

# **Design and Implementation of a General Purpose Fuzzy Control System**

by

Selvakumar A/L Balasupramaniam

Dissertation submitted in partial fulfillment of  
the requirements for the  
Bachelor of Engineering (Hons)  
(Electrical and Electronic Engineering)

JUNE 2004

Universiti Teknologi PETRONAS  
Bandar Seri Iskandar  
31750 Tronoh  
Perak Darul Ridzuan

# CERTIFICATION OF APPROVAL

**Design and Implementation of a General Purpose Fuzzy Control System**

by

Selvakumar A/L Balasupramaniam

A project dissertation submitted to the  
Electrical and Electronic Engineering Programme  
Universiti Teknologi PETRONAS  
in partial fulfillment of the requirement for the  
BACHELOR OF ENGINEERING (Hons)  
(ELECTRICAL & ELECTRONIC ENGINEERING)

Approved by,



(Dr. Mohammad bin Awan)

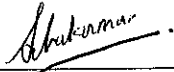
UNIVERSITI TEKNOLOGI PETRONAS

TRONOH, PERAK

June 2004

## CERTIFICATION OF ORIGINALITY

This is to certify that I am responsible for the work submitted in this project, that the original work is my own except as specified in the references and acknowledgements, and the original work contained herein have not been undertaken or done by unspecified sources or persons.



---

SELVAKUMAR A/L BALASUPRAMANIAM

## ABSTRACT

The main objective of this final year project is to design and to implement a general purpose Fuzzy control system that employs IF-THEN fuzzy rules. The design of the general purpose Fuzzy controller has been refined to a Fuzzy air conditioner control system. The Fuzzy control system for an air conditioning system is implemented using two inputs and two outputs variables. The inputs to the system are the room temperature and the room humidity while the outputs are the delta temperature increase/decrease and the relative air flow, controlled by a motor. The Fuzzy air conditioner makes reasoning similar to human decision making based on the room temperature and the room humidity level to adjust the air flow speed and the temperature out from the air conditioner with assumption of that the most comfortable relative humidity level would be at around 45% RH and at room temperature of 27<sup>0</sup> C. The Fuzzy air conditioner has the operating range of 10<sup>0</sup> C to 50<sup>0</sup> C for the room temperature and  $\pm 10\%$  RH for the room humidity. In this project, the Fuzzy Logic Concept is used to implement a Fuzzy control system using 25 main IF-THEN proposition statements. The project was implemented through software (Active-HDL, MATLAB, Fuzzy Logic Toolbox, Project Navigator, and FuzzyTECH) utilization for design development and verification as well as laboratory activities for the hardware realization using Xilinx's VIRTEX II FPGA chip. The author managed to design and implement a simple Fuzzy air conditioning system that able to make decisions based the room temperature and the room humidity at software realization stage only.

## ACKNOWLEDGEMENT

During all the activities performed for this project, the author has received some guidance and assistance which have enabled the author to perform the task smoothly. Therefore, the author would like to take this opportunity to express his highest gratitude mainly to following people in their contribution to the author in this project.

- i) Associate Professor Dr. Mohammad Bin Awan (Supervisor) for providing the necessary guidance to the author when the author faced some problems regarding the project.
- ii) Mr. Balbir Singh (Lecturer) for providing some assistance in certain areas of the project.
- iii) Encik Musa (Laboratory Technician) for providing the necessary assistance for the author in laboratory activities of the project.
- iv) Encik Ramli (Laboratory Technician) for the cooperation and guidance given in using certain equipment.

The author also would like to express his appreciation to those who has assisted the author and contributed effort in the completion of project for this semester.

Thank you very much.

## TABLE OF CONTENTS

CERTIFICATION OF APPROVAL .....	i
CERTIFICATION OF ORIGINALITY .....	ii
ABSTRACT.....	iii
ACKNOWLEDGEMENT .....	iv
CHAPTER 1: INTRODUCTION .....	1
1.1 Background of Study .....	1
1.2 Problem Statement .....	2
1.3 Objectives .....	2
1.4 Scope of Study .....	3
CHAPTER 2: LITERATURE REVIEW AND THEORY .....	4
2.1 Fuzzy Logic Concept.....	4
2.1.1 Fuzzy Model .....	5
2.1.2 Fuzzy Rule Generation .....	6
2.1.3 Membership Functions.....	6
2.2 Control System Design .....	8
2.3 Fuzzy Control System Design.....	10
2.3.1 Procedure in Designing Fuzzy Control System .....	11
CHAPTER 3: METHODOLOGY / PROJECT WORK.....	13
3.1 Procedure Identification.....	13
3.2 Tools required .....	14
CHAPTER 4: RESULTS AND DISCUSSION.....	15
4.1 Results and Findings .....	15
4.1.1 Membership Functions for Fuzzy air conditioner.....	16
4.1.2 Inference system of the Fuzzy air conditioner .....	23
4.1.3 VHDL coding of the Fuzzy control system .....	25
4.1.4 MATLAB Simulations.....	27
4.2 Discussions .....	28
4.2.1 Fuzzy air conditioner and its membership functions .....	28
4.2.2 Comparison between Mamdani and Sugeno Method .....	30
4.2.2 Fuzzy Inference Proposition Rules and Surface Plot.....	31
4.2.3 VHDL programming and MATLAB simulation .....	32

CHAPTER 5: CONCLUSION AND RECOMMENDATIONS .....	33
5.1 Conclusion .....	33
5.2 Recommendation .....	35
REFERENCES .....	36
APPENDICES .....	37

## LIST OF FIGURES

Figure 1: Fuzzy Model.....	5
Figure 2: Fuzzy Rules Model.....	6
Figure 3: Membership function of car speed.....	8
Figure 4: Typical process control system.....	9
Figure 5: Fuzzy air conditioner.....	15
Figure 6: Membership function for room temperature.....	16
Figure 7: Membership function for room humidity.....	17
Figure 8: Membership function for delta temperature increase/decrease.....	18
Figure 9: Membership function for air flow motor.....	19
Figure 10: Membership function for room temperature.....	20
Figure 11: Membership function for room humidity.....	21
Figure 12: Delta Temperature Increase.....	22
Figure 13: Air Flow Motor.....	23
Figure 14: Surface plot of the inputs against delta temperature increase.....	24
Figure 15: Surface plot of the inputs against air flow motor.....	24
Figure 16: A part of Fuzzification VHDL code.....	25
Figure 17: A part of Inference VHDL code.....	26
Figure 18: A part of Defuzzification VHDL code.....	26
Figure 19: A part of Fuzzification MATLAB code.....	27
Figure 20: A part of Inference MATLAB code.....	27
Figure 21: A part of Defuzzification MATLAB code.....	28
Figure 22: Example of non-linear relation between inputs and outputs.....	31
Figure 23: Input membership function - Room temperature.....	37
Figure 24: Input membership function – Room humidity.....	37
Figure 25: Output membership function - Delta temperature increase.....	38
Figure 26: Output membership function – Air flow motor.....	38



## LIST OF APPENDICES

Appendix A	: FIS Simulations.....	37
Appendix B	: Detail Rule Base Model.....	39
Appendix C	: Coding for Fuzzification in VHDL.....	40
Appendix D	: Test bench code for Fuzzification.....	43
Appendix E	: Coding for Rule Base – Inference in VHDL .....	45
Appendix F	: Test bench code for Inference.....	50
Appendix G	: Coding for Defuzzification in VHDL.....	52
Appendix H	: Test bench code for Defuzzification.....	54
Appendix I	: Matlab’s M-file for Fuzzification .....	56
Appendix J	: Matlab’s M-file for Inference .....	58
Appendix K	: Matlab’s M-file for Defuzzification.....	60
Appendix L	: Matlab Results – Fuzzification .....	62
Appendix M	: Matlab Results – Inference .....	63
Appendix N	: Matlab Results – Defuzzification .....	64
Appendix O	: Sugeno method’s sample calculation.....	65

# CHAPTER 1

## INTRODUCTION

### 1.1 Background of Study

Fuzzy logic is used in embedded control and information processing application. Fuzzy provides a simple way to draw definite conclusions from vague, ambiguous or imprecise information [9]. Fuzzy logic resembles human decision making with its ability to work from approximate data and find precise solutions.

Unlike classical logic and conventional control system which requires a deep understanding of a system, exact equations, and precise numeric values, Fuzzy logic provides an alternative way of thinking, which allows modeling complex systems using a higher level of knowledge and experience manipulation [4].

Fuzzy Logic allows expressing this knowledge with subjective, descriptive and imprecise concepts. Some examples are that expressions of very hot, hot, not so hot, warm, chill, cool and cold. Fuzzy Logic has been found to be very suitable for embedded control applications. Fuzzy technology is used to improve quality and reduce development time.

## 1.2 Problem Statement

The conventional control systems that have been implemented are depends on precise range of input [6]. This has caused the conventional control system to be less robust or less flexible. Problem solving using the conventional method solves most of the problems but not all, especially the problems that requires human reasoning. The presence of human during problem solving activities might not be possible all the time as well since the situation would not be favorable for human presence. The problem solving should be solved by only the presence of machine or its equivalent.

Fuzzy logic is based on a simple rule-based IF –THEN [4] proposition statement approach in solving control problem. The Fuzzy logic model is empirically-based, relying on a person’s experience rather than his/her technical understanding of the system. Fuzzy logic utilizes imprecise yet very descriptive set of inputs such as cool, cold, very cold and extremely cold to describe what is actually happening. Fuzzy logic capability in mimicking the human behavior at fast rate enables fuzzy control system to be more robust and reliable.

## 1.3 Objectives

Two main objectives were set to be achieved by the author at the end of the project. The objectives of the project are as follow:

- To design a Fuzzy control system using Fuzzy logic concept for an air conditioning system, or to design a Fuzzy air-conditioner that would able to adjust its operation based on the room temperature and room humidity.
- To implement the design of air conditioner Fuzzy control system using VHDL programming language. VHDL is chosen for the implementation due to the reason that this hardware description language is used more often nowadays in the industry and the availability of the reference resources.

## **1.4 Scope of Study**

The author was able to integrate the theoretical knowledge that has been learned throughout the degree programme with practical approach throughout the project. This project also helped the author to develop his knowledge in new type of control system application, Fuzzy control system which is not taught through the academic syllabus of the programme. Design and implementation stage of this project gave the author a brief view of how integrated chips are designed and implemented at real life situation. The scope of this project included process of learning about the related concept, familiaring with VHDL programming, designing the Fuzzy control system, finalizing the Fuzzy control system design (Fuzzy air conditioner), developing the Fuzzy control system design, implementation of the design and project presentation.

## **CHAPTER 2**

### **LITERATURE REVIEW AND THEORY**

#### **2.1 Fuzzy Logic Concept**

The concept of Fuzzy Logic was introduced by Lotfi Zadeh, a professor at the University of California at Berkeley, and presented as a way of processing data by allowing partial set membership rather than crisp set membership or non-membership [9]. This approach to set theory was not applied to control systems until the 70's due to insufficient small-computer capability prior to that time. Fuzzy logic concept reasoning is that people do not require precise numerical information based inputs, and yet they are capable of highly adaptive control and decision making. If feedback controllers could be programmed to accept noisy, imprecise input, the controllers would be much more effective and perhaps easier to implement.

Fuzzy logic is a problem-solving control system methodology that can be implemented in different kinds of system. It can be implemented in hardware, software, or a combination of both. Fuzzy logic provides a simple way to arrive at a definite conclusion based upon vague, ambiguous, imprecise, noisy, or missing input information. Fuzzy logic is based on a rule-based IF - THEN approach in solving control problem rather than attempting to model a system mathematically. Fuzzy logic's approach is to control problems by mimicking how a human would make decisions, but in a faster rate.

### 2.1.1 Fuzzy Model

Fuzzy model [4] consists of four main elements mainly the inputs, Fuzzification, Defuzzification and the outputs. The inputs and the outputs are the physical devices or results that can be seen and desired by user. The Fuzzification and the Defuzzification is the process performed with the aid of Fuzzy logic and Process logic to relate the inputs to the outputs. Fuzzification process is performed by the Fuzzifier, where crisp inputs are converted into Fuzzy based input representations. Defuzzification is the process performed by the Defuzzifier, where the results of the Fuzzy process logic outputs are converted into crisps values, or desired actions or real time outputs. The exact representation of a Fuzzy Model is shown in Figure 1.

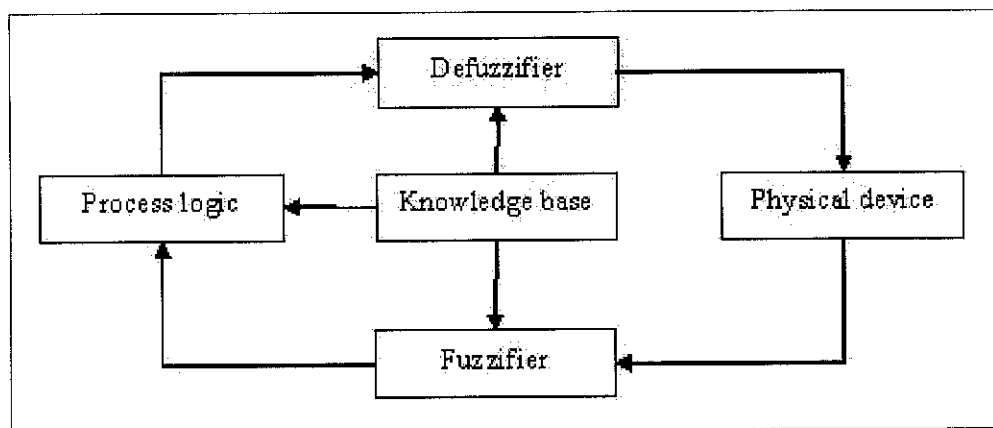


Figure 1: Fuzzy Model

The combination of the Process Logic and the Knowledge Base components is known as the Inference system of a Fuzzy system. The whole Fuzzy system is generally considered as consists of three steps; mainly Fuzzification, Inference and Defuzzification. Fuzzification and Defuzzification are two steps that are complementary to each other while the Inference is the process where the decision making actually takes place. The Inference components differ from one Fuzzy system to another because the architecture depends on the designer of the system.

### 2.1.2 Fuzzy Rule Generation

The Fuzzy rule [4] is basically based on the simple IF – THEN rule which relates the input to the output through combinational logic relationship. The rules are usually known as the proposition statements in layman’s term. An example of the Fuzzy rules is shown below where a Fuzzy system with two inputs, A and B, each having two different situations of 1 and 2. The inputs can be combined through digital logic operations such as the AND or OR operators to produce necessary output conditions.

