can be interpreted as degrees of truth, e.g., "Drew is tall" = 0.38.

Membership functions used in most applications almost never have as simple a shape as tall(*x*). At minimum, they tend to be triangles pointing up, and they can be much more complex than that. Also, the discussion characterizes membership functions as if they always are based on a single criterion, but this isn't always the case, although it is quite common. One could, for example, want to have the membership function for “tall” depend on both a person's height and their age (he's tall for his age). This is perfectly legitimate, and occasionally used in practice. It's referred to as a two-dimensional membership function, or a "fuzzy relation". It's also possible to have even more criteria, or to have the membership function depend on elements from two completely different universes of discourse (Web4).

3.2.4. Logic Operations

Now that we know what a statement like "X is LOW" means in fuzzy logic, how do we interpret a statement like x is low and y is high or (not z is medium).

The standard definitions in fuzzy logic are:

$$\text{truth}(\textit{not } x) = 1.0 - \text{truth}(x) \tag{3.3}$$

$$\text{truth}(x \textit{ and } y) = \text{minimum}(\text{truth}(x), \text{truth}(y)) \tag{3.4}$$

$$\text{truth}(x \textit{ or } y) = \text{maximum}(\text{truth}(x), \text{truth}(y)) \tag{3.5}$$

Some researchers in fuzzy logic have explored the use of other interpretations of the "*and*" and "*or*" operations, but the definition for the "*not*" operation seems to be safe.

Note that if you plug just the values zero and one into these definitions, you get the same truth tables as you would expect from conventional Boolean logic. This is known as the "extension principle", which states that the classical results of Boolean logic are recovered from fuzzy logic operations when all fuzzy membership grades are restricted to the traditional set {0, 1}. This effectively establishes fuzzy subsets and logic as a true generalization of classical set theory and logic. In fact, by this reasoning all crisp (traditional) subsets "are" fuzzy subsets of this very special type; and there is no conflict between fuzzy and crisp methods (Web1).

Some examples; assume the same definition of "tall" as above, and in addition, assume that we have a fuzzy subset OLD defined by the membership function:

$$\begin{cases} \text{old}(x) = 0, & \text{if age}(x) < 18 \text{ yr.} \\ (\text{age}(x) - 18 \text{ yr.})/42 \text{ yr.}, & \text{if } 18 \text{ yr.} \le \text{age}(x) \le 60 \text{ yr.} \\ 1, & \text{if age}(x) > 60 \text{ yr.} \end{cases} \tag{3.6}$$

And for compactness, let

$a = x$ is "tall" and x is "old" (3.7)

$b = x$ is "tall" or x is "old" (3.8)

c = not (x is "tall") (3.9)

Then we can compute the following values:

Table 3.2. Computed degree of membership for height and age (Web1).

height	**age**	***x* is "tall"**	***x* is "old"**	***a***	***b***	***c***
100	65	0.00	1.00	0.00	1.00	1.00
165	30	0.21	0.29	0.21	0.29	0.79
175	27	0.38	0.21	0.21	0.38	0.62
178	32	0.42	0.33	0.33	0.42	0.58
185	31	0.54	0.31	0.31	0.54	0.46
218	45	1.00	0.64	0.64	1.00	0.00
100	4	0.00	0.00	0.00	0.00	1.00

3.2.5. The Fuzzy Inference Process

The general inference process proceeds in three basic steps. This subject will be discussed in the next chapter with all details, but in this chapter it is considered suitable to give general information about the inference process steps.

1. Fuzzification is simply the process of obtaining values for the inputs x_i and computing $\mu(x_i)$ for each of the input membership functions μ (Spooner, 2002).

2. Fuzzy inference engines treat inputs in a form of linguistic variables and gives a response as well as a human-like decision making. The inference procedure derives effective control actions using the fuzzy rules.

3. Under "composition", all of the fuzzy subsets assigned to each output variable are combined together to form a single fuzzy subset for each output variable.

4. Finally is the (optional) "defuzzification", which is used when it is useful to convert the fuzzy output set to a crisp number. There are more defuzzification methods than you can shake a stick at (at least 30). 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 variable value of the center of gravity of the membership function for the fuzzy value. 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 (Web4).

3.2.6. Fuzzy Numbers and Fuzzy Arithmetic

Fuzzy numbers are fuzzy subsets of the real line. They have a peak or plateau with membership grade 1, over which the members of the universe are completely in the set. The membership function is increasing towards the peak and decreasing away from it.

Fuzzy numbers are used very widely in fuzzy control applications. A typical case is the triangular fuzzy number which is one form of the fuzzy number 7. Slope and trapezoidal functions are also used, as are exponential curves similar to Gaussian probability densities.

Nevertheless, fuzzy logic is a mathematical formalism, and a membership grade is a precise number. What's crucial to realize is that fuzzy logic is a logic of fuzziness, not a logic which is itself fuzzy. But that's OK: just as the laws of probability are not random, so the laws of fuzziness are not vague (Kaufmann et al., 1985).

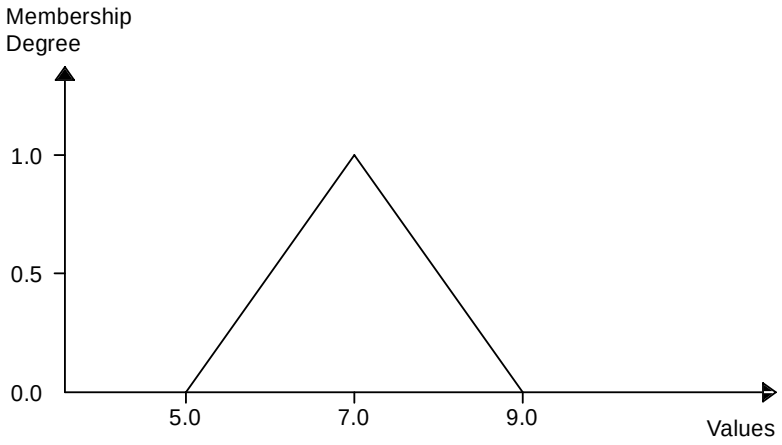

Figure 3.2. Triangular fuzzy number.

3.2.7. What is Possibility Theory?

Possibility theory is a new form of information theory which is related to but independent of both fuzzy sets and probability theory. Technically, a possibility distribution is a normal fuzzy set (at least one membership grade equals 1). For

example, all fuzzy numbers are possibility distributions. However, possibility theory can also be derived without reference to fuzzy sets.

The rules of possibility theory are similar to probability theory, but use either "max/min" or "max/times" calculus, rather than the "plus/times" calculus of probability theory. Also, possibilistic nonspecificity is available as a measure of information similar to the stochastic entropy.

Possibility theory has a methodological advantage over probability theory as a representation of nondeterminism in systems, because the plus/times calculus does not validly generalize nondeterministic processes, while max/min and max/times do. (Rutkowski, 2004).

4. FUZZY CONTROL DESIGN PARAMETERS

4.1. Fuzzy Logic Controller

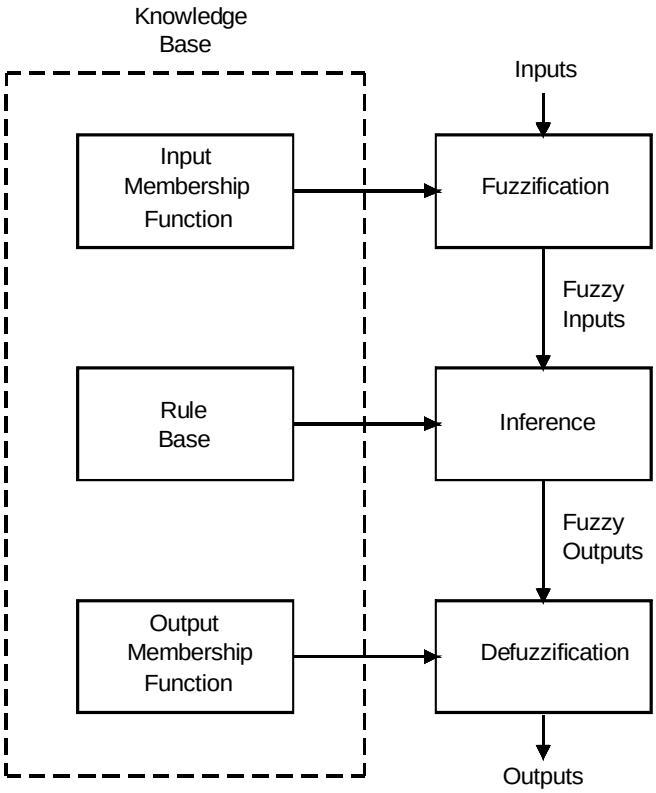

Figure 4.1. Block diagram of a typical fuzzy logic controller.

Figure 4.1 shows the block diagram of a typical fuzzy logic controller (FLC). There are four principal elements to a fuzzy logic controller:

- Fuzzification module (fuzzifier).
- Inference engine.
- Defuzzification module (defuzzifier).
- Knowledge base.

Automatic changes in the design parameters of any of the these elements creates an adaptive fuzzy controller. Fuzzy control systems with fixed parameters are non-adaptive. In this thesis non-adaptive fuzzy controllers are studied. Other non-fuzzy elements which are also part of the control system include the sensors, the

analogue-digital converters, the digital–analogue converters and the normalization circuits.

4.1.1. Fuzzification Module

The fuzzification module converts the crisp values of the control inputs into fuzzy values, so that they are compatible with the fuzzy set representation in the rule base. The choice of fuzzification strategy is dependent on the inference engine, i.e. whether it is composition based or individual-rule-firing based (Driankov et al., 1993).

This module of the controller is facing the real world, takes the data and converts it to a format that can be processed in inference engine. Input variables are first processed in this module according to input membership functions which are addressed in knowledge base of the Fuzzy Logic Controller. These inputs are converted into appropriate fuzzy sets. The fuzzified inputs are then sent to inference engine to be evaluated as per control rules stored in the fuzzy rule base addressed in knowledge base. See figure 3.1. The result is fuzzy sets in the universe of discourse.

The input is always a crisp numerical value limited to the universe of discourse of the input variable and the output is a fuzzy degree of membership (always the interval between 0 and 1). So fuzzification really doesn't amount to anything more than lookup table or functions evaluation (Jang et al., 1997).

The inputs applied to the fuzzification module can be analog or digital if it is taken into account to meet these requirements when design of the controller is implemented. We can call this module as fuzzifier as well and the operation carried out by this unit is fuzzifying.

The converted variables by fuzzifier are fuzzy linguistic variables. Now there are a few more concepts emerged that we need to consider: membership function, universe of discourse and linguistic variables.

4.1.2. Universe of Discourse

For a membership function of input or output the set for all the possible values for the variable is called universe of discourse or universal set. The universe of discourse is always represented by upper-case letters (X), and its members are always denoted by lower-case letters (x). For graphical illustration of a membership function shows the range of possible values for the variable and it is labeled as universe of discourse.

We need to define the universe of discourse for all of the inputs and outputs of the FLC, which are all crisp values (Rao, 1995).

4.1.3. Knowledge Base

The knowledge base consists of a database of the plant. It provides all the necessary definitions for the fuzzification process such as membership functions, fuzzy set representation of the input–output variables and the mapping functions between the physical and fuzzy domain (Cirstea et al., 2002).

The kernel of any expert system consists of a knowledge base (also called a long-term memory), a database (also called a short-term memory or a blackboard interface), and an inference engine (Klir et al., 1995).

