Titre: Title:	Fuzzy expert system for thermal bias compensation of microwave communications equipment
Auteur: Author:	Mark De Payrebrune
Date:	1991
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	De Payrebrune, M. (1991). Fuzzy expert system for thermal bias compensation of microwave communications equipment [Ph.D. thesis, Polytechnique Montréal]. PolyPublie. https://publications.polymtl.ca/57963/

URL de PolyPublie: PolyPublie URL:	https://publications.polymtl.ca/57963/
Directeurs de recherche: Advisors:	
Programme: Program:	Unspecified

UNIVERSITÉ DE MONTRÉAL

FUZZY EXPERT SYSTEM FOR THERMAL BIAS COMPENSATION OF MICROWAVE COMMUNICATIONS EQUIPMENT

par

Mark DE PAYREBRUNE
DÉPARTEMENT DE GÉNIE ÉLECTRIQUE ET DE GÉNIE INFORMATIQUE
ÉCOLE POLYTECHNIQUE

THESE PRESENTÉE EN VUE DE L'OBTENTION
DU GRADE DE PHILOSOPHIAE DOCTOR (Ph.D.)
(GÉNIE ÉLECTRIQUE)
AVRIL 1991

ISBN 0-315-72759-4

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE

Cette thèse intitulée:

FUZZY EXPERT SYSTEM FOR THERMAL BIAS COMPENSATION OF MICROWAVE COMMUNICATIONS EQUIPMENT

présentée par: Mark de Payrebrune

en vue de l'obtention du grade de: Philosophiae Doctor (Ph.D)

a été dûment acceptée par le jury d'examen constitué de:

M. OLIVIER Guy, Ph.D., président

M. MUKHEDKAR Dinkar, D.Sc., membre et directeur de recherche

M. APRIL George-E, M.Sc., membre

M. JASMIN Gilles, Ph.D., membre

Dedicated to my father Ludwik,

he would have been proud.

SOMMAIRE

Le but de cette recherche est de développer un système expert en logique floue pour le contrôle en temps réel du processus d'alignement dans l'équipement des communications par micro-ondes.

Le processus d'alignement consiste en la manipulation allant jusqu'à 10 différentes polarisations de transistors pour parvenir à la performance souhaitée du système. La relation entre la polarisation de transistor et les sorties au niveau de l'équipement de communication est généralement inconnue et ne peut être dénifie mathématiquement. Une difficulté supplémentaire réside dans le fait qu'il existe aussi une interaction complexe entre chaque polarisation de transistors.

Présentement, l'alignement de l'équipement de communications par micro-ondes est réalisé par un personnel expert. La qualification nécessaire au fonctionnement de ce type d'équipement a pu être acquise au terme de longues années et se base sur l'expérience et le développement heuristique. le problème est trop complexe et n'est pas suffisamment bien défini pour être résolu dans le cadre d'un mode algorithmique.

L'objectif de cette recherche est de développer un système expert en logique floue afin de mettre en oeuvre les règles heuristiques utilisées dans l'industrie. L'utilisation de la logique floue a permis au système expert de venir à bout de l'incertitude de relations existant entre les polarisations de transistors individuels.

Un système expert mis au point avec succès fut développé afin d'automatiser les polarisations de transistor et l'alignement de l'équipement de communications par micro-ondes. Les heuristiques utilisées furent définies comme devant faire appel aux techniques de connaissance en ingénierie, et impliquaient des entrevues complètes avec des ingénieurs en design et du personnel hautement qualifié. Les connaissances résultantes furent mises en oeuvre dans un système expert. Ce système expert fut réalisé sur un IBM-PC, et comprenait un contrôle automatique du récepteur de micro-ondes par le biais de convert-isseurs de type numérique/analogique (D/A) et analogique/numérique (A/D).

La logique floue est utilisée dans le but d'intégrer les heuristiques et les données calculées. L'utilisation de la logique floue est impérative pour que le système soit opérationnel puisque les relations entre le signal de sortie mesuré à partir de l'équipement de communication, et les polarisations de transistors individuels étaient inconnues.

ABSTRACT

The goal of this research was to develop a fuzzy logic expert system for the real time control of the alignment process in microwave communications equipment.

The alignment process consists of manipulating up to 10 different transistor biases to obtain a desired system performance. The relationship between transistor bias and communication equipment output is generally unknown and cannot be mathematically defined. A further complication is that there is also a complex interaction between each transistor bias.

Presently, the alignment of microwave communications equipment is done by expert manufacturing personnel. The skill required to tune this type of equipment is developed over a number of years and is based on experience and on the development of a set of heuristics. The problem is too complex and not sufficiently well defined to be solved in an algorithmic fashion.

The aim of this research was to develop a Fuzzy Logic Expert System to implement the heuristic rules used in manufacturing. The use of fuzzy logic allowed the expert system to deal with uncertainty in the relationships between the individual transistor biases.

A successful expert system was developed to automate the biasing and alignment of Microwave Communications Equipment. The heuristics used were defined using the

techniques of Knowledge Engineering and involved detailed interviews with design engineers and expert manufacturing personnel.

The resulting knowledge base was implemented into an expert system. The expert system was implemented on an IBM PC and included the automated control of the microwave receiver through Digital to Analog (D/A) and Analog to Digital (A/D) converters.

Fuzzy logic was used to integrate the heuristics and the measured data. The use of fuzzy logic was imperative to the operation of this system since the relationships between the measured output signal from the communication equipment and the individual transistor biases was unknown.

RÉSUMÉ

La logique floue est une extension de la logique Booléenne. En logique Booléenne, une hypothèse peut être VRAIE ou FAUSSE. Les étapes intermédiaires ne sont pas permises.

En logique floue, ces états intermédiaires sont acceptés et peuvent par conséquent permettre des vérités partielles. Le terme "logique floue" est souvent trompeur car il implique un type de logique imprécise et donc de petite valeur. En réalité, les contrôleurs de logique floue font appel à des entrées très précises, à des calculs performants par un mode mathématique rigoureux, et fournissent des résultats précis qui peuvent être utilisés concrètement.

Les aspects de la logique floue que l'on veut qualifier de "flous" sont les règles et plus spécifiquement la façon par laquelle ils sont codés. Le codage des règles utilise des expressions linguistiques et non pas de représentations numériques. Cette vague interprétation de problèmes complexes est l'aspect qui donne à la logique floue sa puissance. Il permet aux utilisateurs de développer des contrôleurs pour des problèmes complexes sans avoir à remodeler entièrement le problème dans un optique rigoureusement mathématique.

Les contrôleurs de logique floue sont des systèmes règlementés qui font appel à des variables linguistiques pour adopter les approches de notre règle de trois (heuristique) à la résolution d'un problème.

Les contrôleurs de logique floue diffèrent des contrôleurs conventionnels en ce qu'ils se basent sur le concept qu'il existe une certaine imprécision dans la catégorisation des valeurs des variables. Cette caractéristique est utilisée pour améliorer la performance du système expert dans le processus de contrôle en fournissant une certaine flexibilité dans la représentation de la prise de décision humaine. Cette flexibilité est mise en oeuvre en incorporant la théorie d'ensemble de la logique floue dans laquelle des concepts abstraits et subjectifs sont représentés par des variables linguistiques, et contrôler un processus physique.

L'objectif de cette recherche est de développer un tel contrôleur en logique floue afin d'opérer un contrôle en temps réel de l'équipement de communciations par micro-ondes. On visait à développer un système automatisé pour déterminer les valeurs correctes pour les amplificateurs à transistors à effet de champ (FET) par micro-ondes, et on visait également à déterminer comment on peut modifier des valeurs comme fonction de température afin de maintenir constante la puissance de sortie.

Le problème a été cerné avec minutie et une investigation complète des méthodes actuelles de résolution a été évaluée. Le présent système fait appel à une approche manuelle et implique le recours à un expert afin de déterminer empiriquement les conditions de polarisation, et de déterminer la façon par la laquelle la polarisation devrait se modifier en fonction de la température.

Afin d'évaluer la véritable portée du travail effectuée dans le domaine du contrôle en logique floue, une recherche informatisée sur la documentation disponible a été menée.

Les résultats de cette enquête sur la documentation disponible indiquent que la recherche actuelle en ce domaine couvre plusieurs disciplines, et révèlent aussi qu'aucune recherche n'a été publiée dans le domaine du contrôle des micro-ondes.

Afin de déterminer de quelle façon la présente approche manuelle a été exécutée, une série d'entrevues a été faite avec un personnel ingénieur "expert". De ces entrevues, il a été développé un ensemble d'heuristiques qui sont utilisées à la base du système expert en logique floue.

Les principes de la connaissance en génie furent employés afin de soustraire l'heuristique de ces experts. La détermination de l'heuristique fut accompagnée par la compréhension du processus sous-jacent et de la nature itérative du processus. On conclut que la nature non-linéaire du problème de contrôle nécessitait le recours à un expert afin de réussir diverses itérations avant de converger vers une solution acceptable. Les entrevues indiquèrent qu'un grand nombre d'essais et d'erreurs entrait en ligne de compte et se basaient sur diverses règles heuristiques.

Une fois ces heuristiques déterminées, une méthode de mise en application fut recherchée. Le manque de précision des heuristiques ramena le problème à une approche en logique floue. Le compilateur flou "type C" de T.I.L. fut sélectionné pour la réalisation.

Une gamme de fréquence avec 5 paramètres de contrôle fut choisie pour l'expérimentation et le personnel ingénieur expert qui fut impliqué dans le développement et l'essai de cette unité fut interrogé. Le but était de contrôler 5 transistors à effet de champ (FET) et 2 atténuateurs de diode (PIN) et de maintenir un gain constant sur les extrêmes de température. En raison de la complexité du système, l'isolement des effets de chaque appareil et des effets de chacune des relations de paramètres se révéla difficile.

Cette heuristique fut codée dans le système expert en logique floue et fut évaluée pour un système simplifié consistant en un transistor à effet de champ (FET). Dans le but d'augmenter la dynamique du contrôleur en logique floue, une solution par échelon dans la puissance d'entrée fut appliquée. La fonction par échelon fut exécutée par le biais d'un générateur à fonction d'ondes carré, et en le connectant à la modulation d'entrée AM du signal d'entrée RF. Ainsi, l'enveloppe du signal d'entrée RF fut modifiée de 0 à 0.5 db. En utilisant ce système simple, les effets de plusieurs relations de fonctions de paramètres furent évalués avant de procéder à l'intégration des 5 paramètres de contrôle.

Un des aspects les plus difficiles dans la réalisation de ce système de contrôle en logique floue fut de déterminer les différents paramètres pour les fonctions de relation. Plusieurs mois de travail furent passés à analyser empiriquement les différentes combinaisons, et à parvenir à une bonne marche du système.

Le système qui résulte de tout cela, avec les 5 paramètres de contrôle, réussit à maintenir constant le gain de la gamme de fréquence sur la température, et réussit à démontrer que le concept de logique floue peut être appliqué à la polarisation de la gamme de fréquence.

ACKNOWLEDGMENTS

I wish to thank Professor Dinkar Mukhedkar for his dedication, encouragement and his many hours of assistance in the completion of this work.

The generous donation of the software program "Fuzzy-C Compiler" by the company Togai Infra Logic is greatly appreciated and was indispensable in the completion of this work.

I would also like to thank Spar Aerospace Ltd. for the use of their facilities.

TABLE OF CONTENTS

LIST OF FIGURES

INTRODUCTION

The term "fuzzy logic" is often misleading in that it implies a type of logic which is imprecise and therefore of little value. In reality, fuzzy logic controllers use crisp inputs, perform calculations in a rigorous mathematical manner and provide crisp outputs which can be used in the real world.

The aspects of fuzzy logic which are fuzzy are the rules and more specifically the manner in which they are coded. The coding of the rules for fuzzy logic systems utilizes linguistic expressions and not numerical representations. This vague interpretation of complex problems is the aspect which gives fuzzy logic it's power. It enables users to develop controllers for complex problems without having to fully model the problem in a rigorous mathematical sense.

Fuzzy logic is an extension to conventional Boolean logic. In Boolean logic a given premise can be either TRUE or FALSE. No intermediate states are allowed. In fuzzy logic these intermediate states are permitted and can therefore allow partial truths.

Fuzzy logic controllers are rule based systems that use fuzzy linguistic variables to model human rule of thumb (heuristic) approaches to problem solving.

Fuzzy logic controllers differ from conventional controllers in that they are based on the concept that some uncertainty exists in categorizing the values of variables. This feature is used to enhance the performance of expert systems in process control by providing some manner of flexibility in the representation of human decision making. This flexibility is implemented by incorporating fuzzy set theory in which abstract or subjective concepts are represented by linguistic variables and are in turn used to control a physical process.

The goal of this thesis was to design and implement a fuzzy logic controller. The objective was to develop an automated system to determine correct bias values for microwave FET amplifiers and to determine how to modify these values as a function of temperature in order to maintain a constant output power. The research was geared towards the manufacturing environment and addressed those issues relevant to the manufacturing process. The facilities of Spar Aerospace Ltd. in Montreal, Canada were used to perform the experimental portion of the research.

The test bed for the expert system was a microwave amplifier from the Canadian Anik-E spacecraft. The Anik-E spacecraft is a communications satellite which consists of a C-Band and a K-Band transponder. The electronic equipment and microwave amplifiers are mounted on panels which form the outside walls of the spacecraft as shown in Figure 1 and Figure 2.

The amplifier used for the experiments in this thesis was mounted on the K-Band equipment panel and was called a Channel Amplifier. The role of a channel amplifier is to amplify the signal which is to be transmitted to earth to the level required to drive a power amplifier. The output signal from the power amplifier is then fed to the K-Band antenna system and transmitted to earth.

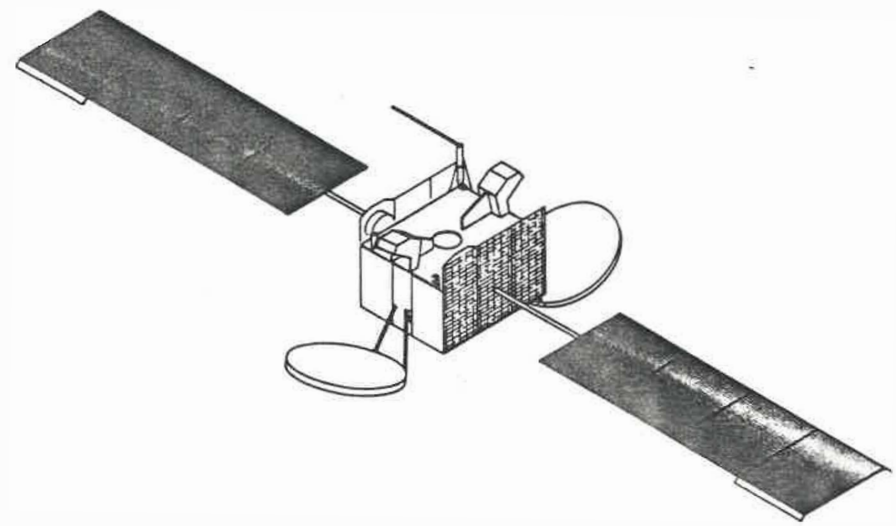


Figure 1 View of Anik-E spacecraft. (with permission of Spar Aerospace Ltd)

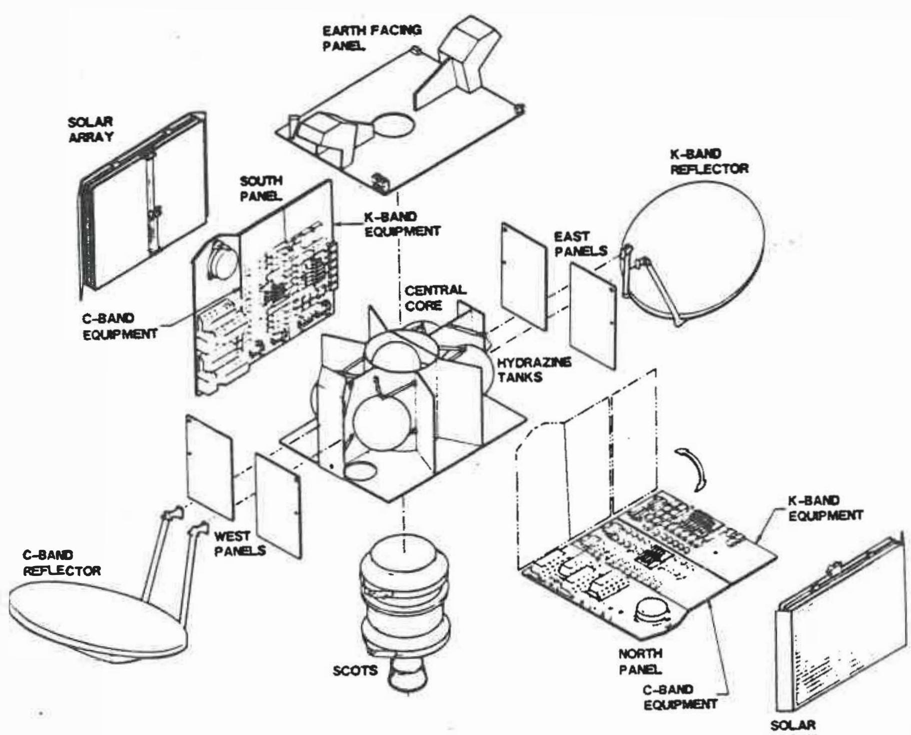


Figure 2 Exploded view of Anik-E spacecraft. (with permission of Spar Aerospace Ltd.)

The channel amplifier is a critical component in the satellite system in that any degradation or failure in the amplifier would affect the entire communication system. For this reason during the manufacturing process special care is taken to ensure that the long term reliability of the amplifier is assured and that the performance will stay within specification limits. The design life of the channel amplifier is 12 years.

The channel amplifier consists of 5 active elements, 2 electronically controlled PIN diode attenuators and 3 FET amplifiers. The gain of the channel amplifier can be varied by changing the bias to these control elements.

In normal operation the channel amplifier will see temperature changes which will vary between -10 and +50 degrees Celsius. This temperature variation is due to the effect of spacecraft heating by the sun. The position of the channel amplifier panel in relation to the sun will determine the temperature of the amplifier.

The temperature changes of the channel amplifier have an effect on the gain of the amplifier. As the temperature is decreased the gain increases and as the temperature increases the gain of the amplifier decreases. Since the channel amplifier is used to drive a power amplifier it is very important to maintain the gain constant in order to provide the correct signal level to the power amplifier. This temperature compensation is accomplished by varying the gate voltage at each FET amplifier and also by varying the bias of the PIN diode attenuators. In this way the gain of the channel amplifier can be increased or decreased.

The present method of temperature compensation is a manual method which involves the use of human experts trained in the compensation process. The human expert accomplishes the temperature compensation by manipulating the bias voltages through a potentiometer. By visually monitoring the gain of the channel amplifier the human expert turns 5 different potentiometers which in turn control the bias values of the 5 active elements in the channel amplifier.

The principles of knowledge engineering were employed to extract the heuristics used by these experts to accomplish this temperature compensation. The determination of heuristics was accompanied by the understanding of the underlying process and the iterative nature of the temperature compensating process.

It was found that the nonlinear nature of the control problem required the expert to perform several iterations before converging to an acceptable solution. The interviews with the experts indicated that a great deal of trial and error was used and was based on several heuristic rules.

Once these heuristics were determined a method of implementing them was investigated. The lack of crispness of the heuristics made the problem amenable to a fuzzy logic approach. The TIL Fuzzy C compiler was selected for the implementation.

These heuristics were coded into the Fuzzy expert system and were evaluated for a simplified system consisting of 1 FET amplifier stage. In order to assess the dynamics of the Fuzzy Logic controller a step response in the input power was used. The step function was implemented by using a square wave function generator and connecting it to the AM modulation input of the RF signal source. In this way the envelope of the RF input signal was varied from 0 to + 0.5 db. Using this simple system the effects of various membership function parameters were evaluated before proceeding to the integration of all 5 control parameters.

One of the more difficult aspects in the implementation of this fuzzy logic control system was the determination of the various parameters for the membership functions. Many months of work were spent analyzing the various combinations empirically and fine tuning the system.

The resulting system, with all 5 control parameters, was successful in maintaining the gain of the RF slice constant over temperature and was successful in demonstrating that the concept of fuzzy logic can be applied to the biasing of RF slices.

CHAPTER 1

THEORY OF FUZZY LOGIC

1.1 INTRODUCTION

Fuzzy logic is an extension to conventional Boolean logic. In Boolean logic a given premise can be either TRUE or FALSE. No intermediate states are allowed.

In fuzzy logic these intermediate states are permitted and can therefore allow partial truths.

For example, a 50 year old man is not OLD and yet he is not NOT(OLD). In fuzzy logic this problem would be implemented by creating a function relating age in years to oldness. In this way one can represent the 50 year old man's situation by quantifying that the 50 year old man is OLD is 60% TRUE. This also implies that the premise that he is NOT(OLD) is 40% TRUE. In fuzzy logic an assertion can be TRUE and FALSE at the same time.

Fuzzy logic is derived from the more general theory of fuzzy sets. If we use fuzzy sets to represent the above example we would create a function relating age to oldness. This curve would relate the age of a person to the degree of membership that person has in the fuzzy set OLD as a function of age given by $\mu_{OLD(age)}$.

In order to be consistent with conventional Boolean logic the values of μ range from 0 to 1. A value of μ=0 indicates that there is no membership in the fuzzy set and a value of μ=1 indicates that there is the highest degree of membership.

An example of a membership function is shown in Figure 3. The membership function relates age to the fuzzy set OLD. As can be seen from Figure 3 the 50 year old man has a membership in the fuzzy set of 0.6. Put in another way, one can say that the 50 year old man is a member of the fuzzy set OLD with a degree of membership $\mu_{OLD=0.6}$.

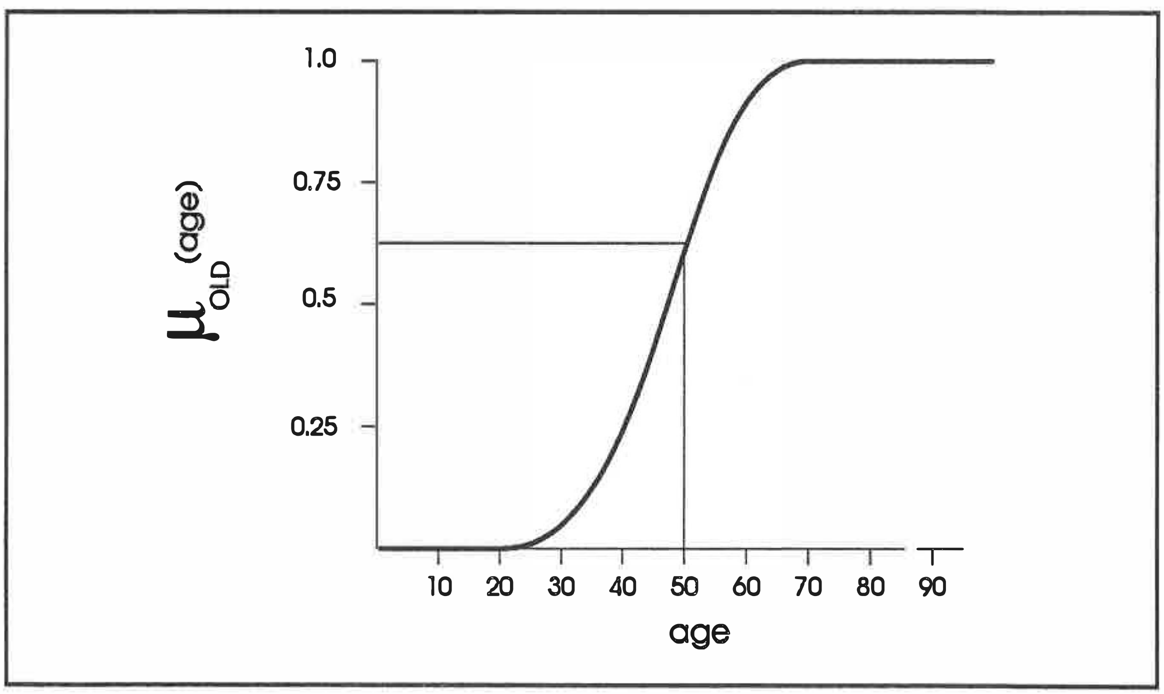


Figure 3 Membership function for the fuzzy set OLD indicating age of 50 year old man.

Manipulations of fuzzy variables are similar to those which can be applied to Boolean variables however they are defined differently. The three standard fuzzy operators are: AND, OR and NOT. These are related to the fuzzy set operators of INTERSECTION, UNION and COMPLIMENT respectively.