**IF A1 and/or B1 THEN H11 ELSE**  
**IF A2 and/or B1 THEN H21 ELSE**  
**IF A1 and/or B2 THEN H12 ELSE**  
**IF A2 and/or B2 THEN H22.**

The IF-THEN rules, when tabulated in a table similar to what shown below, the obtained table, known as the Fuzzy Rules Model [4], shows much more clearly how a Fuzzy Inference system actually works.

A1	H11	H12
A2	H21	H22
	B1	B2

**Figure 2: Fuzzy Rules Model**

### 2.1.3 Membership Functions

Membership Functions [4] profiles are the plots of inputs to the respective degree of truth. The degree of truth is a unitless value that defines how true the condition is. For example, when a person with height of 170cm is considered as tall, the degree of truth for that person is tall is 1.0. However, when another person with the height of 165cm, the degree of truth for that person is tall is less than 1.0, maybe around 0.8. Usually, when the latter condition occurs in a Fuzzy system, it is defined two

linguistic variables. Linguistic variables are the variables used to define a value or a condition with a set of range. Examples of linguistic variables for a category are Tall and Short. Therefore, when the person with the height of 165cm is defined in term Fuzzy system, it would be as 0.8 of Tall and 0.2 of Short. A membership function usually consist more than two linguistic variables and each input and the output of a Fuzzy system have its own membership function. The concepts of linguistic variable and the membership function can be further explained using the example below where the speed of a car is discussed.

Assuming the speed of a car can be categorized by three linguistic variables of slow, moderate and fast. These categories are represented by A, B and C respectively.

A = slow

B = moderate

C = fast

Assuming that the car could travel from the range of speed of 0 to 100km/h and range of each linguistic variable are defined as below.

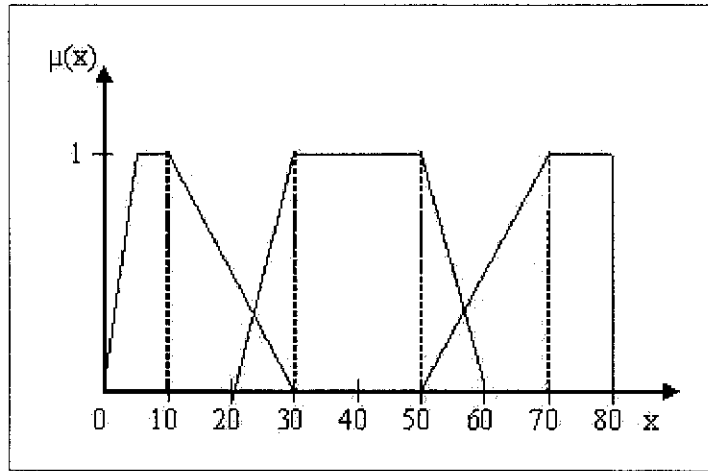
$$A = \{(0, 0), (5, 1), (10, 1), (30, 0)\}$$

$$B = \{(20, 0), (30, 1), (40, 1), (50, 1), (60, 0)\}$$

$$C = \{(50, 0), (70, 1), (80, 1)\}$$

The shape of the membership function for the car speed would be as shown below where the symbol  $\mu(x)$  represents the degree of truth for each linguistic variable at different speed.





**Figure 3: Membership function of car speed**

Membership functions are the most important aspects of a Fuzzy system because the proposition statements are based on the membership functions and the required in Fuzzification, Inference and Defuzzification steps.

## **2.2 Control System Design**

Control systems [6] are basically known as the systems used to maintain the controlled variables at their required value. Typical control system contains following main elements: process, sensor, controller, final controlling element. The typical block diagram of a process that contains any control system is as shown In Figure 4.

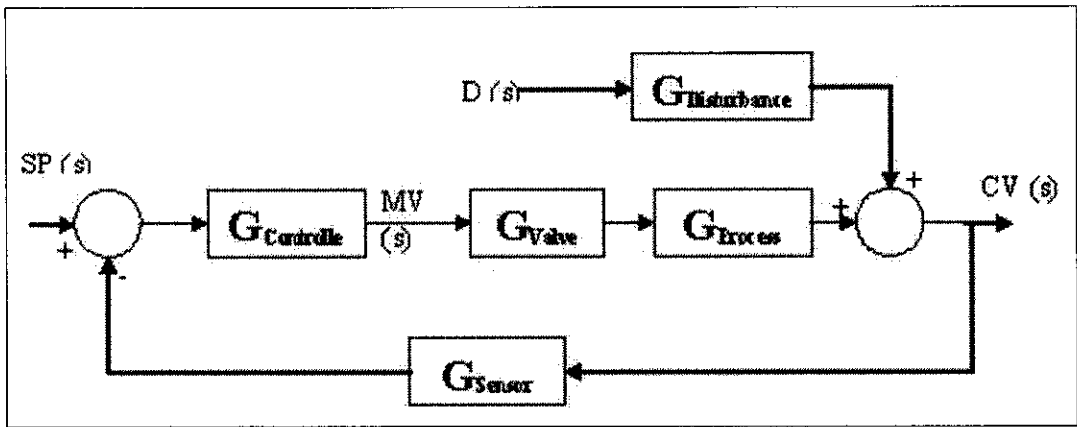


Figure 4: Typical process control system

In the block diagram above, the Controlled Variable (CV) [6] is any process variable that needs to be controlled at desired value or range of values. The value or the range of values that are specified for the controlled variable to achieve is known as the Set Point (SP). The difference between Set Point and the Controlled Variable is known as the Error (E) or offset. The main objective of any control system is to maintain the particular system with zero value of error. To do so, the controller will get the error value and perform some calculations to compensate the error. The output of the controller, which is known as the Manipulated Variable (MV), will be used to control the final controlling element, where the final controlling element controls some variables that have causal relationship with the Controlled Variable. The controller will compensate the error by changes the value of the final controlling elements so that the error is reduced or eliminated. The disturbance variable (D) is any variable that cannot be controlled but influences the output of process. Disturbances tend to introduce error. Therefore, the controller must be able to compensate the error introduced by the disturbance as well.

The control system above is specifically known as the Feedback Control System [6]. Feedback control system uses the output to maintain the controlled variable at its desired value. There are other types of control system or strategy, such as Cascade Control System, Feed Forward Control System, Feed Back -Feed Forward Control System, Ratio Control System and etc. However, the main objective of mentioned

control systems is to maintain the controlled variables at their desired value in spite of disturbance occurrence.

### **2.3 Fuzzy Control System Design**

Fuzzy control system (FCS) [5] is the control system that utilizes the Fuzzy logic reasoning concept. Fuzzy control system is assumed to be more robust and flexible since it has the characteristic offered by the Fuzzy logic concept. Fuzzy control system uses the available membership function of its inputs and its outputs to make the decision. Unlike conventional control system which usually uses the PID system and produces almost the same results if compared to another conventional control system, a Fuzzy control system would not produce the same results compared to another Fuzzy control system due to difference in the membership functions and the IF-THEN rules design of the Fuzzy control system.

An example of Fuzzy control system is the Fuzzy washing machine [9]. Fuzzy washing machine's inputs usually are the level of dirtiness, number of cloth and the water level. The outputs of the Fuzzy washing machine would be amount of detergent needed and the washing time. Assuming in ideal case of one cloth with 100% level of dirtiness and 100% water level, the Fuzzy control system will then set the detergent amount as 50 grams and washing time as 5 minutes. In another case of where the level of dirtiness is less than 100%, the Fuzzy control system might set different amount of detergent amount and washing time.

### **2.3.1 General Procedure in Designing Fuzzy Control System**

There are five main steps involved in designing a Fuzzy Control System [4]. The steps are listed below in sequence.

- i) Define Fuzzy problem in detail.
- ii) Identify all important variable and their ranges.
- iii) Determine membership profiles for each variable ranges.
- iv) Determine rules (propositional statements) including action needed.
- v) Select Defuzzification methodology.

#### ***Define Fuzzy problem in detail.***

First step of any problem solving is the definition of the problem. Similar in this procedure, the problem statement is defined and understood. The scope of the problem is analyzed for feasibility. The objective of the problem solution is defined. The important requirements such as the inputs and the outputs are noticed.

#### ***Identify all important variable and their ranges.***

Once the inputs and the outputs are located, their values or their range of values are set or identified according to the problem statement or scope of the problem solution.

#### ***Determine membership profiles for each variable ranges.***

Once the inputs and outputs ranges have been identified, the membership functions for each variable (input and output) is defined or determined. The membership function determines the degree of truth of each variable at different conditions.

***Determine rules (propositional statements) including action needed.***

The Fuzzy rules model which consists of IF-THEN statements must be generated. The IF-THEN statements are based on their respective membership functions. The IF-THEN statements related the input membership functions to the output membership functions.

***Select Defuzzification methodology.***

The final step of designing the Fuzzy control system is the Defuzzification where the output statement (THEN) is directly mapped to some kind of physical representation or action. The step is also refers to the actual Fuzzy control system output realization.

## **CHAPTER 3**

### **METHODOLOGY / PROJECT WORK**

#### **3.1 Procedure Identification**

At the initial stage of the project which covers the entire semester one period, literature review was carried out to familiarize with the Fuzzy logic concept. Once the intended concept has been fully understood, the literature review was then proceeded to focus more on the application of Fuzzy logic in designing and implementation the required Fuzzy control system. The implementation process is carried out through digital logic design, where digital components such as the shift registers, carry-select adders and min/max combinational logic was also studied so that the exact characteristics of the digital design of the Fuzzy control system could be understood during the implementation. Since the project associated with two inputs; room temperature and the room humidity, a literature review to understand the correlation between the two inputs was also made so that the author could just use only one input device to get two different types of inputs. The design implementation required VHDL programming to construct the working code for the final Fuzzy control system.

### 3.2 Tools required

The project involves VHDL Programming to design and implement the Fuzzy controller.

Due to this, the following tools have been used in performing all the related activities of the project:

- ALDEC's Active-HDL Verilog, VHDL and EDIF Simulation software – This software was used to write and compile the VHDL programs that were used to realize the Fuzzy control system. VHDL was utilized to design, write and simulate the logic design of the Fuzzy control system. Once the programs have been constructed, compiled and verified, the programs are tested using the test bench program codes to simulate the intended output of the programs.
- MATLAB – This software was used as an alternative to the ALDEC's Active-HDL Verilog, VHDL and EDIF Simulation software. The Fuzzy control system component programs were constructed tested and simulated using this software to validate the design. However, the programs designed using this software could not be used for the implementation of the Fuzzy control system on the intended hardware.
- Fuzzy Logic Toolbox of MATLAB (Fuzzy Inference System –FIS Editor Viewer) – This toolbox of MATLAB software was used specifically to assist in the Fuzzy Inference system of the Fuzzy control system. Though the toolbox is specifically for the Fuzzy Inference system design, other component designs of the Fuzzy control system are also possible, with limited applications.
- FuzzyTECH 5.5 – This software was used to automatically generate the proposition statements based on the membership functions of the Fuzzy control system.
- Xilinx's Project Navigator – An important software that was used to synthesize and to generate the net list files from the created VHDL programs before the design could be downloaded to the FPGA.

## CHAPTER 4

### RESULTS AND DISCUSSION

#### 4.1 Results and Findings

When the Fuzzy control system design was finalized to a Fuzzy air conditioning system, two inputs and two outputs were defined. The inputs are the room temperature and room humidity. The outputs are the delta temperature increase required to be made by the control system and the air flow motor, or the fan speed. In this project, it is assumed that the room humidity could be controlled by the rate of air flow to the room as evaporation increases with high air movement and decreases with low air movement. The overall Fuzzy air conditioner would be as illustrated by Figure 5.

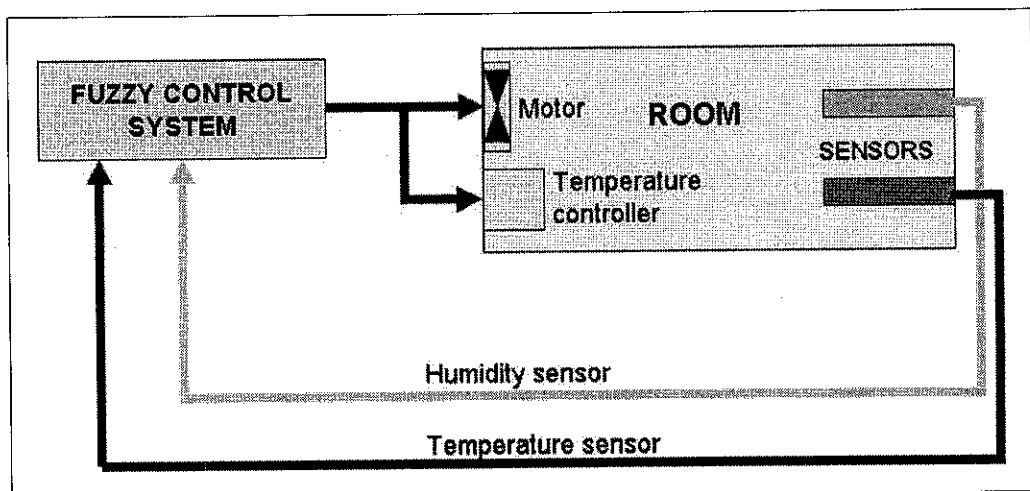


Figure 5: Fuzzy air conditioner



#### 4.1.1 Membership Functions for Fuzzy air conditioner

There are four membership functions for the Fuzzy air conditioner. Two of the membership functions are for the inputs; room temperature and the room humidity, while the two are for the outputs; air flow and the delta temperature increase. The membership functions are shown Figure 6.