The knowledge base contains general knowledge pertaining to the problem domain. In fuzzy expert systems, the knowledge is usually represented by a set of fuzzy production rules, which connect antecedents with consequences, premises with conclusions, or conditions with actions. They most commonly have the form "If A, then B," where A and B are fuzzy sets.

The inference engine may also use knowledge regarding the fuzzy production rules in the knowledge base. This type of knowledge, whose appropriate name is-rule base, is located in the unit called rule base base. This unit contains rules about the use of production rules in the knowledge base.

For the knowledge base, the expert defines the input and output observation (the descriptive words) and the range (the fuzzy number range). The expert also

defines the consequent output for each input (the rule). The designer defines the membership functions for inputs and outputs. The knowledge base is then put into action in an inference engine–a computer program that can take actual inputs, let them fire the rules, and export outputs to the domain system (McNeill et al., 1994).

Creating a knowledge base with the Fuzzy Knowledge Builder follows the five-step design

1. Identify the inputs and their ranges and name them.

2. Identify the output and its ranges and name it.

3. Create the fuzzy membership function for each input and output.

4. Translate the interaction of the inputs and outputs into As–Then rules. If all are and rules, this interaction can be represented as a matrix. (The Fuzzy Knowledge Builder was the first product to use a graphical matrix representation of the rules.) If and and or rules are allowed, fewer rules are required, but the clarity of matrix representation is lost.

5. Decide on the inference engine that will act on the specific inputs and the knowledge base to produce the specific defuzzified output (McNeill et al., 1994).

4.1.4. Inference Engine

In general, fuzzy controllers are special expert systems. Each employs a knowledge base, expressed in terms of relevant fuzzy inference" rules, and an appropriate inference engine to solve a given control problem. Fuzzy controllers vary substantially according to the nature of the control problems they are (Klir et al., 1995).

The inference engine of a fuzzy expert system operates on a series of production rules and makes fuzzy inferences. There exist two approaches to evaluating relevant production rules. The first is data-driven and is exemplified by the generalized modus ponens. In this case, available data are supplied to the expert system, which then uses them to evaluate relevant production rules and draw all possible conclusions. An alternative method of evaluation is goal-driven; it is exemplified by the generalized modus tollens form of logical inference.

4.1.5. Membership Functions

The present design utilizes three types of functions – Γ-function, L-function and Λ- function – all of which have already been presented. These functions have been proven to produce good results for control applications and can be easily implemented into hardware. The universe of discourse of the input variables is partitioned into five fuzzy sets or linguistic values (B_1 to B_5), while the output variable can take any of the seven linguistic values (D_1 to D_7). Graphical representations of the membership functions are shown in Figs. 4.5 and 4.6.

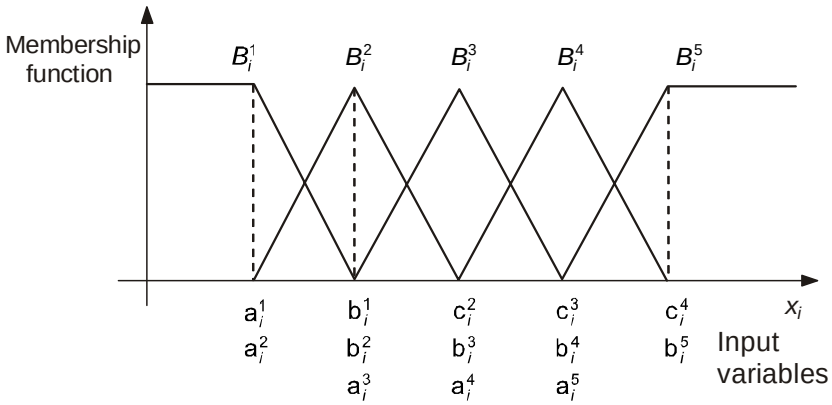

Figure 4.2. Membership function of input variables.

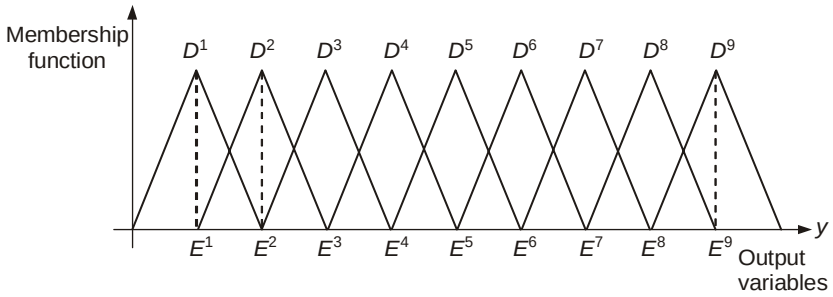

Figure 4.3. Membership function of output variables.

The following equations define the membership functions for the linguistic values associated with the input variables.

$$B_i^j = \begin{cases} 1 & x_i \langle a_i^j \\ \dfrac{(x_i - b_i^j)}{(a_i^j - b_i^j)} & a_i^j \le x_i \le b_i^j \quad \text{for j} = 1 \\ 0 & x_i \rangle\ b_i^j \end{cases} \tag{4.1}$$

$$B_i^j = \begin{cases} 0 & x_i \langle a_i^j \\ \dfrac{(x_i - a_i^j)}{(b_i^j - a_i^j)} & a_i^j \le x_i \le b_i^j \quad \text{for } j = 5 \\ 1 & x_i \rangle\ b_i^j \end{cases} \tag{4.2}$$

$$B_i^j = \begin{cases} 0 & x_i \langle a_i^j \\ \dfrac{(x_i - a_i^j)}{(b_i^j - a_i^j)} & a_i^j \le x_i \le b_i^j \\ & \text{for } j = 2,3,4 \\ \dfrac{(x_i - c_i^j)}{(b_i^j - c_i^j)} & b_i^j \le x_i \le c_i^j \\ 0 & x_i \rangle\ c_i^j \end{cases} \tag{4.3}$$

where i = 1, 2 and j = 1, . . . , 5.

The crisp values of the input variables are mapped onto the fuzzy plane using the equations above. It gives each input variable a membership function relating to the fuzzy sets, B_i^1 to B_i^5. It has to be pointed out that in these equations, B_i^j is used to denote the membership function. The symbol B_i^j is used for both, the linguistic value as well as the membership function. Strictly speaking, the linguistic value should be denoted using B_i^j, while the membership function using $\mu B_i, j(x_i)$.

In this example, 'μ' and '(x_i)' are dropped from the denotation of membership function, thus,

$$\mu_{\boldsymbol{B}i,j}(x_i) = x) = B_i^j = 0.5 \tag{4.4}$$

is used to indicate that *xi* belongs to the linguistic value B_i^j by a membership function of the value 0.5. The universe of discourse of the output variable is divided into seven linguistic values. The membership functions of the output values are intentionally made to be symmetrical, as this will simplify the defuzzification computation. E^1 to E^7 are the mean of each function and act as the weightings to the weighted average method of defuzzification.

4.1.6. Types of Membership Functions

Various types of membership functions are used in Fuzzy Logic Control Systems. Below you can find the most common membership functions. This is also used as one to the criteria for determining the suitable microcontroller type and the related hardware for the Fuzzy Logic Control System. In fuzzy control applications, Gaussian or bell-shaped functions and S-functions are not normally used. Functions such as Γ-function, L-function and Λ-function are far more common (Cirstea, M.N., 2002).

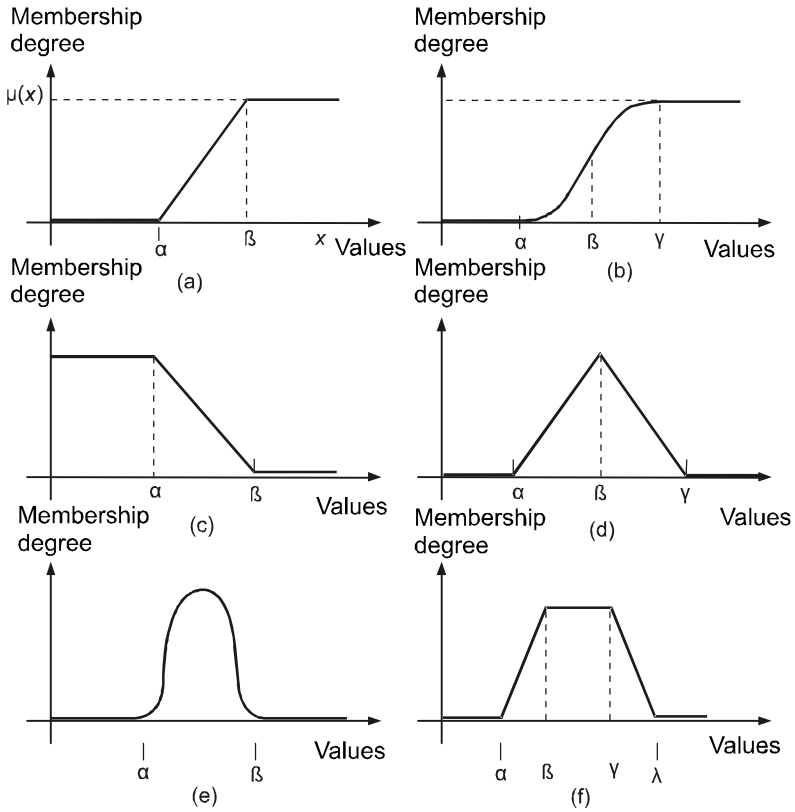

Figure 4.4. Types of membership functions: (a) Γ-function; (b) S-function; (c) Lfunction; (d) Λ-function; (e) Gaussian function and (f) Π-function.

4.1.7. Fuzzy Rule Base

The rule base is essentially the control strategy of the system. It is usually obtained from expert knowledge or heuristics and expressed as a set of IF-THEN rules. The rules are based on the fuzzy inference concept and the antecedents and consequents are associated with linguistic variables. For example:

IF error (*e*) is Positive Big (PB) THEN output (*u*) is Negative Big (NB)

rule antecedent rule consequent

Error (*e*) and output (*u*) are linguistic variables while Positive Big (PB) and Negative Big (NB) are the linguistic values. The rules are interpreted using a fuzzy

implication technique. In fuzzy control theory, this is normally Mamdani's implication technique (Cirstea et al., 2002).

Each input variable can take any of the five linguistic values, therefore 25 (= 5 x 5) rules are formulated. The rules have the typical fuzzy rule structure, using linguistic variables in both the antecedent and consequent, and are expressed in IF-"then" manner. They map the input states onto 25 output conditions (C_1 to C_{25}). The fuzzy rules have the general form,

$$R^k : \text{IF } x_1 \text{ is } A_1^k \text{ AND is } x_2 \text{ is } A_2^k \text{ , THEN is } C^k \tag{4.5}$$

where R_k (k = 1, 2, …, 25) is the kth rule of the fuzzy system, x_1, x_2 are the input linguistic variables, y is the output linguistic variable, A_i^k (i = 1, 2; k= 1, 2,…, 25) is the kth fuzzy set defined in the input space and A_i^k can take any linguistic value associated with x_i, C^k is the output condition inferred by the kth rule. If the denotation is such that the linguistic variables x_i for i = 1, 2 have the following linguistic values:

$$B_i^k \text{ for } i = 1, 2; j = 1 \text{ to } 5 \tag{4.6}$$

Then the rule base can be represented by a fuzzy associative memory (FAM) table (4.1).