Given two fuzzy variables A and B the following definitions for fuzzy operators are:

A AND B = min(μA, μB)

A OR B = max(μA, μB)

NOT A = 1 - μA

The implementation of these operators into fuzzy logic controllers will be covered in this chapter as well as the translation between crisp and fuzzy variables through membership functions, how rule inferences are implemented and how a crisp output is obtained from the fuzzy variables. A literature survey is also presented to indicate the type of work that has been done in the field of fuzzy logic.

1.2 LITERATURE SURVEY

In order to assess the true scope of the work done in the field of fuzzy logic control, a computer literature search was performed. The search was limited to the past 15 years and yielded 221 papers all relating to the theory and implementation of fuzzy logic systems.

The results of this literature survey indicated that the present work in this area covers many disciplines and also revealed that no work has been published in the microwave control area which form the basis for this thesis.

The following is a short survey of some of the more interesting papers and spans a very wide range of application. It is presented here to provide the reader with a feeling for the scope of the present applications in the fuzzy logic area and does not represent a complete and exhaustive literature search.

1.2.1 Medical Applications

The role of fuzzy logic as an aid to the medical diagnostic problem has been a very popular one since the number of medical variables and the inaccuracies of medical tests result in the extensive use of heuristics by the physician in pronouncing a diagnosis.

Based on this premise Adlassnig, Kolarz and Scheithauer (1985) have developed a medical diagnostic program called CADIAG. It is a fact that uncertainty of knowledge about the patient and about medical relationships are generally accepted and considered to be an inherent concept in medicine. The physician, however, is capable of drawing conclusions from this uncertain data. The use of a fuzzy logic expert system provided the possibility of using linguistic concepts and yielded excellent approximations to medical results.

In a second paper Adlassnig, Kolarz, Scheithauer, Effenberger and Grabner (1985) describe the performance of the CADIAG system. A total of 426 cases with rheumatic and 47 cases with pancreatic diseases were tested. The overall accuracy was 91.1% for rheumatic diseases and 100% for pancreatic diseases. These high success rates indicate the power of fuzzy logic in dealing with uncertainty.

A powerful concept in fuzzy logic is the use of weighting factors to integrate fuzzy values into crisp responses. The concept of weighted factors was introduced in the MYCIN program by Franczyk (1988). Through the use of a flow chart he described how the input ill defined knowledge was dealt with using the concept of probability weighted factors, combination functions and certainty factors.

A system for controlling human arterial pressure was described by Hao et al (1988). Human arterial pressure was controlled by regulating the infusion rate of the drug sodium nitroprusside. The system resulted in satisfactory results and was also used to study the effects of different fuzzy controller parameters

The analysis of chest pain was described by Hudson (1987). Techniques for the analysis of uncertainty were analyzed and a demonstration of these fuzzy logic techniques were applied to the analysis of chest pain.

A general analysis of the representation of uncertainty in medical systems was described by Hughes (1989).

Linkens et al (1988) developed a fuzzy logic expert system to act as an advisor for patient anaesthesia. The authors aimed to develop an expert system which would advise an anaesthetist on the current depth of the anaesthetic state of the patient in an operating theatre. The role of the expert system was to suggest measures necessary to regain a desired state of anaesthesia. Initial results were successful and the system has been further improved with the addition of speech output and a more user friendly interface.

The use of fuzzy expert systems in the orthodontic area were presented by Sims et al (1989). The system was designed to assist non specialized dentists with orthodontic problems. The system was found to be useful in modelling the thought processes of orthodontic specialists and crystallizing this knowledge

1.2.2 Control System Applications

The following papers were focused on the control of various process parameters and illustrate the use of fuzzy logic expert systems in the solution of complex control problems.

Aoyagi et al (1988) described a fuzzy logic expert system used to perform the start up sequence of a fossil fuel power plant. The expert system involved the marriage of quantitative calculations and qualitative knowledge. The quantitative calculations were based on plant dynamics models. The qualitative knowledge consisted of schedule modification rules with inherent fuzziness, which represent the relationships between stress margins and modification rates of the system parameters. The system was verified and showed to provide quick and accurate plant start ups.

A similar approach was used by Bare et al (1988) in the design of a fuzzy logic expert system to perform the self tuning of a gasoline refinery. The system was divided into two parts the supervisory control and system optimization. The knowledge bases for the controllers was established through the use of knowledge engineering and from basic engineering knowledge. The system was found to perform well.

Batur and Kasparian (1989) reported success in controlling the essential parameters of a DC motor through the use of a fuzzy logic control system.

The special problems of real time control were addressed by Chiu and Togai (1988). The development of a practical fuzzy logic implementation were addressed and the problems associated with the implementation of the fuzzy expert system into a single chip computer were also addressed.

Clema et al (1985) described another expert system for real time applications specifically the "Pilots Associate" for the U.S. air force. The pilots associate integrates the highly complex knowledge of a fighter pilot and presents the information in a manner and at a time when it is most needed. The expert database was derived from knowledge engineering and the specific flight and system characteristics of the airplane.

The implementation of rule based expert systems were described by Efstathiou (1987) and focused on the implementation aspects of simple control systems. The limitations of some of the control schemes was addressed.

Hirota (1989) describes the development and use of VLSI fuzzy logic chips for process control.

An example of a practical fuzzy logic expert system which is now coming on the market is the automotive anti skid system. Matsumoto et al (1987) developed such a system and based the control values on the road surface conditions. This autonomous system is capable of detecting the present state of the wheel and to detect the road surface and provide the appropriate control parameters. This is a classic example where the inputs are fuzzy yet a crisp output value is generated.

Another practical application is the control of a rotary dryer described by Pietranski et al (1987).

The subject of servo control is addressed by Sripada et al (1987). The results indicated that the fuzzy logic controller performed better than a conventional PID controller. The fuzzy logic controller was implemented in three phases depending on where the system was. Smooth control was implemented near the set point, 'typical' feedback control in the intermediate region and, far away from the set point, as much control as necessary was permitted to prevent the controlled variable from exceeding user specified limits.

Another VLSI chip to implement fuzzy logic was described by Togai and Watanabe (1986). The design of an inference architecture and its implementation on a VLSI chip was described.

A unique application of fuzzy logic was in the control of an elevator system described by Tsugi et al (1989). The optimum elevator car to answer a call was determined

A unique application of fuzzy logic was in the control of an elevator system described by Tsugi et al (1989). The optimum elevator car to answer a call was determined by using knowledge bases, production rules and fuzzy rules extracted from the knowledge of experts. The use of fuzzy logic resulted in considerable improvement, reducing the average waiting time by 15-20% and long waits by 30-50% compared to conventional methods.

1.2.3 Nuclear Power Station Applications

A special case of the role of fuzzy logic is in the supervision and control of nuclear power stations.

Bernard (1988) designed a fuzzy logic expert system to control the MIT 5 MW experimental reactor. A comparison between the fuzzy logic rule based approach and the conventional analytical approach was given. The main conclusion was that the rule based approach is more robust than the analytical approach.

Beuenaflor and Finch (1988) describe a simulation for a nuclear power plant. The role of simulation is very important in the nuclear plant since the consequences of an error during normal operation are very severe. This fuzzy logic expert system was used to explore the various facets of control in a nuclear power plant and was used during on site exercises.

An application in establishing a constant heat balance in a nuclear reactor was given by Guth (1987). The interface between the heat exchanger and the core of a nuclear power plant was used as the control problem. Based on rules generated by interviews with experts, Kitoedki and Ksiazek (1985) described the diagnostic use of fuzzy logic as well as that of fault detection in a nuclear power plant. The complexity of the plant puts a great deal of stress on the operator and it was found that this expert system could be used to supplement the operator's work.

Lee et al (1987) discussed the general problem of coding rules and described the use of an expert system to diagnose component failure in a nuclear power plant.

1.2.4 General Theory Papers

This section describes some general papers that deal with the theory of fuzzy logic and with the various techniques for implementing them.

The subject of diagnosis was well described by Kitowski and Bargiel (1987). The problem of diagnosing physical system failures was also addressed.

Togai (1985), Buckley (1988), Jingqiu and Hong (1989) and Graham (1987) present good review and theoretical papers describing the various techniques used to implement fuzzy logic expert systems.

The concept of modelling was addressed by Kara-Zaitri (1989). The methodology presented dealt with cases where detailed standard probabilistic assessments are not justified, quantitative data is not available and a very large number of failure modes are possible. The use of fuzzy logic was described as a solution to overcome some of these problems.

possible. The use of fuzzy logic was described as a solution to overcome some of these problems.

The father of fuzzy logic, L.A. Zadeh, has published a number of papers all dealing with the theoretical background of the mathematical basis for fuzzy logic. These papers are listed in the bibliography.

1.2.5 Oil

The use of fuzzy logic is well suited to the oil prospecting area since the prospector must deal with many unknown parameters and must rely on a large heuristic knowledge base. An example of the use of fuzzy logic is given by Lebailly et al (1987) who dealt with the special problems of the interpreting the data generated by prospect appraisal experiments.

1.2.6 Pattern Recognition

Schneider et al (1988) described how the use of fuzzy logic has improved the pattern recognition process by reducing the search space for each feature. This has resulted in a direct increase in the speed of the system.

1.2.7 Damage Assessment

Shiraishi et al (1989) developed a system to assess the damage states of bridge structures. The use of a fuzzy logic system made it possible to deal with various kinds of uncertainty and ambiguity in the data, rules and inference processes.

Vaija et al (1985) described a failure detection system designed for complex processes with ill defined mathematical models and poor measurement accuracy.

1.2.8 Space Station

Vachtesevanos and Davey (1987) and Vachtesevanos et al (1986) described the work being performed on the U.S. space station thermal subsystem.

1.2.9 Management

Whalen (1984) explored the role of fuzzy logic in the decision making process in a management role. Some financial application were explored by Ganoe et al (1986). Chang and Ibbs (1989) have looked into the problem of resource allocation in the construction industry.

An application of fuzzy logic expert systems to scheduling was shown by Kerr and Walker (1989) in the scheduling of a machine shop. The dynamic nature of the type of work flowing through the machine shop made any schedule quickly out of date. The use of an expert system with the concept of fuzzy logic overcame the problem of one of a kind type work.

1.3 FUZZY LOGIC CONTROLLERS

1.3.1 Definition

Fuzzy logic controllers are rule based systems that use fuzzy linguistic variables to model human rule of thumb (heuristic) approaches to problem solving.

Fuzzy logic controllers differ from conventional controllers in that they are based on the concept that some uncertainty exists in categorizing the values of variables. This feature is used to enhance the performance of expert systems in process control by providing some manner of flexibility in the representation of human decision making. This flexibility is implemented by incorporating fuzzy set theory in which abstract or subjective concepts are represented by linguistic variables and are in turn used to control a physical process.

The field of fuzzy logic controllers is relatively new with several Japanese systems in full scale production and relatively few North American systems in place although several are under development. Some of the more successful applications of fuzzy logic controls are: Photographic Automatic Camera Focusing, Anti lock braking systems, air conditioner temperature controllers and even washing machine controllers which detect how dirty your clothes are and adjust the detergent quantity, water temperature and wash time.

The following sections will describe the steps normally required to implement a fuzzy logic controller. In addition to describing these steps, some of the theory of fuzzy logic will be described and the implementation of this theory will be illustrated.

1.3.2 Determination of Decision Variables

The first step in implementing a fuzzy logic controller is to determine which decision variables are most important and will be most effective in producing the desired control action. It is extremely important that only the most important decision variables are chosen since the selection of a large number of variables will greatly increase the size of the rule base and will also make the fine tuning of the controller very difficult. The decision variables normally consist of the parameters which are measured by an operator. Typical examples of decision variables are temperature, pressure, voltage, and current.

1.3.3 Determination of Control Variables

Once the decision variables have been selected, the control variables must be defined. There are no restrictions on the number of control variables since they do not effect the size of the rule base. Their selection is usually quite simple in that they represent the 'knobs' of the system that the human operator must turn to effect control of the system.

1.3.4 Determination of Fuzzy Rules

The next step in the process is to determine the necessary actions required to control the system. These actions are derived from the knowledge of experts and are coded into rules which employ linguistic variables. A rule is provided by a human expert for each possible condition of input variables and provides a fuzzy action for each of these conditions. This rule coding is similar to a conventional expert system except that the rules employ linguistic variables and are therefore easier to code, more closely reflect the knowledge of the expert and are more efficient.

These rules describe how the decision variables are processed and how they are linked to the control variables. This step is a most difficult one, in that the human decision making process must be modeled and then implemented into a series of linguistic rules. The use of these linguistic rules is what gives fuzzy logic controllers their power and their advantage over conventional controllers.

The rules are written in a form so that they encompass all possible conditions which may exist in the physical system. The rules are written in the form:

IF[$input_1$ is ($Input\_Condition_1$) and $input_2$ is ...]

then [$control_1$ is ($Control\_Condition_1$) and $control_2$ is ....]

Where $input_1$ is the input decision variable and $Input\_Condition_1$ is the fuzzy set characterizing the decision variable. $Control_1$ is the output control variable and $Control\_Condition_1$ is the fuzzy set characterizing the output control variable.

1.3.5 Fuzzy Membership Function

In order to implement these fuzzy rules in a practical expert system some way of inferring a crisp output from the fuzzy system is required. The Fuzzy Membership Function performs this translation.

A fuzzy membership function is an approximation of the degree of confidence, or membership, that the value of the crisp input variable belongs to the fuzzy set. Fuzzy membership values (f) are numerical representations of these confidence values. The range of values that a membership function value can have fall between 0 and 1. When f= 1 maximum confidence is given and indicates that the crisp value is very well represented by the fuzzy set. When f=0 a minimum confidence value is assigned and indicates that the fuzzy set does not provide a good representation of the crisp value. For each crisp input variable, each fuzzy set has a membership function value although in some cases this value may be 0.

An example of this concept is shown in Figure 4. The example shows how a crisp input, in this case the temperature of a room is converted to a degree of membership. In this case there are three fuzzy membership functions being used; Low, Medium and High.

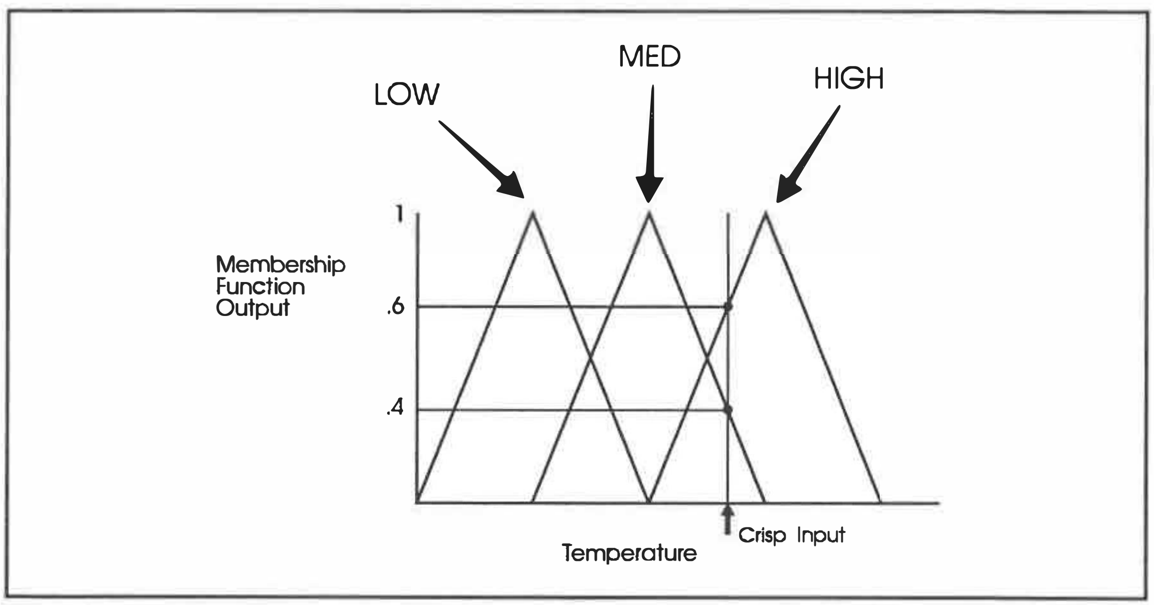


Figure 4 Example of translation between crisp input and fuzzy membership function.

Each fuzzy membership function overlaps the adjacent one in order to provide some continuity. This continuity is required to prevent the control system from oscillating between two states and provides for smoother control between system states. In a binary control system the control output would be switched from completely OFF to fully ON as in a house thermostat. In the fuzzy logic implementation this smooth transition gives a greater level of control to the system and provides for a smoother response.

In the example the crisp input falls between the fuzzy membership functions medium and high. The degree of membership for each membership function is computed by determining the level of membership that the crisp value has in each set. In the example the crisp value has 0 membership in the Low fuzzy set, a degree of membership of .4 to the Medium fuzzy set and .6 to the High fuzzy set. From this example it can be seen that for any crisp input all fuzzy sets are activated but to different degrees. This degree of activation is measured by the membership function output.

1.3.6 System Control and Defuzzification

The final phase in the implementation of a fuzzy logic controller is the conversion of the fuzzy control output to a crisp value that can be used to control a physical process. This process, called defuzzification, selects the centre of gravity (centroid) of the output membership functions as the crisp output value. An example of this process is shown in Figure 5.

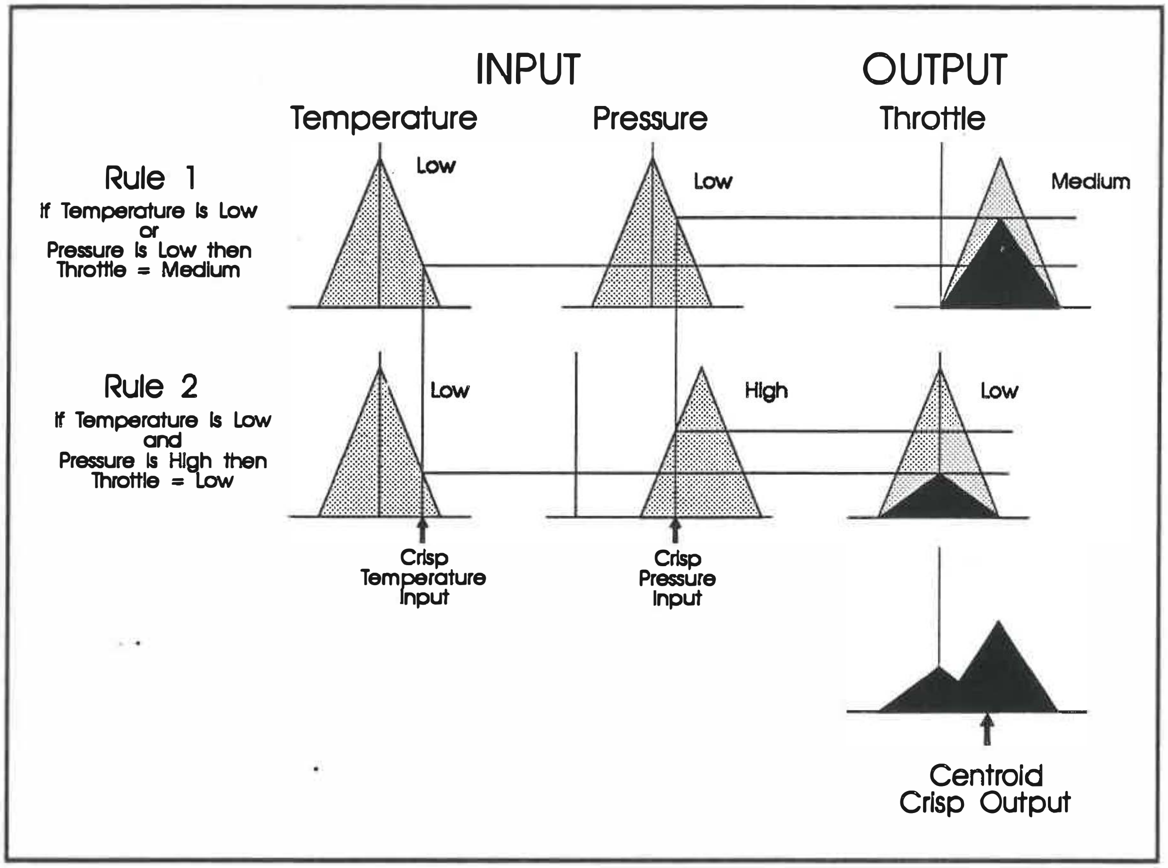


Figure 5 Example of fuzzy inference process and of defuzzification

This example illustrates a simple throttle control for an automobile engine. The inputs consist of the measured values of temperature and pressure and the output is the position of the motor throttle. Two rules are shown but in a real system many more rules would be used.

The LOW input membership function for temperature represents temperatures close to zero and is therefore centred about the Y axis. A temperature of 0 degrees would yield a degree of membership of 1.0. Any other temperature would give a degree of membership less than 1 and this degree of membership would be a function of the slope of the LOW membership function.

The membership functions for pressure are LOW and HIGH and are centred about 0 for LOW and about a higher value of pressure for HIGH. The HIGH membership function is defined in a similar manner to the LOW membership function except that it is not centred about the Y axis but about an intermediate pressure value. This intermediate pressure value is the value at which the HIGH membership function would yield a degree of membership of 1.

The output membership functions consist of LOW throttle settings and MEDIUM throttle settings. The output membership function determines the final throttle setting.

The first rule implements the union of the LOW temperature membership function and the LOW pressure membership function. The union (OR) operation takes the maximum of the degree of membership of each membership function as the value to scale the output membership function. In Figure 5 the crisp temperature input lies close to the edge of the triangular LOW temperature membership function whereas the crisp pressure input lies closer to the centre of the LOW pressure membership function. Therefore, the crisp pressure input has a greater degree of membership to the LOW pressure membership runction than the crisp temperature input had to the LOW temperature membership function. In the rule governing these inputs the OR operator is used. The OR operator is defined as the maximum of the two degree of memberships. Using this definition the crisp pressure input has resulted in the greatest degree of membership and therefore the output throttle MEDIUM membership function is scaled to this degree of membership.

This scaled output membership function is shown in Figure 5 as a black triangle that is a scaled version of the original throttle MEDIUM membership function.

In a similar fashion the second rule implements the intersection of the temperature and pressure membership functions. The intersection (AND) operation selects the minimum of the two input membership functions. In this case the Temperature membership function gave tne smallest degree of membership and therefore the output Throttle membership function was scaled to this value.

In order to produce a crisp output from the two Throttle membership functions a process called defuzzification is applied. There are several techniques for defuzzication. The method used in this thesis is the centroid defuzzification method. This method selects the output of the fuzzy expert system corresponding to the centroid or centre of gravity of the output membership function.

A numerical example of the centroid defuzzification method is shown in Figure 6. In Figure 6 two membership functions for temperature are shown. One representing a LOW temperature and one representing a Medium temperature. The height of each membership function has been scaled as a result of the application of the fuzzy rules. The objective now is to determine a crisp output for the system.

The first step in performing a centroid defuzzification is to compute the area and the moment of each triangular membership function. The moment is computed about the

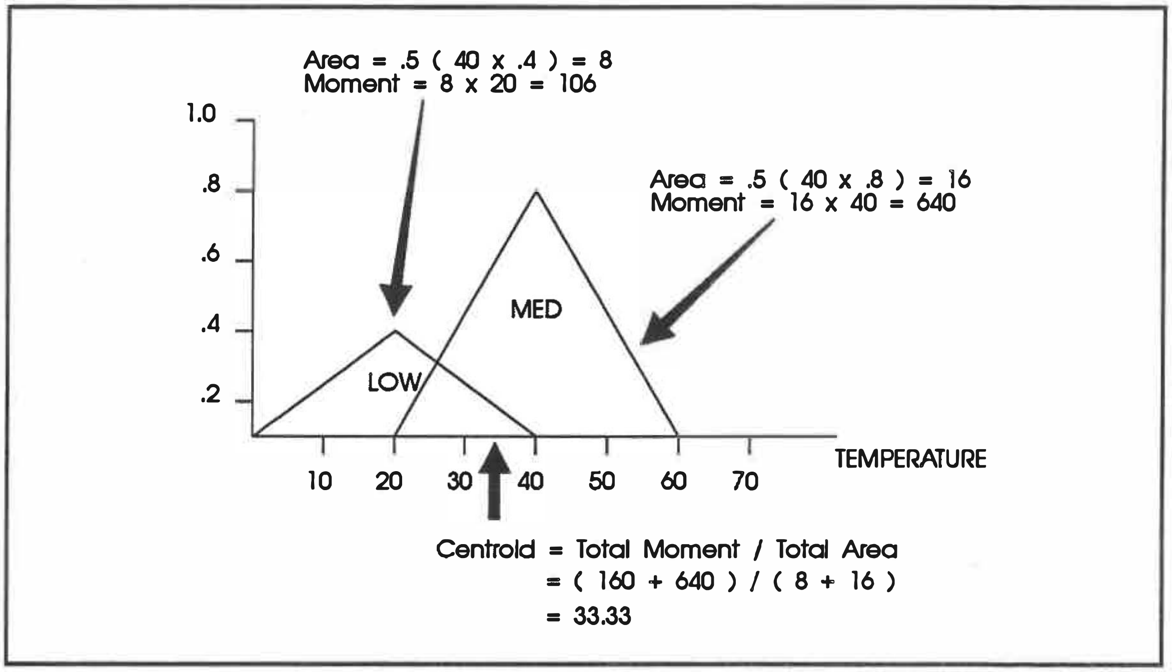


Figure 6 Numerical example of defuzzification process.

midpoint of the base of the triangle and gives an area of 8 and a moment of 106 for the LOW membership function and an area of 16 and a moment of 640 for the MEDIUM membership function. In order to compute the centroid of the two membership functions the areas and the moments are added together and then the centroid is computed by dividing the total moment by the total area as shown in Figure 6.