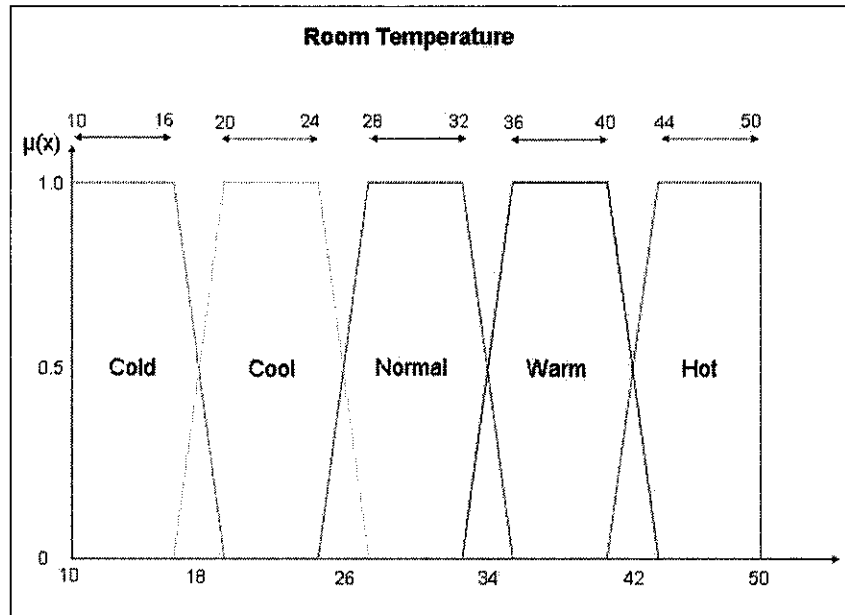


Figure 6: Membership function for room temperature

The membership function for the room temperature is defined for a temperature range of from 10<sup>0</sup>C to 50<sup>0</sup>C. Five linguistic variables are used in the membership function to defined five different ranges of room temperature. The linguistic variables are cold, cool, normal, warm and hot. The temperature range for each linguistic variable is shown below.

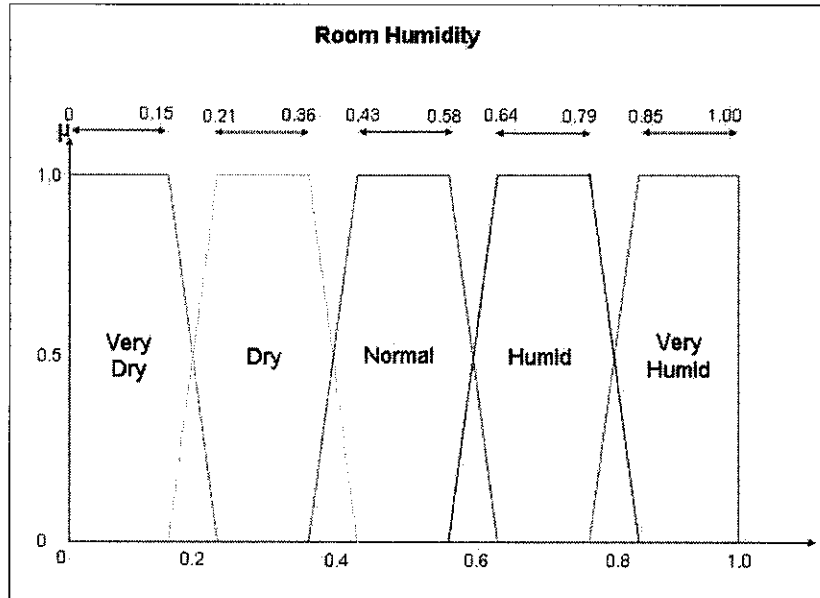
$$\text{Cold} = \{(10, 1), (16, 1), (20, 0)\}$$

$$\text{Cool} = \{(16, 0), (20, 1), (24, 1), (28, 0)\}$$

$$\text{Normal} = \{(24, 0), (28, 1), (32, 1), (36, 0)\}$$

$$\text{Warm} = \{(32, 0), (36, 1), (40, 1), (44, 0)\}$$

$$\text{Hot} = \{(40, 1), (44, 1), (50, 1)\}$$



**Figure 7: Membership function for room humidity**

The membership function for the room humidity is defined for a range of 0 to 1.0 of scale. Five linguistic variables are used in the membership function to defined five different ranges of room humidity. The most comfortable room humidity is assumed at 0.5 of scale. The linguistic variables used are very dry, dry, normal, humid and very humid. The range for each linguistic variable is shown below.

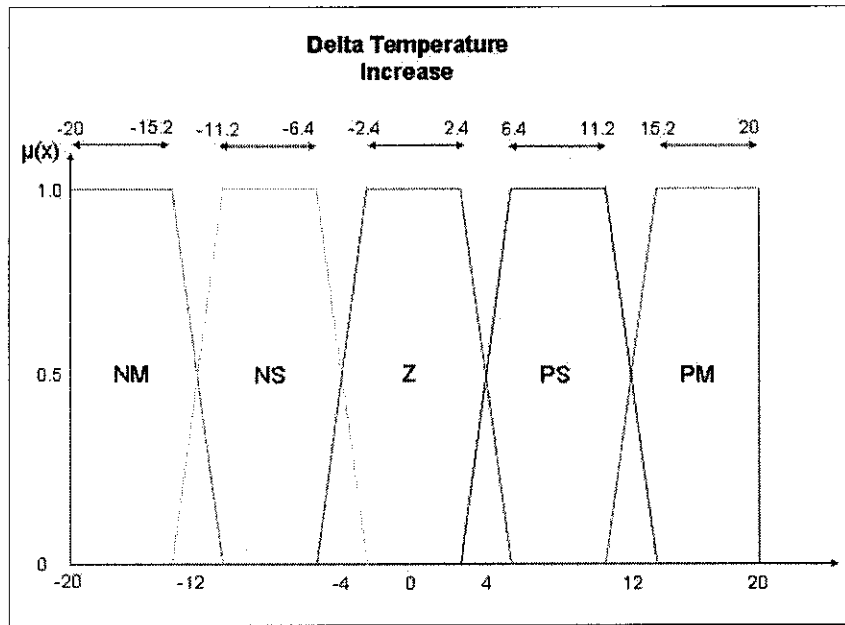
$$\text{Very dry} = \{(0, 1), (0.15, 1), (0.21, 0)\}$$

$$\text{Dry} = \{(0.15, 0), (0.21, 1), (0.36, 1), (0.43, 0)\}$$

$$\text{Normal} = \{(0.36, 0), (0.43, 1), (0.58, 1), (0.64, 0)\}$$

$$\text{Humid} = \{(0.58, 0), (0.64, 1), (0.79, 1), (0.85, 0)\}$$

$$\text{Very humid} = \{(0.79, 1), (0.85, 1), (1.0, 1)\}$$



**Figure 8: Membership function for delta temperature increase/decrease**

The membership function for the delta temperature increase is defined for a temperature range of from  $-20^{\circ}\text{C}$  to  $20^{\circ}\text{C}$ . Five linguistic variables are used in the membership function to defined five different ranges of delta temperature increase. The linguistic variables are NM (negative medium), NS (negative small), Z (zero), PS (positive small) and PM (positive medium). The membership function for variable differs from the input variables in term of their range where in this membership function, the range consist of both negative and positive values. The main purpose of this type of range is to enable proper corrective action that would lead to a stable condition, which is a principle of Bounded Input, Bounded Output (BIBO). The temperature range for each linguistic variable is shown below.

$$\text{NM} = \{(-20, 1), (-15.2, 1), (-11.2, 0)\}$$

$$\text{NS} = \{(-15.2, 0), (-11.2, 1), (-6.4, 1), (-2.4, 0)\}$$

$$\text{Z} = \{(-6.4, 0), (-2.4, 1), (2.4, 1), (6.4, 0)\}$$

$$\text{PS} = \{(2.4, 0), (6.4, 1), (11.2, 1), (15.2, 0)\}$$

$$\text{PM} = \{(11.2, 1), (15.2, 1), (20, 1)\}$$

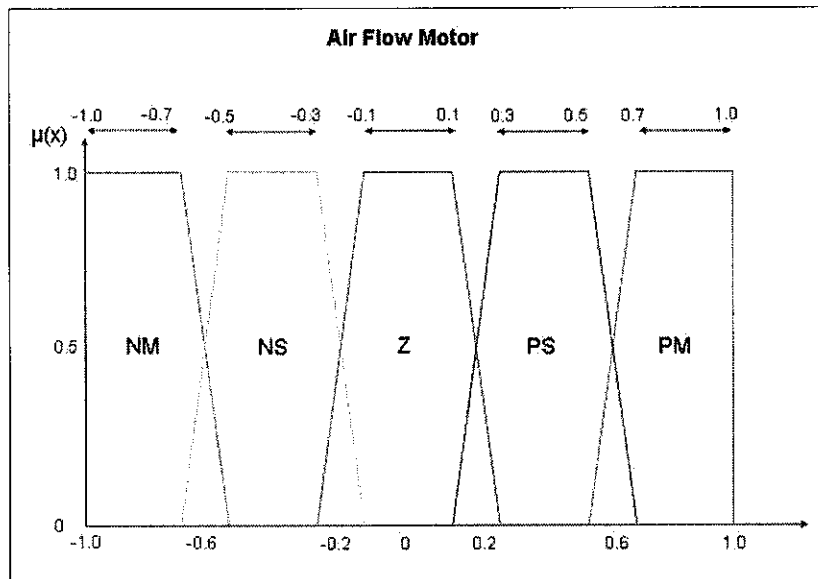


Figure 9: Membership function for air flow motor

The membership function for the air flow motor is defined for a range of from -1.0 to 1.0 of scale. Five linguistic variables are used in the membership function to defined five different ranges of motor speed or air flow. The linguistic variables are NM (negative medium), NS (negative small), Z (zero), PS (positive small) and PM (positive medium). The membership function for this variable similar to that of delta temperature increase in term of their range where the range consists of both negative and positive values. The main purpose of this type of range is to enable proper corrective action that would lead to a stable condition. The relative action range for each linguistic variable is shown below.

$$NM = \{(-1.0, 1), (-0.7, 1), (-0.5, 0)\}$$

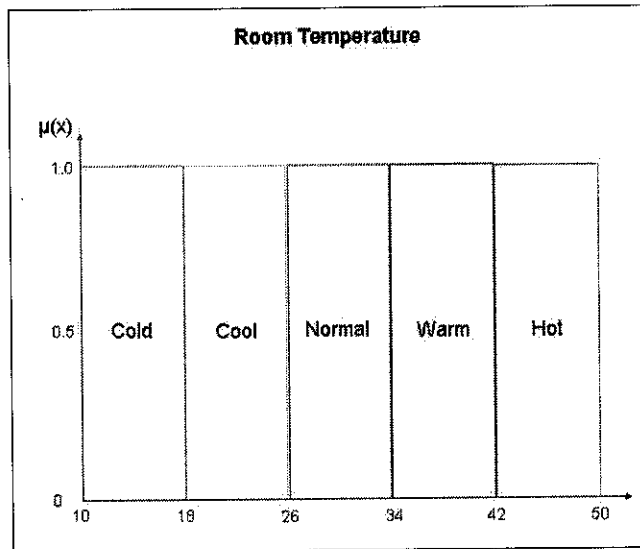
$$NS = \{(-0.7, 0), (-0.5, 1), (-0.3, 1), (-0.1, 0)\}$$

$$Z = \{(-0.3, 0), (-0.1, 1), (0.1, 1), (0.3, 0)\}$$

$$PS = \{(0.1, 0), (0.3, 1), (0.5, 1), (0.7, 0)\}$$

$$PM = \{(0.5, 1), (0.7, 1), (1.0, 1)\}$$

However, to ease the code generation for the Fuzzification, Inference and Defuzzification using VHDL language, the obtained membership functions were simplified from type trapezoidal to type rectangular, as shown in Figure 10.



**Figure 10: Membership function for room temperature**

The temperature range for each linguistic variable is shown below.

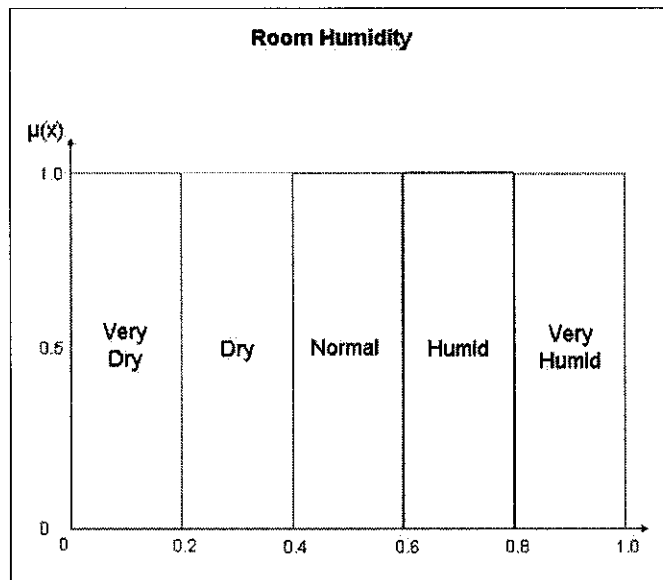
$$\text{Cold} = \{(10, 1), (18, 1), (18, 0)\}$$

$$\text{Cool} = \{(18, 0), (18, 1), (26, 1), (26, 0)\}$$

$$\text{Normal} = \{(26, 0), (26, 1), (34, 1), (34, 0)\}$$

$$\text{Warm} = \{(34, 0), (34, 1), (42, 1), (42, 0)\}$$

$$\text{Hot} = \{(42, 1), (42, 1), (50, 1)\}$$



**Figure 11: Membership function for room humidity**

The range for each linguistic variable is shown below.