Table 4.1. Fuzzy associative memory table.

x_1 \ x_2	B_2^1	B_2^2	B_2^3	B_2^4	B_2^5
B_1^1	C^1	C^2	C^3	C^4	C^5
B_1^2	C^6	C^7	C^8	C^9	C^{10}
B_1^3	C^{11}	C^{12}	C^{13}	C^{14}	C^{15}
B_1^4	C^{16}	C^{17}	C^{18}	C^{19}	C^{20}
B_1^5	C^{21}	C^{22}	C^{23}	C^{24}	C^{25}

4.1.8. Rule Antecedent and Consequent

The clause (“the light is green”, or more generally “this is true”) is called the antecedent; the clause (“cross the street”, or more generally “do that”) is called the consequent. In applying this rule, we first evaluate the truth of the antecedent. If the antecedent is sufficiently true, we execute the consequent instructions. We suppose that we have previously learned that the rule is valid. Assuming we wish to cross the street, we first see if the antecedent is true by comparing the observed data (the color of the traffic light) to the value given in the rule (red). If the comparison in the antecedent holds, then we execute the consequent instruction and cross the street. (Our rule here is not very good; it neglects to check for a car, which might be turning right directly in front of us.)

4.1.8.1. Antecedents

In formal terms, the antecedent is a logical proposition whose truth can be determined. The antecedent may be complex, made up of several simple propositions whose individual truths are combined by “and”, “or”, and “not” connectives, such as

“if” (speed is fast) “and” (it is raining) “and not” (windshield wipers on)

“then” turn on windshield wipers, reduce speed (2:3)

Most antecedent clauses will be comparisons between a data value for the case at hand and a value specified for a particular rule. Clearly, the range of circumstances, which a rule could cover, will depend on the types of data admissible in the syntax of our rules. In a conventional non-fuzzy expert system, data types commonly provided include numbers (if x is 35) and character strings (if name is “John”). Fuzzy expert systems provide additional data types: fuzzy numbers (if age is about 30) and fuzzy sets (if speed is fast) where speed could be slow, medium, or fast; the degree of truth value we have in a specific value (if we are only 0.9 sure that name is “John”, then the truth value in that value is 0.9 and can be tested in a rule antecedent. Other words that can be used in a rule antecedent are modifiers called hedges, and are used to modify truth values. For example, if we have a fuzzy set called age that could take on values of young, middle-aged, or old, we can ask “if age is young”, but can also ask “if age is somewhat young” or “if age is very young”. In this way, fuzzy expert systems provide a very flexible syntax for rules compared to non-fuzzy expert systems (Siler et al., 2005).

4.1.8.2. Consequents

There is a great variety of consequent instructions available in fuzzy expert systems. There are instructions for input of data and output of data and conclusions; for modifying old data and creation of new data; and instructions to control the rule firing process itself. The most controversial instructions in a fuzzy expert system are those that modify data and the truth values in data. The whole process of deriving new data and conclusions from given data is the inference process in logic; since

fuzzy logic is quite a new branch of mathematics, some of the inference methods presented in this book are controversial. (However, they have all been tested in practice over the past 15 years.) Rules with consequent instructions that control the rule firing process are called meta rules in AI parlance, and are of great importance in an inference process that involves several steps. (Most real-world problems require multi-step inference.)

"Flops" has about 70 different instructions, called commands. (We have just met one "Flops" command, declare, which identifies a data declaration.) Any of these instructions can be used in the consequent of a rule. One important command is rule, which identifies the definition of a "Flops" rule. Since any "Flops" command is legal in the consequent of a rule, a rule can generate other rules. "Flops" rules can also generate data declarations (Siler et al., 2005).

4.1.9. Defuzzifying Module

The diagram in Fig. 3.2 shows the membership functions related to a typical fuzzy controller's output variable defined over its universe of discourse. The FLC will process the input data and map the output to one or more of these linguistic values (LU_1 to LU_5). Depending on the conditions, the membership functions of the linguistic values may be clipped. Figure 6.5 shows an output condition with two significant (clipped above zero) output linguistic values. The union of the membership functions forms the fuzzy output value of the controller.

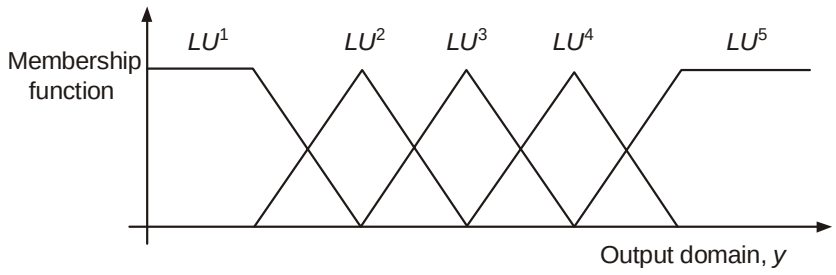

Figure 4.5. Membership function of the output linguistic values.

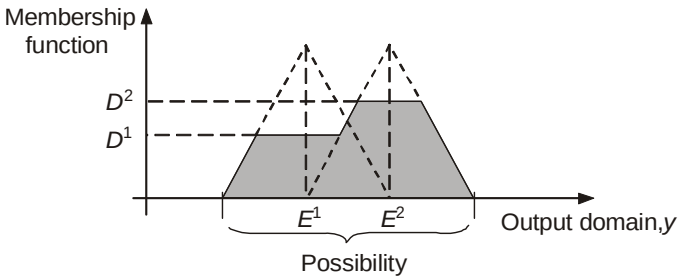

Figure 4.6. Possibility distribution of an output condition.

This is represented by the shaded area in Fig. 3.3 and is expressed by the fuzzy set equation:

$$S = \bigcup_{i=1}^{k} S_i, \mu_s(y) = m\underset{i}{a}x[\mu_{S_i}(y)], i = 1, 2, \ldots k \tag{4.7}$$

where:

S is the union of all the output linguistic values.

S_i is an output linguistic value with clipped membership function.

k is the total number of output linguistic values defined in the universe of discourse.

In most cases, the fuzzy output value *S* has very little practical use as most applications require non-fuzzy (crisp) control actions. Therefore, it is necessary to produce a crisp value to represent the possibility distribution of the output. The mathematical procedure of converting fuzzy values into crisp values is known as 'defuzzification'. A number of defuzzification methods have been suggested. The different methods produce similar but not always the same results for a given input condition. The choice of defuzzification methods usually depends on the application and the available processing power.

4.1.9.1. Weighted Average Method

The defuzzification method used in the example presented in the second part is the weighted average method. This method requires relatively little processing power

and is ideal for FPGA implementation where 'area space' is a major consideration. However, it is only valid for symmetrical membership functions. Each membership function is assigned with a weighting, which is the output point where the membership value is maximum. Based on the diagram in Fig. 3.3, the defuzzification process can be expressed by:

$$f(y) = \frac{\sum \mu(y).y}{\sum \mu(y)} \tag{4.8}$$

and using the weighted average method becomes:

$$f(y) = \frac{\sum_{m=1}^{n} E^m.D^m}{\sum_{m=1}^{n} D^m} \tag{4.9}$$

where:

f(y) is the crisp output value.

E_m is the crisp weighting for the linguistic value LU_m.

D_m is the membership value of y with relation to the linguistic value LU_m.

The crisp defuzzifier output is used as it is or via an interfacing block, to control the plant.

4.1.9.2. Centroid Method

This method is also known as the center of mass, or center of gravity method. It is probably the most commonly used defuzzification method. The defuzzified output, x^* is defined by

$$x^* = \frac{\int \mu_F(x).x dx}{\int \mu_F(x) dx} \tag{4.10}$$

where the symbol ∫ denotes algebraic integration.

4.1.10. Fuzzy Implementation Techniques

The most popular designs of neuro-fuzzy structures fall into one of the following categories, depending on the connective between the antecedent and the consequent in fuzzy rules:

- takagi-sugeno method – consequents are functions of inputs,
- mamdani-type reasoning method – consequents and antecedents are related by the min operator or generally by a t-norm,
- logical-type reasoning method - consequents and antecedents are related by fuzzy implications, e.g. binary, Zadeh and others.

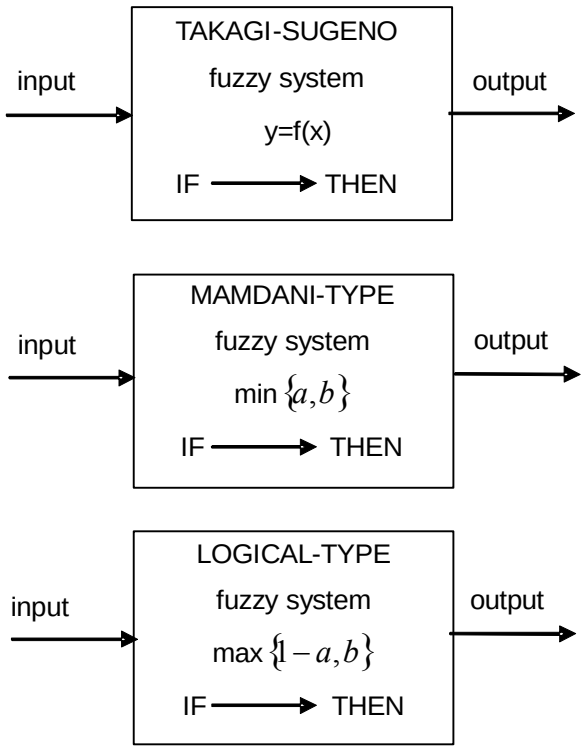

Figure 4.7. Examples of typical fuzzy systems (Rutkowski, 2004).

It should be noted that most applications are dominated by the Mamdani-type fuzzy reasoning. Moreover, in a specific singleton case, the Takagi-Sugeno method is reduced to the Mamdani method. On the other hand, there is a widely held belief

about the inferiority of the logical method comparing with the Mamdani method. However, it was emphasized by Yager that "no formal reason exists for the preponderant use of the Mamdani method in fuzzy logic control as opposed to the logical methodother than inertia." Moreover, "as a matter of fact the Mamdani approach has some disadvantages: its inability to distinguish more specific information in the face of vague information and the requirement of having the antecedents of the rules span the whole input space." This statement was an inspiration for the author to determine the type of fuzzy inference (Mamdani or logical) in the process of learning (Rutkowski, 2004).

4.1.11. Fuzzy Logic Implementation Example

In the design of an FLC system it is assumed that:

- A solution exists.
- The input and output variables can be observed and measured.
- An adequate solution (not necessarily an optimum one) is acceptable.
- A linguistic model can be created based on the knowledge of a human expert. In order to model a system linguistically, one needs to:
- Identify the input and output variables of the process to be controlled (the plant). For example: speed, temperature, humidity, etc.
- Define subsets that cover the universe of discourse of each variable and assign a linguistic label to each one. For example, the linguistic variable speed may be defined as three fuzzy subsets: slow, medium, and fast.
- Form a rule-base by assigning relationships between inputs and outputs.
- Determine a fuzzification method.
- Determine a defuzzification method to be used to generate a crisp output from the fuzzy outputs generated from the rule-base. The section to follow elaborates on the process of defuzzification and its numerous methods (Ibrahim, 2004).