Defuzzification, in Figure 5, is implemented by combining the two throttle membership functions and finding the centre of gravity or centroid of the composite structure. This is shown in Figure 5 with the result of the two rules being a centroid between a Low and Medium throttle setting and slightly closer to the Medium throttle value. It is this centroid value which is sent to the throttle controller and determines the position of the engine throttle.

This simple example has illustrated the manner in which fuzzy logic controllers operate. In a real system the number of rules would be larger but the basic operation of the system would be the same.

CHAPTER 2

DEVELOPMENT OF EXPERT SYSTEM

2.1 DEFINITION OF THE PROBLEM

The objective of this research was to develop an automated system to determine the correct bias values for microwave FET amplifiers and to determine how to modify these values as a function of temperature in order to maintain a constant output power. The research was performed in a manufacturing environment and addressed those issues relevant to the manufacturing process. The facilities of Spar Aerospace Ltd. in Montreal, Canada were used to perform the experimental portion of the research.

The channel amplifier used for these experiments consisted of 5 active elements, 2 electronically controlled PIN diode attenuators and 3 FET amplifiers. The gain of the channel amplifier was varied by changing the bias to these control elements. A photograph of the channel amplifier is shown in Figure 7.

In addition to the RF amplifier chain the channel amplifier also included a voltage regulator, telemetry circuit and a command circuit. The function of the voltage regulator was to condition the spacecraft bus voltage and to generate several secondary voltages to bias the FET amplifiers and the PIN diode attenuators.

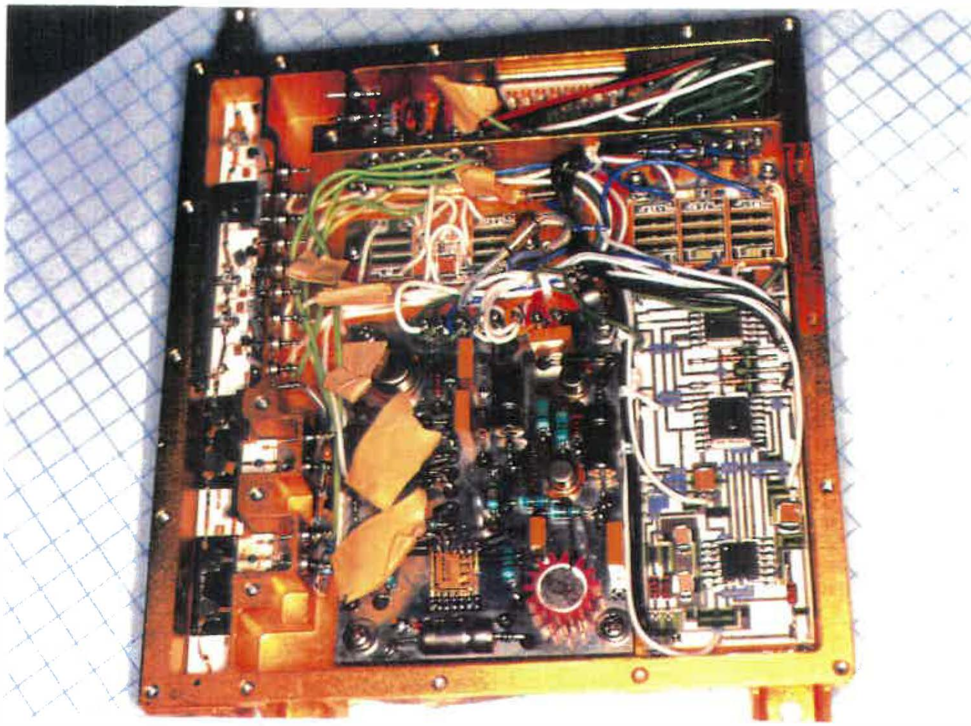


Figure 7 Channel amplifier used for experimentation.

The command and telemetry circuits were used to receive commands from the spacecraft bus and to send telemetry to the spacecraft bus. Commands for turning the channel amplifier ON and OFF were relayed through the command circuit.

In normal operation the channel amplifier will see temperature changes which will vary between -10 and +50 degrees Celsius. This temperature variation is due to the effect of spacecraft heating by the sun. The position of the channel amplifier panel in relation to the sun will determine the temperature of the amplifier.

The temperature changes of the channel amplifier have an effect on the gain of the amplifier. As the temperature is decreased the gain increases and as the temperature increases the gain of the amplifier decreases. Since the channel amplifier is used to drive a power amplifier it is very important to maintain the gain constant in order to provide the correct signal level to the power amplifier. This temperature compensation is accomplished by varying the gate voltage at each FET amplifier and also by varying the bias of the PIN diode attenuators. In this way the gain of the channel amplifier can be increased or decreased.

The present method of temperature compensation is a manual method which involves the use of human experts trained in the compensation process. The human expert accomplishes the temperature compensation by manipulating the bias voltages through a series of potentiometers mounted in a bias box. The bias box is connected to the channel amplifier and controls each gate and PIN diode voltage independently. A photograph of the channel amplifier bias box is shown in Figure 8. The bias box consists of 5 multi turn potentiometers which are manipulated by the human expert.

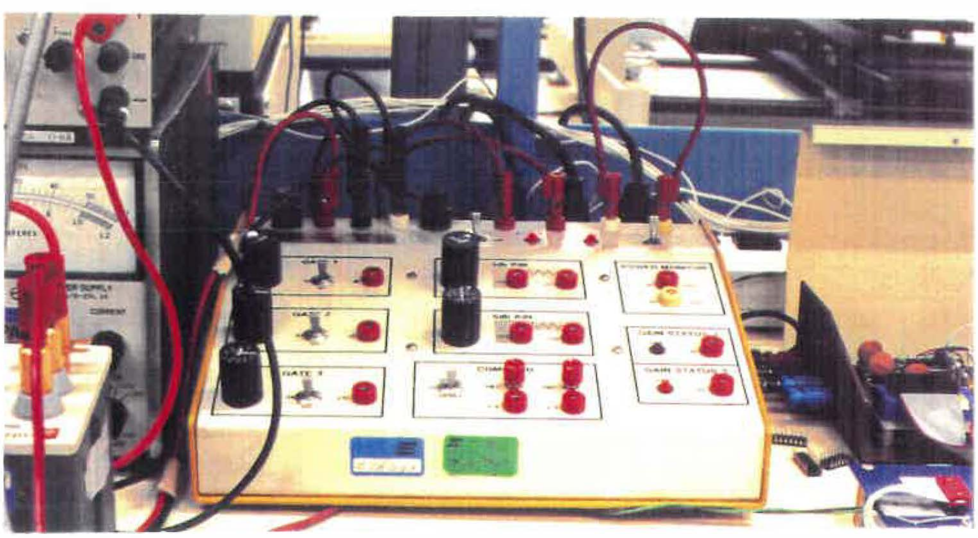


Figure 8 Bias box used for manual control of channel amplifier.

By visually monitoring the gain of the channel amplifier the human expert turns these 5 different potentiometers to change the gain of each amplification stage and thereby control the gain of the channel amplifier. Once these potentiometer settings are determined the required bias compensation is implemented through a thermistor bias circuit.

2.2 KNOWLEDGE ENGINEERING

In order to determine the ways the present manual approach is implemented a series of interviews were held with "expert" engineering personnel. The interviews consisted of an introductory dialogue to discuss the problem in general terms and to put the interview into perspective. It was very important to put the "expert" at ease and to make sure that he understood that the objective was not to put into question his expertise or to gauge how much of an expert he really was.

The next step in the interviews was to explore the thought processes that the experts went through in solving the thermal bias compensation problem. This was done by having the experts perform a typical bias compensation on the channel amplifier and noting the types of responses that they had when confronted with different situations.

In addition to these interview a series of experiments was performed with the channel amplifier to explore the various temperature compensation schemes and to observe first hand the though process that the expert personnel went through.

From these interviews and experiments the following major heuristics were derived.

2.2.1 Heuristic #1

The level of each FET is initially set at room temperature to obtain an equal drain current of 15 mA in each FET. The gate voltages for each FET required to obtain this level of current were noted by the expert and were used as a starting point for bias compensation.

2.2.2 Heuristic #2

During thermal transitions the output power was monitored and the effect of each FET bias voltage was tried in bringing the response back to the reference setting. When questioned carefully the expert would indicate that he first determined the sensitivities of each FET to the gain of the channel amplifier and would adjust the potentiometers accordingly. The expert would use phrases such as "gate 1 is a little less sensitive than gate 2 so I will not turn the potentiometer as much for gate 2".

2.2.3 Heuristic #3

During thermal transitions the expert would monitor the rate of change of the channel amplifier gain and would modify the amount of gate voltage change he induced at each gate based on the rate of change of channel amplifier gain.

2.2.4 Heuristic #4

The expert monitored the deviation that each FET and PIN potentiometer control made from the its nominal settings. He had a rule that each gate value should deviate as little as possible from the nominal setting.

The experts rationale for this rule was that there were other parameters which were effected by extreme bias compensation values and in order to avoid these extreme bias values the change of each control potentiometer were minimized. The expert tried to spread out the compensation amongst all 5 control parameters. An example of such a parameter is the intermodulation distortion which would degrade at an extreme bias value.

2.3 SOFTWARE DEVELOPMENT

Based on the above heuristics a fuzzy logic expert system was developed. The implementation was greatly enhanced through the use of the software package Togai Infra Logic Fuzzy C Compiler. This compiler is designed to assist in the writing, testing and debugging of fuzzy logic expert systems. The rule bases are written in a simple high level language called the TIL Fuzzy Programming Language and are then compiled into standard ANSI C source code. This C source code can then be compiled and linked to the main C program. In this way the Fuzzy C expert system can be incorporated into a standard C language program.

The structure of the expert system is shown in the flow chart in Figure 9. The main C language program called Biasdrv.C was used to control the hardware and to call the fuzzy logic expert system routines. A listing of the program is contained in Appendix E.

The main program starts by initializing a data acquisition board (Data Translation DT2801A) contained in the PC. This data acquisition board was used to measure the temperature and gain of the channel amplifier. The DT2801A board was controlled through the low level routines contained in the program DT2801.C. A listing of this program is contained in Appendix A. After the DT2801 board has been initialized the main program opens a ramdisk file to store measurement data. At this point the control loop starts with the main program reading in the temperature and gain of the channel amplifier. This information is passed to the fuzzy logic expert systems which return the amount that each of the 5 control outputs should be changed. The main program modifies the bias of each control output through the use of the program Setbias.C. A listing of this program is contained in Appendix B. In order to determine the uncompensated gain of the channel amplifier the nominal settings for each of the 5 control outputs is set and the uncompensated gain is read. The compensated outputs are restored and the measurements are stored in the ramdisk file. At this point the program loops back and reads the new gain. This loop continues until it is stopped by the operator.

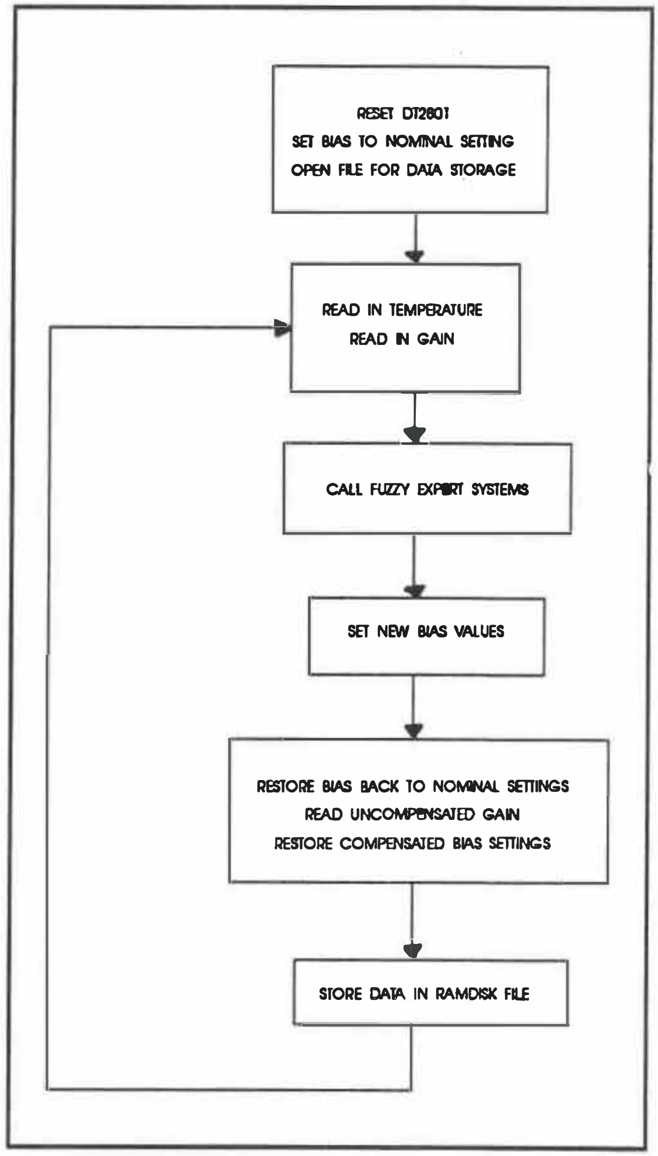


Figure 9 Flow chart of main control program.

The fuzzy logic expert system was developed based on the heuristics derived from the experts. A block diagram of the fuzzy expert system is shown in Figure 10. From Figure 10 it can be seen that there are a total of 6 fuzzy expert systems each with two inputs and one output.

The two inputs to the first expert system consist of the gain of the channel amplifier and the rate of change of gain (Delta Gain). The output of the first expert system was designed to indicate the amount by which the gain of the channel amplifier must be changed in order to maintain a constant gain.

The remaining 5 expert systems serve the purpose of taking the output from the first expert system and dividing up the control value amongst the 5 control parameters. The

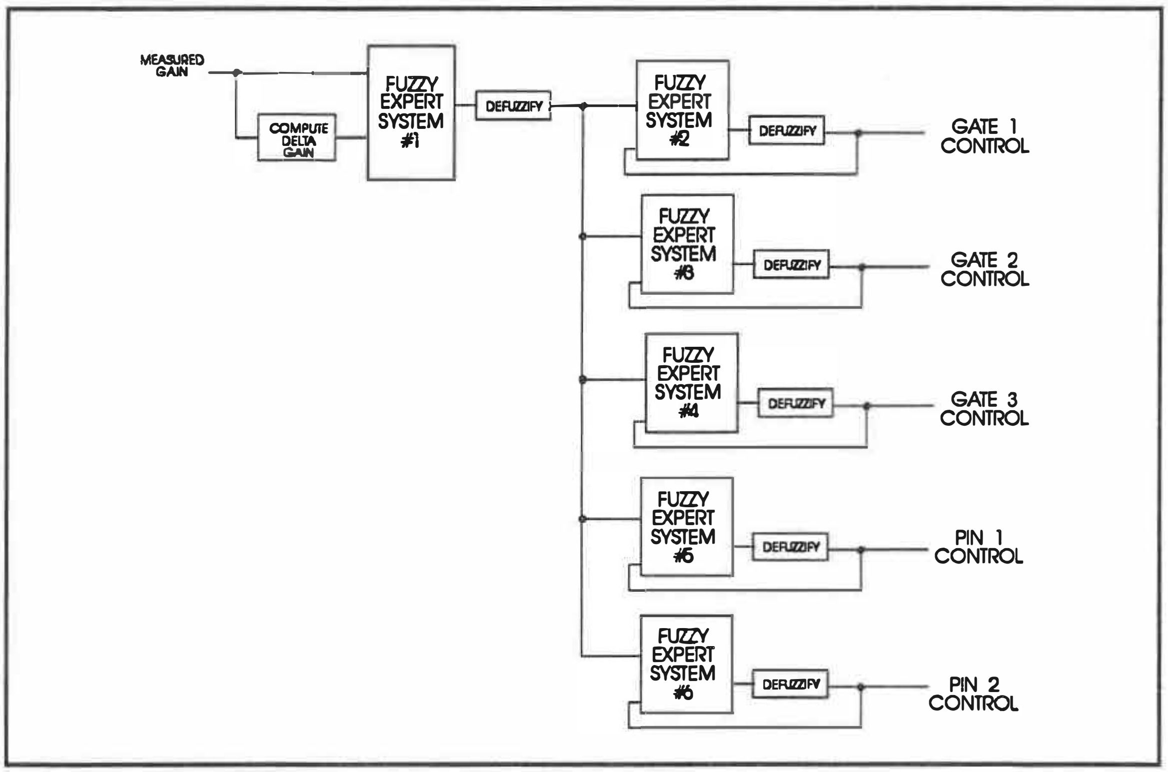


Figure 10 Block diagram of fuzzy logic expert systems.

expert system must take into consideration the present state of the control output and the direction in which the gain must be changed. The objective of this expert system is to minimize the deviation of each control output from their nominal settings.

2.3.1 Expert System #1

The function of this expert system is to determine the amount by which the channel amplifier's bias must be changed in order to maintain a constant gain. In order to implement this expert system three membership functions were defined, Gain, Delta Gain and Controller. A listing of the TIL source code for this fuzzy logic expert system is contained in Appendix D.

2.3.1.1 Gain Membership Function

The Gain membership function was used to classify the gain measured from the channel amplifier into 7 categories as shown in Figure 11. The 7 categories of gain range from the extreme values of gain described by the Extremely Low and Extremely High members to the OK member used to represent a stable condition. The boundaries for each of the 7 members was determined empirically and was found to be very sensitive to the overall system response.

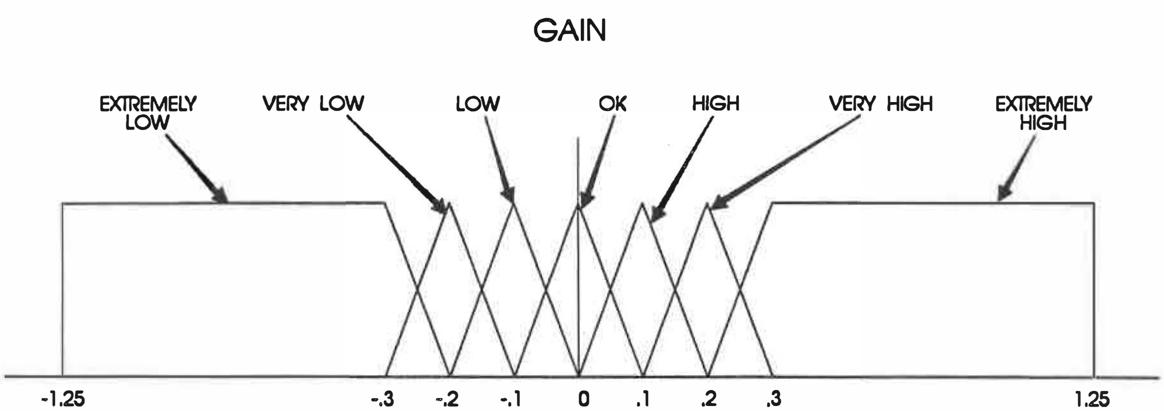


Figure 11 Membership function for Gain.

In a similar fashion the membership functions for Low, Very Low, Extremely Low, High, Very High and Extremely High were defined.

From Figure 11 it can be seen that there is an overlap between each membership function to provide continuity. For example when the gain is between the members OK and Low both membership functions are activated and their corresponding rules govern the output from the system.

The OK membership function limits were set at .1 dB. The other members Low, High, Very Low and Very High were grouped close to the OK member to provide strong positive control about the OK member. Since deviations outside of 0.3 dB were not expected the Extremely Low and Extremely High members dealt with the extreme cases. A number of experiments, some of which are described in Chapter 3, were performed to determine the final membership function limits.

2.3.1.2 Delta Gain Membership Function

The Delta Gain membership function, shown in Figure 12, was used to determine the rate of change of gain and contains the members; decreasing, small and increasing. Initially this membership function had only two members but through detailed system tests, described in Chapter 3, a third membership function Small was added. The addition of the Small member permitted the control of the system when the rate of change of gain was less than .05 db per sample. This value of .05 was empirically determined. It was found that the fuzzy expert system response was very sensitive to the limits set in the Delta Gain membership function.

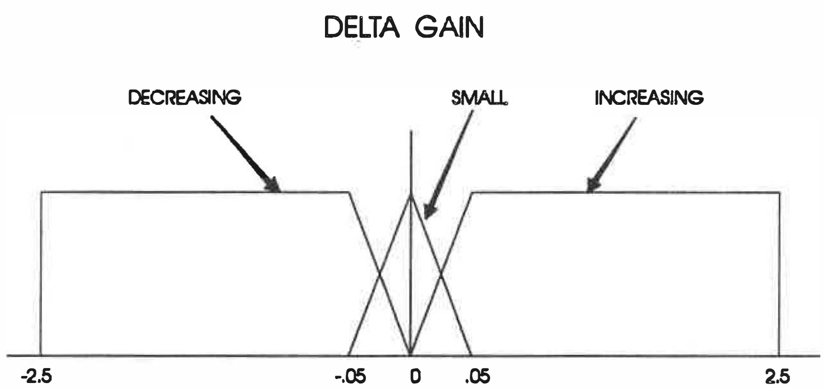


Figure 12 Membership function for Delta Gain.

2.3.1.3 Controller Membership Function

The Controller membership function was used to control the output of the fuzzy logic expert system. A diagram of this membership function is shown in Figure 13. The Controller membership function consisted of 5 members. The Idle member was used to control the output when a very small deviation was required. The limits for the Idle member were set to 1. This value was experimentally determined and was initially set higher resulting in the oscillation of the output. After several experimental runs, described in Chapter 3, the membership function value was set to 1. This membership function was most difficult to fine tune in that the expert system was very sensitive to the values in this membership function.

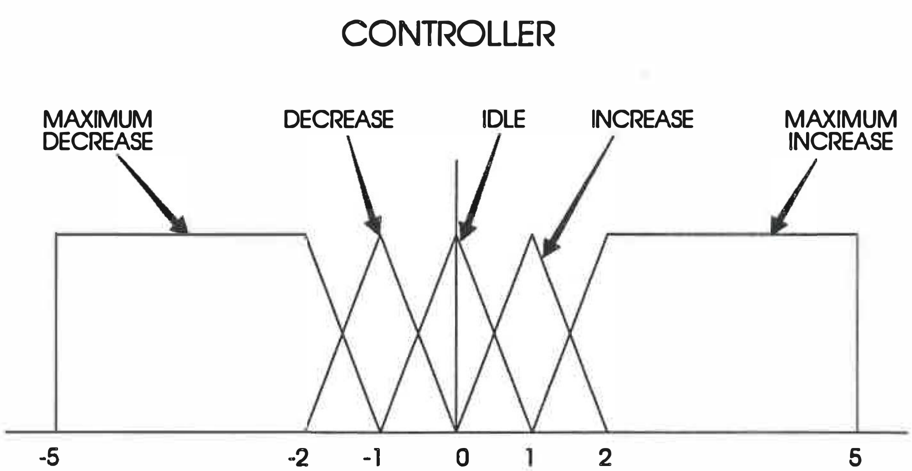


Figure 13 Membership function for Controller.