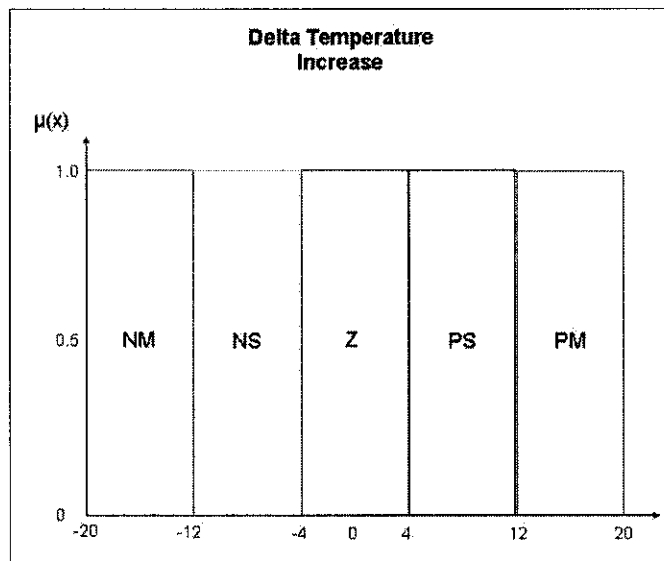
$$\text{Very dry} = \{(0, 1), (0.2, 1), (0.2, 0)\}$$

$$\text{Dry} = \{(0.2, 0), (0.2, 1), (0.4, 1), (0.4, 0)\}$$

$$\text{Normal} = \{(0.4, 0), (0.4, 1), (0.6, 1), (0.6, 0)\}$$

$$\text{Humid} = \{(0.6, 0), (0.6, 1), (0.8, 1), (0.8, 0)\}$$

$$\text{Very humid} = \{(0.8, 1), (0.8, 1), (1.0, 1)\}$$



**Figure 12: Delta Temperature Increase**

The temperature range for each linguistic variable is shown below.

$$NM = \{(-20, 1), (-12, 1), (-12, 0)\}$$

$$NS = \{(-12, 0), (-12, 1), (-4, 1), (-4, 0)\}$$

$$Z = \{(-4, 0), (-4, 1), (4, 1), (4, 0)\}$$

$$PS = \{(4, 0), (4, 1), (12, 1), (12, 0)\}$$

$$PM = \{(12, 1), (12, 1), (20, 1)\}$$

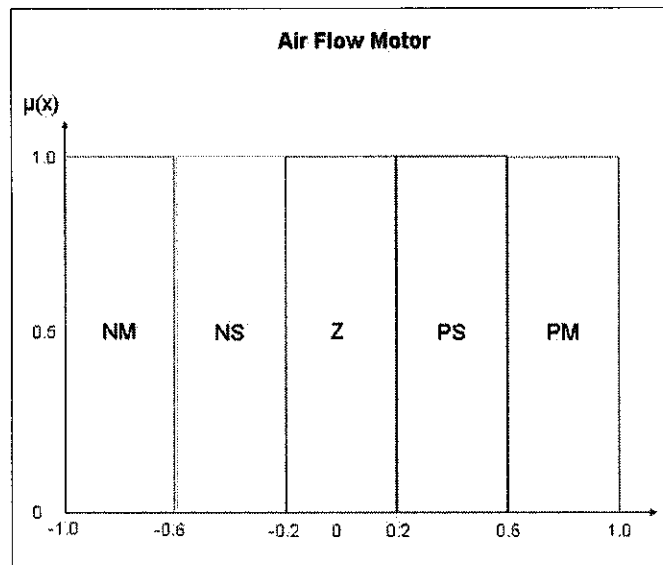


Figure 13: Air Flow Motor

The relative action range for each linguistic variable is shown below.

$$NM = \{(-1.0, 1), (-0.6, 1), (-0.6, 0)\}$$

$$NS = \{(-0.6, 0), (-0.6, 1), (-0.2, 1), (-0.2, 0)\}$$

$$Z = \{(-0.2, 0), (-0.2, 1), (0.2, 1), (0.2, 0)\}$$

$$PS = \{(0.2, 0), (0.2, 1), (0.6, 1), (0.6, 0)\}$$

$$PM = \{(0.6, 1), (0.6, 1), (1.0, 1)\}$$

#### 4.1.2 Inference system of the Fuzzy air conditioner

For generating and designing the Fuzzy control system's Inference system, the Fuzzy Toolbox of MATLAB was utilized. With the Fuzzy Inference System (FIS) Editor Viewer program [7], the following Inference system was developed for the Fuzzy control system. The toolbox requires for the membership function of the system, both inputs and the outputs to be defined. Once this has been done, the proposition statements of the Fuzzy control system that have been decided earlier are used to generate the Inference system. The Inference system eventually used to map the Fuzzy control system's inputs to their respective outputs. Figure 23 till Figure 26 in Appendix A show the FIS simulation process. The Fuzzy Rules model for the Fuzzy air conditioner system is shown in Appendix B. The Inference system is then plotted



using FIS to view the surface plot. The results are shown below where the Inference system of the Fuzzy control system is plotted against each output variable.

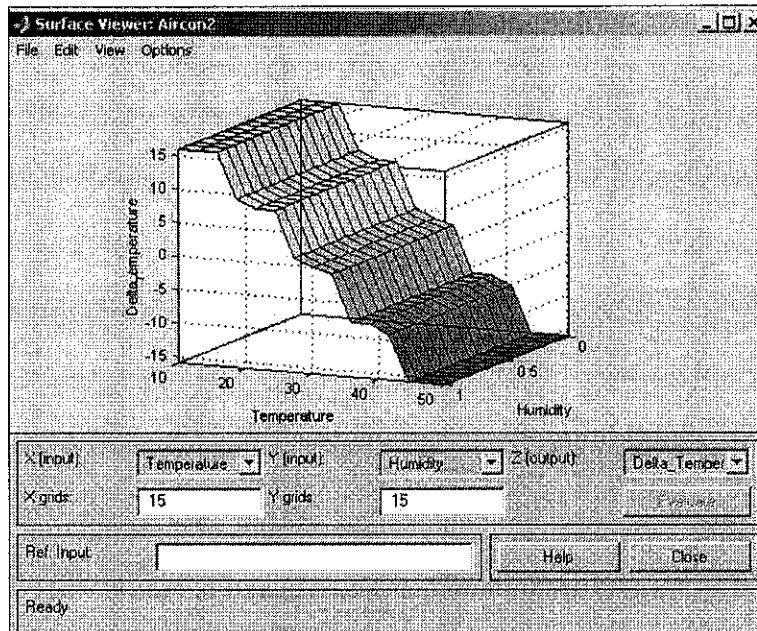


Figure 14: Surface plot of the inputs against delta temperature increase

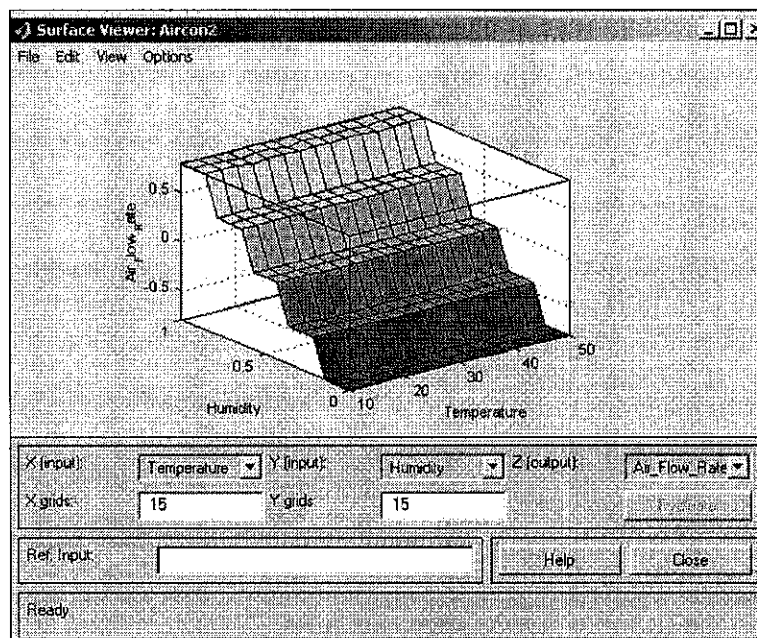


Figure 15: Surface plot of the inputs against air flow motor

### 4.1.3 VHDL coding of the Fuzzy control system

#### *Code Generation for Fuzzification*

Based on the Fuzzification membership functions that the author has obtained, the following VHDL codes were generated. The Fuzzification code is based on the membership functions that have 5 linguistic variables for each of its inputs (two inputs). A part of the coding for the Fuzzification is attached in shown in Figure 16. The full VHDL code and its respective test bench code are attached in Appendix C and D.

```
set-up the fuzzy ranges. The membership function that will be used
is a simple rectangle.

y_pm( fuzzyWidth - 1 downto 0 ) <= membership( fuzzyWidth * 1 - 1 downto fuzzyWidth * 0 );
y_ps( fuzzyWidth - 1 downto 0 ) <= membership( fuzzyWidth * 2 - 1 downto fuzzyWidth * 1 );
y_z ( fuzzyWidth - 1 downto 0 ) <= membership( fuzzyWidth * 3 - 1 downto fuzzyWidth * 2 );
y_ns( fuzzyWidth - 1 downto 0 ) <= membership( fuzzyWidth * 4 - 1 downto fuzzyWidth * 3 );
y_nm( fuzzyWidth - 1 downto 0 ) <= membership( fuzzyWidth * 5 - 1 downto fuzzyWidth * 4 );
```

Figure 16: A part of Fuzzification VHDL code

#### *Code Generation for Inference (Rule Base)*

The component of the Fuzzy control system is where the design IF-THEN proposition statement is utilized. Rules are set by the designer so that the control system would able to map the inputs to their respective outputs, depending on the conditions and requirements. A part of the coding for the Inference and is attached in shown in Figure 17. The full VHDL code its respective test bench code is attached in Appendix E and F.

```

if difference = pmedium and integral = plarge then
    internal_control_pm <= nmedium;
elsif difference = pmedium and integral = pmedium then
    internal_control_pm <= zero;
elsif difference = pmedium and integral = psmall then
    internal_control_pm <= psmall;
elsif difference = pmedium and integral = zero then
    internal_control_pm <= pmedium;
elsif difference = pmedium and integral = nsmall then
    internal_control_pm <= pmedium;

```

Figure 17: A part of Inference VHDL code

### *Code Generation for Defuzzification*

The Defuzzification code that has been generated uses the inferred values from the Inference rule bases to produce the Defuzzification output. The output of the code generation for the following code is in such that the only one output, which is the motor speed (flow). A part of the coding for the Defuzzification is attached in shown in Figure 18. The full VHDL code and its respective test bench code are attached in Appendix G and H.

```

if fuzzy_output = pmedium then
    save_output := defuzzify_pm;
elsif fuzzy_output = psmall then
    save_output := defuzzify_ps;
elsif fuzzy_output = zero then
    save_output := defuzzify_z;
elsif fuzzy_output = nsmall then
    save_output := defuzzify_ns;
elsif fuzzy_output = nmedium then
    save_output := defuzzify_nm;

```

Figure 18: A part of Defuzzification VHDL code

#### 4.1.4 MATLAB Simulations

Due to compilation difficulties in the ALDEC's Active HDL software, the intended result from the VHDL program could not be obtained. To produce the exactly same type of results, the MATLAB software was utilized. The software was used write all three components of Fuzzy control system. The MATLAB M-file coding for the Fuzzification, Inference and the Defuzzification programs are shown in Appendix I, J and K. The MATLAB simulation results are shown in Appendix L, M and N. A partial program containing the main interface of the M-files are shown in Figure 19 to Figure 21.

```
function [] = fuzzy (Temp,Hum)

Temp = input('Enter room temperature value = ');
Hum = input('Enter room humidity value = ');

if ((Temp>=10)&(Temp<=16))
    Temp_LV1='cold'
    Temp_tv1=1
elseif ((Temp>=20)&(Temp<=24))
    Temp_LV1='cool'
    Temp_tv1=1
```

Figure 19: A part of Fuzzification MATLAB code

```
Temp_LV1 = input('Enter room temperature linguistic variable = ','s');
Temp_LV2 = input('Enter room temperature linguistic variable = ','s');
Hum_LV1 = input('Enter room humidity linguistic variable = ','s');
Hum_LV2 = input('Enter room humidity linguistic variable = ','s');

if (Temp_LV1=='NH')
    Delta_temp_LV1='PH'
elseif (Temp_LV1=='NS')
    Delta_temp_LV1='PS'
elseif (Temp_LV1=='Z')
    Delta_temp_LV1='Z'
elseif (Temp_LV1=='PS')
    Delta_temp_LV1='NS'
else (Temp_LV1=='PH')
    Delta_temp_LV1='NH'
end
```

Figure 20: A part of Inference MATLAB code

```

Temp = input('Enter room temperature value = ');
Hum = input('Enter room humidity value = ');
Temp_LV1 = input('Enter room temperature linguistic variable = ','s');
Temp_LV2 = input('Enter room temperature linguistic variable = ','s');
Hum_LV1 = input('Enter room humidity linguistic variable = ','s');
Hum_LV2 = input('Enter room humidity linguistic variable = ','s');
Delta_temp_LV1 = input('Enter delta room temperature linguistic variable = ','s');
Delta_temp_LV2 = input('Enter delta room temperature linguistic variable = ','s');
Airflow_LV1 = input('Enter room air flow linguistic variable = ','s');
Airflow_LV2 = input('Enter room air flow linguistic variable = ','s');
Temp_tv1 = input('Enter room temperature truth value = ');
Temp_tv2 = input('Enter room temperature truth value = ');
Hum_tv1 = input('Enter room humidity truth value = ');
Hum_tv2 = input('Enter room humidity truth value = ');

if ((Delta_temp_LV1=='NH') & (Temp_tv1==1))
    c=(Temp-10)*(1/6);
    output_Delta_temp=(-20)+(c*4.8)

```

Figure 21: A part of Defuzzification MATLAB code

## 4.2 Discussions

A Fuzzy control system with two inputs and two outputs were designed. How this two inputs and the two outputs are related through the membership functions that have been defined, is discussed in following section. Other than that, the Mamdani's method [7] of Inference used for the design is also explained. As a mean of comparison, another method of Inference known as the Sugeno method [7] is also discussed to clarify the different between these two methods.

### 4.2.1 Fuzzy air conditioner and its membership functions

The Fuzzy control system designed is a Fuzzy air conditioner which takes the room temperature and the room humidity as the inputs to the system. The control system will then try to maintain the room condition such that the room temperature and room humidity would be at a comfortable level. In order to do so, the control system will make use two parameters; the delta temperature increase and the air flow motor. The room temperature is increased or decreased accordingly using the delta temperature increase parameter while the room humidity would be controlled by air flow to the room by the fan speed.

Studies were made on finding the correlation between the temperature and the humidity to simplify the hardware design of the Fuzzy control system. Studies that have been conducted showed that there is some correlation between temperature and

relative humidity (RH). This correlation can be made through a chart, known as the psychrometric [10] chart. A psychrometric chart graphically illustrates the relationships between air temperature and relative humidity as well as other properties. However, in this project, the correlation is assumed to be unity and linear along the temperature range specified for Fuzzy control system. The desired or most comfortable humidity level for human is around 45% RH [11]. In the Fuzzy control design, this level is assumed to be represented by the 0.5 of scale. The maximum and the minimum humidity that the air flow motor would be able to control are assumed around  $\pm 10\%$  RH.