Figure 4.8 shows a block diagram of the FLC and a stand-alone generator set. There are two inputs to the FLC. Its final functionality is determined by the choice of

inputs and the definition of the fuzzy rule base. This allows the system to be studied under different control strategies and specifications using the same hardware. Only three sections have to be modified: the two interfacing blocks and the rule base. In this experiment, the DC (Direct Current) voltage at the output of the rectifier is used as an input to the FLC. The plan is to test the system with a basic control strategy: maintaining the DC voltage within a small range, over varying load currents. The control output *u* is used to control the rate of flow of the fuel to the diesel engine.

The tasks involved in designing a typical FLC can be loosely summarized as follows:

- Decide on an overall strategy based on the design criteria.
- Identify an implementation technology.
- Identify the I/O variables (knowledge base).
- Define the membership functions for the variables (knowledge base).
- Formulate a fuzzy rule base.
- Choose a method of inference.
- Choose a defuzzification technique.

This design procedure forms a general guide and may well vary from one design to another, depending on the individual aims and requirements (Cirstea et al., 2002).

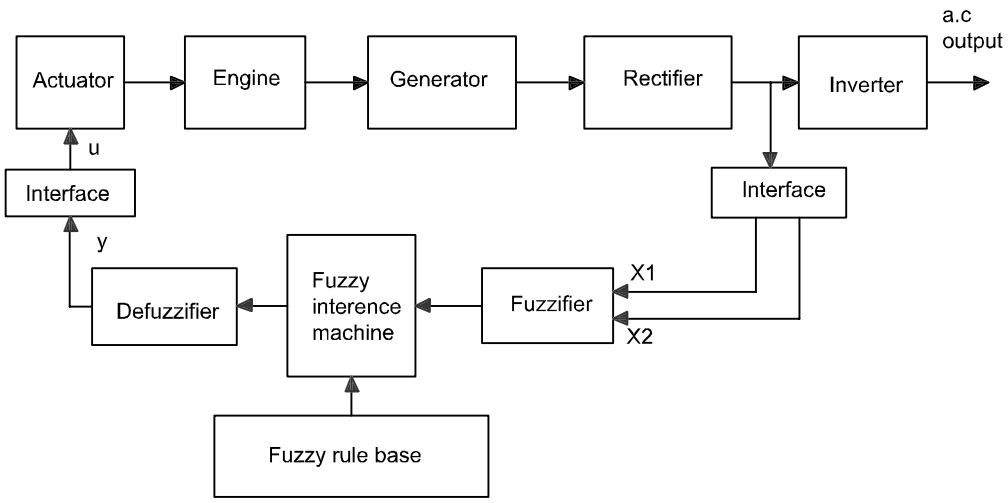

Figure 4.8. Block diagram of fuzzy logic controller and control plant (Cirstea et al., 2002).

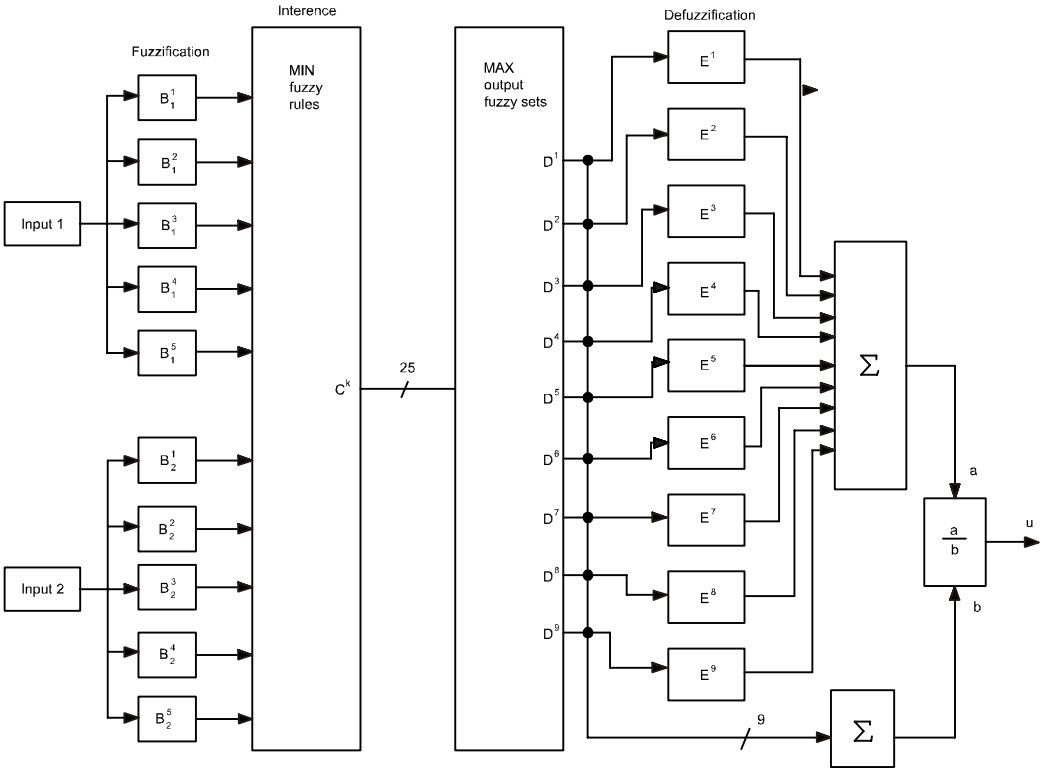

Figure 4.9. Block diagram of the operations in a fuzzy logic controller (Cirstea et al., 2002).

5. MICROCONTROLLER SELECTION CRITERIA

5.1. Overview

In this chapter the architecture of a general purpose microcontroller (μC) is given. And each functional block is defined. To decide the μC features for our specific application we need to take into account the fuzzy control system parameters which have been discussed in the last section. These parameters are listed below and the effects of them on μC selection criteria are discussed in this section.

5.2. Types of Microcontrollers Used in FLC Applications

There are several types of microcontrollers that are used in Fuzzy Logic Control applications. Here the general purpose microcontrollers and specialized hardware for FLC applications will be introduced.

5.2.1. Architecture of a General Purpose Microcontroller

The study for developing a software, on selecting microcontroller by determining some basic specifications of the microcontroller such as processing speed, program memory is aimed to describe a general purpose μC. Except for giving a multi processing option the program has been prepared at this point of view. But however we can not neglect that there are many different types of specialized μC developed specially for FLCS. Though this is not in the scope of this developed software general information will be as well given about these hardware.

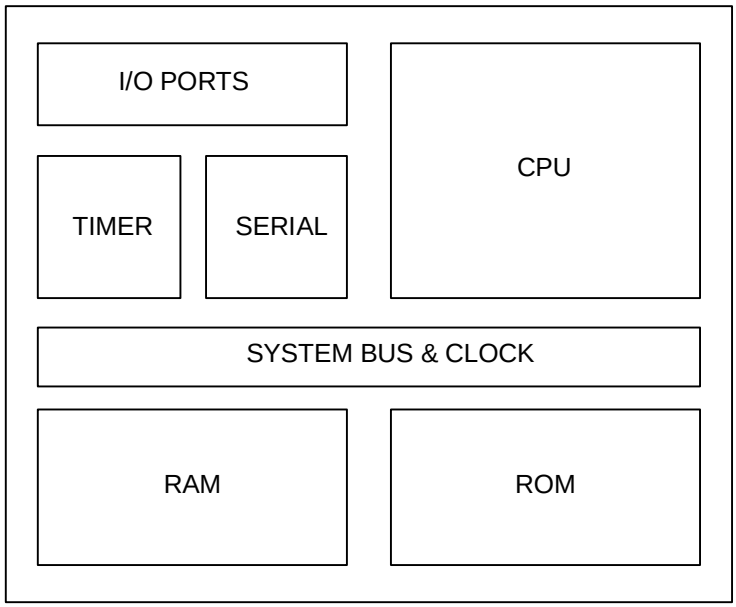

Figure 5.1. Block diagram of a simple microcontroller (Bannatyne et al., 1997).

A microcontroller is a single integrated circuit that at least contains the necessary elements of a complete computer system: CPU, memory, a clock oscillator, and input & output. Microcontrollers commonly contain additional peripheral modules, such as serial and timer units. An example of a simple microcontroller is shown in Figure 5.1.

The CPU fetches instructions from program memory via the address and data bus. Instructions are fetched from successive memory locations until the execution of a branch or jump instruction. Once fetched, an instruction waits in the instruction register or pipe unit until the CPU control logic is ready to decode it. An instruction is decoded by control logic to produce control signals. These control signals are fed into the execution unit to produce micro-operations that perform the function of the instruction. The execution unit contains 1) a set of registers (programmer's model, temporary registers, address, data & instruction buffers), 2) functional units, such as an ALU and shifter, and 3) internal busses to connect the registers and functional units (Bannatyne et al., 1997).

5.2.2. Specialized Microcontrollers for FLCS

Although the scope of this thesis only covers the selection criteria for general purpose microcontrollers I want to give brief information about other microcontroller types for FLCSs. Especially about that μCs which are specially developed for fuzzy systems. The performance and memory demand parameters will be compared and this will give a horizon about selecting such special purpose hardware for our application. By using the developed software you can calculate the performance and memory demand and by using the given values below you can also have idea about the selection criteria for such types.

5.2.3. Some Selected Fuzzy Functions

If the instructions set of a microcontroller is developed particularly for Fuzzy Logic Control Systems. This will facilitate you for your software development job. For specialized instruction commands will be given to the controller easily and with shorter sentences. This will support μC for performance and memory usage. Less programming memory and less processing power will suffice for the same application if we compare it to general purpose μC.

Below you can see some selected fuzzy functions which are required to use in a FLC system. No matter which microcontroller you use you need to fulfill these functions. If the instructions set of the selected μC is suitable to perform these functions this will be an important advantage.

Table 5.1. Fuzzy Functions.

No	Function	No	Function
1	Addition	10	Dilation
2	Subtraction	11	Data shift
3	Multiplication	12	Max-min composition
4	Division	13	Data movement
5	Minimum	14	Data exchange
6	Maximum	15	Push into stack
7	Fuzzification	16	Pop out of stack
8	Defuzzification	17	Two fuzzy numbers' distance
9	Centralization	18	Two fuzzy numbers' cross point

5.2.4. FLC Implementation Preferences

As a general taxonomy, we can identify four classes among the different implementation alternatives:

1- Software and hardware solutions with general purpose components, namely general purpose microcontrollers.

2- General purpose processors with instructions for specialized computations.

3- Dedicated fuzzy coprocessors.

4- Fuzzy ASIC's capable of stand-alone operations (Costa et al., 1995).

Software implementation of fuzzy algorithms on standard microcontrollers (MC68HC11, i8051, ST9, etc.) are today the most widely used techniques. While this approach well suites nontime-critical applications it becomes inadequate whenever processes requiring high or medium throughput appear.

The following table shows advantages and disadvantages of the approach.

Pros: complete flexibility, support of non-fuzzy computations by itself, availability of development systems, and availability of system support tools.

Cons: slow speed (Costa et al., 1995).

Typically, a Fuzzy Controller is composed by a set of if-then rules involving three basic operations (Dualibe et al., 2003). These basic operations for FLCSs were explained in the 4. chapter.

The computational complexity of a fuzzy rule base can be quantified by means of several parameters such as: the number of inputs, the number of outputs, the number and shape of the membership functions per input/output, the number of rules, the rule inference method, the defuzzification algorithm, and the precision needed. The system response time (i.e. input to output delay) is also used as a performance parameter for comparisons. Figure 5.2 shows a classification of typical control problems depending on the complexity (i.e. number of rules) and the system response time (Dualibe et al., 2003).