2.3.1.4 Description of Rules

There are 17 rules governing this first expert system. The rules were derived from the heuristics of the human experts and through a great deal of experimentation with the channel amplifier system. The rules are as follows:

Rule 1

If gain is extremely_low then

controller = max_increase

This first rule covers the case when the input is extremely low and therefore a maximum controller output is required to bring the gain back to a nominal setting.

Rule2

If gain is very_low and dgain is decreasing then

controller = max_increase

Rule3

If gain is very_low and dgain is increasing then

controller = increase

Rule4

If gain is very_low and dgain is small then

controller = max_increase

Rules 2, 3 and 4 cover the case then the measured gain has been categorized as being very low. The rules are modified by the Delta Gain membership function. If the Delta

Gain is decreasing then the already low gain value is getting lower and therefore drastic action must be taken. Rule #2 shows this case and the Controller is set to maximum increase.

When the Delta Gain is increasing the low gain value is getting larger and therefore the system is moving in the right direction. In this case, Rule #3, the controller is commanded with a small increase.

In the last case, Rule #4, if the Delta Gain is small the system has stabilized at the low gain setting and therefore maximum controller output is required to increase the gain of the channel amplifier.

Rule5

If gain is low and dgain is decreasing then

controller = increase

Rule6

If gain is low and dgain is increasing then

controller = idle

Rule7

If gain is low and dgain is small then

controller = increase

Rules 5 through 7 take care of the case when the gain is LOW. Again the three cases for delta gain are considered but since the gain is close to the nominal setting the controller outputs will be smaller than for the extremely low gain case. In this case the controller output is set to Increase for the Delta Gain decreasing and small cases and to Idle for the Delta Gain increasing case.

Rule8

If gain is ok and dgain is increasing then

controller = decrease

Rule9

If gain is ok and dgain is decreasing then

controller = increase

Rule10

If gain is ok then

controller = idle

In rules 8 through 10 the gain is OK and the Delta Gain varies. Rule #8 covers the case when the Delta Gain is increasing. Since the gain is OK an increasing Delta Gain will tend to pull the gain away from the OK setting. For this reason the controller is set to Decrease.

In Rule #9 the delta gain is decreasing and therefore the controller is set to increase to pull the gain response back to the nominal setting.

In Rule #10 the gain is OK and therefore the controller is set to Idle. No fur her change is required in this case.

Rule11

If gain is high and dgain is decreasing then controller = idle

Rule12

If gain is high and dgain is increasing then

controller = decrease

Rule13

If gain is high and dgain is small then

controller = decrease

Rule14

If gain is very_high and dgain is decreasing then

controller = decrease

Rule15

If gain is very_high and dgain is increasing then

controller = max_decrease

Rule16

If gain is very_high and dgain is small then

controller = max_decrease

Rule17

If gain is extremely_high then

controller = max_decrease

Rules 11 through 17 cover the same cases as rules 1 through 7 but for a positive change in gain instead of a negative change. The rules are symmetrical for both positive and negative gain changes.

2.3.2 Expert Systems #2 Through #6

The role of these expert systems is to provide the scaling for each control output. One expert system is assigned to each output due to the different sensitivities of each FET and PIN diode attenuator.

In order to evaluate the level of control required at each output a sensitivity analysis was performed. The bias of each FET and PIN diode attenuator was set to it's nominal value and then each control was changed by a small amount while keeping all other bias values constant. The change in gain was recorded and the sensitivity computed. The sensitivities of each control parameter are shown in Figure 14.

	SENSETIVITY (dB/bit)	NORMALIZED
GATE 1	.0065	1.0
GATE 2	.0375	.173
GATE 3	.0215	.302
PIN 1	.0845	.077
PIN 2	.0310	.210

Figure 14 Sensitivity of control parameters.

It can be seen from Figure 14 that the most sensitive parameter was PIN diode attenuator #1 and the least sensitive parameter was the Gate #1 bias. Also shown in Figure 14 is the sensitivity of each control to a 1 bit change. Since a Digital to Analog (D/A) converter was used to drive the control outputs these values represent the minimum control gain change in dB that can be accomplished with this system.

A total of 3 membership functions were used in these expert systems. These membership functions were the GATE VALUE, CONTROL VALUE and the GATE FACTOR. The inputs to the system were the GATE VALUE and the CONTROL VALUE and the output was the GATE FACTOR. A software listing of the 5 fuzzy logic expert systems used to control the channel amplifier are contained in Appendix C.

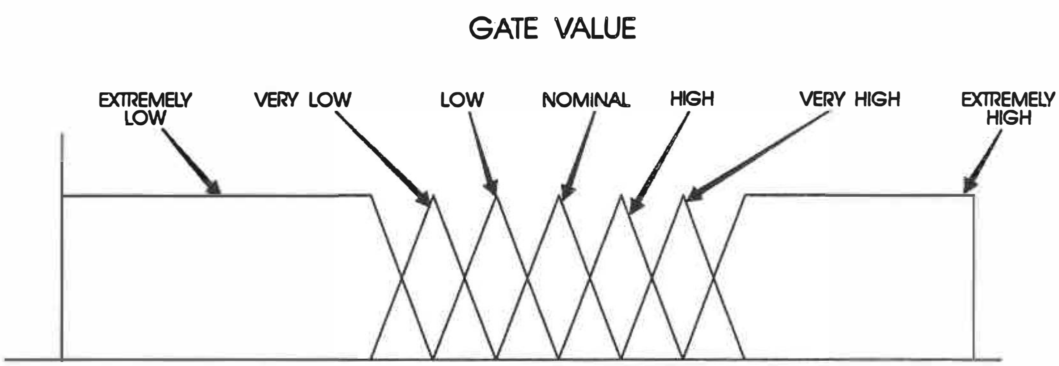
The function of this expert system is to take the output of the first expert system and to compute the control output for each of the 5 control parameters. The control value from the first expert system must be scaled in such a way as to provide the total control output required without substantial deviation from the nominal values of each control output.

This criteria is accomplished by scaling the control output as a function of the deviation from the nominal value of each control output. The more a given control output deviates from its nominal setting the less will its value be changed by the expert system. This approach provides an equalizing effect on the control outputs. In addition the outputs are scaled based on their control sensitivity.

The nominal values for each control output are defined in the GATE VALUE input membership function and the sensitivity scale factors are defined in the GATE FACTOR output membership function.

2.3.2.1 Gate Value Membership Function

The GATE VALUE membership function for each of the 5 fuzzy expert systems is shown in Figure 15. The fuzzy expert systems are called Factor 1 through Factor 5 and control the outputs of the 5 control parameters respectively.



FACTOR 1	0	88	108	128	148	168	188	208	255
FACTOR 2	0	72	92	112	132	152	172	192	255
FACTOR 3	0	95	115	135	155	175	195	215	255
FACTOR 4	0	170	180	190	200	210	220	230	255
FACTOR 5	0	170	180	190	200	210	220	230	255

Figure 15 Membership function for Gate Value.

The Nominal member is centred about the nominal value of each of the 5 control parameters. As the gate value deviates from this nominal value it is categorized as being Low if it below the nominal value and High if it is above. In all 7 members describe this function.

The exact transition points between members for each of the 5 control outputs were difficult to determine and required much experimentation and fine tuning.

2.3.2.2 Control Value Membership Function

In addition to the GATE VALUE input membership function the CONTROL VALUE input membership function is also used and is shown in Figure 16. The function of the CONTROL VALUE membership function is to determine the slope of the input control value. The two members Negative and Positive overlap about the Y axis to permit a smooth transition between the two states. When a control value is equal to zero both the positive and the negative members are activated cancelling out their effects. The transition point of .1 was chosen empirically in that it yielded good system performance.

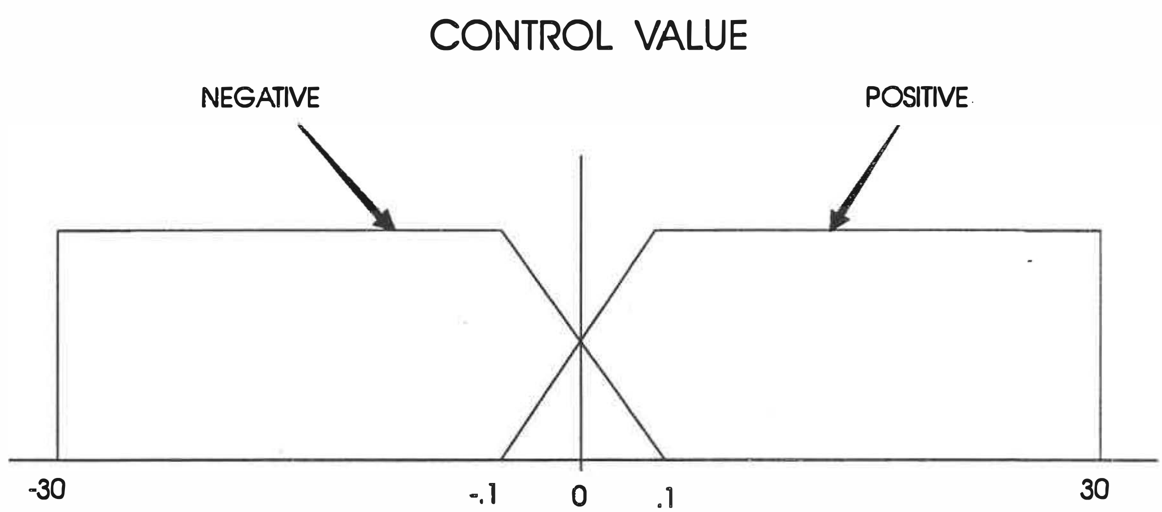
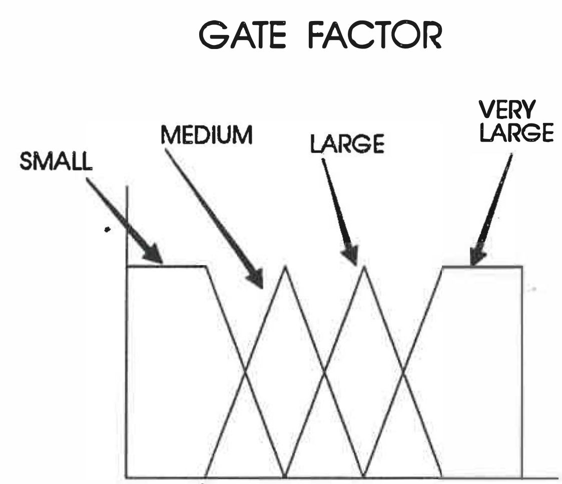


Figure 16 Membership function for Control value.

2.3.2.3 Gate Factor Membership Function

The GATE FACTOR membership function was used to provide the output scale factor. This was accomplished through the use of 4 members as shown in Figure 17. The limits of Factor 1 for the Gate 1 output are normalized and therefore range from 0 to 1. The other factors are scaled to indicate the relative sensitivity of the control parameters. The Gate 1 control was found to be the most sensitive and therefore has a maximum factor value of 1. Factor 4 which represents the most sensitive parameter and corresponds to the control signal for the PIN diode attenuator has the smallest range of scale factor with a maximum value of 0.077. The scale factors are based purely on the sensitivities of each of the control parameters.



FACTOR 1	0	.2	.4	.6	.8	1.0
FACTOR 2	0	.035	.069	.104	.138	.173
FACTOR 3	0	.060	.121	.182	.242	.302
FACTOR 4	0	.015	.031	.046	.062	.077
FACTOR 5	0	.042	.084	.126	.168	.210

Figure 17 Membership function for Gate Factor.

2.3.2.4 Description of Rules

There are 13 rules which govern the output of this expert system. The rules are the same for each of the 5 expert systems except that the membership functions have different parameters for each of the control outputs.

The following are the rules used in the fuzzy logic expert systems.

Rule1

If gate_value is extremely_low and control_value is positive then

gate_factor = very_large

Rule2

If gate_value is extremely_low and control_value is negative then

gate_factor = small

These first two rules address the situation when the control value has deviated from its nominal setting. The rule is modified depending on the polarity of the control value. If the control value is positive then the gate factor is maximized by assigning it to the Very Large member. If the control value is negative then the control output would tend to deviate even further from its nominal setting so a very small scale factor is assigned through the use of the member Small.

Rule3

If gate_value is very_low and control_value is positive then

gate_factor = large

Rule4

If gate_value is very_low and control_value is negative then

gate_factor = small

Rule5

If gate_value is low and control_value is positive then

gate_factor = medium

Rule6

If gate_value is low and control_value is negative then

gate_factor = small

The same type of reasoning is applied to the rules for very low and low except that the scale factor value is decreased as the response approaches the nominal setting.

Rule 7

If gate_value is nominal then

gate_factor = small

When the gate value is at the nominal setting then very little action is required and therefore the gate factor is set to Small.

Rule8

If gate_value is high and control_value is negative then

gate_factor = medium

Rule9

If gate_value is high and control_value is positive then

gate_factor = small

Rule10

If gate_value is very_high and control_value is negative then

gate_factor = large

Rule11

If gate_value is very_high and control_value is positive then

gate_factor = small

Rule12

If gate_value is extremely_high and control_value is negative then

gate_factor = very_large

Rule13

If gate_value is extremely_high and control_value is positive then

gate_factor = small

The rules for positive gate values are identical to the case for negative gate values. The rules are symmetrical for the high and low gate value cases.

2.4 Summary

This chapter has described the development of the fuzzy logic expert system. In addition to the expert system routines a number of programs were written to control the channel amplifier and to monitor its temperature and gain. During the operation of this system all critical parameters were logged into a ramdrive file for later viewing and analysis.

The following chapter will present the details of the hardware used for these experiments and will present the data gathered during the experimental runs.

CHAPTER 3

EXPERIMENTAL RESULTS

3.1 INTRODUCTION

This chapter describes the tests performed to validate the fuzzy logic expert system as well as the steps taken to fine tune its response. This fine tuning procedure was accomplished through the manipulation of membership function parameters while observing their effects on the overall system performance.

The tests were performed on a modified breadboard amplifier, shown in Figure 18, taken from the development phase of the Anik-E spacecraft programm. This amplifier

Figure 18 Detail of channel amplifier showing RF amplifier chain.

consisted of three FET amplification stages and two PIN diode attenuators and operated in the 11.7 to 12.2 GHz frequency band.

The tests were initially performed on a single stage of this amplifier to fine tune the process and were later demonstrated on the complete amplifier.

This chapter will describe the experimental setup used to perform these experiments. A detailed review of the data will be presented along with some of the intermediate results from the fine tuning experiments performed on a single amplification stage. The final experimental results on the complete channel amplifier will be presented and an analysis of the results will be discussed.

3.2 EXPERIMENTAL SETUP

The experiments were conducted in the laboratories of Spar Aerospace Limited in Montreal, Quebec. The laboratory was located in a class 10,000 clean room and therefore special protective clothing had to be worn.

The experimental setup consisted of three main sections, Digital, RF and Thermal. A block diagram of the test setup is shown in Figure 19.

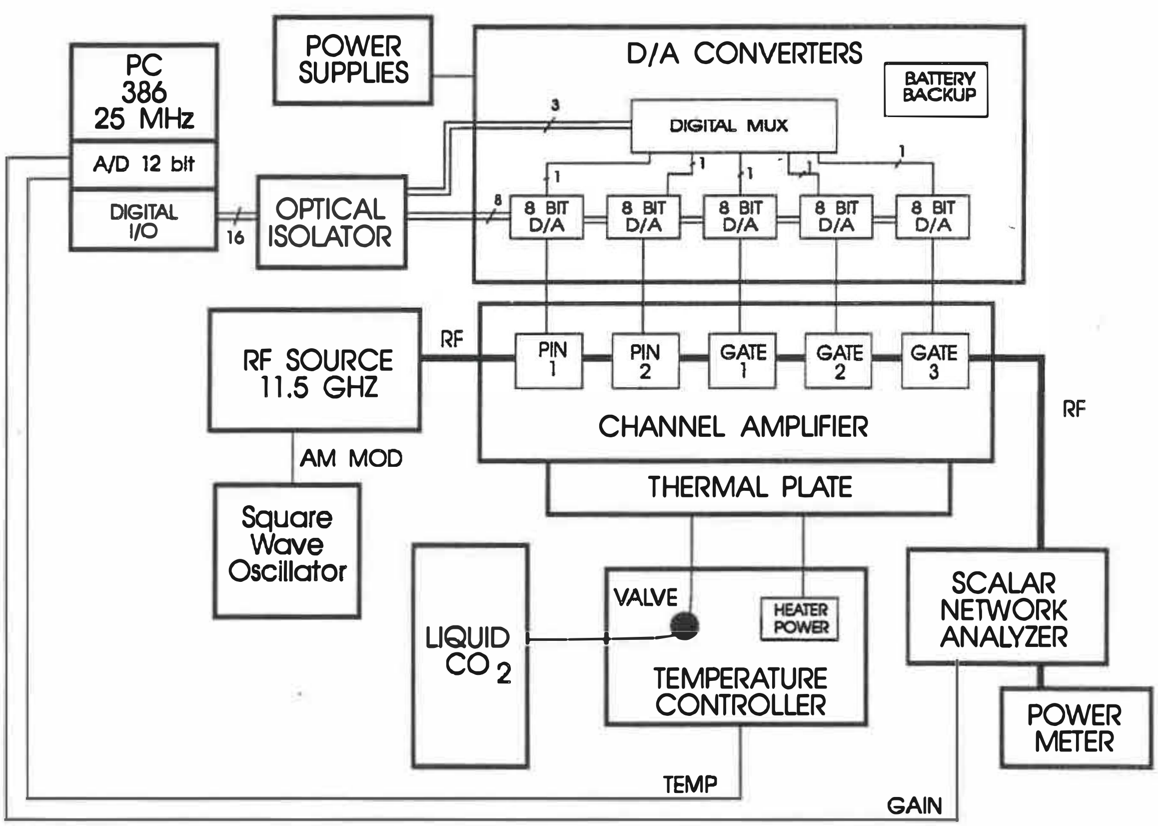


Figure 19 Block diagram of experimental setup.

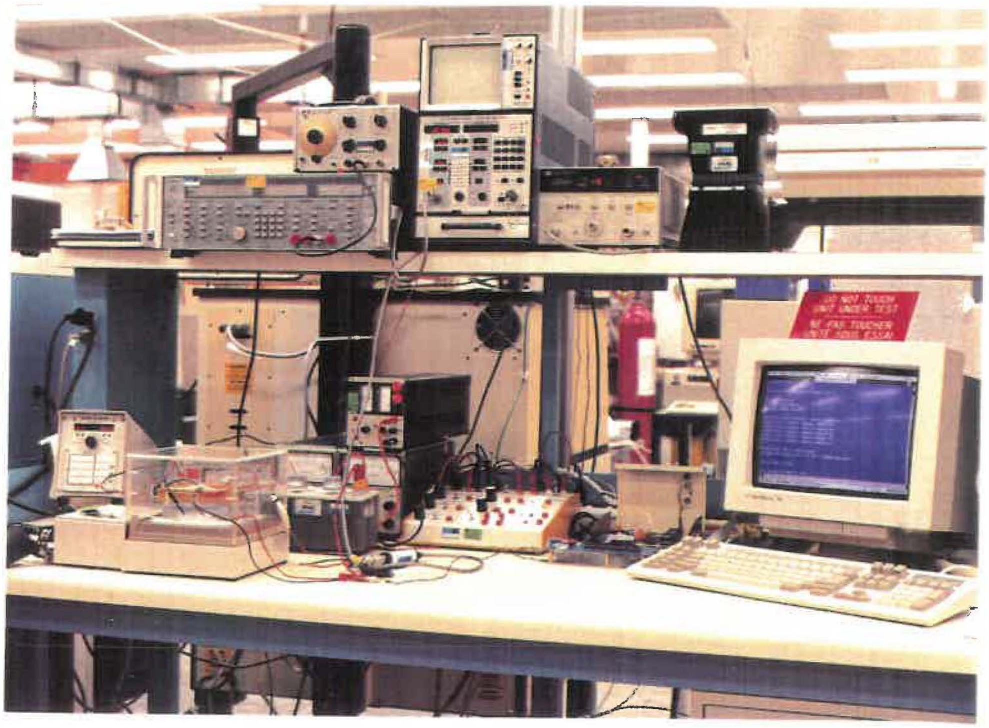


Figure 20 Test bench in class 10,000 clean room.

A photograph of this test setup is shown in Figure 20. The RF equipment is shown on the upper part of the test bench with the thermal control system and channel amplifier on the left hand side and the digital portion on the right hand side. The following sections will describe these parts in more detail.

3.2.1 Digital Setup

The digital portion of the setup consisted of a 386 based PC computer operating at 25 MHz. The computer contained a Data Translation data acquisition card which provided 16 analog inputs with a resolution of 12 bits and 16 digital input output lines.

These digital I/O lines were optically isolated from the RF unit under test to prevent noise problems and as a safety feature in the event of catastrophic failure of the PC. This was important since the value of the channel amplifier was $70K.

The optically isolated digital I/O lines were fed to a custom built D/A module which was used to control the bias levels of the RF slice. The role of the D/A converter module was to provide the interface normally accomplished by a human when he turns the control potentiometers. The D/A converter accomplished this by providing the voltage which would normally be produced by a potentiometer. The interface to the PC allowed this manipulation to occur under computer control. A photograph of the PC used for these experiments is shown in Figure 21.

A series of programs were written to provide the interface between the PC and the D/A converter. The first program was written to interface the PC to the Data Translation data acquisition board. A listing of this program is contained in Appendix A. These low level drivers were written in Microsoft Quick C and provided routines to control the A/D converters and the digital I/O lines. The Data Translation board also contained 2 D/A converters but these were not used since they could not easily be converted to battery backup operation which was essential in this case.

Figure 21 PC 386 computer used to implement fuzzy expert system.

The custom built D/A converter consisted of a battery backup system to maintain the bias in the event of a power failure. This was important since the small signal FETs (NEC 673) in the channel amplifier could be damaged if the gate voltage is reduced to 0 volts with the drain voltage still present. In normal production of the channel amplifier a battery backup system is used to prevent this occurrence and therefore it was also

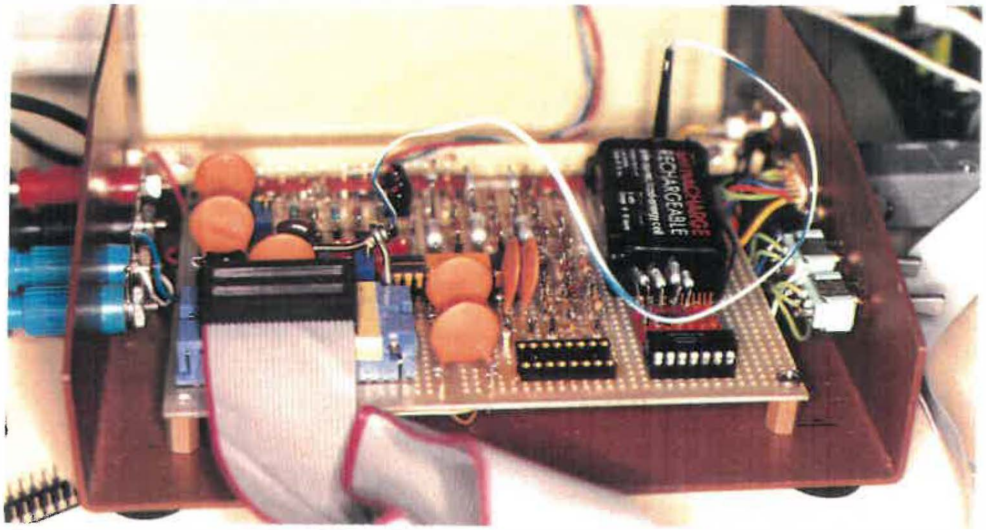


Figure 22 Detail of D/A converter module showing battery backup.