Two types of membership functions have been utilized in the project; type trapezoidal and type rectangular. Membership function type trapezoidal has been seen and used commonly for the most of the Fuzzy application. This type of membership function defines the transition between two adjacent linguistic variables smoothly without abrupt changes. Therefore, Fuzzy calculation can be assumed to be more accurate and the results resemble more to human reasoning. Membership function type rectangular that has been used as an alternative to simplify the VHDL code generation is not widely used to the reason that there is no smooth transition between the two adjacent linguistic variables in the membership functions. However, due to its simplicity, the code generation of the component is less complicated than the other types of membership functions. Though the membership function type rectangular has been used for the VHDL code generation, for the MATLAB simulations, membership function type rectangular was used to simulate Fuzzy decision making.

There are few types of Fuzzy Inference calculation methods. Two most commonly used calculation methods are the Mamdani method [7], which was introduced by Ebrahim Mamdani, and Sugeno method [7]. In this project, a method similar to Mamdani was used because the method is simpler to be executed or designed for the programming activities. The details of the both Inference methods are explained in the following subsection. Though the Fuzzy Inference is process of formulating the mapping from a given input to an output using Fuzzy logic, these two types of Inference systems differ in the way outputs are calculated.

## 4.2.2 Comparison between Mamdani and Sugeno Inference Method

The mapping provides a basis from which decisions can be made.

### *Mamdani Method*

Mamdani Fuzzy Inference [7] method is the most commonly seen Fuzzy methodology. Mamdani-type Inference expects the output membership functions to be fuzzy sets. After the aggregation process, there will be Fuzzy set for each output variable that needs to be defuzzified. In many cases, for more efficient Fuzzy Inference, a single spike is used as the output membership functions rather than a distributed fuzzy set. It enhances the efficiency of the Defuzzification process because it greatly simplifies the computation required by the more general Mamdani method, which finds the centroid of a two-dimensional function. Rather than integrating across the two-dimensional function to find the centroid, the weighted average of a few data points are used. Time and cost is reduced by this method since less processing are being done but with accurate results.

### *Sugeno Method*

Sugeno Inference method [7], also known as Takagi-Sugeno-Kang method of Fuzzy Inference It is similar to the Mamdani method in many respects. In fact the first two parts of the fuzzy Inference process, fuzzifying the inputs and applying the Fuzzy operator, are exactly the same. The main difference between Mamdani-type of Fuzzy Inference and Sugeno-type is that the output membership functions are only linear or constant for Sugeno-type fuzzy Inference.

A typical fuzzy rule in a zero-order Sugeno fuzzy model has the form

$$\text{if } x \text{ is } A \text{ and } y \text{ is } B \text{ then } z = k$$

where  $A$  and  $B$  are fuzzy sets in the antecedent, while  $k$  is a crisply defined constant in the consequent. When the output of each rule is a constant, the similarity with Mamdani's method is striking. The only distinctions are the fact that all output membership functions are singleton spikes, and the implication and aggregation methods are fixed and can not be edited. The implication method is simply

multiplication, and the aggregation operator just includes all of the singletons. More details and a sample Inference calculation using the Sugeno method are attached in Appendix O.

#### 4.2.3 Fuzzy Inference Proposition Rules and the Surface Plot

From Appendix B, it can be seen that the two input variables are influencing the output variables linearly. The room temperature is linearly controlling the delta temperature increase and the room humidity uses the air flow motor to change the humidity level. This is illustrated by the generated surface plot shown in Figure 15. The proposition statements are made in such a condition so that the design of the Fuzzy control system would be possible using the VHDL programming and MATLAB simulation. For industrial solutions, type of Fuzzy control system are not linear. The inputs are non-linearly mapped by the outputs. An example of a surface plot of an industrial-based Fuzzy Inference system is Figure 22.

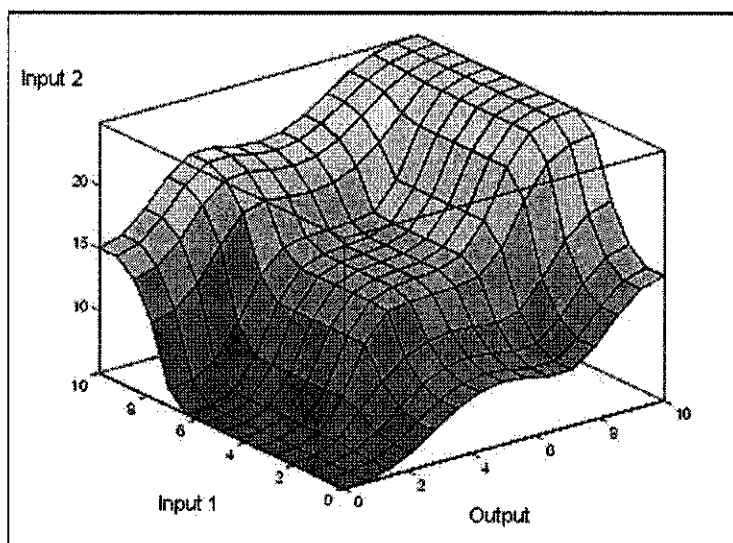


Figure 22: Example of non-linear relation between inputs and outputs

This type of Inference system is considered more 'Fuzzy' and resembles more similar to human reasoning compared to the one generated in the project. The linearity of the Inference system is controlled by the assignment of degree of truth on each proposition statement. In this project, a degree of truth of 1.0 is assigned to every



proposition statements. If these assignments are changed accordingly, the Fuzzy Inference system of this Fuzzy air conditioner would be able to make a better decision in controlling the room temperature and room humidity.

#### **4.2.4 VHDL programming and MATLAB simulation**

The VHDL programs [1], [2] of the Fuzzy control system were developed so that it could be used to realize the system on the hardware (FPGA). Three different components of the Fuzzy control system were coded using VHDL code. Rectangular type membership functions were utilized for ease of code generation. The VHDL code would basically get the inputs, perform Fuzzification on the inputs based on the 5 linguistic variables that have been defined, perform the Inference operation and return the corresponding outputs. The outputs of the Inference system then will be used in the Defuzzification operation to convert the outputs to crisp values. However, due to errors in compilation due to missing libraries of the software, results could not be obtained. The programs were later converted into M-file and simulated using MATLAB to get the results. The results of the MATLAB simulation can be considered more accurate since the programs were developed based on the trapezoidal type membership functions.

## **CHAPTER 5**

### **CONCLUSION AND RECOMMENDATIONS**

#### **5.1 Conclusion**

Fuzzy control system is a system that resembles human reasoning and decision making. The main advantage of the Fuzzy control system is that it can make decision based on the vague and imprecise inputs or data. This allows it to be utilized in areas or applications that require human reasoning and decision making.

Besides that, Fuzzy control system is based on empirical decision making. One does not to fully understand every aspect of any system before implementing the control system. Once a problem is carefully understood and objectives have been defined, the Fuzzy control system can be developed straight away based on the understanding or the required system. No detail knowledge is required of every single aspect of the system since the objectives of a Fuzzy control system is get the desired outputs based on the inputs. This is proven in this project where the author does not have detail knowledge on how an air conditioner works. However, since the objective is to control the room temperature and the room humidity, the Fuzzy control system was developed based on controlling or adjusting the temperature increase and the air flow motor.

Since it does not require detail understanding of a system, the development Fuzzy control system is less time consuming as compared to other conventional control system. There is no necessity to fully understand the exact operation of an air

conditioner. However, it would be the designer's advantage if the exact operation of the system.

In this project of designing and developing a Fuzzy control system, a two inputs and two outputs were specified to the control system. The Fuzzy control system are required to make decision based on the room temperature and the room humidity to control the level of air flow in the room and the amount of temperature that need to be increased. As a result of this project, the required system was able to be designed. The system is able to make the decision based on the specified variables which are mainly utilizing the room temperature. However, much can be done to improve the Fuzzy controller and the way to implement the Fuzzy controller. The recommendations for the Fuzzy controller are mentioned in the following section.

## **5.2 Recommendation**

The Fuzzy control system designed for the air conditioner contains three main steps; Fuzzification, Inference and Defuzzification. In this project, all the codes were generated manually, based on the author's understanding on the Fuzzy control system. There are chances that the generated codes would not function or performed exactly as intended by the designer. To avoid this problem, the author would like to recommend of using the tailored software that are available in the industry to be used since there are industry-certified and the design development process would consume less time. When the design development consume less time as time consumed in this project, the design implementation on the hardware can be considered and possible. The hardware realization would show better and solid results. Besides that, the results from the software utilization, a more accurate and precise results can be expected.

In this project, the correlation between the room temperature and the room humidity was assumed to be linear and unity. However, this would not be the case in the real situation where at different Relative Humidity, the temperature changes or depends non-linear. When in the future, this project is implemented through hardware realization, either two different sensors are to be used or a program to correlate the two inputs should be used. Besides that, other parameters can be included in the future design development so that more accurate of control is possible.

## REFERENCES

- [1] S. Brown and Z. Vranesic, 2000, "FUNDAMENTALS OF DIGITAL LOGIC WITH VHDL DESIGN", McGraw-Hill International Editions
- [2] K.C. Chang, 1999, "Digital System Design with VHDL and Synthesis: An Integrated Approach", IEEE COMPUTER SOCIETY PRESS
- [3] S. Dillen, F. Rashid and T. Chepyha, "Fuzzy Logic Controller in VHDL", WaveRider
- [4] M.R. Kaimal, S Dasgupta, M Harishankar, 1997, "Neuro-Fuzzy Control Systems", Narosa Publishing House
- [5] A. Kandel and G. Langkolz, 1993, "Fuzzy Control System", CRC Press
- [6] T.E. Marlin, 2000, "Process Control: Designing Processes and Control Systems for Dynamic Performance ", McGraw-Hill International Edition
- [7] MATLAB Release 6.1 Help, "Mamdani and Sugeno Inference Method", MathWorks Inc.
- [8] V. Salapura and V. Hamann, "Implementing Fuzzy Control Systems Using and Statecharts", Technische Universita't Wien
- [9] <<http://www.aptronix.com/fide/whatFuzzy.htm>>
- [10] <<http://www.ianrpubs.unl.edu/generalag/g626.htm>>
- [11] <<http://www.science.howstuffworks.com/question651.htm>>