Apart from the always-possible software implementation of fuzzy inference rules in a workstation, which offers the highest flexibility but the largest response time, the relative simplicity of the fuzzy algorithms makes attractive the use of hardware structures for implementing Fuzzy Controllers.

Figure 5.2 identifies four classes among the different hardware implementation alternatives:

Implementation of fuzzy algorithms on standard microcontrollers is the most widely used technique.

Still maintains general-purpose computation capabilities by employing general-purpose processors (i.e. CISC and RISC) with the addition of a few specialized instructions to accelerate the fuzzy operations.

These are special-purpose processors dedicated to fuzzy computations. They cannot implement the entire control system by themselves due to the lack of general-purpose computation capability, but still provide some flexibility and configurability features.

Direct implementation on silicon of fuzzy algorithms by using high-level or full-custom synthesis techniques (i.e. analog, digital or mixed-signal). The FPGA-based implementations also fall in this category (Dualibe et al., 2003).

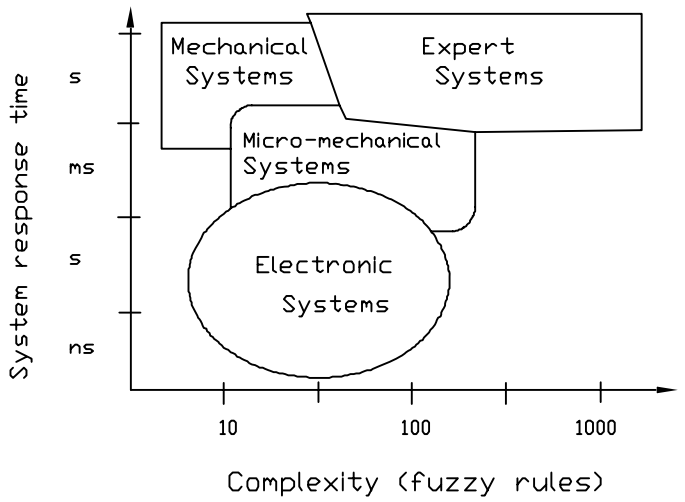

Figure 5.2. Allocation of the typical control problems in the complexity-time response (Dualibe et al., 2003).

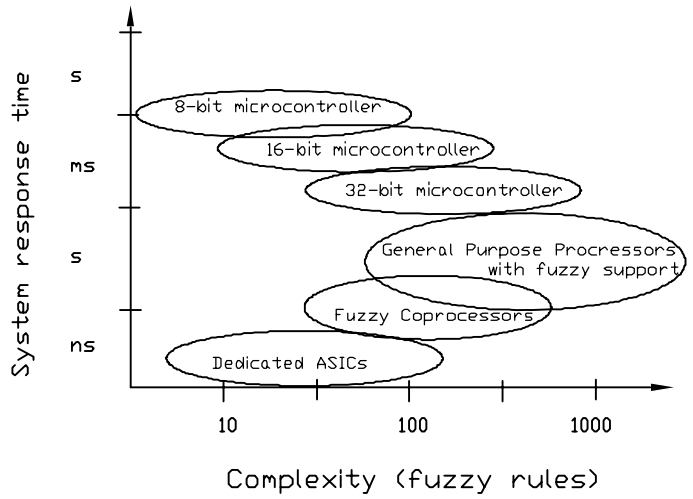

Figure 5.3. Allocation of the possible hardware solutions in the complexity-time response space (Dualibe et al., 2003).

Figure 5.2 shows a qualitative allocation of the different classes in the space complexity-time response. Table 5.2 summarizes the main advantages and disadvantages in each alternative.

It is evident that speed, complexity, flexibility, interaction requirements, real-time constraints, prototyping and production times will strongly influence the choice of the implementation option. Certainly, an optimal solution for the whole

application range does not exist, but the different approaches should be considered according to the requirements of a given design.

Table 5.2. Advantages and disadvantages of microcontroller types. (Dualibe et al., 2003).

Alternative	Advantages	Disadvantages
1. General-Purpose Processors:	-Complete flexibility -Short prototyping time -For high complexity FLCs -Availability of development systems	-Low speed (control systems with slow dynamics: 0.1-1 KHz)
2. General-Purpose Processors with Dedicated Fuzzy Instructions:	-Complete flexibility -Short prototyping time -For high complexity FLCs -Higher speed compared with 1: 1-10KHZ (CISC), 1 -50KHz (RISC) -Simple extension of existing CPU cores	-Higher cost than 1 (due to small production volume) -Low performance for high-end applications
3. Special-Purpose Coprocessors:	-Higher speed than 1 and 2 (i.e. 10-100KHz) -For moderate complexity FLCs (256-rule, 16-I/16-0) -Short prototyping time -Modularity	-Need of general purpose processor to support nonfuzzy operations -Limited flexibility
4. Dedicated ASICs:	-Fast processing, tuned to the application (i.e. 0.1-1MHz) -Availability of high level synthesis tools -Low cost in terms of die area -Available as a subsystem block for embedded applications -Cost-effective for high production volumes	-Fixed applications -Low flexibility (minimal programmability) -Limited complexity FLCs -Only for ASIC or FPGA based systems -Large prototyping time

5.2.5. Digital Techniques

Although fuzzy chips may have limited input/output capabilities, they have particularly useful applications in real-time control systems. Analog fuzzy values must be converted to binary digital signals. Analog-to-digital conversion can lead to quantization errors in both input signals and membership values. Thus, an deterioration in the fuzzy processing may occur if an insufficient number of bits is used to represent the analog signals. On the other hand, using a large number of bits can slow down the process. This is the trade-off between precision and speed.

Fuzzy dedicated circuits are characterized by:

- The number of inputs and outputs.
- The number and shapes of membership functions.
- Inference techniques including operators, consequences, and size of the premises.
- Defuzzification method.
- The number of fuzzy logic inferences per second, FLIPS.
- Physical size.
- Power consumption.
- Software available to support the design.

Numerous approaches to digital fuzzy ICs have been reported.

5.2.6. Digital Fuzzy IC

Togai et al. proposed the first fuzzy logic digital integrated circuit implementation in the mid-1980s. In the first prototype, they emphasized simplicity, making it particularly useful as a starting point in understanding custom digital IC implementation of fuzzy logic inference. The implementation is based on the analysis summarized here.

The inference mechanism analyzed lends itself to VLSI realization by a logical architecture with two-level hierarchy using min and max operations. Ordinary OR and AND gates were used and a custom CMOS technology was employed. The prototype implemented 16 rules with one antecedent and one consequent. It represented each fuzzy label by 31 elements with 16 possible values of membership. The maximum inference speed was 80,000 FLIPS.

5.3. Microcontroller “on-chip” Hardware Demands

For Fuzzy Control applications the parameters mentioned in the last chapter will be effective on our hardware choice.

In this section the on-chip hardware demand criteria for any predefined application is defined. This demand is discussed by classifying it into two main categories: Memory demand and processing speed demand. The architecture of a general purpose microcontroller is taken into consideration when this taxonomy is done.

In Fuzzy Logic Control Systems all the data is stored in a block which is called knowledge base as you can see in figure 4.1. And the detailed information has been given about KB. We can say if we can define the memory requirement for KB we will have defined the memory requirement for the FLCS.

For processing speed demand there is a concept which has been explained above in this thesis FLIPS. FLIPS is the measure of processing power requirement. Another main criteria which is effective on this issue is number of bits. These two main criteria will define us the dimension of performance requirement for the hardware.

5.3.1. Memory Demand

Two types of memory demand will be discussed. Random Access Memory, which called commonly as RAM and Reading Only Memory which is labeled as ROM.

ROM will keep the data which is one time programmed at the beginning and only used for reading the pre-stored data later while system is being used. Our knowledge base will be kept in this memory. So this will be the largest memory in our system. Since neuro-fuzzy systems are not included in the scope of this thesis. We don't need to store any extra data permanently while the program is running on our system. We will take into consideration only a memory of ROM.

RAM will have less volume in the FLCS as compared to ROM. All the data will not be required to be kept at the same time in RAM. Only the data which is under operation will be temporarily kept in this memory. So this volume of this memory will very little. Only in some extreme situation the amount of this memory demand will increase significantly.

This parameter determines the memory demand for the application. As explained in the last section knowledge base keeps all the required data for the application. As you can see in Figure 3.1 It consist of Input membership functions, output membership functions and fuzzy rule base, number of rules. Resolution should also be taken into account since it is effective on memory demand (Eichfeld et al., 1995).

$$M = R_s \, x \, [\sum_{k=1}^{KB} [256xNi + (Ni + Nr)xNo] + 264xNo] \tag{5.1}$$

Here R_s is the resolution, KB is the number of Knowledge Base, N_i is Number of input, N_r is number of rules, N_o is number of output. And M is the memory demand for the proposed microcontroller.

5.3.1.1. Using a Look-up Table

If you prefer to use look-up table for your application then the memory demand and performance demand of the μC will vary. In this approach, the fuzzy controller is reduced to a look-up table that can be described by a set of Boolean equations. The size of the look-up table for typical fuzzy controllers turned out to be reasonably small. This is due to the fact that the number of unique fuzzy inputs to a fuzzy controller is equal to the product of the dimensions of the inputs universes of discourse (Manzoul et al., 1992).

5.3.2. Performance Demand

Since the then-part chip can execute the consequent inference including defuzzification every 4 clock cycles and the clock frequency is Fc MHz, the inference speed of the then-part chip is F_c / 4 MFLIPS (Fuzzy Logic Inference Per second). While, the inference speed of the if-part chip is dependent on the numbers of inputs, outputs and rules. First, let us consider the dependence on the numbers of inputs and outputs. When the number of inputs and outputs are N_i and N_o respectively, (N_i . N_o) clock cycles are needed at least for the process. So, the inference speed is F_c / (N_i . N_o) MFLIPS. Next, let us consider the dependence on the number of rules. Since each processing element executes one antecedent per one clock cycle, F_c (clock frequency) x N_p (the number of processing elements) M antecedents can be executed per second. So, since N_i . N_o . N_r antecedents are processed in case of N_i-input, N_o- output and N_r-rule, the inference speed is F_c x N_p / (N_i . N_o . N_r) MFLIPS. Therefore, the inference speed (Is) of the system can be expressed as follows (Sasaki et al., 1991):

$$I_s = min\ \{(F_c, F_c / (N_i . N_o),\ F_c\ x\ N_p / (N_i . N_o . N_r)\}\ \text{MFLIP} \qquad (5.2)$$

Here we can say Is (Inference speed) can be equal to Fc, Fc / (Ni . No) or FcxNp / (Ni . No . Nr) but for as per the minimum law we must determine the minimum value and take it into consideration.

If we want to calculate the maximum clock frequency of microcontroller for the given application then equation (5.1) can be reorganized as:

$$F_c = Max\ \{I_s, I_s\ x\ (N_i . N_o), I_s\ x\ (N_i . N_o . N_r) / N_p\} \qquad (5.3)$$

Here F_c denotes maximum required μC speed in MHz for our application. Since this formula will be used in our software to compute the required processing speed we need to take it as the worst case that we can meet for an ordinary general purpose μC. If you are planning to use a special purpose μC developed for Fuzzy

Logic Control Systems with special instructions set this value might change specifically.