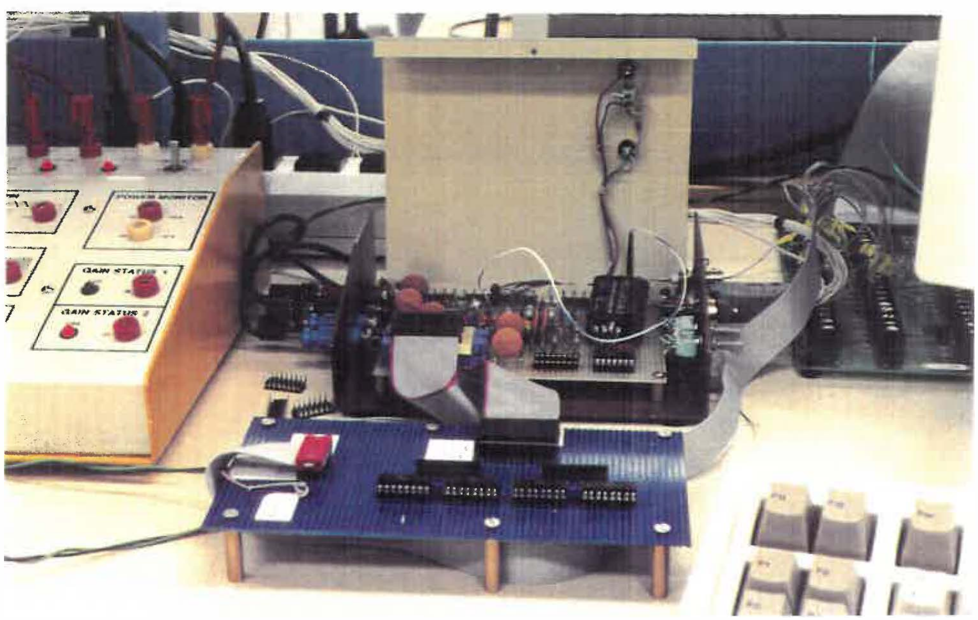


Figure 23 D/A converter and optical isolator modules.

incorporated into the experimental setup. A photograph of the D/A converter is shown in Figure 23. The optical isolator board is shown in the foreground and the D/A module is in

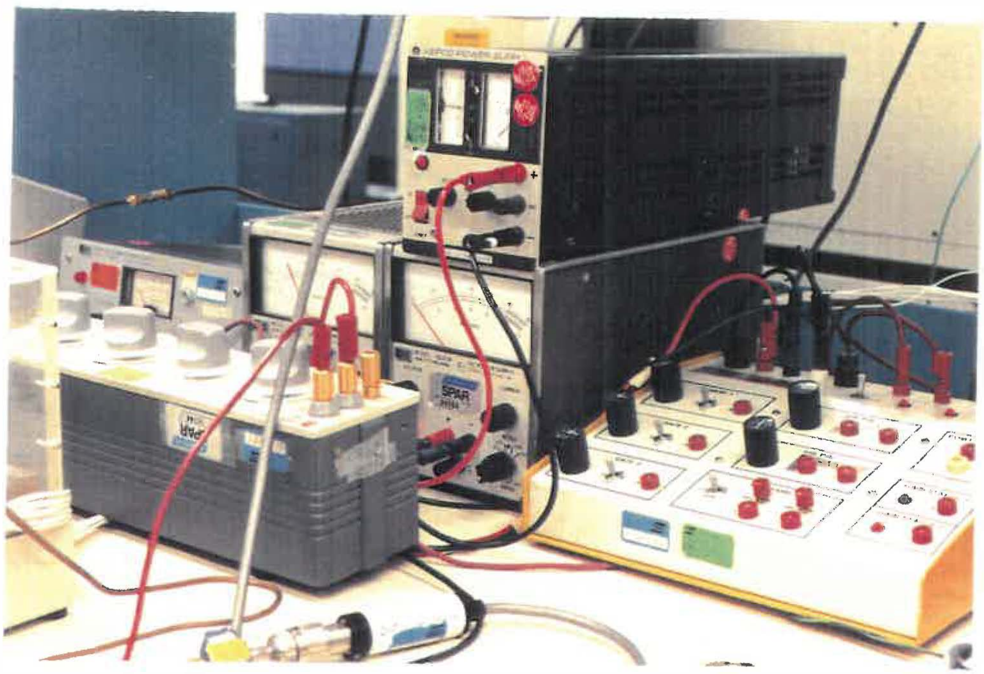


Figure 24 Power supplies used to power D/A converter and channel amplifier.

the background. A close up of the D/A converter module is shown in Figure 22 and shows the battery backup system. Due to the low current drawn by the FET gates and the low power consumption of the D/A converter module the battery size could be made small.

In addition to the D/A converter several Laboratory power supplies were used to provide the drain voltage for the RF slice under test and to power the D/A converter module. These supplies are shown in Figure 24 along with the manual control box which would normally be used if the bias were implemented manually.

3.2.2 RF Setup

The RF setup consisted of an RF signal source, a low frequency square wave generator, a scalar network analyzer and a power meter.

The RF source was connected to the input of the channel amplifier and provided the RF signal input. The frequency of this input was 11.5 Ghz with a signal strength of -12 dBm.

In order to perform some of the initial evaluations of the expert system it was

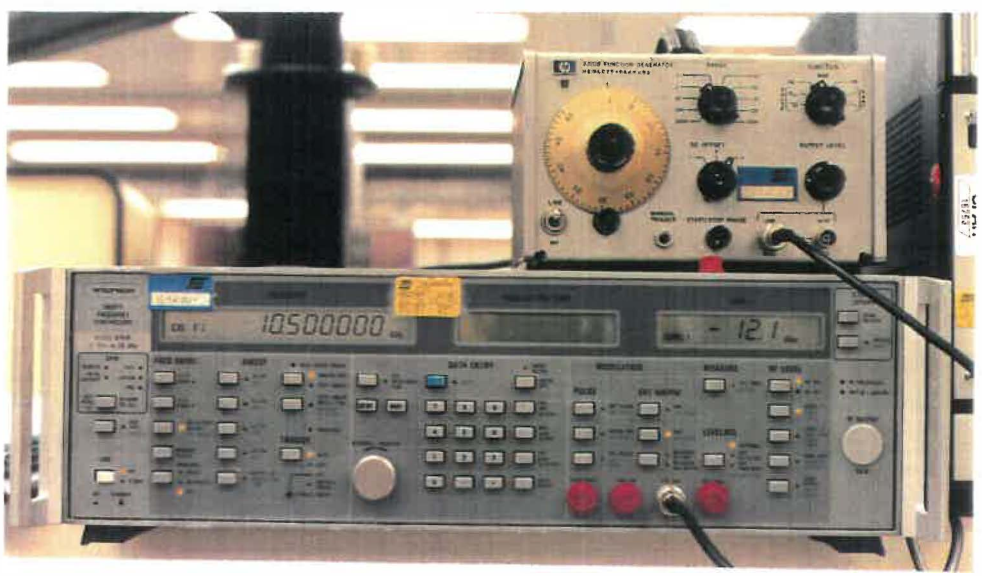


Figure 25 RF signal source showing external square wave generator used for AM modulation of RF signal.

necessary to modulate the RF signal strength by a square wave to simulate a step input to the system.

A square wave oscillator was connected to the RF signal source at its external AM modulation input. In this way the RF could be modulated by 0.5 dB from its nominal setting. A photograph of the RF signal source and the external AM modulation source is shown in Figure 25.

The output of the channel amplifier was connected to a scalar network analyzer. This analyzer was used to measure the signal strength coming out of the LNA and had the capability of subtracting the input signal applied to the LNA. In this way the output seen on the scalar network analyzer represented the gain of the channel amplifier. The scalar

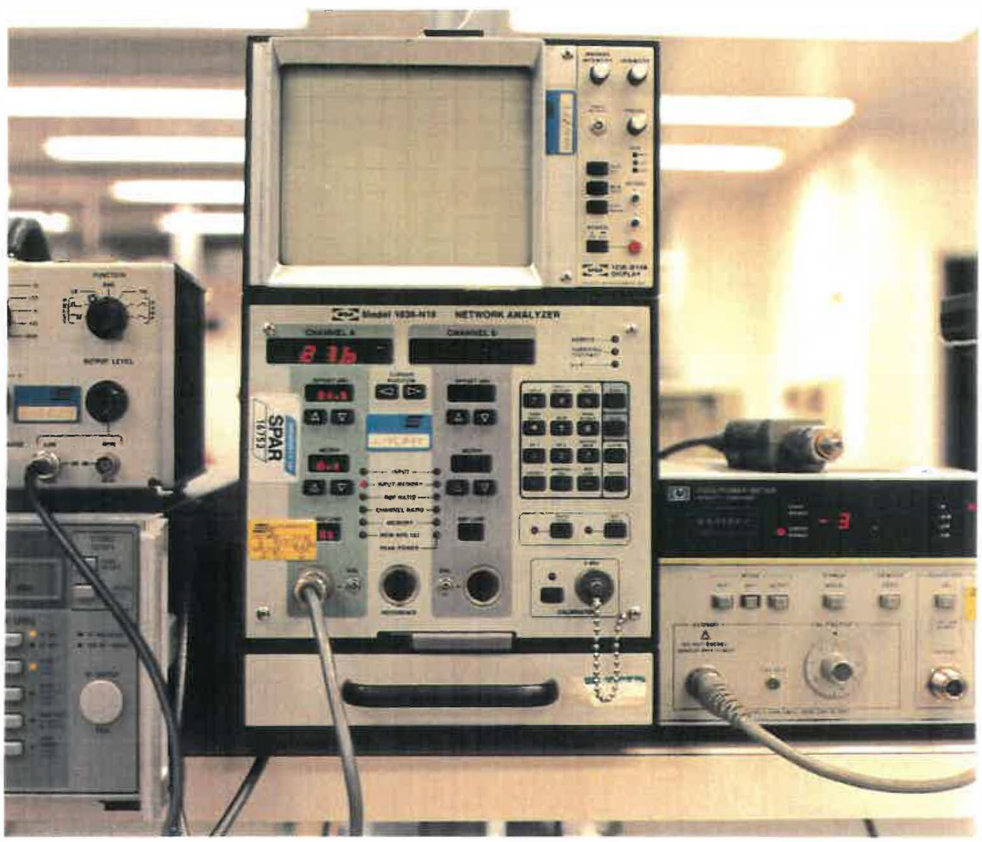


Figure 26 Scalar network analyzer used to measure the gain of the channel amplifier.

network analyzer had an analog output which provided a voltage proportional to the measured gain. This analog output was connected to one of the A/D input channels of the Data Translation data acquisition card contained in the PC. In this way the gain of the channel amplifier could be sampled and recorded by the PC. A photograph of the scalar network analyzer used in this experimental setup is shown in Figure 26. Beside the network analyzer in Figure 26 is a power meter which was used to calibrate the RF system and to ensure that the correct input levels were used to drive the channel amplifier.

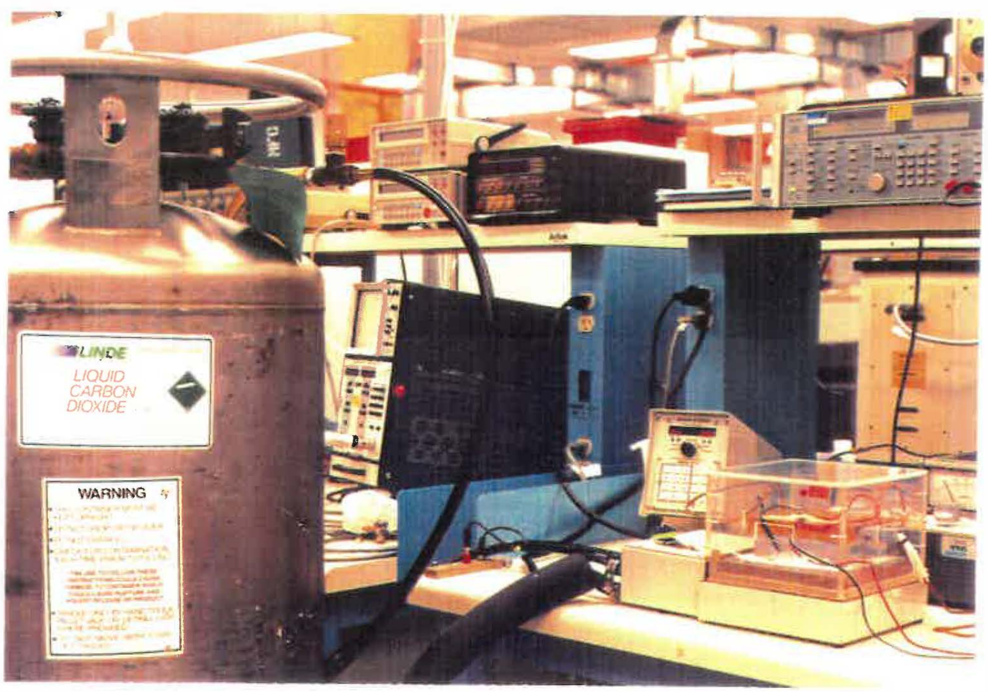


Figure 27 Thermal system showing liquid carbon dioxide tank , thermal controller and thermal plate.

3.2.3 Thermal Setup

In order to control the temperature of the channel amplifier a computer controlled thermal plate, shown in Figure 27, was used. The thermal plate operated in two modes, heating and cooling. In the heating mode resistive heaters under the thermal plate were activated and the amount of current passed through the heater was regulated by the thermal

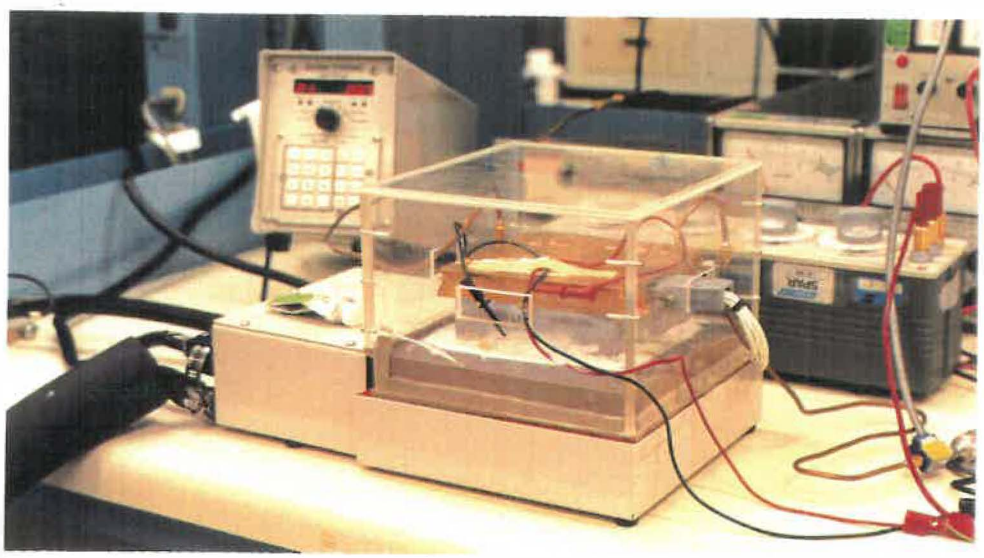


Figure 28 Detail of thermal plate and of thermal controller.

controller. In the cooling mode liquid CO_2 was passed through a heat exchanger contained within the thermal plate. The amount of CO_2, and therefore the amount and the rate of cooling, was modulated by a valve controlled by the thermal controller.

The system was independently controlled by a microprocessor based controller shown in Figure 28. This controller was programmed via a keypad on the front panel of

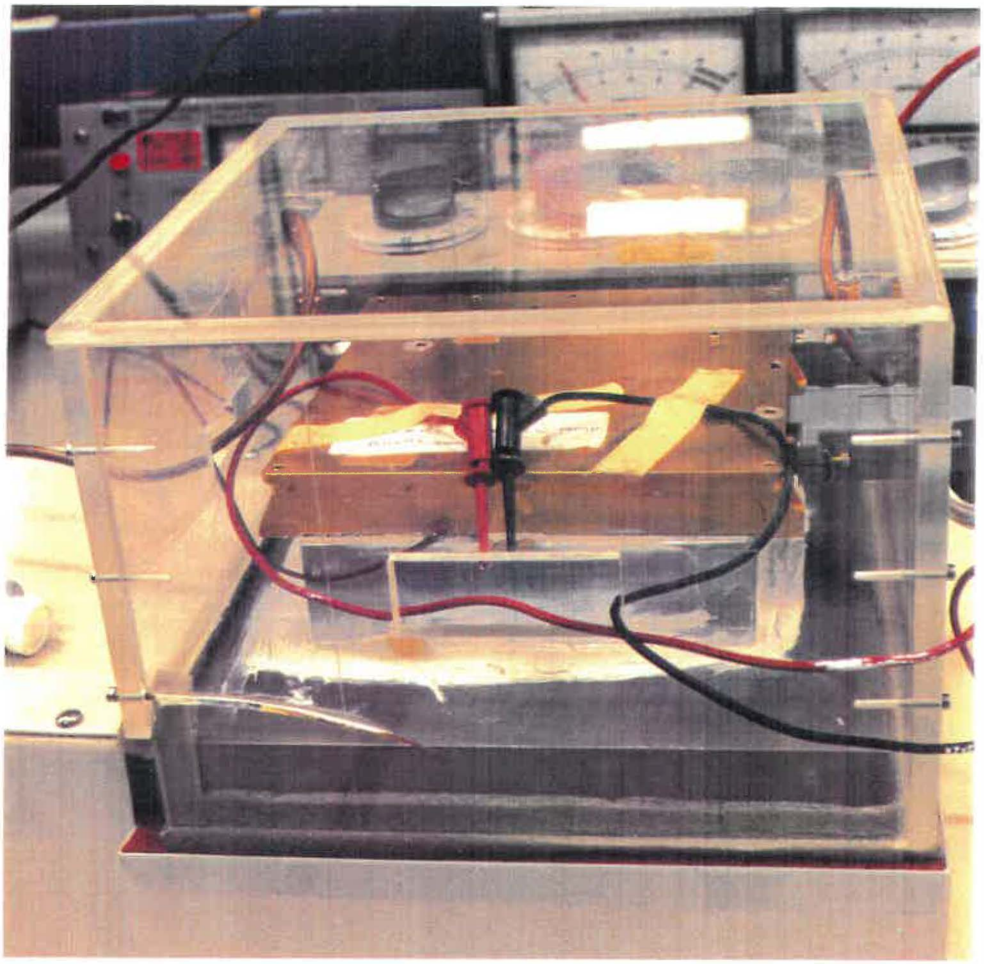


Figure 29 Detail of thermal plate showing mounting configuration of the channel amplifier.

the unit. The programming consisted of the start and stop temperatures, dwell times at each temperature and the rate of change of temperature as a function of time.

The channel amplifier was mounted on the thermal plate shown in Figure 29. In order to provide a good thermal interface heat sink compound was used between the thermal plate and the base of the channel amplifier.

A temperature sensor was attached to the base of the channel amplifier and was monitored by one of the A/D channels of the Data Translation data acquisition board contained in the PC. In this way the temperature of the channel amplifier could be monitored and recorded by the PC.

This thermal plate was used for the final tests of the expert system to verify the thermal performance of the channel amplifier.

3.3 INITIAL TESTS

One of the more difficult aspects of designing this fuzzy logic control system was the determination of the various parameters for the membership functions. Many months of work were spent analyzing the various combinations of membership functions empirically. This section describes some of these empirical trials.

3.3.1 Step response

The final goal of this project was to control 3 FETs and 2 PIN diode attenuators and to maintain a constant gain. Due to the complexity of the channel amplifier system it was difficult to isolate the effects that each amplification stage had on the response of the channel amplifier system. For this reason a single FET was electrically isolated in the channel amplifier and the fuzzy logic controller was simplified to allow the control of only the one FET amplification stage.

In order to evaluate the effect of different membership function parameters a step response in the input power was used. This permitted the characterization of the dynamics of the single FET system. Using this simple system the effects of various membership function parameters were evaluated before proceeding to integrate all 5 channel amplifier control parameters.

The step function in input power was implemented by using a square wave function generator and connecting it to the AM modulation input of the RF signal source. The amplitude of the square wave source was set so that the RF was modulated by 0.5 dB. The advantage of using a step function was that the effects of the various membership functions could be determined on the dynamics of the control response in a clear and rapid way. Much time was saved in debugging the single FET system with a step function input. It would have been very difficult to determine the effects of various parameters if all 5 control values would have been used.

3.3.1.1 Effect of Gain Slope Membership Function

The delta gain membership function processes the change of gain between samples and modifies the firing of rules between a positive and a negative slope. The rationale behind this membership function was not only to provide a sharp transition zone but also to give a buffer between the two when the slope was very close to zero. This membership function was one of the more difficult to optimize.

Initially only two members were selected, decreasing and increasing, as shown in Figure 30. The end point values were chosen to provide a very narrow transition zone. There were two members one for decreasing gain and one for increasing gain. The transition points were put close together and set to a value of 0.005 dB. In order to evaluate the effect of only this membership function all other membership functions were untouched and remained at the values described in Chapter 2.

The gain response of this system to a step input is shown in Figure 31. The response of the system was extremely slow and the settling time was very long. Initially when the delta gain was large the gain of the system responded quickly but as the delta gain decreased

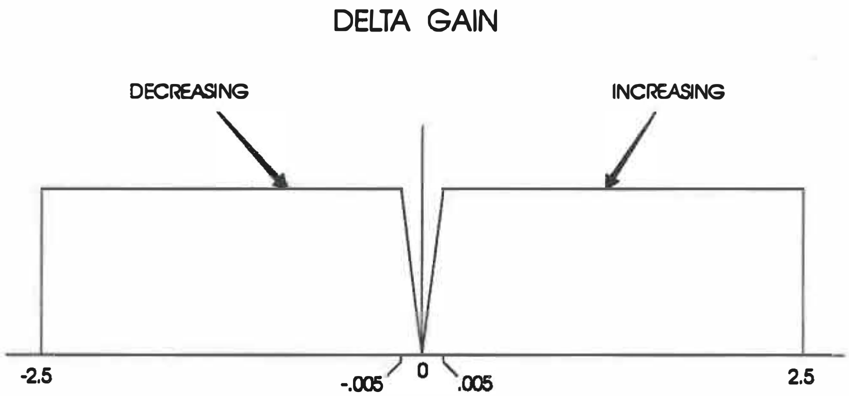


Figure 30 Initial setting for Delta gain membership function.

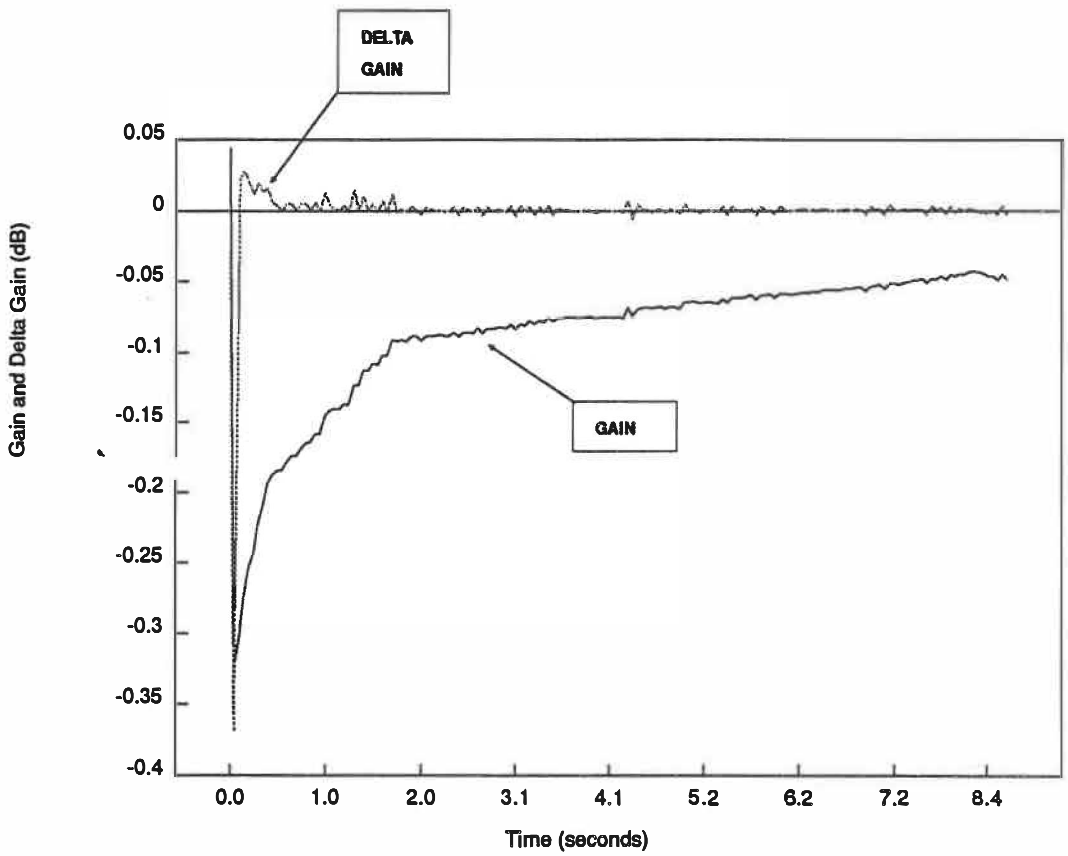


Figure 31 Gain and Delta Gain response for a step input wih modified Delta Gain membership function.