# APPENDICES

## Appendix A: FIS Simulations

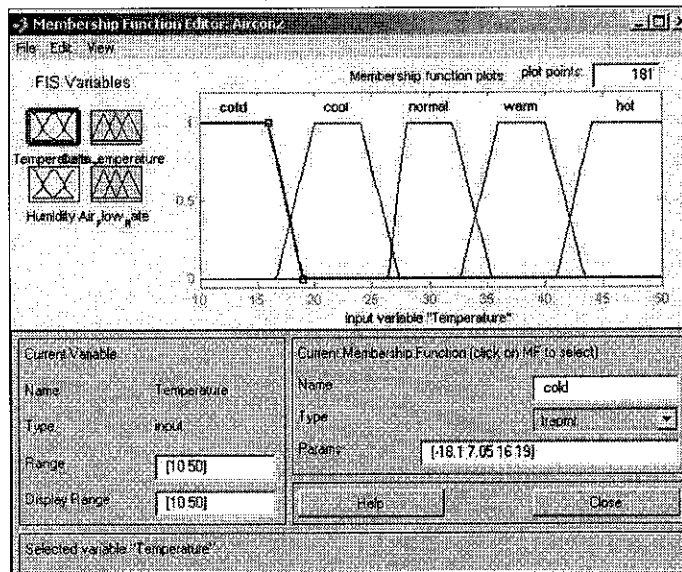


Figure 23: Input membership function - Room temperature

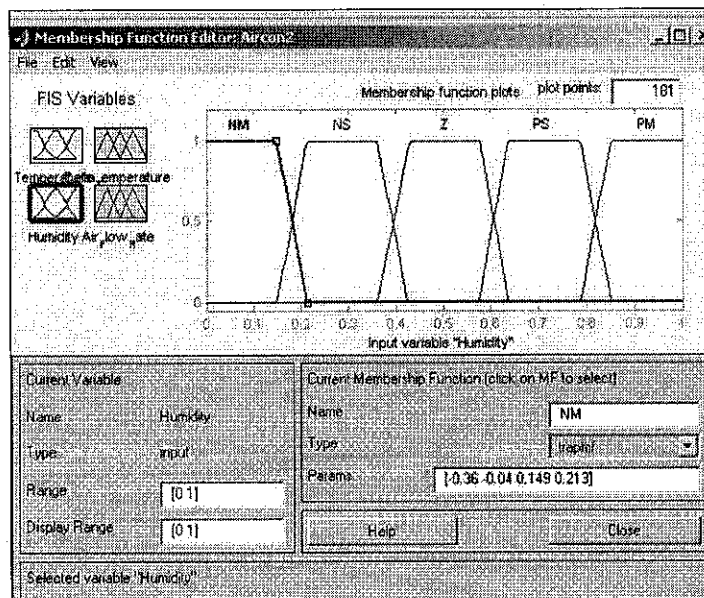


Figure 24: Input membership function – Room humidity

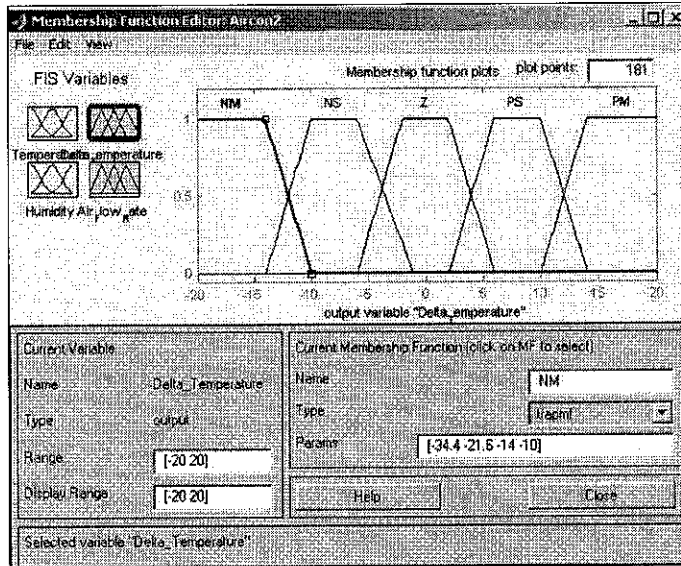


Figure 25: Output membership function - Delta temperature increase

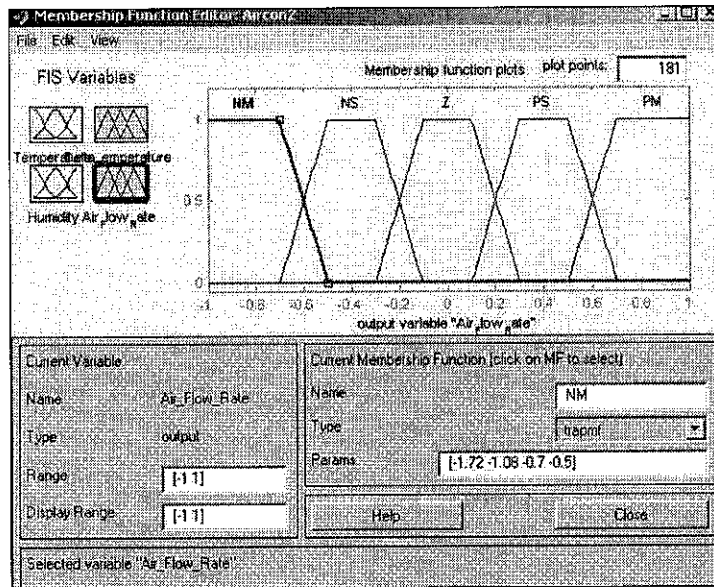


Figure 26: Output membership function – Air flow motor

## Appendix B: Detail Rule Base Model

IF		THEN	
Room Temperature	Room Humidity	Air Flow Motor	Delta Temperature Increase
Cold	Very Dry	PM	PM
Cool	Very Dry	PM	PS
Normal	Very Dry	PM	Z
Warm	Very Dry	PM	NS
Hot	Very Dry	PM	NM
Cold	Dry	PS	PM
Cool	Dry	PS	PS
Normal	Dry	PS	Z
Warm	Dry	PS	NS
Hot	Dry	PS	NM
Cold	Normal	Z	PM
Cool	Normal	Z	PS
Normal	Normal	Z	Z
Warm	Normal	Z	NS
Hot	Normal	Z	NM
Cold	Humid	NS	PM
Cool	Humid	NS	PS
Normal	Humid	NS	Z
Warm	Humid	NS	NS
Hot	Humid	NS	NM
Cold	Very Humid	NM	PM
Cool	Very Humid	NM	PS
Normal	Very Humid	NM	Z
Warm	Very Humid	NM	NS
Hot	Very Humid	NM	NM



## Appendix C: Coding for Fuzzification in VHDL

```
library ieee;
use ieee.std_logic_1164.all;

library work;
use work.config.all;

entity Fuzzification is

    generic ( busWidth : positive := 4;

              fuzzyWidth : positive := 3
            );

    port ( clock : in std_logic;
          reset : in std_logic;
          data : in std_logic_vector( busWidth - 1 downto 0 );
          membership : in std_logic_vector( 5 * fuzzyWidth - 1 downto 0 );
          valid_in : in std_logic;
          valid_out : out std_logic;
          fuzzy_data : out std_logic_vector( 2 downto 0 )
        );

end Fuzzification;

architecture membership_function of Fuzzification is

    -- Define the membership signals

    signal fuzzy_pm : std_logic_vector( busWidth - 1 downto 0 );
    signal fuzzy_ps : std_logic_vector( busWidth - 1 downto 0 );
    signal fuzzy_z : std_logic_vector( busWidth - 1 downto 0 );
    signal fuzzy_ns : std_logic_vector( busWidth - 1 downto 0 );
    signal fuzzy_nm : std_logic_vector( busWidth - 1 downto 0 );

    -- Define the fuzzy results, there are only 5 of them since there is only 5 membership functions

    signal fuzzy_result : std_logic_vector( 4 downto 0 );

    signal internal_data : std_logic_vector( 2 downto 0 );

    signal extend_ones : std_logic_vector( busWidth - 1 downto fuzzyWidth );
    signal extend_zeros : std_logic_vector( busWidth - 1 downto fuzzyWidth );

begin

    extend_ones <= ( others => '1' );
    extend_zeros <= ( others => '0' );

    -- Set-up the fuzzy ranges. The membership function that will be used is a simple rectangle.

    fuzzy_pm( fuzzyWidth - 1 downto 0 ) <= membership( fuzzyWidth * 1 - 1 downto fuzzyWidth * 0 );
    fuzzy_ps( fuzzyWidth - 1 downto 0 ) <= membership( fuzzyWidth * 2 - 1 downto fuzzyWidth * 1 );
    fuzzy_z ( fuzzyWidth - 1 downto 0 ) <= membership( fuzzyWidth * 3 - 1 downto fuzzyWidth * 2 );
    fuzzy_ns( fuzzyWidth - 1 downto 0 ) <= membership( fuzzyWidth * 4 - 1 downto fuzzyWidth * 3 );
    fuzzy_nm( fuzzyWidth - 1 downto 0 ) <= membership( fuzzyWidth * 5 - 1 downto fuzzyWidth * 4 );

    fuzzy_pm( busWidth - 1 downto fuzzyWidth ) <= extend_zeros;
    fuzzy_ps( busWidth - 1 downto fuzzyWidth ) <= extend_zeros;
    fuzzy_z ( busWidth - 1 downto fuzzyWidth ) <= extend_zeros;
    fuzzy_ns( busWidth - 1 downto fuzzyWidth ) <= extend_ones;
    fuzzy_nm( busWidth - 1 downto fuzzyWidth ) <= extend_ones;

    compare_pm : lpm_compare generic map (
        lpm_width => busWidth,
        lpm_representation => "signed",
        lpm_pipeline => 1
    )
    port map (
```

```

        dataa    => data,
        datab   => fuzzy_pm,
        aclr     => reset,
        clock    => clock,
        ageb     => fuzzy_result(0)
    );

compare_ps : lpm_compare generic map (
    lpm_width    => busWidth,
    lpm_representation => "signed",
    lpm_pipeline => 1
)
port map (
    dataa    => data,
    datab   => fuzzy_ps,
    aclr     => reset,
    clock    => clock,
    ageb     => fuzzy_result(1)
);

compare_z : lpm_compare generic map (
    lpm_width    => busWidth,
    lpm_representation => "signed",
    lpm_pipeline => 1
)
port map (
    dataa    => data,
    datab   => fuzzy_z,
    aclr     => reset,
    clock    => clock,
    aeb     => fuzzy_result(2)
);

compare_ns : lpm_compare generic map (
    lpm_width    => busWidth,
    lpm_representation => "signed",
    lpm_pipeline => 1
)
port map (
    dataa    => data,
    datab   => fuzzy_ns,
    aclr     => reset,
    clock    => clock,
    aleb     => fuzzy_result(3)
);

compare_nm : lpm_compare generic map (
    lpm_width    => busWidth,
    lpm_representation => "signed",
    lpm_pipeline => 1
)
port map (
    dataa    => data,
    datab   => fuzzy_nm,
    aclr     => reset,
    clock    => clock,
    aleb     => fuzzy_result(4)
);

```

```

output : process
begin

    wait until rising_edge( clock );

    if reset = '1' then

        internal_data <= zero;

    else

        if fuzzy_result(0) = '1' then

            internal_data <= pmedium;

```

```

elseif fuzzy_result(1) = '1' then
    internal_data <= psmall;
elseif fuzzy_result(2) = '1' then
    internal_data <= zero;

    elseif fuzzy_result(4) = '1' then
        internal_data <= nmedium;
elseif fuzzy_result(3) = '1' then
    internal_data <= nsmall;
else
    internal_data <= zero;
end if;
end if;
end process output;

flop_output : lpm_ff generic map ( lpm_width => busWidth )
    port map (
        data    => internal_data,
        clock   => clock,
        sclr    => reset,
        q       => fuzzy_data
    );

flop_valid : synchronizer generic map ( numberOfLevels => 3 )
    port map (
        clock    => clock,
        reset    => reset,
        input    => valid_in,
        output   => valid_out
    );
end membership_function;

```

## Appendix D: Test bench code for Fuzzification

```
library ieee;
use ieee.std_logic_1164.all;

entity Fuzzification_tb is
end Fuzzification_tb;

architecture mixed of Fuzzification_tb is

    component Fuzzification is

        port ( clock    : in std_logic;
              reset    : in std_logic;
              data      : in std_logic_vector(4 - 1 downto 0);
              membership : in std_logic_vector(5 * 3 - 1 downto 0);
              valid_in  : in std_logic;
              valid_out  : out std_logic;
              fuzzy_data : out std_logic_vector(2 downto 0)
            );

    end component Fuzzification;

    constant T_pw    : time := 20 ns;
    constant delay_time : time := 10 ns;
    constant busWidth : positive := 2;

    constant pmedium : std_logic_vector := "000";
    constant psmall  : std_logic_vector := "001";
    constant zero    : std_logic_vector := "010";
    constant nsmall  : std_logic_vector := "011";
    constant nmedium : std_logic_vector := "100";

    signal clock      : std_logic;
    signal reset      : std_logic;

    signal data        : std_logic_vector(4 - 1 downto 0);
    signal membership  : std_logic_vector(5 * 3 - 1 downto 0);
    signal valid_in    : std_logic;

    signal data_delayed   : std_logic_vector(4 - 1 downto 0);
    signal membership_delayed : std_logic_vector(5 * 3 - 1 downto 0);
    signal valid_in_delayed : std_logic;

    signal valid_out : std_logic;
    signal fuzzy_data : std_logic_vector(2 downto 0);

begin

    data_delayed <= data after delay_time;
    membership_delayed <= membership after delay_time;
    valid_in_delayed <= valid_in after delay_time;

    dut : Fuzzification port map (
        clock => clock,
        reset => reset,
        data => data_delayed,
        membership => membership_delayed,
        valid_in => valid_in_delayed,
        valid_out => valid_out,
        fuzzy_data => fuzzy_data
    );

    clock_gen : process
    begin

        clock <= '0';
```

```

wait for T_pw;
clock <= '1';
wait for T_pw;

    end process clock_gen;

    reset_control : process
    begin

        reset <= '1';
        wait for 4 * T_pw;

    reset <= '0';

        wait;
    end process reset_control;

    input_control : process
    begin

data <= ( others => '0' );
valid_in <= '0';

    membership <= "01110000001010";

    wait for T_pw;

wait for 10 * 2 * T_pw;

valid_in <= '1';
data <= "0000";
wait for 4 * 2 * T_pw;

data <= "0001";
wait for 4 * 2 * T_pw;

data <= "0010";
wait for 4 * 2 * T_pw;

data <= "0011";
wait for 4 * 2 * T_pw;

data <= "1100";
wait for 4 * 2 * T_pw;

data <= "1101";
wait for 4 * 2 * T_pw;

data <= "1110";
wait for 4 * 2 * T_pw;

data <= "1111";
wait for 4 * 2 * T_pw;

data <= "0011";
valid_in <= '0';
    wait;
    end process input_control;
end mixed;

```

## Appendix E: Coding for Rule Base – Inference in VHDL

```
library ieee;
use ieee.std_logic_1164.all;

library work;
use work.config.all;

entity fuzzy_rulebase is

port ( clock    : in std_logic;
      reset    : in std_logic;
      difference : in std_logic_vector( 2 downto 0 );
      integral  : in std_logic_vector( 2 downto 0 );
      valid_in  : in std_logic;
      valid_out : out std_logic;
      control   : out std_logic_vector( 2 downto 0 )
    );

end fuzzy_rulebase;

architecture rules of fuzzy_rulebase is

signal internal_control : std_logic_vector( 2 downto 0 );

signal internal_control_pm : std_logic_vector( 2 downto 0 );
signal internal_control_ps : std_logic_vector( 2 downto 0 );
signal internal_control_z  : std_logic_vector( 2 downto 0 );
signal internal_control_ns : std_logic_vector( 2 downto 0 );
signal internal_control_nm : std_logic_vector( 2 downto 0 );

begin

control_rules_pm : process
begin

wait until rising_edge( clock );

if reset = '1' then

internal_control_pm <= zero;

else

if difference = pmedium and integral = pmedium then
internal_control_pm <= zero;
elsif difference = pmedium and integral = psmall then
internal_control_pm <= psmall;
elsif difference = pmedium and integral = zero then
```

```

    internal_control_pm <= pmedium;
elseif difference = pmedium and integral = nsmall then
    internal_control_pm <= pmedium;
elseif difference = pmedium and integral = nmedium then
    internal_control_pm <= pmedium;
else
    internal_control_pm <= zero;
end if;
end if;

end process control_rules_pm;

control_rules_ps : process
begin

    wait until rising_edge( clock );

    if reset = '1' then
        internal_control_ps <= zero;
    else
        if difference = psmall and integral = pmedium then
            internal_control_ps <= nsmall;
        elseif difference = psmall and integral = psmall then
            internal_control_ps <= zero;
        elseif difference = psmall and integral = zero then
            internal_control_ps <= psmall;
        elseif difference = psmall and integral = nsmall then
            internal_control_ps <= psmall;
        elseif difference = psmall and integral = nmedium then
            internal_control_ps <= pmedium;
        else
            internal_control_ps <= zero;
        end if;
    end if;

end process control_rules_ps;

control_rules_z : process
begin

    wait until rising_edge( clock );

    if reset = '1' then

        internal_control_z <= zero;
    else
        if difference = psmall and integral = pmedium then
            internal_control_z <= nmedium;
        elseif difference = psmall and integral = psmall then
            internal_control_z <= nsmall;
        elseif difference = psmall and integral = zero then

```

```

    internal_control_z <= zero;
    elsif difference = psmall and integral = nsmall then
        internal_control_z <= psmall;
    elsif difference = psmall and integral = nmedium then
        internal_control_z <= psmall;
    else
        internal_control_z <= zero;
    end if;
end if;
end process control_rules_z;

```

```

control_rules_ns : process
begin

```

```

    wait until rising_edge( clock );

```

```

    if reset = '1' then

```

```

        internal_control_ns <= zero;
    else

```

```

        if difference = nsmall and integral = pmedium then
            internal_control_ns <= nmedium;

```

```

        elsif difference = nsmall and integral = psmall then
            internal_control_ns <= nmedium;

```

```

        elsif difference = nsmall and integral = zero then
            internal_control_ns <= nsmall;

```

```

        elsif difference = nsmall and integral = nsmall then
            internal_control_ns <= zero;

```

```

        elsif difference = nsmall and integral = nmedium then
            internal_control_ns <= psmall;

```

```

        else
            internal_control_ns <= zero;

```

```

        end if;
    end if;

```

```

end process control_rules_ns;

```

```

control_rules_nm : process
begin

```

```

    wait until rising_edge( clock );