As you can see in the figure 5.4 the most processing power of the μC is spent at defuzzification function. So we have to take it into consideration while calculating the required μC performance demand.

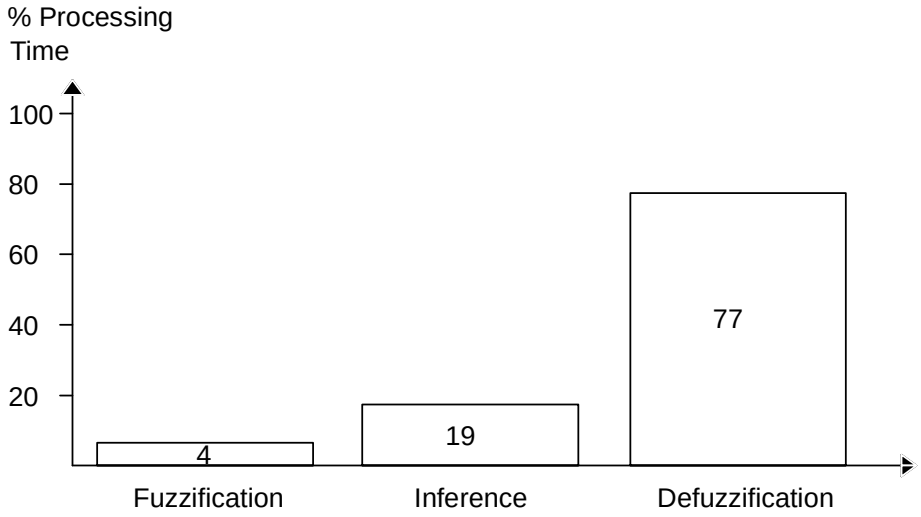

Figure 5.4. Percentile of the processing time (Ungering et al, 1994).

5.4. Developing the Software

The software development has been realized in three phases:

1. Preparing the flowchart of the software.
2. Preparing the program algorithm
3. Writing the program codes in Visual Basic (VB) programming language.

5.4.1. The Flowchart of the Software

In Figure 5.5 the flowchart of the software can be seen. Preparing the flowchart is the basic step for software development. The operations which are implemented in the developed software are defined graphically. Loops can bee seen easily since they are drawn graphically.

We can summarize the operations shown in the flowchart as follows:

1. Ask for required variables to be entered.
2. Examine these variables by comparing the given reference values.
3. If the entered variables are within the range of reference values they are then assigned and stored.
4. Execute the calculations as per the given equations.
5. Display the results.

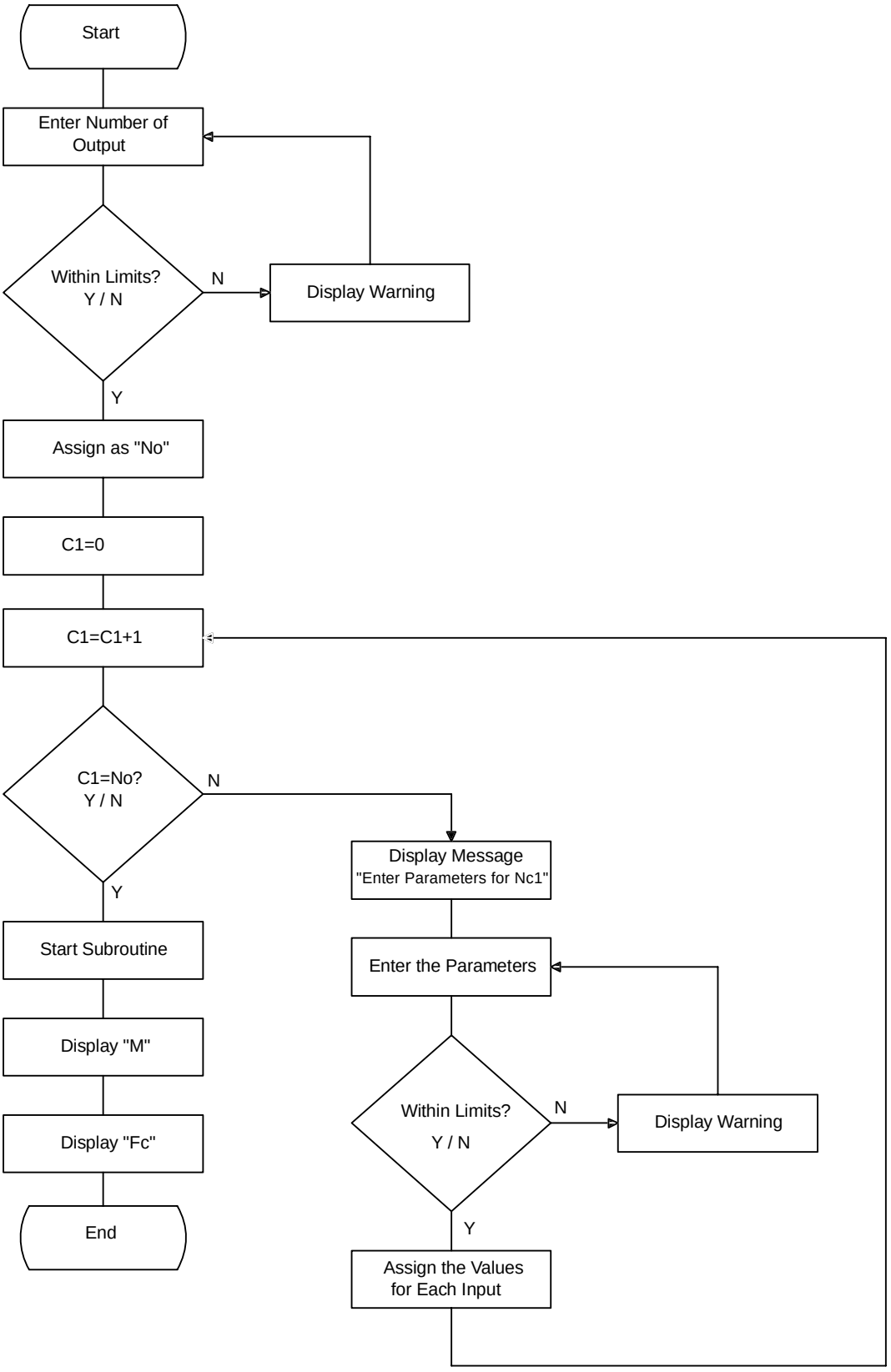

Figure 5.5. Flowchart of the developed software.

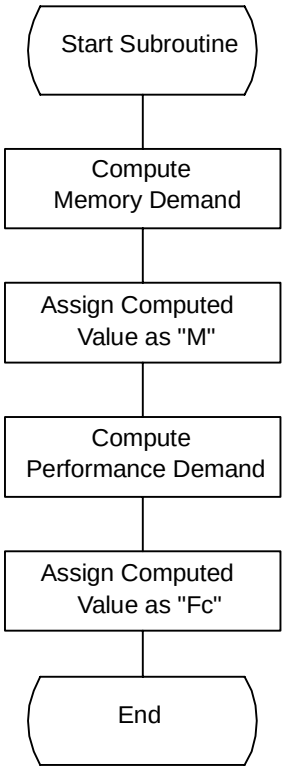

Figure 5.6. Flowchart of the subroutine.

5.4.2. Program Algorithm

Step 1: Start the program

Step 2: Enter number of output

Step 3: Assign this value as N_o

Step 4: Is the entered value within limits (Y/N)?

Step 5: If the answer is Y go to Step 8

Step 6: Display warning message

Step 7: Go to Step 2

Step 8: Reset C_1 (Counter 1)

Step 9: $C_1=C_1+1$

Step 10: Is $C_1=N_o+1$ (Y/N)?

Step 11: If the answer is 'Y' go to Step 26

Step 12: Display "Enter parameters for output C_1"

Step 13: Enter number of inputs

Step 14: Assign this value as N_i

Step 15: Is the entered value within limits (Y/N)?

Step 16: If the answer is Y go to Step 19

Step 17: Display warning message

Step 18: Go to Step 15

Step 19: Enter number of rules

Step 20: Assign this value as N_r

Step 21: Is the entered value within limits (Y/N)?

Step 22: If the answer is Y go to Step 25

Step 23: Display warning message

Step 24: Go to Step 19

Step 25: Go to Step 9

Step 26: Start Subroutine

Step 27: Compute results according to Subroutine

Step 28: Display the results

Step 29: End

Subroutine:

Step 1: Compute the amount of memory demand according to:

$$M = R_s \, x \, [\sum_{k=1}^{KB}[256xNi + (Ni + Nr)xNo] + 264xNo]$$

Step 2: Compute the performance demand according to:

$$F_c = Max \{I_s, I_s \, x \, (N_i \, . \, N_o), I_s \, x \, (N_i \, . \, N_o \, . \, N_r) / N_p\} \; (F_c)$$

5.4.3. User Defined Parameters

First of all user should enter some parameters as row data for the software. In the 4th chapter a detailed explanation has been given about the parameters for FLCSs. Then this data will be used by the software to make some computation as per user's selections. The screen-shot of the first screen of the developed software can be seen in figure 5.6 As you can see the software has a graphical user interface as to provide a user friendly face to the user.

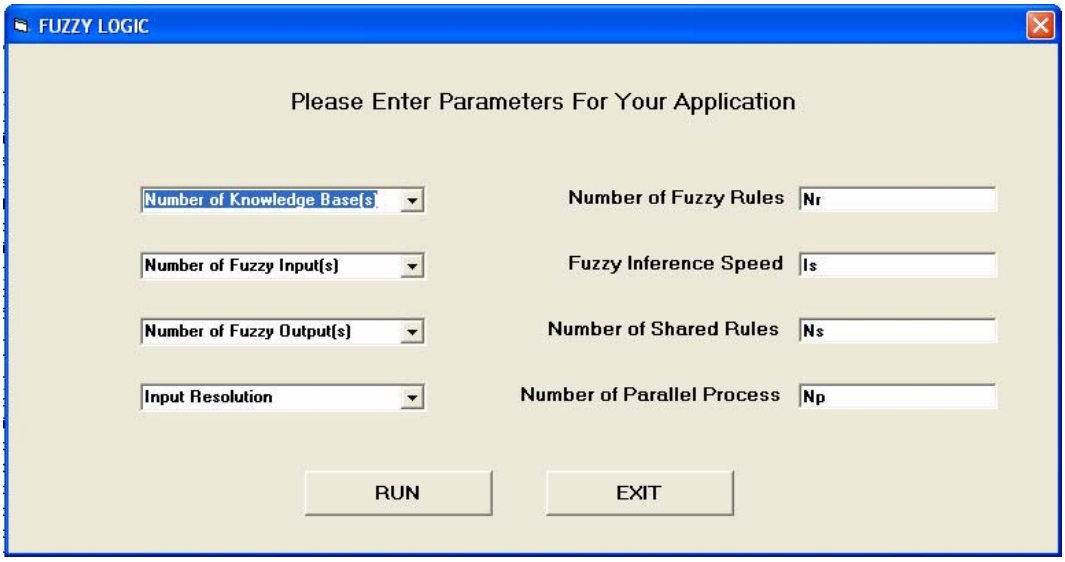

Figure 5.7. Screenshot before parameters are defined.

Above you can see a screenshot of the developed software and the parameters which should be entered by the user.