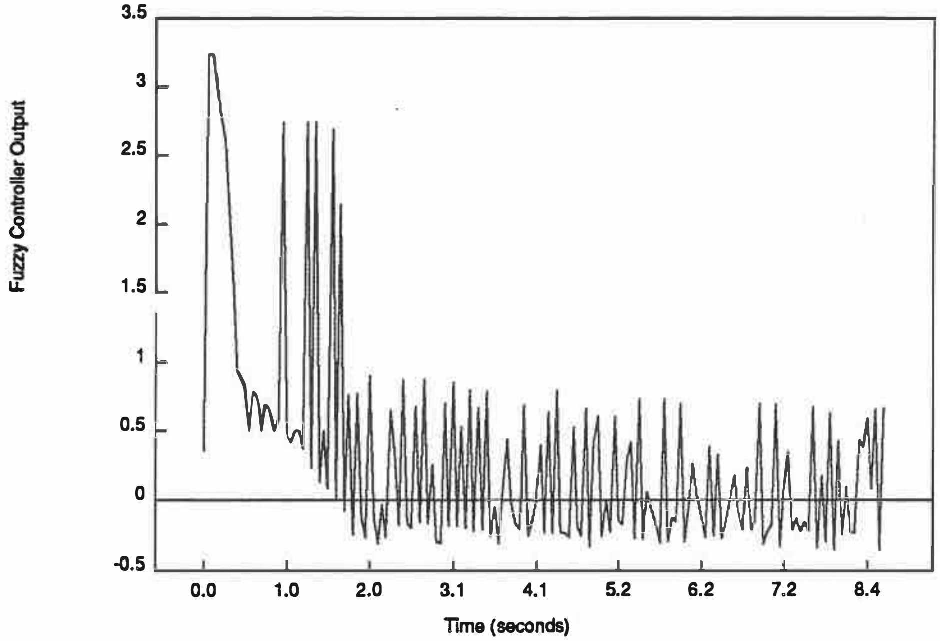


Figure 32 Control output of first fuzzy logic expert system.

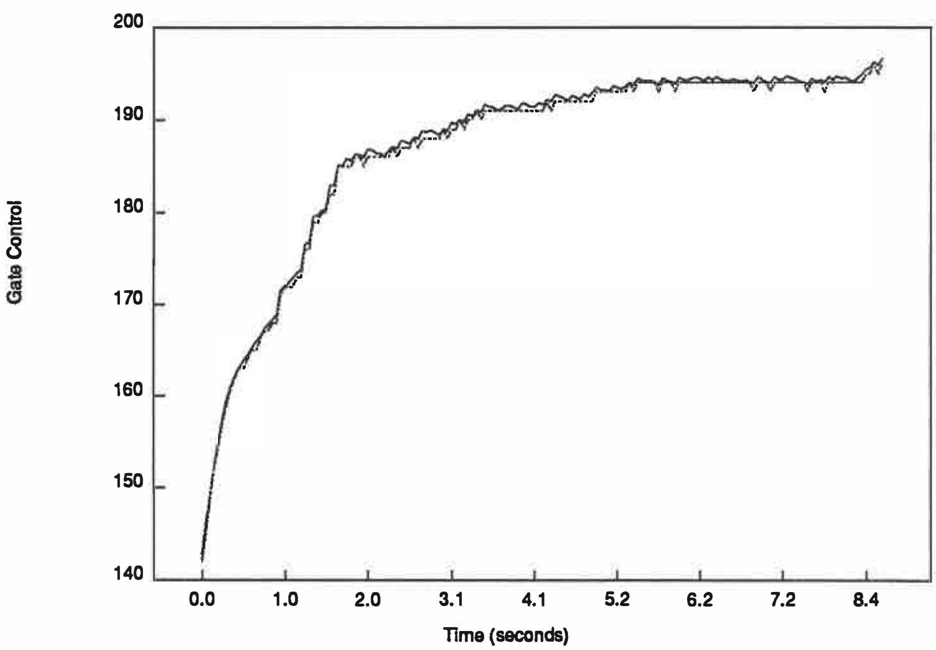


Figure 33 Gate control output. Solid line represents floating point value and dotted line represents the corresponding byte output

the rate of gain change of gain also decreased and made the settling time of the system to a step input very long. For this reason another member was added, SMALL, in order to provide a transition between these two extreme states. The addition of this membership function increased the number of rules from 14 to 17 rules.

The effect of the same step input on the gate control value and the fuzzy controller output are shown in Figures 32 and 33. The 8 bit gate control value tracked the floating point value very well but the settling time of the control response was very long. The fuzzy controllers output was very noisy due to the rapid transition between rules governing the negative and positive DGAIN functions. The addition of the member SMALL prevented this oscillation between states and made the response smoother.

3.3.1.2 Effect of Controller Scaling

The membership function which directly modified the output to the channel amplifier was the CONTROLLER membership function. The parameters of this membership function were varied to determine the required values for good system performance. The controller was a difficult membership function to model in that it was very sensitive to system inputs.

The controller membership function was divided into 5 members as shown in Figure 34. These membership function limits were empirically determined and were finally set to a maximum of 5 as described in Chapter 2. In order to demonstrate the sensitivity of this parameter the membership function values were increased by a factor of 10.

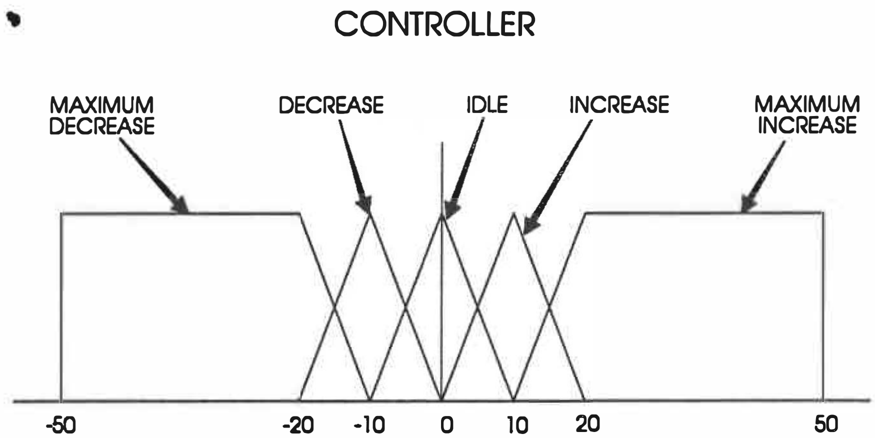


Figure 34 Modified membership function for Controller.

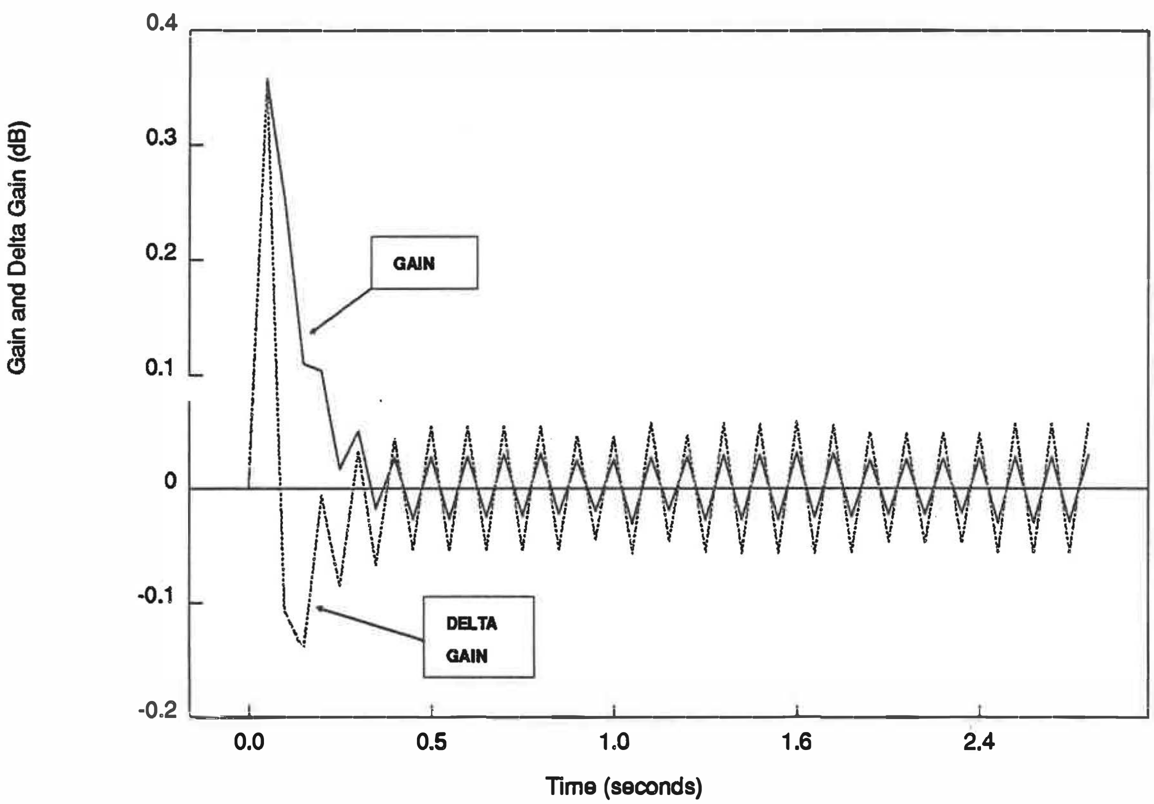


Figure 35 Gain and Delta gain response with modified Controller parameters

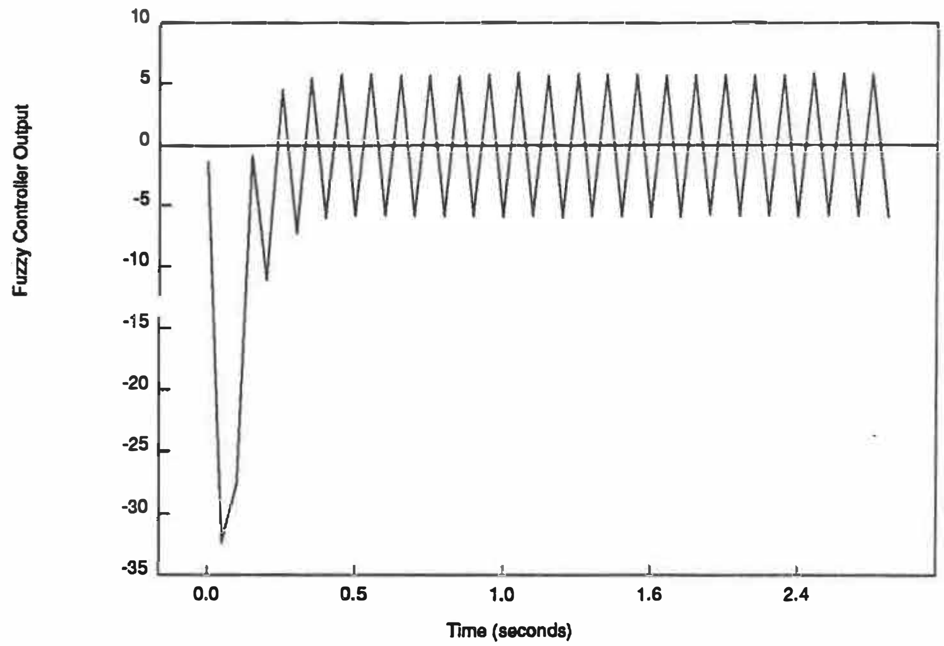


Figure 36 Control output from first fuzzy logic expert system.

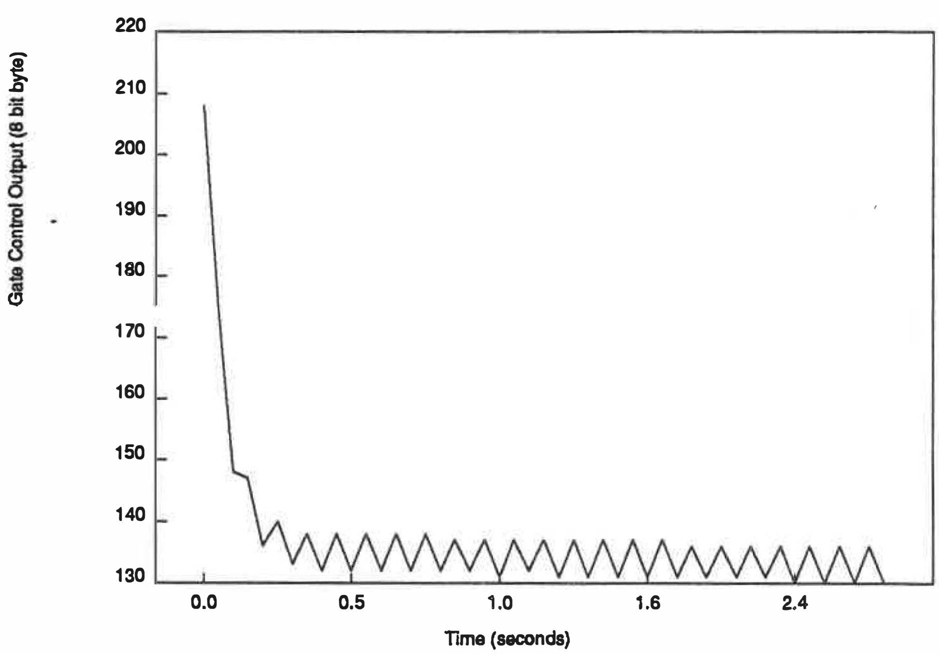


Figure 37 Byte control output for step input.

The gain response of this system to a step input in power is shown in Figure 35. Due to the large extreme values of the controller the gain response is seen to settle very quickly but then starts to oscillate around the 0 level. This oscillation can also be seen in the 8 bit gate control output shown in Figure 36. The reason for this oscillation is that the fuzzy controller output, shown in Figure 37, was over responding to the changes in the delta gain parameter. When the delta gain was negative the controller would command a large increase in the gain. This would result in an overshoot on the positive side of the gain curve resulting in further overreaction of the fuzzy logic controller.

This membership function was fine tuned based on the step response outputs and it was determined that a maximum value of 5 yielded good system performance

3.3.2 Final Settings for Single FET System

The gain response of a single FET system to a step input, using the final membership function values described in Chapter 2, is shown in Figure 38. From the figure it can be seen that the initial response to the step is rapid and that the system settles to its final output within 6 seconds. This time constant was found to be desirable since a time constant which was too fast would respond to noise and a time constant which was too slow would not be able to track the gain as the temperature was changed.

The gain response of the fuzzy controller is shown in Figure 39 and demonstrates the effect that the DGAIN membership function can have on rule switching. The initial part of the control curve is very steep and corresponds to the region of rapid change of input

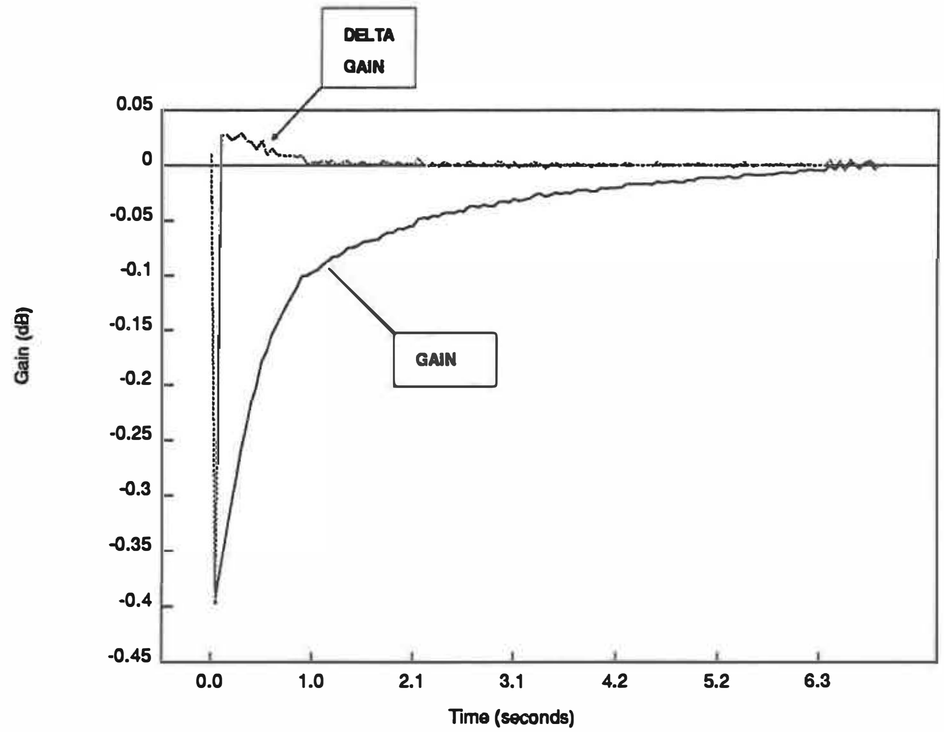


Figure 38 Gain and Delta gain response using final membership function values.

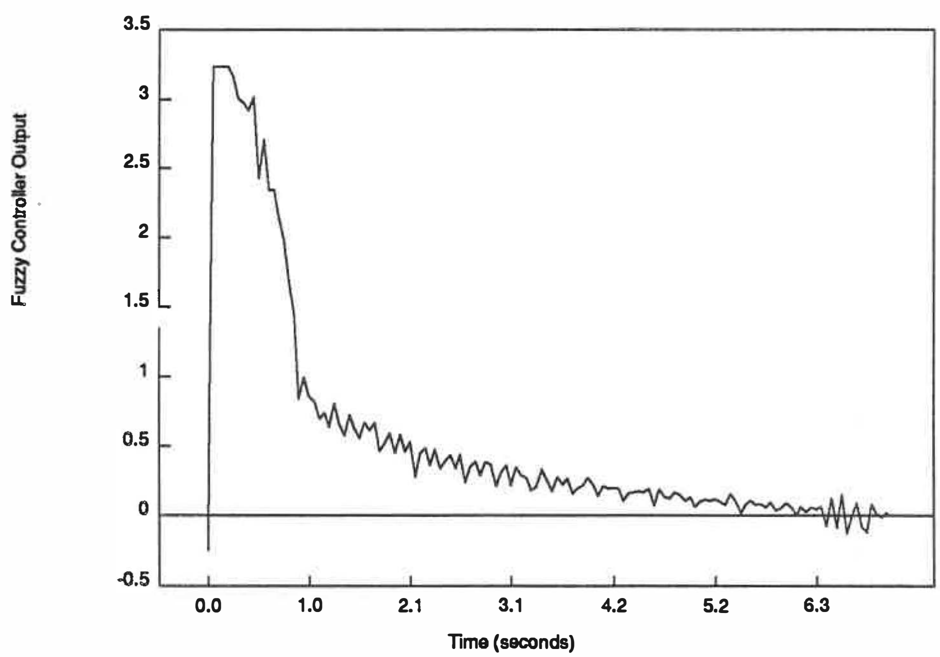


Figure 39 Control parameter from first fuzzy logic expert system.

power. As the rate of change of gain is reduced the control output decreases. As the rate of change of gain decreases and the gain approaches the reference value the rules that are fired change smoothly from state to state. This can be seen from Figure 39 as a smooth transition from a rapid change in controller output to a more gradual one as the desired value is reached.

3.4 FINAL TESTS

After all the membership function values were evaluated using the step response on a single FET system all 5 control parameters were integrated and the thermal performance of the system was tested.

These thermal tests were used to simulate the type of application that the channel amplifier would see in the space environment. The temperature extremes over which the channel amplifier must operate are between -10 and +50 degrees celsius. The thermal plate was programmed to transition between these two extremes at a rate of 3 degrees per minute. This slow rate of change is characteristic of what the unit would see in orbit. Another reason for not using a rapid rate of change is that there exists a possibility of thermally shocking the unit and causing some mechanical overstress.

The channel amplifier was thermally cycled while the fuzzy logic controller monitored the gain of the channel amplifier and controlled each of the five control parameters to maintain the gain at the reference level.

The temperature profile used for these experiments is shown in Figure 40. The gain of the channel amplifier is shown in Figure 41. In this figure both the uncompensated gain and the gain compensated by the fuzzy logic controller are shown. The fuzzy logic controller was able to maintain the gain of the channel amplifier to within 0.1 dB from the reference gain. The deviations which appear were due to the finite resolution of the 8 bit D/A converter.

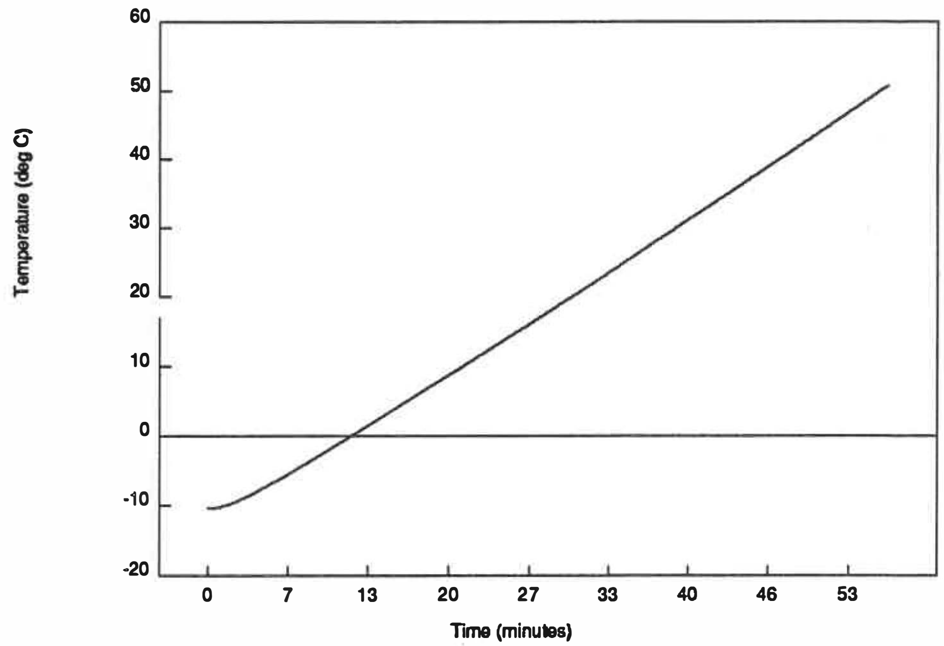


Figure 40 Temperature profile of thermal run.

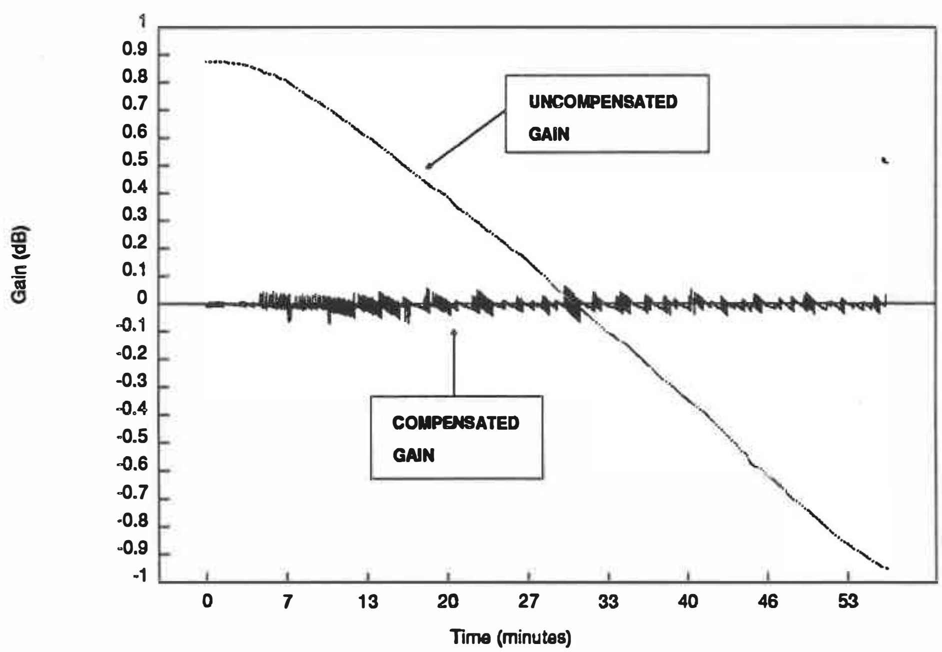


Figure 41 Compensated and uncompensated gain response during thermal run.