```

```

    if reset = '1' then

```

```

        internal_control_nm <= zero;
    else

```

```

        if difference = nmedium and integral = pmedium then
            internal_control_nm <= nmedium;

```

```

        elsif difference = nmedium and integral = psmall then
            internal_control_nm <= nmedium;

```

```

        elsif difference = nmedium and integral = zero then
            internal_control_nm <= nmedium;

```

```

        elsif difference = nmedium and integral = nsmall then

```



```

    internal_control_nm <= nsmall;
elseif difference = nmedium and integral = nmedium then
    internal_control_nm <= zero;
else
    internal_control_nm <= zero;
end if;
end if;

end process control_rules_nm;

control_out : process

function check( value : in std_logic_vector( 2 downto 0 ) ) return std_logic is

    variable return_value : std_logic;

begin

    if internal_control_pm = value or
       internal_control_ps = value or
       internal_control_z = value or
       internal_control_ns = value or
       internal_control_nm = value then

        return_value := '1';

    else
        return_value := '0';
    end if;
    return return_value;
end function check;

begin

    wait until rising_edge( clock );

    if reset = '1' then
        control <= zero;
    else
        if check( pmedium ) = '1' then
            control <= pmedium;
        elsif check( psmall ) = '1' then
            control <= psmall;
        elsif check( nsmall ) = '1' then
            control <= nsmall;
        elsif check( nmedium ) = '1' then
            control <= nmedium;
        else
            control <= zero;
        end if;
    end if;
end if;

```

```
end process control_out;

flop_valid : synchronizer generic map ( numberOfLevels => 2 )
    port map (
        clock    => clock,
        reset    => reset,
        input    => valid_in,
        output   => valid_out
    );

end rules;
```

## Appendix F: Test bench code for Inference

```
library ieee;
use ieee.std_logic_1164.all;

entity fuzzy_rulebase_tb is
end fuzzy_rulebase_tb;

architecture mixed of fuzzy_rulebase_tb is

  component fuzzy_rulebase is

    port ( clock      : in std_logic;
          reset      : in std_logic;

          difference : in std_logic_vector( 2 downto 0 );
          integral   : in std_logic_vector( 2 downto 0 );
          valid_in   : in std_logic;
          valid_out  : out std_logic;
          control    : out std_logic_vector( 2 downto 0 )
        );

  end component fuzzy_rulebase;

  constant T_pw      : time := 20 ns;
  constant delay_time : time := 10 ns;
  constant busWidth  : positive := 2;

  constant pmedium : std_logic_vector := "000";
  constant psmall  : std_logic_vector := "001";
  constant zero    : std_logic_vector := "010";
  constant nsmall  : std_logic_vector := "011";
  constant nmedium : std_logic_vector := "100";

  signal clock      : std_logic;
  signal reset      : std_logic;

  signal difference : std_logic_vector( 2 downto 0 );
  signal integral   : std_logic_vector( 2 downto 0 );
  signal valid_in   : std_logic;

  signal difference_delayed : std_logic_vector( 2 downto 0 );
  signal integral_delayed  : std_logic_vector( 2 downto 0 );
  signal valid_in_delayed  : std_logic;

  signal valid_out : std_logic;
  signal control   : std_logic_vector( 2 downto 0 );

begin

  difference_delayed <= difference after delay_time;
  integral_delayed  <= integral   after delay_time;
  valid_in_delayed  <= valid_in   after delay_time;

  dut : fuzzy_rulebase port map (
    clock      => clock,
    reset      => reset,
    difference => difference_delayed,
    integral   => integral_delayed,
    valid_in   => valid_in_delayed,
    valid_out  => valid_out,
    control    => control
  );

  clock_gen : process
  begin
    clock <= '0';

    wait for T_pw;
    clock <= '1';
    wait for T_pw;
```

```

        end process clock_gen;

        reset_control : process
        begin
            reset <= '1';
            wait for 4 * T_pw;

reset <= '0';
            wait;

        end process reset_control;

        input_control : process

        procedure test_rule( input_1 : in std_logic_vector( 2 downto 0 );
            input_2 : in std_logic_vector( 2 downto 0 )
            ) is
        begin

            difference <= input_1;
            integral <= input_2;

            wait for 4 * 2 * T_pw;
            valid_in <= '1';
            wait for 2 * T_pw;
            valid_in <= '0';

        end procedure test_rule;

        begin
            difference <= pmedium;
            integral <= psmall;
            valid_in <= '0';

            wait for T_pw;

            test_rule( pmedium, pmedium );
            test_rule( pmedium, psmall );
            test_rule( pmedium, zero );
            test_rule( pmedium, nsmall );
            test_rule( pmedium, nmedium );

            test_rule( psmall, pmedium );
            test_rule( psmall, psmall );
            test_rule( psmall, zero );
            test_rule( psmall, nsmall );
            test_rule( psmall, nmedium );

            test_rule( zero, pmedium );
            test_rule( zero, psmall );
            test_rule( zero, zero );
            test_rule( zero, nsmall );
            test_rule( zero, nmedium );

            test_rule( nsmall, pmedium );
            test_rule( nsmall, psmall );
            test_rule( nsmall, zero );
            test_rule( nsmall, nsmall );
            test_rule( nsmall, nmedium );

            test_rule( nmedium, pmedium );
            test_rule( nmedium, psmall );
            test_rule( nmedium, zero );
            test_rule( nmedium, nsmall );
            test_rule( nmedium, nmedium );

            wait;

        end process input_control;

    end mixed;

```

## Appendix G: Coding for Defuzzification in VHDL

```
library ieee;
use ieee.std_logic_1164.all;

library work;
use work.config.all;

entity Defuzzification is

    generic ( busWidth    : positive := 4;

             fuzzyWidth  : positive := 3
            );

    port ( clock    : in std_logic;
          reset    : in std_logic;
          fuzzy_output : in std_logic_vector( 2 downto 0 );
          membership : in std_logic_vector( 5 * fuzzyWidth - 1 downto 0 );
          valid_in  : in std_logic;
          valid_out  : out std_logic;
          output    : out std_logic_vector( busWidth - 1 downto 0 )
        );

end Defuzzification;

architecture membership_function of Defuzzification is

    signal defuzzify_pm  : std_logic_vector( fuzzyWidth - 1 downto 0 );
    signal defuzzify_ps  : std_logic_vector( fuzzyWidth - 1 downto 0 );
    signal defuzzify_z   : std_logic_vector( fuzzyWidth - 1 downto 0 );
    signal defuzzify_ns  : std_logic_vector( fuzzyWidth - 1 downto 0 );
    signal defuzzify_nm  : std_logic_vector( fuzzyWidth - 1 downto 0 );

    signal fuzzy_result : std_logic_vector( 4 downto 0 );

    signal internal_output : std_logic_vector( busWidth - 1 downto 0 );

    signal extend_ones : std_logic_vector( busWidth - 1 downto fuzzyWidth );
    signal extend_zeros : std_logic_vector( busWidth - 1 downto fuzzyWidth );

begin

    extend_ones <= ( others => '1' );
    extend_zeros <= ( others => '0' );

    defuzzify_pm <= membership( fuzzyWidth * 1 - 1 downto fuzzyWidth * 0 );
    defuzzify_ps <= membership( fuzzyWidth * 2 - 1 downto fuzzyWidth * 1 );
    defuzzify_z  <= membership( fuzzyWidth * 3 - 1 downto fuzzyWidth * 2 );
    defuzzify_ns <= membership( fuzzyWidth * 4 - 1 downto fuzzyWidth * 3 );
    defuzzify_nm <= membership( fuzzyWidth * 5 - 1 downto fuzzyWidth * 4 );

    defuzzify : process

        variable save_output : std_logic_vector( fuzzyWidth - 1 downto 0 );

    begin

        wait until rising_edge( clock );

        if reset = '1' then

            save_output := defuzzify_z;

        else

            if fuzzy_output = pmedium then
                save_output := defuzzify_pm;
            elsif fuzzy_output = psmall then
                save_output := defuzzify_ps;
            elsif fuzzy_output = zero then
                save_output := defuzzify_z;
            elsif fuzzy_output = nsmall then
```

```

    save_output := defuzzify_ns;
  elsif fuzzy_output = nmedium then
    save_output := defuzzify_nm;
  else
    save_output := defuzzify_z;
  end if;
end if;

internal_output( fuzzyWidth - 1 downto 0 ) <= save_output;

if save_output( fuzzyWidth - 1 ) = '1' then

  internal_output( busWidth - 1 downto fuzzyWidth ) <= extend_ones;

else

  internal_output( busWidth - 1 downto fuzzyWidth ) <= extend_zeros;

end if;

end process defuzzify;

sync_output : lpm_ff generic map ( lpm_width => busWidth )
  port map (
    data    => internal_output,
    clock   => clock,
    sclr    => reset,
    q       => output
  );

flop_valid : synchronizor generic map ( numberOfLevels => 2 )
  port map (
    clock    => clock,
    reset    => reset,
    input    => valid_in,
    output   => valid_out
  );

end membership_function;

```

## Appendix H: Test bench code for Defuzzification

```
library ieee;
use ieee.std_logic_1164.all;

entity Defuzzification_tb is
end Defuzzification_tb;

architecture mixed of Defuzzification_tb is

  component Defuzzification is

    port ( clock      : in std_logic;
          reset       : in std_logic;
          fuzzy_output : in std_logic_vector( 2 downto 0 );
          membership   : in std_logic_vector( 5 * 3 - 1 downto 0 );
          valid_in     : in std_logic;
          valid_out    : out std_logic;
          output       : out std_logic_vector( 4 - 1 downto 0 )
        );

  end component Defuzzification;

  constant T_pw      : time := 20 ns;
  constant delay_time : time := 10 ns;
  constant busWidth  : positive := 2;

  constant pmedium : std_logic_vector := "000";
  constant psmall  : std_logic_vector := "001";
  constant zero    : std_logic_vector := "010";
  constant nsmall  : std_logic_vector := "011";
  constant nmedium : std_logic_vector := "100";

  signal clock      : std_logic;
  signal reset      : std_logic;

  signal fuzzy_output : std_logic_vector( 2 downto 0 );
  signal membership   : std_logic_vector( 5 * 3 - 1 downto 0 );
  signal valid_in     : std_logic;

  signal fuzzy_output_delayed : std_logic_vector( 2 downto 0 );
  signal membership_delayed   : std_logic_vector( 5 * 3 - 1 downto 0 );
  signal valid_in_delayed     : std_logic;

  signal valid_out : std_logic;
  signal output    : std_logic_vector( 4 - 1 downto 0 );

begin

  fuzzy_output_delayed <= fuzzy_output after delay_time;
  membership_delayed   <= membership   after delay_time;
  valid_in_delayed     <= valid_in     after delay_time;

  dut : Defuzzification port map (
    clock      => clock,
    reset      => reset,
    fuzzy_output => fuzzy_output_delayed,
    membership  => membership_delayed,
    valid_in    => valid_in_delayed,
    valid_out   => valid_out,
    output      => output
  );

  clock_gen : process
  begin

    clock <= '0';

    wait for T_pw;
    clock <= '1';
    wait for T_pw;

    end process clock_gen;

end mixed;
```

```

reset_control : process
begin

    reset <= '1';
    wait for 4 * T_pw;

    reset <= '0';
    wait for 76 * T_pw;

    reset <= '1';
    wait for 10 * T_pw;

reset <= '0';
    wait;

    end process reset_control;

    input_control : process
    begin

fuzzy_output <= ( others => '0' );
valid_in <= '0';

membership <= "01110000001010";
wait for T_pw;

wait for 10 * 2 * T_pw;

fuzzy_output <= pmedium;
wait for 4 * 2 * T_pw;

fuzzy_output <= psmall;
wait for 4 * 2 * T_pw;

fuzzy_output <= zero;
wait for 4 * 2 * T_pw;

fuzzy_output <= nsmall;
wait for 4 * 2 * T_pw;

fuzzy_output <= nmedium;
wait for 4 * 2 * T_pw;

fuzzy_output <= pmedium;
valid_in <= '0';

wait;

    end process input_control;

end mixed;

```



## Appendix I: Matlab's M-file for Fuzzification

```
function [] = fuzzy (Temp,Hum)

Temp = input('Enter room temperature value = ');
Hum = input('Enter room humidity value = ');

if ((Temp>=10)&(Temp<=16))
    Temp_LV1='cold'
    Temp_tv1=1
elseif ((Temp>=20)&(Temp<=24))
    Temp_LV1='cool'
    Temp_tv1=1
elseif ((Temp>=28)&(Temp<=32))
    Temp_LV1='normal'
    Temp_tv1=1
elseif ((Temp>=36)&(Temp<=40))
    Temp_LV1='warm'
    Temp_tv1=1
elseif ((Temp>=44)&(Temp<=50))
    Temp_LV1='hot'
    Temp_tv1=1
elseif ((Temp>16)&(Temp<20))
    Temp_LV1='cold'
    Temp_LV2='cool'
    a = Temp-16;
    Temp_tv1= 1-0.25*a
    Temp_tv2= 0.25*a
elseif ((Temp>24)&(Temp<28))
    Temp_LV1='cool'
    Temp_LV2='normal'
    a = Temp-24;
    Temp_tv1= 1-0.25*a
    Temp_tv2= 0.25*a
elseif ((Temp>32)&(Temp<36))
    Temp_LV1='normal'
    Temp_LV2='warm'
    a = Temp-32;
    Temp_tv1= 1-0.25*a
    Temp_tv2= 0.25*a
else ((Temp>40)&(Temp<44))
    Temp_LV1='warm'
    Temp_LV2='hot'
    a = Temp-40;
    Temp_tv1= 1-0.25*a
    Temp_tv2= 0.25*a
end

if((Hum>=0)&(Hum<=0.15))
```

```

Hum_LV1='Negative Medium'
Hum_tv1=1
elseif ((Hum>=0.2125)&(Hum<=0.3625))
Hum_LV1='Negative Small'
Hum_tv1=1
elseif ((Hum>=0.425)&(Hum<=0.575))
Hum_LV1='Zero'
Hum_tv1=1
elseif ((Hum>=0.6375)&(Hum<=0.7875))
Hum_LV1='Positive Small'
Hum_tv1=1
elseif ((Hum>=0.85)&(Hum<=1.0))
Hum_LV1='Positive Medium'
Hum_tv1=1
elseif ((Hum>0.15)&(Hum<0.2125))
Hum_LV1='Negative Medium'
Hum_LV2='Negative Small'
b = Hum-0.15;
Hum_tv1= 1-16*b
Hum_tv2= 16*b
elseif ((Hum>0.3625)&(Hum<0.425))
Hum_LV1='Negative Small'
Hum_LV2='Zero'
b = Hum-0.3625;
Hum_tv1= 1-16*b
Hum_tv2= 16*b
elseif ((Hum>0.575)&(Hum<0.6375))
Hum_LV1='Zero'
Hum_LV2='Positive Small'
b = Hum-0.575;
Hum_tv1= 1-16*b
Hum_tv2= 16*b
else ((Hum>0.7875)&(Hum<0.85))
Hum_LV1='Positive Small'
Hum_LV2='Positive Medium'
b = Hum-0.7875;
Hum_tv1= 1-16*b
Hum_tv2= 16*b