The number of fuzzy logical inferences per second (FLIPS) is equal to the Fuzzy logic inference value required for the application. For practical systems, a speed of 50M FLIPS can be achieved (Manzoul et al., 1992).

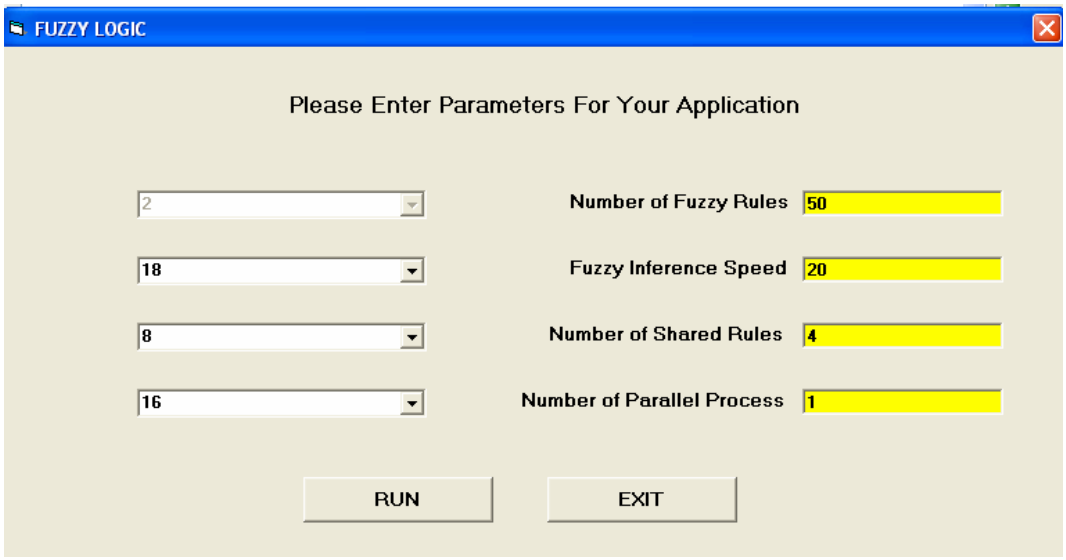

Figure 5.8. Screenshot after parameters are defined.

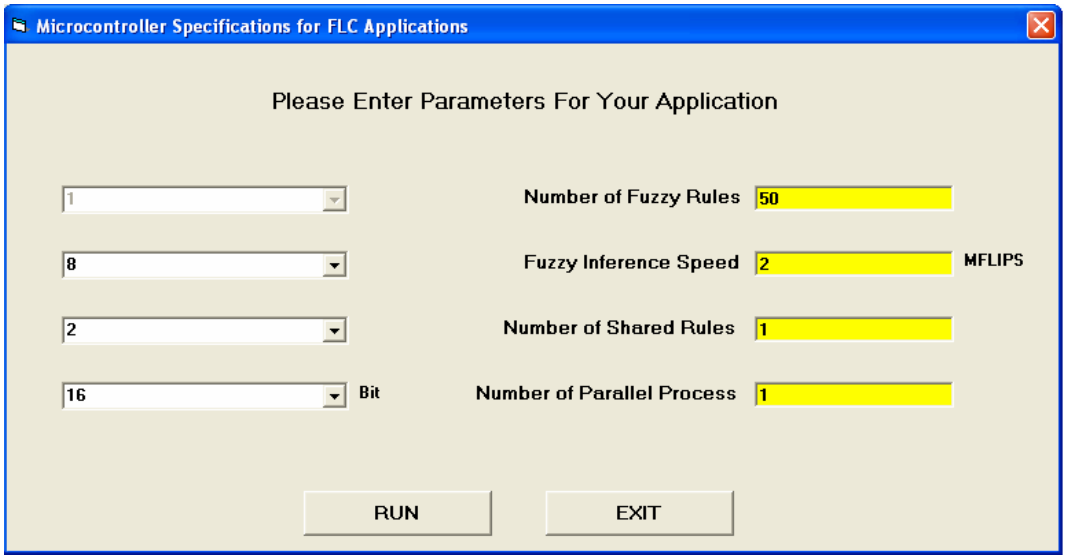

Figure 5.9. Numerical example.

For the given values above, the results has been provided as you can see below.

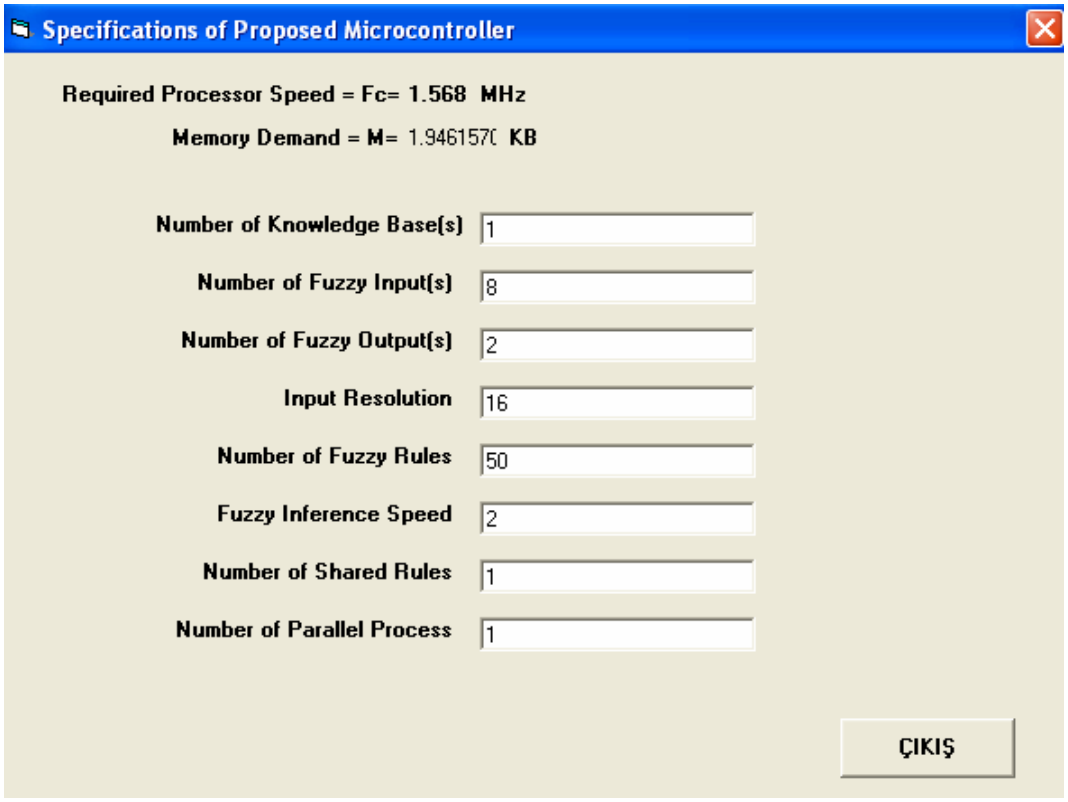

Figure 5.10. The results for the given values.

5.5. Numerical Examples

We need to use the parameters of two different systems and compute the memory and performance demand of the required microcontroller by using the developed software. To see the effectiveness of the developed software one of the systems is chosen with less fuzzy rules and requires less processing speed such as a fuzzy logic temperature controller. The second system has many rules and requires much more processing speed such as auto-focusing of a photograph machine.

The input parameters for the slow system are listed below:

- Number of knowledge base: 1
- Number of fuzzy inputs: 4
- Number of fuzzy outputs: 2
- Input resolution: 8 bit
- Number of fuzzy rules: 40
- Required fuzzy inference speed: 1 MFLIPS
- Number of shared rules: 0
- Number of parallel processors: 1

The results for the slow system:

- Performance Demand: 0.32 Mhz = 320 KHz
- Memory Demand: 0.37 KB = 379 B

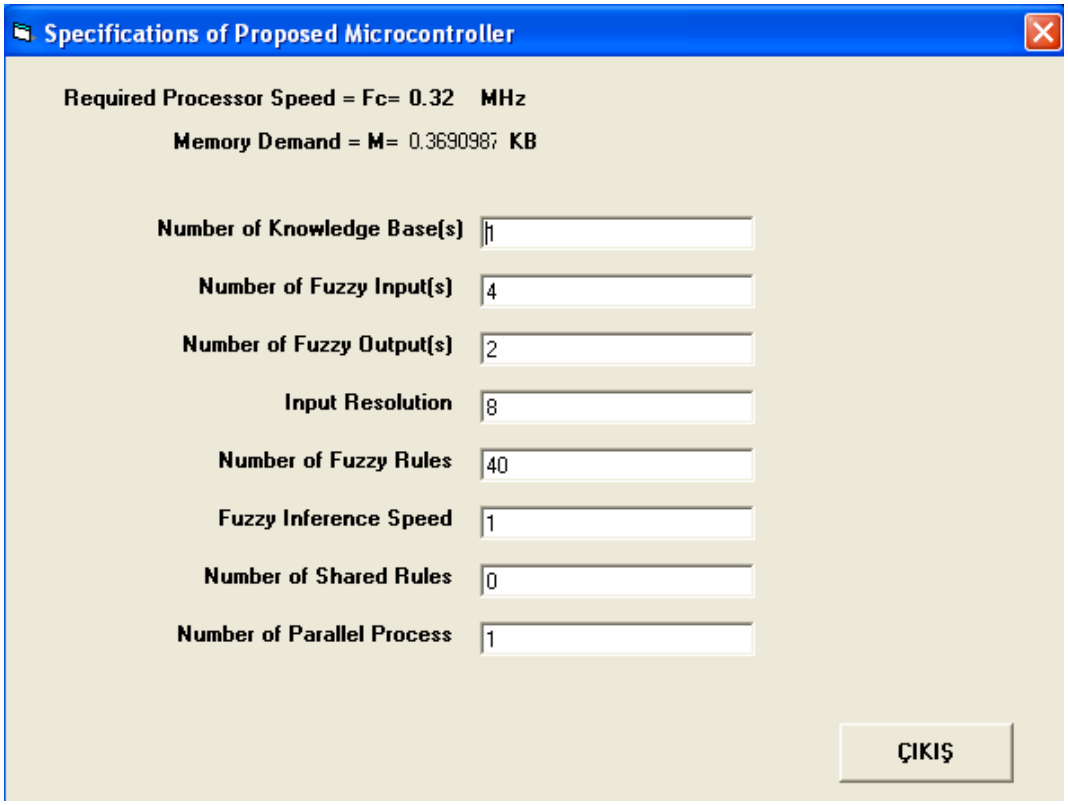

Figure 5.11. The results for a simple system

Now let us compute the results for the second system. This will show the increasing needs of memory and performance as the input parameters increase.