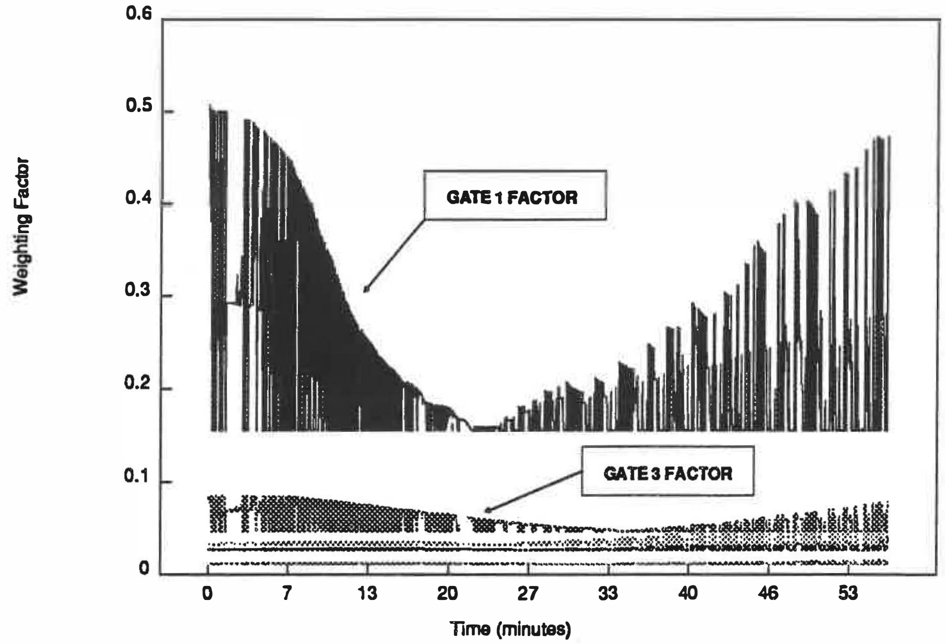


Figure 42 Scale factors for final thermal run for each of the 5 control parameters.

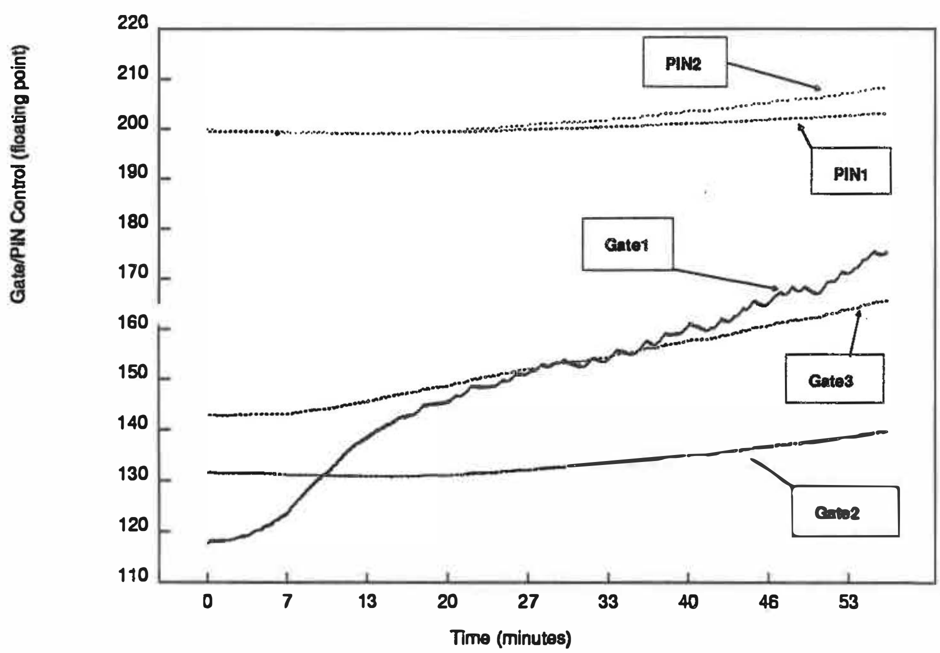


Figure 43 Floating point output for each of the 5 control parameters.

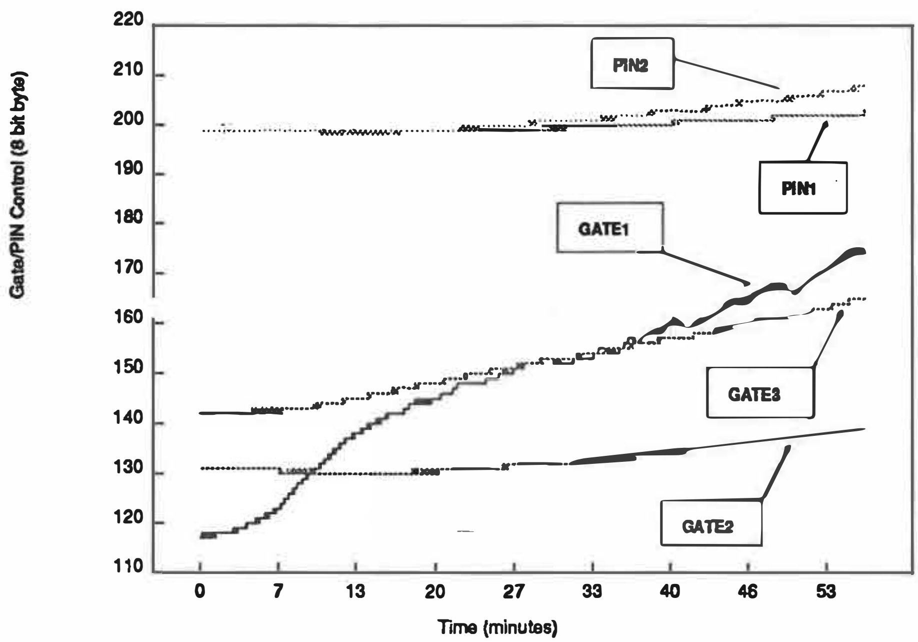


Figure 44 Final thermal run, byte output for 5 control outputs.

Each of the 5 control outputs was assigned a scaling factor as was shown in Chapter 2. The response of these scaling factors is shown in Figure 42. Since the Gate 1 control was the least sensitive to control inputs it has the largest scale factor. The other control parameters had smaller scale factors and therefore were less active. The shape of the weighting factors indicates the goal of the weighting factor which was to maintain each control as close as possible to the nominal control values. This produced a U shaped curve for the GATE 1 weighting factor. The bottom of the U represents the area in which the GATE 1 output is very close to the nominal value. As the gate 1 control value deviates from this nominal value the weighting factor increased. This increase in weighting factor is governed by the rules for the weighting factors discussed in Chapter 2.

The floating point output values for the 5 channel amplifier controls are shown in Figure 43. From the figure it can be seen that the Gate 1 control output was exercised the most. The other control output were much more sensitive to control input and therefore were exercised less.

The final outputs to the 8 bit D/A converters is shown in Figure 44. These curves are similar to those of Figure 43 except that the floating point values are now converted to 8 bit byte values. The resulting quantization can be seen as sharp transitions which oscillate between the two binary states before transitioning to the new state. The quantization is most apparent on the Gate 1 control output since it was exercised the most.

3.5 DISCUSSION OF RESULTS

As can be seen from the experimental results the effect of quantization error on the final system response is quite severe. Although the final response remained quite close to the reference gain (< 0.1 dB) it could have been substantially improved through the use of a D/A converter with more resolution.

The effect of the quantization of control output also effected the fuzzy logic control system in that misleading information was fed to the controller. When the floating point value of a particular control value was increased very slightly, for instance when the system output was very close to the desired reference gain, this increase could trigger the transition to a new 8 bit binary state causing a rather large change in the gain of the channel amplifier. This change would be fed back through the fuzzy logic controller as a change in the delta gain and the gain parameters. The fuzzy logic controller would respond to this change by commanding a reduction of gain. This would lead to a slight oscillation centred about the gain level of the least significant bit being toggled.

The most sensitive control parameter was the first pin diode attenuator PIN1 which had a gain sensitivity of 0.085 dB per bit. If more than on output was transitioning from one binary state to another the resulting quantization noise would be additive. For instance if PIN1 and Gate1 were both transitioning at the same time then the change in channel amplifier gain would be .085 + .007 = .092 dB. This level of quantization noise was observed in the system gain response in Figure 41.

CONCLUSIONS

The objective of this work was to develop an automated system to determine correct bias values for microwave FET amplifiers and to determine how to modify these values as a function of temperature in order to maintain a constant system gain.

The development of this automated system used the principles of knowledge engineering, artificial intelligence and fuzzy logic. By using a combination of these principles, and some low level C language programs, a successful fuzzy expert system was implemented and tested on a 12 GHz channel amplifier system.

During the development phase of this work the nature of the bias compensation problem was explored. A thorough investigation of the present solution methods was evaluated. The present system utilizes a manual approach and involves the use of a human expert to empirically determine the amplifier bias conditions and to determine the way in which the bias should be changed as a function of temperature. These interviews were complemented by a series of laboratory experiments with a typical amplifier system.

The interviews and laboratory experiments resulted in a set of heuristics which governed the bias compensation process. The heuristics revealed that the solution to the bias compensation problem was ill defined, nonlinear in nature and required the acquired skill of an expert to successfully accomplish the task.

In order to determine the nature of this acquired expert knowledge, the principles of knowledge engineering were employed to extract the heuristics from several experts. The determination of these heuristics resulted in an understanding of the underlying process of thermal bias compensation and in the iterative nature of the process. It was found that the nonlinear nature of the control problem required the expert to perform several iterations before converging to an acceptable solution.

Once these heuristics were determined a method of implementing them was investigated. The lack of crispness of the heuristics made the problem amenable to a fuzzy logic approach. The TIL Fuzzy C compiler was used to implement the fuzzy logic expert system. Several other programs were written in C to interface with the microwave test equipment and to control the bias of the channel amplifier. A circuit to interface the channel amplifier to a PC was designed and constructed and was used to control the individual bias levels of each active element in the channel amplifier.

The concept of gain measurement is fundamental to the alignment and testing of a microwave system. In this thesis the gain was compensated for temperature variations but the system could easily be extended to the control of other parameters which affect the gain. This thesis has developed the foundation in both the software and hardware to further develop this application in the future.

The present trend in the design and development of microwave communications equipment is towards greater and greater miniaturization. This miniaturization results in lower manufacturing costs, higher reliability and lower system weight. A side benefit to this miniaturization is the ability to integrate more functions to the RF system then was possible using conventional technology. As more of these functions are added the degree of complexity increases and therefore the difficulty in aligning and testing also increases.

This thesis has successfully demonstrated that the techniques of artificial intelligence and specifically fuzzy logic can be applied to the manufacture of microwave amplifiers and could be further developed to assist in the manufacturing of more complex microwave products.

BIBLIOGRAPHY

Adlassnig, K.P.; Kolarz, G.; Scheithauer, W., "Present State of the Medical Expert System Cadiag-2", ***Methods Inf. Med.***, Vol. 24, No. 1, pp. 13-20, 1985.

Adlassnig, K.P.; Kolarz, G.; Scheithauer, W.; Effenberger, H.; Grabner, G., Cadiag: "Approaches to Computer Assisted Medical Diagnosis", ***Comput. Biol. & Med.***, Vol. 15, No. 5, pp.: 315-35, 1985.

Aoyagi, K.; Tanemura, K.; Matsumoto, H.; Eki, Y.; Nigawara, S., "An Expert System for Startup Scheduling and Operation Support in Fossil Power Plant", ***Proceedings of the International Workshop on Artificial Intelligence for Industrial Applications: IEEE AI '88***, pp.: 167-72, 1988.

Bare, W.H.; Mulholland, R.J.; Sofer, S.S., "Design of a Self-Tuning Expert Controller for a Gasoline Refinery Catalytic Reformer", ***Proceedings of the 1988 American Control Conference***, pp.: 247-53, 1988.

Batur, C.; Kasparian, V., "A Real Time Fuzzy Self Tuning Control", ***Proceedings of the 1989 American Control Conference***, pp.: 1810-15, 1989.

Bernard, J.A., "Use of a Rule-Based System for Process Control", ***IEEE Control Syst. Mag.***, Vol. 8, No. 5, pp.: 3-13, 1988.

Buckley, J.J., "Managing Uncertainty in a Fuzzy Expert System, ***Int. J. Man-Mach. Stud.***, Vol. 29, No. 2, pp.: 129-48, 1988.

Buenaflor, M.; Finch, H.G., "'Training' a Simulation for Emergency Response", ***Artificial Intelligence and Simulation: The Diversity of Applications. Proceedings of the SCS Multiconference on Artificial Intelligence and Simulation: The Diversity of Applications***, pp.: 221-4, 1988.

Cai Jingqiu; Guo Hong, "The Structure and Design of a Fuzzy Expert System", ***Mini-Micro Syst.***, Vol. 10, No. 2, pp.: 7-11, 1989.

Chang, T.C.; Ibbs, C.W., "A Fuzzy Expert System for Priority Ranking in Network Resource Allocation", ***Artificial Intelligence Techniques and Applications for Civil and Structural Engineers***, pp.: 101-8, 1989.

Chiu, S.; Togai, M., "A Fuzzy Logic Programming Environment for Real-Time Control", ***Int. J. Aprox. Reason.***, Vol. 2, No. 2, pp.: 163-75, 1988.

Clema, J.K.; Werling, R.; Chande, A., "Expert Systems for Realtime Applications", ***Proceedings of the IEEE 1985 National Aerospace and Electronics Conference NAECON 1985***, Vol. 2, pp.: 1322-9, 1985.

Efstathiou, J., "Rule-Based Process Control Using Fuzzy Logic, Approximate Reasoning In Intelligent Systems", ***Decision and Control***, pp.: 145-58, 1987.

Franczyk, B., "Methods for Dealing with Vague Knowledge in Expert Medical Systems", ***Wiss. Z. Tech. Hochsch. Ilmenau***, Vol. 34, No. 1, pp.: 25-40, 1988.

Ganoe, F.J.; Whalen, T.H.; Tabor, C.D., "Evaluation Of the Cause Diagnosis Function of a Prototype Fuzzy-Logic-Based Knowledge System for Financial Arrays", ***Proceedings of the 1986 IEEE International Conference on Systems, Man, and Cybernetics***, pp.: 644-9 Vol. 1, 1986.

Graham, I., "Fuzzy Sets Towards Higher Order Fuzzy Logic", ***Fuzzy Sets & Syst.***, Vol. 23, No. 1, Pp.: 19-32, 1987.

Guth, M.A.S., "An Expert System Design Incorporating Fuzzy Logic for Diagnosing Heat Balances in a Nuclear Power Plant", ***Third Annual Expert Systems In Government Conference Proceedings***, pp.: 210-16, 1987.

Hao Ying; Sheppard, L.; Tucker, D., "Expert-System-Based Fuzzy Control of Arterial Pressure by Drug Infusion", ***Med. Prog. Through Technol.***, Vol. 13, No. 4, pp.: 203-15, 1988.

Hirota, K., "Recent Trends On Fuzzy Control Applications", ***J. Soc. Instrum. Control Eng.***, Vol. 28, No. 11, pp.: 970-5, 1989.

Hudson, D.L.; Cohen, M.E., "Approximate Reasoning in a Diagnostic Program for Chest Pain Analysis", ***Recent Developments in the Theory and Applications of Fuzzy Sets. Proceedings of NAFIPS '86 - 1986 Conference of the North American Fuzzy Information Processing Society***, pp.: 249-59, 1986.

Hudson, D.L.; Cohen, M.E., "Fuzzy Logic in Medical Expert Systems", ***Proceedings of the 26th IEEE Conference on Decision and Control***, pp.: 337-42, Vol. 1, 1987.

Hughes, C., "The Representation of Uncertainty in Medical Expert Systems", ***Med. Inform.***, Vol. 14, No. 4, pp.: 269-79, 1989.

Kara-Zaitri, C., "Application of Fuzzy Logic to Safety Assessment", ***Reliability 89 Proceedings***, pp.: 2-31, Vol.1, 1989.

Kerr, R.M.; Walker, R.N., "A Job Shop Scheduling System Based on Fuzzy Arithmetic", ***Proceedings of the Third International Conference. Expert Systems and the Leading Edge in Production and Operations Management***, pp.: 433-50, 1989.

Kitowski, J.; Bargiel, M., "Diagnostics Of Faulty States In Complex Physical Systems Using Fuzzy Relational Equations, Approximate Reasoning in Intelligent Systems", ***Decision and Control***, pp.: 175-94, 1987.

Lebailly, J.; Martin-Clouaire, R.; Prade, A., "Use of Fuzzy Logic in a Rule-Based System in Petroleum Geology", ***Approximate Reasoning in Intelligent Systems, Decision and Control***, pp.: 125-44, 1987.

Lee, J.C.; Gmytrasiewicz, P.; Hassberger, J.A.; Reifman, J., "Generating Rules in Expert Systems for Nuclear Power Plant Diagnostics", ***Trans. Am. Nucl. Soc.***, Vol. 55, pp.: 62-4, 1987.

Linkens, D.A.; Greenhow, S.G.; Asbury, A.J.; Rob, H.M., "Recent Advances in an Expert Advisor for Anaesthesia", ***British Medical Informatics Society Expert Systems in Medicine Fifth Annual Meeting***, pp.: 2, 1988.

Matsumoto, N.; Kuraoka, H.; Ohka, N.; Ohba, M.; Tabe, T., "Expert Anti Skid System", ***Proceedings of IECON 87: 1987 International Conference on Industrial Electronics, Control, and Instrumentation***, pp.: 810-16, Vol. 2, 1987.

Pietranski, J.F.; Marsolan, N.F.; King, K.H., "Expert Fuzzy Process Control of a Rotary Dryer", ***Proceedings of the 1987 American Control Conference***, pp.: 1359-62, Vol. 2, 1987.

Schneider, M.; Friedman, M.; Kandel, A., "The Use of Fuzzy Relations in Pattern Recognition", ***Proceedings of the 1988 IEEE International Conference on Systems, Man, And Cybernetics***, pp.: 66-9 Vol. 1, 1988.

Shiraishi, N.; Furuta, H.; Umano, M.; Kawakami, K., "Knowledge Based Expert System for Damage Assessment Based on Fuzzy Reasoning", ***Artificial Intelligence Techniques and Applications for Civil and Structural Engineers***, pp.: 211-16, 1989.

Sims Williams, J.; Matthewman, A.; Brown, D., "An Orthodontic Expert System", ***Fuzzy Sets Syst.***, Vol. 30, No. 2, pp.: 121-33, 1989.

Sripada, N.R.; Fisher, D.G.; Morris, A.J., "AI Application For Process Regulation And Servo Control", ***IEE Proc.***, Vol. 134, No. 4, pp.: 251-9, 1987.

Togai, M.; Watanabe, H., "Expert System on a Chip: An Engine for Real-Time Approximate Reasoning", ***IEEE Expert***, Vol. 1, No. 3, pp.: 55-62, 1986.

Togai, M., "A Fuzzy Inverse Relation Based on Godelian Logic and its Applications", ***Fuzzy Sets & Syst.***, Vol. 17, No. 2, pp.: 211-19, 1985.

Tsuji, S.; Amano, M.; Hikita, S., "Application of the Expert System to Elevator Group-Supervisory Control", ***Proceedings. The Fifth Conference on Artificial Intelligence Applications***, pp.: 287-94, 1989.

Vachtsevanos, G.; Davey, K., "Fault Diagnostics For The Space Station Thermal Control System Using A Hybrid Analytical/Intelligent Approach", ***Proceedings of the IEEE International Symposium on Intelligent Control***, pp.: 54-8, 1987.

Vachtsevanos, G.; Hexmoor, H.; Purves, B., "On the Intelligent Control of the Space Station Common Module Thermal System", ***Proceedings of the 1986 IEEE International Conference on Systems, Man, and Cybernetics***, pp.: 1390-4, Vol. 2, 1986.

Vaija, P.; Jarvelainen, M.; Dohnal, M., "Multilevel Failure Detection System", ***Comput. Ind.***, Vol. 6, No. 4, pp.:253-63, 1985.

Whalen, T., "Fuzzy Knowledge Based Systems in Management", ***Hum. Syst. Manage.***, Vol. 4, pp.: 263-74, 1984.

Zadeh, L.A., "Fuzzy Sets", ***Information and Control***, Vol. 8, pp.: 338-353, 1965.

Zadeh, L.A., "The Role of Fuzzy Logic in the Management of Uncertainty in Expert Systems", ***Fuzzy Sets and Syst.***, Vol. 11, pp. 199-227, 1983.

Zadeh, L.A., "Syllogistic Reasoning in Fuzzy Logic and its Application to Reasoning With Dispositions", ***Proceedings of the Fourteenth International Symposium on Multiple-Valued Logic***, pp.: 148-53, 1984.

Zadeh, L.A., "Making Computers Think Like People", ***IEEE Spectrum***, Vol. 21, pp.: 26-32, 1984.

Zadeh, L.A., "Syllogistic Reasoning in Fuzzy Logic and its Application to Usuality and Reasoning with Dispositions", ***IEEE Trans. Syst. Man & Cybern.***, Vol. 15, pp.: 754-63, 1985.

Zadeh, L.A., "A Theory of Usuality Based on Fuzzy Logic, Recent Developments in the Theory and Applications of Fuzzy Sets", ***Proceedings of Nafips '86 - 1986 Conference of the North American Fuzzy Information Processing Society***, pp.: 642, 1986.

Zadeh, L.A., "Management of Uncertainty in Expert Systems", ***Theory and Application of Expert Systems in Emergency Management Operations***, pp.: 22-39, 1986.

Zadeh, L.A., "Theory of Evidential Reasoning: A Critical Analysis", ***Proceedings-2nd International Symposium on Artificial Intelligence and Expert Systems***, pp. 329-75, 1988.