```

End

## Appendix J: Matlab's M-file for Inference

```
function [] = infer (x);

Temp_LV1 = input('Enter room temperature linguistic variable = ','s');
Temp_LV2 = input('Enter room temperature linguistic variable = ','s');
Hum_LV1 = input('Enter room humidity linguistic variable = ','s');
Hum_LV2 = input('Enter room humidity linguistic variable = ','s');

if (Temp_LV1=='NM')
    Delta_temp_LV1='PM'
elseif (Temp_LV1=='NS')
    Delta_temp_LV1='PS'
elseif (Temp_LV1=='Z')
    Delta_temp_LV1='Z'
elseif (Temp_LV1=='PS')
    Delta_temp_LV1='NS'
else (Temp_LV1=='PM')
    Delta_temp_LV1='NM'
end

if (Temp_LV2=='NM')
    Delta_temp_LV2='PM'
elseif (Temp_LV2=='NS')
    Delta_temp_LV2='PS'
elseif (Temp_LV2=='Z')
    Delta_temp_LV2='Z'
elseif (Temp_LV2=='PS')
    Delta_temp_LV2='NS'
else (Temp_LV2=='PM')
    Delta_temp_LV2='NM'
end

if (Hum_LV1=='NM')
    Airflow_LV1='NM'
elseif (Hum_LV1=='NS')
    Airflow_LV1='NS'
elseif (Hum_LV1=='Z')
    Airflow_LV1='Z'
elseif (Hum_LV1=='PS')
    Airflow_LV1='PS'
else (Hum_LV1=='PM')
    Airflow_LV1='PM'
end

if (Hum_LV2=='NM')
    Airflow_LV2='NM'
elseif (Hum_LV2=='NS')
    Airflow_LV2='NS'
elseif (Hum_LV2=='Z')
```

```
Airflow_LV2='Z'  
elseif (Hum_LV2=='PS')  
    Airflow_LV2='PS'  
else (Hum_LV2=='PM')  
    Airflow_LV2='PM'  
end
```

## Appendix K: Matlab's M-file for Defuzzification

```
function [] = defuzzy (x)

Temp = input('Enter room temperature value = ');
Hum = input('Enter room humidity value = ');
Temp_LV1 = input('Enter room temperature linguistic variable = ','s');
Temp_LV2 = input('Enter room temperature linguistic variable = ','s');
Hum_LV1 = input('Enter room humidity linguistic variable = ','s');
Hum_LV2 = input('Enter room humidity linguistic variable = ','s');
Delta_temp_LV1 = input('Enter delta room temperature linguistic variable = ','s');
Delta_temp_LV2 = input('Enter delta room temperature linguistic variable = ','s');
Airflow_LV1 = input('Enter room air flow linguistic variable = ','s');
Airflow_LV2 = input('Enter room air flow linguistic variable = ','s');
Temp_tv1 = input('Enter room temperature truth value = ');
Temp_tv2 = input('Enter room temperature truth value = ');
Hum_tv1 = input('Enter room humidity truth value = ');
Hum_tv2 = input('Enter room humidity truth value = ');

if ((Delta_temp_LV1=='NM')&(Temp_tv1==1))
    c=(Temp-10)*0.2;
    output_Delta_temp=c*6
elseif ((Delta_temp_LV1=='NS')&(Temp_tv1==1))
    c=(Temp-20)*0.2;
    output_Delta_temp=(c*4)+10
elseif ((Delta_temp_LV1=='Z')&(Temp_tv1==1))
    c=(Temp-28)*0.2;
    output_Delta_temp=(c*4)+18
elseif ((Delta_temp_LV1=='PS')&(Temp_tv1==1))
    c=(Temp-36)*0.2;
    output_Delta_temp=(c*4)+26
else((Delta_temp_LV1=='PM')&(Temp_tv1==1))
    c=(Temp-44)*0.2;
    output_Delta_temp=(c*6)+34
end

if ((Airflow_LV1=='NM')&(Hum_tv1==1))
    d=(Hum-0)*(1/0.15);
    output_Airflow=d*0.15
elseif ((Airflow_LV1=='NS')&(Hum_tv1==1))
    d=(Hum-0.2125)*(1/0.15);
    output_Airflow=(d*0.15)+0.2125
elseif ((Airflow_LV1=='Z')&(Hum_tv1==1))
    d=(Hum-0.425)*(1/0.15);
    output_Airflow=(d*0.15)+0.425

elseif ((Airflow_LV1=='PS')&(Hum_tv1==1))
    d=(Hum-0.6375)*(1/0.15);
    output_Airflow=(d*0.15)+0.6375
else ((Airflow_LV1=='PM')&(Hum_tv1==1))
```

```

d=(Hum-0.85)*(1/0.15);
output_Airflow=(d*0.15)+0.85
end

if (((Delta_temp_LV1=='NM')&(Delta_temp_LV2=='NS'))&(Temp_tv1~=1))
    tvf=min(Temp_tv1,Temp_tv2);
    output_Delta_temp=10-4*tvf
elseif (((Delta_temp_LV1=='NS')&(Delta_temp_LV2=='Z'))&(Temp_tv1~=1))
    tvf=min(Temp_tv1,Temp_tv2);
    output_Delta_temp=18-4*tvf
elseif (((Delta_temp_LV1=='Z')&(Delta_temp_LV2=='PS'))&(Temp_tv1~=1))
    tvf=min(Temp_tv1,Temp_tv2);
    output_Delta_temp=26-4*tvf
else (((Delta_temp_LV1=='PS')&(Delta_temp_LV2=='PM'))&(Temp_tv1~=1))
    tvf=min(Temp_tv1,Temp_tv2);
    output_Delta_temp=34-4*tvf
end

if (((Airflow_LV1=='NM')&(Airflow_LV2=='NS'))&(Hum_tv1~=1))
    tvf=min(Hum_tv1,Hum_tv2);
    output_Airflow=0.2125-0.0625*tvf
elseif (((Airflow_LV1=='NS')&(Airflow_LV2=='Z'))&(Hum_tv1~=1))
    tvf=min(Hum_tv1,Hum_tv2);
    output_Airflow=0.425-0.0625*tvf
elseif (((Airflow_LV1=='Z')&(Airflow_LV2=='PS'))&(Hum_tv1~=1))
    tvf=min(Hum_tv1,Hum_tv2);
    output_Airflow=0.6375-0.0625*tvf
else (((Airflow_LV1=='PS')&(Airflow_LV2=='PM'))&(Hum_tv1~=1))
    tvf=min(Hum_tv1,Hum_tv2);
    output_Airflow=0.85-0.0625*tvf
end

```

## Appendix L: Matlab Results – Fuzzification

```
>> fuzzyinput
```

```
Enter room temperature value = 35
```

```
Enter room humidity value = 0.65
```

```
Temp_LV1 =
```

```
normal
```

```
Temp_LV2 =
```

```
warm
```

```
Temp_tv1 =
```

```
0.2500
```

```
Temp_tv2 =
```

```
0.7500
```

```
Hum_LV1 =
```

```
Positive Small
```

```
Hum_tv1 =
```

```
1
```

## Appendix M: Matlab Results – Inference

```
>> Inference
```

```
Enter room temperature linguistic variable = Z  
Enter room temperature linguistic variable = PS  
Enter room humidity linguistic variable = PS  
Enter room humidity linguistic variable =
```

```
Delta_temp1 =
```

```
Z
```

```
Delta_temp2 =
```

```
NS
```

```
Airflow1 =
```

```
PS
```

```
Airflow2 =
```

```
PM
```



## Appendix N: Matlab Results – Defuzzification

```
>> Defuzzification
```

```
Enter room temperature value = 35  
Enter room humidity value = 0.65  
Enter room temperature linguistic variable = Z  
Enter room temperature linguistic variable = PS  
Enter room humidity linguistic variable = PS  
Enter room humidity linguistic variable =  
Enter delta room temperature linguistic variable = Z  
Enter delta room temperature linguistic variable = NS  
Enter room air flow linguistic variable = PM  
Enter room air flow linguistic variable =  
Enter room temperature truth value = 0.25  
Enter room temperature truth value = 0.75  
Enter room humidity truth value = 1  
Enter room humidity truth value =
```

```
output_Delta_temp =
```

```
2.2000
```

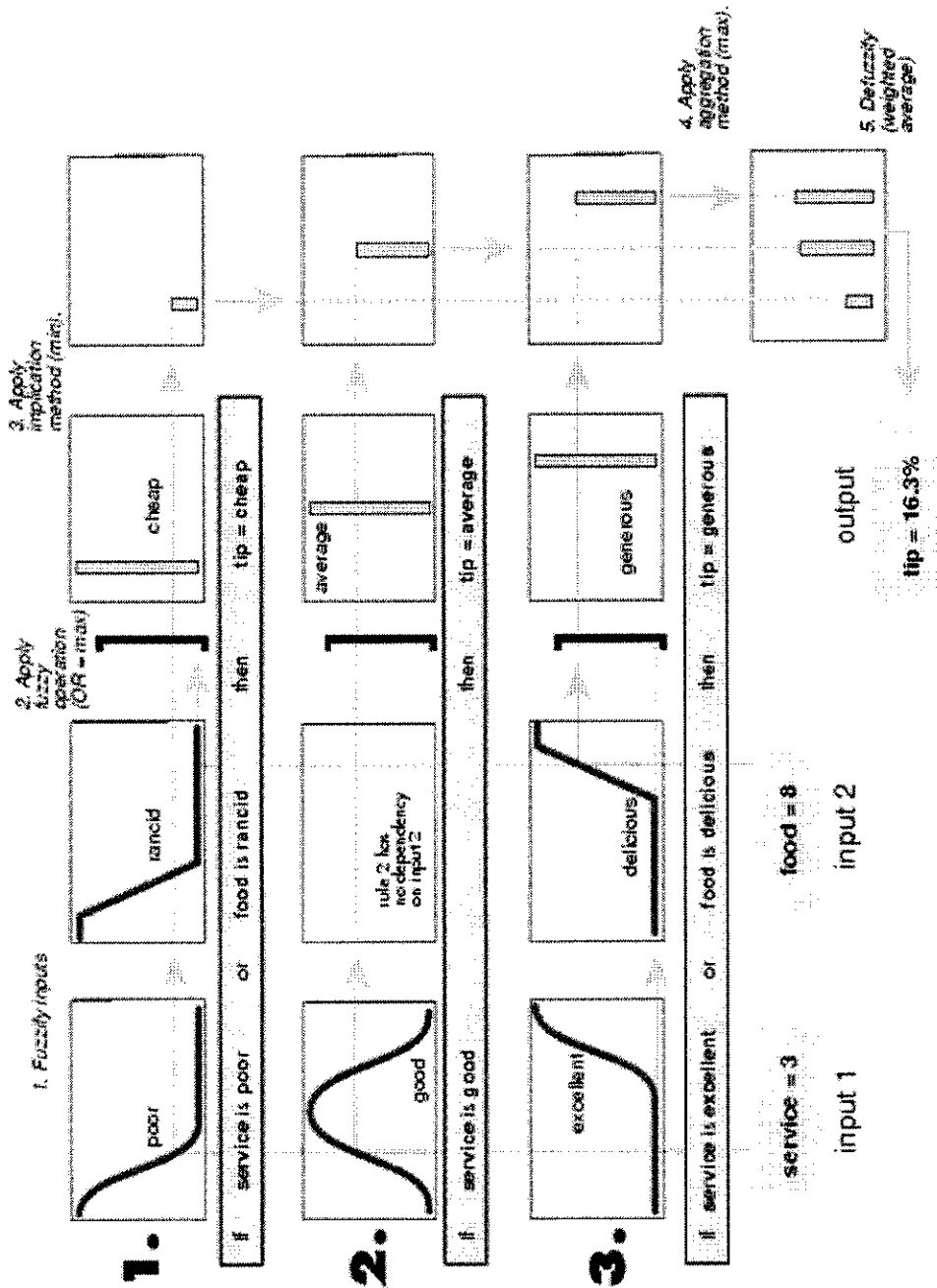
```
output_Airflow =
```

```
0.3000
```

```
output_Delta_temp =
```

```
14.2000
```

## Appendix O: Sugeno method's sample calculation



The figure above shows the Fuzzy tipping model developed for use as a zero-order Sugeno system. Fortunately it is frequently the case that singleton output functions

are completely sufficient for a given problem's needs. The more general first-order Sugeno fuzzy model has rules of the form

$$\text{if } x \text{ is } A \text{ and } y \text{ is } B \text{ then } z = p*x + q*y + r$$

where  $A$  and  $B$  are fuzzy sets in the antecedent, while  $p$ ,  $q$ , and  $r$  are all constants. The easiest way to visualize the first-order system is to think of each rule as defining the location of a "moving singleton." That is, the singleton output spikes can move around in a linear fashion in the output space, depending on what the input is. This also tends to make the system notation very compact and efficient. Higher order Sugeno fuzzy models are possible, but they introduce significant complexity with little obvious merit.

Because of the linear dependence of each rule on the system's input variables, the Sugeno method is ideal for acting as an interpolating supervisor of multiple linear controllers that are to be applied, respectively, to different operating conditions of a dynamic nonlinear system. For example, the performance of an aircraft may change dramatically with altitude and Mach number. Linear controllers, though easy to compute and well-suited to any given flight condition, must be updated regularly and smoothly to keep up with the changing state of the flight vehicle. A Sugeno fuzzy Inference system is extremely well suited to the task of smoothly interpolating the linear gains that would be applied across the input space; it's a natural and efficient gain scheduler. Similarly, a Sugeno system is suited for modeling nonlinear systems by interpolating multiple linear models.