The input parameters for the fast system are listed below:

- Number of knowledge base: 1
- Number of fuzzy inputs: 8
- Number of fuzzy outputs: 2
- Input resolution: 16 bit
- Number of fuzzy rules: 100
- Required fuzzy inference speed: 20 MFLIPS
- Number of shared rules: 0
- Number of parallel processors: 1

The results for the fast system:

- Performance Demand: 32 MHz
- Memory Demand: 3.62 KB = 3711 B

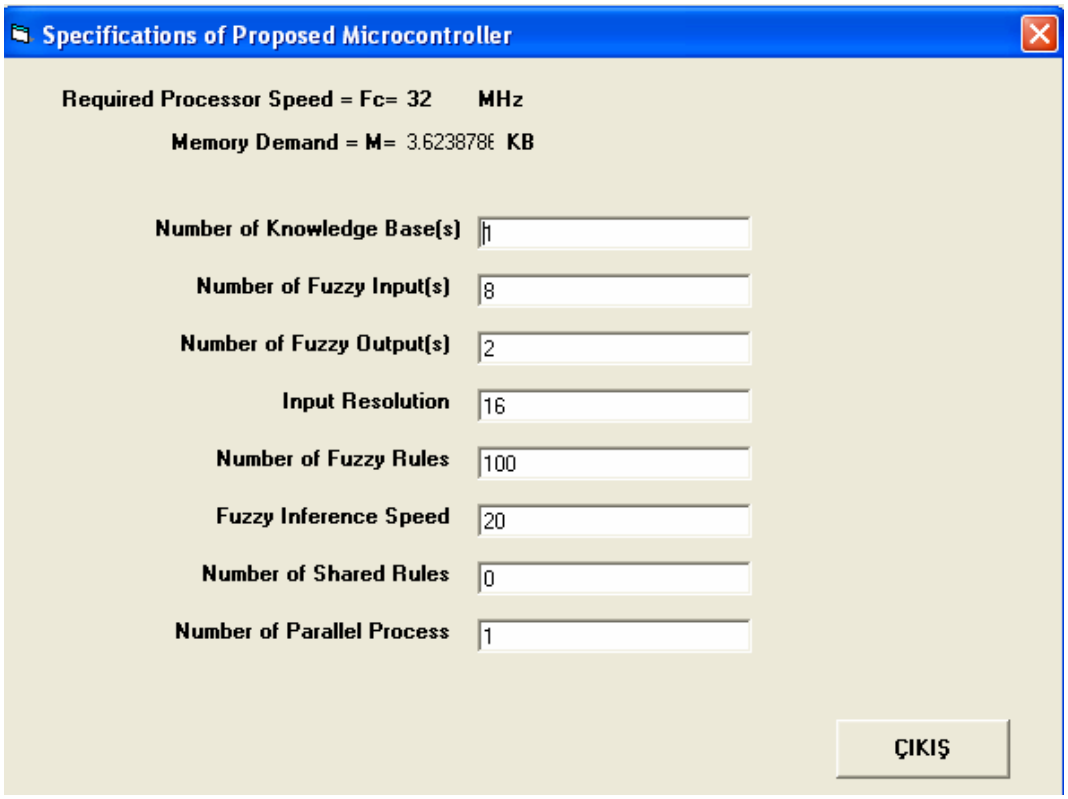

Figure 5.12. The results for a complex system

6. RESULTS AND CONCLUSION

Fuzzy logic control systems can be categorized by considering the parameters of the system. You can see the parameters used by the developed software in figure 5.7. If the values for each parameters are small, we can say it is a simple system, if values for each parameters are high, then we can say it is a more complex system. As the complexity of the system increases the memory demand and the performance demand increases as you can see in equation (5.1) and equation (5.3). This has been also shown by selected examples in chapter 5. In figure 5.11 the parameters and the demands of a slow and simple system can be seen. Since the effective parameters values are small, the demand for both performance and memory are small. But if we consider a more complex system and calculate the demand for performance and memory, the results are noticeably higher then the first example. You can see the parameters and results screen of the developed software in figure 5.12. The calculations have been carried out according to equations (5.1) and (5.2). The effect of each parameter is explained in 5th chapter.

The results of the numerical examples in the 5th chapter proved that certain parameters have certain effects on the amount of program memory and required processor speed. As you can see in the examples the results are practical. According to the given parameters values, two basic microcontroller selection criteria, namely memory and performance demands, is computed and monitored. When we consider the function of the developed software we can call it “FLCS memory and performance demand monitor”.

Fuzzy logic control was not a preferable method before 1980s. But nowadays it’s very popular even for small electronic goods which we use at home. So it is obvious that its importance is growing rapidly day by day.

In this thesis, it is aimed to propose a user-friendly software. It is developed by means of a high level visual programming language, namely Visual Basic 6.0 of Microsoft. It is considered that, this programming language is the best choice for developing such software with a visual interface. All the parameters may be entered in the same window, all the warnings or information messages even the results can

be displayed in the same window. This provides the ability of simply viewing all the data in the same window. This is useful and practical.

The proposed software is prepared only for the purpose of selecting the proper microcontroller for FLC applications. This software can be as well extended by adding some more solutions for FL. VB is suitable for developing future revisions of the software by adding desired extensions.

Most of the applications of fuzzy logic are digital applications, digital embedded applications. So specialized or general purpose microcontrollers are used. Microcontrollers are various and the best performance and the cost effective choice should be done by the designer. Designers can also utilize this study for the hardware selection if they don't prefer to use all embedded system, or prefer to use a microprocessor instead.

Specialized microcontrollers for FLCSs should be considered for future work. Although it is mentioned in the 5th chapter it is not included in the developed software since this is not in the scope of this thesis. Notice that you can decrease the need for processing speed, even for the memory demand, by using specialized microcontrollers for FLC applications. They can execute the required FLC operations with less instruction as compared to general-purpose microcontrollers because they have specialized constructions set. So this reduces the program memory needs and since operations are executed with less instructions this reduce the number of instructions and so the required processing speed demand, too. In table 5.1 some specialized fuzzy functions can be seen. The specialized controllers have special architecture developed for the specific confined purpose, this support the performance of the controller. Their higher cost and non-flexibility can be counted as their main drawbacks. As this specialized hardware becomes more widespread, the price of them will also drop down.

Along the first three chapters fuzzy logic was introduced with its all aspects. This was also a detailed tutorial for fuzzy logic. This will be useful for inexperienced designers.

In the 2nd chapter some software tools which are most widespread and can be provided via internet. These software tools are useful for designers.

In the 3rd chapter design parameters for fuzzy logic system were detailed. These parameters should be known to develop a fuzzy logic system and they are also required for hardware selection. In the 4th section a simple example of fuzzy logic control system has been retold. So it should be easier to understand some aspects by studying on it.

The criteria required selecting the microcontroller type is given in the 5th chapter. Two application examples were added in this section.

In the 6th chapter, the last chapter the results obtained from the numerical examples were compared and considered. Conclusions of the overall study have been commented. Some advice for future works also has been given.

REFERENCES

BANNATYNE, R., VIOT, G., 1997. Introduction to Microcontrollers, Conference Proceedings Wescon/97., 564-574.

CIRSTEA, M.N., KHOR, J.G., and MCCORMICK, M., 2002. Neural and Fuzzy Logic Control of Drives and Power Systems, Newnes an imprint of Elsevier Science, Woburn p.399.

COSTA, A., GLORIA, A.D., FARABOSCHI, P., PAGNI A., and RIZZOTTO, G., 1995. Hardware Solutions for FLC, Proceedings of the IEEE, 85(3): 422-434.

DRIANKOV, D., HELLENDOORN, H., and REINFRANK, M., 1996. An Introduction to Fuzzy Control. Springer, p.316.

DUALIBE, C., VERLEYSEN, M., and JESPERS, P.G.A., 2003. Design of Analog Fuzzy Logic Controllers in CMOS Technologies, Kluwer Academic Publishers, New York, 1-10.

EICHFELD, H., KLIMKE, M., MENKE, M., NOLLES J., and KÜNEMUND T., 1995. A General-Purpose Fuzzy Inference Processor, IEEE Micro,15(3):12-17.

IBRAHIM, A.M., 2004. FUZZY LOGIC for Embedded Systems Applications, Newnes an Imprint of Elsevier Science, Burlington, p.289.

JACKSON, A., 1994. Fuzzy Logic vs Traditional Approaches To TheDesign of Microcontroller-Based Systems, IEEE Aerospace Applications Conference, 491-503.

JANG, J.S., and ROGER, G., 1997. MATLAB Fuzzy Logic Toolbox, the MathWorks Inc., 1.1-2.27.

KAUFMANN, A., AND GUPTA, M.M., 1985. Fuzzy Sets and Systems "Introduction to Fuzzy Arithmetic", Reinhold, New York.

KASABOV, N.K., 1998. Foundations of Neural Networks, Fuzzy Systems, and Knowledge Eng., Massachusetts Institute of Technology, London p.547.

KANDEL, A., and LANGHOLZ, G., 1998. "Fuzzy Hardware, Architectures and Applications", Kluwer Academic Publishers.

KLIR, G.J., YUAN B., 1995. Fuzzy Sets and Fuzzy Logic, Theory and Applications, Prentice Hall, p.591.

LARSEN, HENRIK LEGIND, 1997. Fundamentals of Fuzzy Sets and Fuzzy Logic. AAUE Computer Science, 1-2.

LEE, Y., JANG, S., CHUNG, K., LEE, D., KIM, W., and LEE, C., 1994. A Fuzzy-Control Processor For Automatic Focusing, IEEE Micro, 40(2): 138-144.

MCNEILL, F.M., and THRO, E., 1994. Fuzzy Logic: A Practical Approach. Academic Press, Boston, p.279.

MANZOUL, M.A., and JAYABHARATHI, D., 1992. Fuzzy Controller On FPGA Chip, IEEE International Conference, 1309-1316.

REZNIK, L., 1997. Fuzzy Controllers, Newnes, Melburne, p.285.

RAO, V.B., 1995.C++ Neural Networks and Fuzzy Logic, IDG Books Worldwide, Inc., 70-490.

RUTKOWSKI, L., 2004. Flexible Neuro-Fuzzy Systems Structures, Learning and Performance Evaluation, Kluwer Academic Publishers, New York, p.277.

SASAKI, M., UENO, F., and INOUE, T., 1991. 7.5MFLIPS Fuzzy Microprocessor Using SIMD and Logic-in-Memory Structure, 527-534.

SILER, W., BUCKLEY J.J., 2005. Fuzzy Expert Systems and Fuzzy Reasoning. John Wiley & Sons, Inc., New Jersey, 5-20.

SPOONER, J.T., MAGGIORE, M., ORDONEZ, R., and PASSINO, K. M., 2002. Stable Adaptive Control and Estimation for Nonlinear Systems (Editor: S. HAYKIN). Adaptive and Learning Systems for Signal Processing, Communications, and Control. John Wiley & Sons, Inc, 184-185.

UNGERING, A.P., BAUER, H., GOSER K., 1994. Architecture of a Fuzzy-Processor based on an 8-bit Microprocessor, IEEE World Cong., 1: 297-301.

YAN, J., RYAN, M., and POWER, J., 1994. Using Fuzzy Logic, Prentice-Hall, New York.

YAGER, R.R., FILEV, D.P., 1994. Essentials of Fuzzy Modeling and Control. John Wiley & Sons, New York, p.388.

WEB1: www.transfertech.de/www/soft_e.htm

WEB2: www.programmersheaven.com/zone22/cat167/1244.htm

WEB3: www.fuzzytech.com

WEB4: www.cs.cmu.edu

BIOGRAPHY

I was born in Tarsus, Turkey, in 1975. I completed the high school education in Adana. I received the B.S. degree in Electrical and Electronics Engineering from İnönü University, 2002. After completion my B.S. training, I have started MSc degree in the department of Electrical and Electronics Engineering in Çukurova University.

Nowadays I'm working in Automotive sector as a project engineer.

My areas of interest include Fuzzy logic control, microcontrollers communication protocols, multiplexing, specialization of microcontroller based control systems for automotive.