APPENDIX A

PROGRAM LISTING DT2801.C

This Appendix contains the listing of the program DT2801 which was used to control a Data Translation data acquisition card. This card was contained in the PC and was used to interface from the unit under test to the Fuzzy C program.

```
/****************************************************************************/
/*                                                                          */
/*    FILE:   DT2801.C                                                      */
/*                                                                          */
/*    PURPOSE:  To control the DT2801 Data Acquisition board.               */
/*                      *** CAUTION ***                                     */
/*              The DT2801 board's hardware jumpers must be                 */
/*              configured in the following way:                            */
/*                   A/D ... Differential, Bipolar +/- 10 Volts             */
/*                   D/A ... Bipolar +/- 10 Volts                           */
/*              Any other configuration will require modifications to       */
/*              this program.                                               */
/*                                                                          */
/*    LATEST UPDATE:  Oct. 4, 1990                                          */
/*                                                                          */
/*    LANGUAGE:  Microsoft Quick C version 2.1                              */
/*                                                                          */
/*    Copyright (c)  Mark de Payrebrune, 1990                               */
/*    All Rights Reserved                                                   */
/*                                                                          */
/*    USAGE:                                                                */
/*            The routines that can be called by the user are:              */
/*                                                                          */
/*            RESET DT2801 BOARD                                            */
/*                  void reset_board( void )                                */
/*                     Initializatin of the DT2801 board is performed.      */
/*                                                                          */
/*            READ FROM A/D CONVERTER                                       */
/*                   float read_a_to_d( int channel, int gain)              */
/*                      The A/D converter channel (0-7) and the A/D         */
/*                      gain (1,2,4,8) are set.  A conversion is initiated */
/*                      on the specified channel.  This function returns    */
/*                      the measured voltage as a floating point number     */
/*                      between -10 and +10 volts with 12 bit resolution.   */
/*                                                                          */
/*                   int read_a_to_d_bin( int channel, int gain)            */
/*                      Same as above but the value returned is the         */
/*                      12 bit binary value read from the A/D converter.    */
/*                                                                          */
/*             WRITE TO D/A CONVERTER                                       */
/*                   void d_to_a_out( int channel, float data_value )       */
/*                      The D/A converter channel (0,1) is selected and     */
/*                      the floating point data value (+/-10.)is sent to    */
/*                      the selected D/A port.  The output range of the     */
/*                      D/A converter is +10 to -10 volts with 12 bit       */
/*                      resolution.                                         */
/*                                                                          */
/*              READ FROM DIGITAL PORT                                      */
/*                   int digital_in( int channel )                          */
/*                      Set digital channel (0,1) for input and read        */
```

```
/*                      an 8 bit byte from the channel.  The function     */
/*                      returns an integer containing the digital data.   */
/*                                                                        */
/*              WRITE TO DIGITAL PORT                                     */
/*                   void digital_out( int channel, int output_data )     */
/*                      Set digital channel (0,1) for output and write    */
/*                      an 8 bit byte (0-255) to the selected channel.    */
/*                                                                        */
/*     THE FOLLOWING SAMPLE PROGRAM ILLUSTRATES THE USE OF THESE ROUTINES. */
/*                                                                        */
/*          #include <stdio.h>                                            */
/*          #include "dt2801.h"                                           */
/*                                                                        */
/*          main()                                                        */
/*          {                                                             */
/*             int channel, gain, data;                                   */
/*             float volts;                                               */
/*                                                                        */
/*             reset_board();                                             */
/*                                                                        */
/*             channel = 0;                                               */
/*             gain = 1;                                                  */
/*             volts = read_a_to_d( channel, gain );                      */
/*             printf( "A/D channel %d = %f \n", channel, volts );        */
/*                                                                        */
/*             channel = 1;                                               */
/*             volts = 1.2;                                               */
/*             d_to_a_out( channel, volts );                              */
/*                                                                        */
/*             channel = 0;                                               */
/*             data = digital_in( channel );                              */
/*             printf( "Digital channel %d = %x \n", channel, data );     */
/*                                                                        */
/*             channel = 1;                                               */
/*             data = 0x81;                                               */
/*             digital_out( channel, 0x81 );                              */
/*          }                                                             */
/*                                                                        */
/**************************************************************************/


#include "dt2801.h"   /* Definitions for DT2801 board */
```

```
/**************************************************************************/
/*                                                                        */
/*    READ_A_TO_D( channel, gain )                                        */
/*                                                                        */
/*    Read specified A/D channel with specified gain, convert measured    */
/*    12 bit integer into a floating point voltage and return.  Permitted */
/*    values of channel number are 0 - 7 and gain are 1,2,4 or 8.  The    */
/*    A/D converter measures bipolar inputs in the range of +/- 10 Volts. */
/*    This routine returns a measured floating point voltage.             */
/**************************************************************************/

float read_a_to_d( int channel, int gain)
{
   int low;               /* Low byte of 12 bit word read back from A/D   */
   int high;              /* High  "   "  "  "   "     "    "   "    "   */
   int gain_code;         /* Gain converted to a value between 0 and 3    */

   float data_value;      /* Measured 12 bit A/D data (High + Low bytes)  */
   float factor;          /* Resolution of A/D converter in Volts         */
   float volts;           /* Measure A/D voltage in Volts                 */

   switch( gain )         /* Check that gain is within permitted    */
   {                      /* limits and convert gain # to a gain    */
   case 1:                /* code that can be passed to the DT2801. */
      gain_code = 0;
      break;
   case 2:
      gain_code = 1;
      break;
   case 4:
      gain_code = 2;
      break;
   case 8:
      gain_code = 3;
      break;
   default:
      printf( "\n\n\n Gain out of range in A/D command \n\n\n" );
      abort();
      break;
   }

   if( (channel  0) || (channel  7) )    /* Is Channel # OK? */
   {
     printf( "\n\n\n Channel out of range in A/D command \n\n\n" );
     abort();
   }

   wait_data_in();             /* Check that previous commands are finished */
   wait_ready();
   outp( COMMAND_REG, READ_A_TO_D );  /* Send READ A/D command */
```

```
    wait_data_in();
    outp( DATA_REG, gain_code );        /* Set A/D gain */
    wait_data_in();
    outp(DATA_REG, channel );           /* Set channel# and start conversion */
    wait_data_out();
    low = inp( DATA_REG );              /* Read low byte of mesured value */
    wait_data_out();
    high = inp( DATA_REG );             /* Read high byte of measured value */
    check_for_error();                  /* Make sure no error occurred */

    data_value = high * 256 + low;      /* Combine low and high data bytes */
    factor = (10./4096.)/gain;          /* Compute voltage resolution */

    volts = data_value * factor * 2. - (10. / gain);     /* Compute voltage */

    return( volts );                    /* Return measured voltage */
}
```

```
/***************************************************************************/
/*                                                                         */
/*    D_TO_A_OUT( channel, data_value )                                    */
/*                                                                         */
/*    Send a floating point number to the specified D/A channel.  The      */
/*    permitted D/A channel numbers are 0 and 1.  The D/A converter is     */
/*    configured in a bipolar mode with 12 bit resoution and +/- 10 Volt   */
/*    range.  The floating point data is converted to offset binary coding */
/*    prior to being sent to the D/A converter.                            */
/***************************************************************************/

void d_to_a_out( int channel, float data_value )
{
  int number;      /* Data value converted to an integer */
  int high;        /* High byte of data value            */
  int low;         /* Low   "   "  "   "                 */

  if( (channel  0) || (channel  1) )     /* Is channel # OK? */
  {
    printf( "\n\n\n D/A channel selection error \n\n\n" );
     abort();
  }

  if( (data_value  10.) | (data_value  -10.) )   /* Is data in range */
  {
     printf( "\n\n\n D/A data value is out of range \n\n\n" );
     abort();
  }

  number = ( data_value + 10. ) * 4096. / 20.;    /* Conv data to integer */
  high = number / 256;                            /* Get high byte        */
  low = number - high * 256;                      /* Get low byte         */

  wait_data_in();                       /* Check if previous cmd finished */
  wait_ready();
  outp( COMMAND_REG, WRITE_D_TO_A );    /* Send write command to D/A */
  wait_data_in();
  outp( DATA_REG, channel );            /* Set D/A channel # */
  wait_data_in();
  outp( DATA_REG, low );                /* Send low byte of voltge */
  wait_data_in();
  outp( DATA_REG, high );               /* Send high byte of voltage */
  wait_data_in();
  wait_ready();
  check_for_error();                    /* Check that no error occurred */
}
```

```
/***************************************************************************/
/*                                                                         */
/*    DIGITAL_IN( channel )                                                */
/*                                                                         */
/*    Read an 8 bit byte from the specified digital input channel.  There  */
/*    are 2 channels ( 0 and 1 ).  The routine sets the specified channel  */
/*    to input mode and then returns the data present on the port.         */
/***************************************************************************/

int digital_in( int channel )
{
   int read_data;                    /* The data read from the digital port */

   if( (channel  0 ) || (channel  1) )          /* Is channel # valid ? */
    {
      printf( "\n\n\n Channel out of range in DIGITAL_IN call \n\n\n" );
      abort();
    }

   wait_data_in();            /* Check that previous command is finished */
   wait_ready();
   outp( COMMAND_REG, SET_DIG_PORT_INPUT );  /* Set digital port for input */
   wait_data_in();
   outp( DATA_REG, channel );   /* Send digital port # */
   wait_data_in();
   wait_ready();
   check_for_error();           /* Check that no error occurred */

   outp( COMMAND_REG, READ_DIG_INPUT );   /* Send read port command */
   wait_data_in();
   outp( DATA_REG, channel );             /* Set port # to read from */
   wait_data_out();
   read_data = inp( DATA_REG );           /* Read data */
   wait_data_in();
   wait_ready();
   check_for_error();                     /* Check that no error occurred */

   return( read_data );
}
```

```
/****************************************************************************/
/*                                                                          */
/*    DIGITAL_OUT( channel, output_data )                                   */
/*                                                                          */
/*    Send an 8 bit byte to the specified digital output channel.  There    */
/*    are two output channels (0 and 1).  This routine sets the specified   */
/*    channel to output mode and then sends the output data to the channel */
/****************************************************************************/

void digital_out( int channel, int output_data )
{
   if( (channel  0 ) || (channel  1) )     /* Is channel # valid ? */
    {
       printf( "\n\n\n Channel out of range in DIGITAL_OUT call \n\n\n" );
       abort();
    }

   if( (output_data  0 ) || (output_data  255) )  /* Is data valid ? */
    {
      printf( "\n\n\n Data is out of range in DIGITAL_OUT call \n\n\n" );
       abort();
    }

   wait_data_in();    /* Check that previous command is finished */
   wait_ready();
   outp( COMMAND_REG, SET_DIG_PORT_OUTPUT );   /* Set port for output */
   wait_data_in();
   outp( DATA_REG, channel );    /* Set port # */
   wait_data_in();
   wait_ready();
   check_for_error();            /* Check that no errors occurred */

   outp( COMMAND_REG, WRITE_DIG_OUTPUT );  /* Send write port command */
   wait_data_in();
   outp( DATA_REG, channel );              /* Set digital port */
   wait_data_in();
   outp( DATA_REG, output_data );          /* Send output data to port */
   wait_data_in();
   wait_ready();
   check_for_error();                 /* Check that no errors occurred */
}
```

```
/****************************************************************************/
/*                                                                          */
/*    RESET_BOARD()                                                         */
/*                                                                          */
/*    The reset command initializes the DT2801 board.  A stop command is   */
/*    sent to terminate all execution followed by the reset command.  A    */
/*    functional test of the board is performed by the TEST command.       */
/*    The test command puts a 1 in the data register and increments this   */
/*    value each time the data register is read.  The DT2801 board is      */
/*    verified by comparing the expected value to the value read from      */
/*    the data register.  If an error should occurr the program is         */
/*    aborted.  The test command is terminated after 256 values are read   */
/*    and the error registers are then cleared.  The DT2801 board is       */
/*    now reset and verified.                                              */
/****************************************************************************/

reset_board()
{
   int temp;         /* Dummy variable used to read data after stop command */
   int i;            /* Loop increment variable */
   int count = 0;    /* Data read in from the DT2801 during TEST command */

   outp( COMMAND_REG, STOP_COMMAND );   /* Stop all board execution */
   temp = inp( DATA_REG );              /* Read in garbage */
   wait_data_in();
   wait_ready();

   outp( COMMAND_REG, RESET );      /* Send RESET command */
   wait_data_out();
   temp = inp( DATA_REG );          /* Read in garbage */
   wait_ready();
   check_for_error();               /* Make sure that no error occurred */

   outp( COMMAND_REG, TEST );       /* Send TEST command */

   for( i=1; i<=255; i++ )          /* Loop 255 times and compare DATA_REG */
   {                                /* values with 'i' loop values.  If    */
     wait_data_out();              /* they are not = then flag an error.  */
      count = inp( DATA_REG );
      if( i != count )
      {
    printf( "\n\n\n Communication test with DT2801 Board FAILED \n\n\n" );
    abort();
      }
   }

   outp( COMMAND_REG, STOP_COMMAND );   /* Stop the TEST command */
   temp = inp( DATA_REG );              /* Read in garbage */
   wait_data_in();
   wait_ready();
```

```
    outp( COMMAND_REG, CLEAR_ERROR );    /* Clear the error generated when   */
    wait_data_in();                      /* the test command was stopped.    */
    wait_ready();
}
```

```
/***************************************************************************/
/*                                                                         */
/*    CHECK_FOR_ERROR()                                                    */
/*                                                                         */
/*    Read the status register and check if the MSB is set.  If it is set  */
/*    then issue the STOP command, print out the error and abort the       */
/*    program.                                                             */
/***************************************************************************/

check_for_error()
{
   int temp;         /* Dummy variable used to read data after stop command */
   int status;       /* Value read from status register */
   int err1;         /* First byte read from error register */
   int err2;         /* Second byte read from error register */

   status = inp( STATUS_REG );                      /* Read present status */
   if( (status & ERROR_FLAG) == ERROR_FLAG )        /* Check for error bit set */
   {
      outp( COMMAND_REG, STOP_COMMAND );           /* Issue stop command */
      temp = inp( DATA_REG );                      /* Read in garbage */
      wait_data_in();
      wait_ready();

      outp( COMMAND_REG, READ_ERROR_REG );         /* Issue read error command */
      wait_data_out();
      err1 = inp( DATA_REG );                      /* Read in first error byte */
      wait_data_out();
      err2 = inp( DATA_REG );                      /* Read in second error byte */

      if( err1 & 1 )
         printf( "\n\n\n Reserved error message \n\n\n" );
      if( err1 & 2 )
         printf( "\n\n\n Command overwrite error \n\n\n" );
      if( err1 & 4 )
         printf( "\n\n\n Clock set error \n\n\n" );
      if( err1 & 8 )
         printf( "\n\n\n Digital port select error \n\n\n" );
      if( err1 & 16 )
         printf( "\n\n\n Digital port set error \n\n\n" );
      if( err1 & 32 )
         printf( "\n\n\n DAC select error \n\n\n" );
      if( err1 & 64 )
         printf( "\n\n\n DAC clock error \n\n\n" );
      if( err1 & 128 )
         printf( "\n\n\n DAC # conversions error \n\n\n" );

      if( err2 & 1 )
         printf( "\n\n\n A/D channel error \n\n\n" );
      if( err2 & 2 )
```

```
      printf( "\n\n\n A/D gain error \n\n\n" );
    if( err2 & 4 )
      printf( "\n\n\n A/D clock error \n\n\n" );
    if( err2 & 8 )
      printf( "\n\n\n A/D multiplexer error \n\n\n" );
    if( err2 & 16 )
      printf( "\n\n\n A/D # conversions error \n\n\n" );
    if( err2 & 32 )
      printf( "\n\n\n Data where command expected error \n\n\n" );
    if( err2 & 64 )
      printf( "\n\n\n Reserved error \n\n\n" );
    if( err2 & 128 )
      printf( "\n\n\n Reserved error \n\n\n" );

    abort();      /* Abort program */
  }
}
```

```
/*************************************************************************/
/*                                                                       */
/*    WAIT_READY()          (loop until flag is set)                     */
/*    WAIT_DATA_OUT()       ( "     "     "   "  " )                     */
/*    WAIT_DATA_IN()        (loop until flag is cleared)                 */
/*                                                                       */
/*    The above functions are low level routines used to check status    */
/*    register flags and to loop until they are set.  In the routine     */
/*    WAIT_DATA_OUT an error check is performed on the READY flag.  If the */
/*    READY flag is set while the program is waiting for data to come in, */
/*    it implies that the command has already finished and could reslt in */
/*    in an infinite loop.  If this condition is detected the error flag  */
/*    is checked and the program is aborted.                             */
/*************************************************************************/

wait_ready()
{
   int status;        /* Data read from status register */
   int flag = 0;        /* Flag is set to 1 when READY flag is set */

   while( flag == 0 )
   {
     status = inp( STATUS_REG );                  /* Read status register */
     if( (status & READY_FLAG) == READY_FLAG )    /* Check READY flag */
        flag = 1;       /* Exit loop, READY flag is set */
     else
        flag = 0;       /* Continue looping */
   }
}

wait_data_out()
{
   int status;        /* Data read from status register */
   int flag = 0;        /* Flag is set to 1 when DATA_OUT_READY flag is set */

   while( flag == 0 )
   {
     status = inp( STATUS_REG );                  /* Read status reg */
     if( (status & READY_FLAG ) == READY_FLAG )   /* READY flag set ? */
     {
       printf( "\n\n\n READY flag set in wait_data_out() \n\n\n" );
       check_for_error();
       abort();
     }
     else
       if( (status & DATA_OUT_READY_FLAG) == DATA_OUT_READY_FLAG )
          flag = 1;     /* Exit loop, Data_out_ready flag is set */
       else
          flag = 0;     /* Continue looping */
   }
```

```
}

wait_data_in()
{
   int status;       /* Data read from status register */
   int flag = 0;       /* Flag is set to 1 when DATA_IN_FULL flag is clear */

   while( flag == 0 )
   {
     status = inp( STATUS_REG );     /* Get status */
      if( (status & DATA_IN_FULL_FLAG) == DATA_IN_FULL_FLAG )
        flag = 0;    /* Continue looping */
      else
        flag = 1;    /* Exit loop, Data_in_full flag is now cleared */
   }
}
```

APPENDIX B

PROGRAM LISTING SETBIAS.C

The function of this program is to provide direct control over the RF slice under test through the control of the D/A interface connected to the RF slice.

This program enables the user to send an 8 bit byte to any of 5 D/A converters. The D/A converters are configured as follows:

D/A #1 ... FET #1 Gate voltage

D/A #2 ... FET #2 Gate voltage

D/A #3 ... FET #3 Gate voltage

D/A #4 ... PIN diode #1 voltage

D/A #5 ... PIN diode #2 voltage

```
/*************************************************************************/
/*                                                                       */
/*    FILE:    SET_BIAS.C                                                */
/*                                                                       */
/*    PURPOSE:  To control the ACTIVE BIAS Breadboard.  This program     */
/*              permits the user to send an 8 bit byte to any of         */
/*              5 D/A converters in order to control the RF slice.       */
/*                                                                       */
/*    LATEST UPDATE:  Oct. 30, 1990                                      */
/*                                                                       */
/*    LANGUAGE:  Microsoft Quick C version 2.1                           */
/*                                                                       */
/*    Copyright (c)  Mark de Payrebrune, 1990                            */
/*    All Rights Reserved                                                */
/*                                                                       */
/*    USAGE:                                                             */
/*            The ACTIVE BIAS breadboard is configured as follows:       */
/*                                                                       */
/*                 D/A #1 ... FET #1 Gate voltage                        */
/*                 D/A #2 ... FET #2 Gate voltage                        */
/*                 D/A #3 ... FET #3 Gate voltage                        */
/*                 D/A #4 ... PIN diode #1 voltage                       */
/*                 D/A #5 ... PIN diode #2 voltage                       */
/*                                                                       */
/*            An example of the use of this routine is shown below:      */
/*                                                                       */
/*                #include <stdio.h>                                     */
/*                #include "dt2801.h"                                    */
/*                void set_bias( int channel, int data );                */
/*                                                                       */
/*                main()                                                 */
/*                {                                                      */
/*                   int foo, channel, data;                             */
/*                                                                       */
/*                   reset_board();                                      */
/*                                                                       */
/*                   channel = 1;                                        */
/*                   data = 128;                                         */
/*                   set_bias( channel, data );                          */
/*                }                                                      */
/*                                                                       */
/*    NOTES:                                                             */
/*            The active bias board is connected to the DT2801 data      */
/*            acquisition board through an EPROM socket.  The data lines */
/*            are connected to port 0 of the DT2801 and the address      */
/*            lines are connected to port 1.  The WR line is the highest */
/*            order bit of port 1.  An optical isolator on each line is  */
/*            used to isolate the active bias box from the computer and to */
/*            translate from TTL to CMOS levels.  The opto               */
/*            isolators invert the data and therefore another inversion  */
```

```
/*              is done in software before the data is sent to the active    */
/*              bias board.  The data is written to the appropriate D/A      */
/*              converter when the write line (WR) is changed from high      */
/*              to low state and back to high.  The channel number must be   */
/*              between 1 and 7 and the data must be between 0 and 255.      */
/*                                                                           */
/*****************************************************************************/

#include <stdio.h>
#include "dt2801.h"
void set_bias( int channel, int data );

void set_bias( int channel, int data )
{

   channel = 7 - (channel-1);   /* Subtract 1 from channel and invert */

   if( (channel  0) || (channel  7) )    /* Is Channel # OK? */
   {
      printf( "\n\n\n Channel out of range in SET_BIAS \n\n\n" );
      abort();
   }

   data = 255 - data;             /* Invert data bits */

   if( (data  0) || (data  255) )    /* Is Channel # OK? */
   {
      printf( "\n\n\n Data out of range in SET_BIAS \n\n\n" );
      abort();
   }

   digital_out( 1, 0x00 | channel);   /* Set WR pin high AD7228 */
   digital_out( 0, data );            /* Send data to AD7228 */
   digital_out( 1, 0x80 | channel);   /* Toggle WR pin low to set-data reg */
   digital_out( 1, 0x00 | channel);   /* Set WR pin back to high */
}
```

APPENDIX C

PROGRAM LISTINGS

FACTOR.TIL

This Appendix contains a listing of the files FACTOR1, FACTOR2, FACTOR3, FACTOR4 and FACTOR5.TIL. These programs are written in the Togai fuzzy C compiler language and are used to scale control parameters before they are used to control the RF slice under test.

Program FACTOR1.TIL

```
PROJECT FACTOR1

  VAR gate_value
    TYPE  float
    MIN    0.
    MAX    255.

    MEMBER extremely_low
      POINTS 0. 1 88. 1 108. 0
        END

    MEMBER very_low
      POINTS 88. 0 108. 1 128. 0
        END

    MEMBER low
      POINTS 108. 0 128. 1 148. 0
        END

    MEMBER nominal
      POINTS 128. 0 148. 1 168. 0
        END

    MEMBER high
      POINTS 148. 0 168. 1 188. 0
        END

    MEMBER very_high
      POINTS 168. 0 188. 1 208. 0
        END

    MEMBER extremely_high
      POINTS 188. 0 208. 1 255. 1
        END
  END

  VAR control_value
    TYPE float
    MIN  -30.0
    MAX   30.0

    MEMBER negative
      POINTS -30. 1 0.0 1 .1 0
```

```
        END

   MEMBER positive
      POINTS -.1 0 0.0 1 30. 1
         END
END

VAR gate_factor
   TYPE float
   MIN  0.0
   MAX  1.0
   MEMBER small
      POINTS 0. 1 .2 1 .4 0
         END

   MEMBER medium
      POINTS .2 0 .4 1 .6 0
         END

   MEMBER large
      POINTS .4 0 .6 1 .8 0
         END

   MEMBER very_large
      POINTS .6 0 .8 1 1.0 1
         END
END

FUZZY factor_1
   OPTIONS
      OUTPUTSCOPE="PUBLIC"
         END

   RULE Rule1
      If gate_value is extremely_low and control_value is positive then
         gate_factor = very_large
            END
   RULE Rule2
      If gate_value is extremely_low and control_value is negative then
         gate_factor = small
            END

   RULE Rule3
      If gate_value is very_low and control_value is positive then
         gate_factor = large
            END
   RULE Rule4
      If gate_value is very_low and control_value is negative then
         gate_factor = small
            END
```

```
    RULE Rule5
      If gate_value is low and control_value is positive then
        gate_factor = medium
          END
    RULE Rule6
      If gate_value is low and control_value is negative then
        gate_factor = small
          END
    RULE Rule7
      If gate_value is nominal then
        gate_factor = small
          END

    RULE Rule8
      If gate_value is high and control_value is negative then
        gate_factor = medium
          END
    RULE Rule9
      If gate_value is high and control_value is positive then
        gate_factor = small
          END

    RULE Rule10
      If gate_value is very_high and control_value is negative then
        gate_factor = large
          END
    RULE Rule11
      If gate_value is very_high and control_value is positive then
        gate_factor = small
          END

    RULE Rule12
      If gate_value is extremely_high and control_value is negative then
        gate_factor = very_large
          END
    RULE Rule13
      If gate_value is extremely_high and control_value is positive then
        gate_factor = small
          END
  END

  CONNECT from gate_value to factor_1 END
  CONNECT from control_value to factor_1 END
  CONNECT from factor_1 to gate_factor END

END
```

Program FACTOR2.TIL

```
PROJECT FACTOR2

  VAR gate_value
    TYPE  float
    MIN    0.
    MAX    255.

    MEMBER extremely_low
      POINTS 0. 1 72. 1 92. 0
        END

    MEMBER very_low
      POINTS 72. 0 92. 1 112. 0
        END

    MEMBER low
      POINTS 92. 0 112. 1 132. 0
        END

    MEMBER nominal
      POINTS 112. 0 132. 1 152. 0
        END

    MEMBER high
      POINTS 132. 0 152. 1 172. 0
        END

    MEMBER very_high
      POINTS 152. 0 172. 1 192. 0
        END

    MEMBER extremely_high
      POINTS 172. 0 192. 1 255. 1
        END
  END

  VAR control_value
    TYPE float
    MIN  -30.0
    MAX   30.0

    MEMBER negative
      POINTS -30. 1 0.0 1 .1 0
```

```
        END

    MEMBER positive
      POINTS -.1 0 0.0 1 30. 1
        END
END

VAR gate_factor
    TYPE float
    MIN  0.0
    MAX  .173

    MEMBER small
      POINTS 0. 1 .035 1 .069 0
        END

    MEMBER medium
      POINTS .035 0 .069 1 .104 0
        END

    MEMBER large
      POINTS .069 0 .104 1 .138 0
        END

    MEMBER very_large
      POINTS .104 0 .138 1 .173 1
        END
END

FUZZY factor_2
    OPTIONS
      OUTPUTSCOPE="PUBLIC"
        END

    RULE Rule1
      If gate_value is extremely_low and control_value is positive then
        gate_factor = very_large
          END
    RULE Rule2
      If gate_value is extremely_low and control_value is negative then
        gate_factor = small
          END

    RULE Rule3
      If gate_value is very_low and control_value is positive then
        gate_factor = large
          END
    RULE Rule4
      If gate_value is very_low and control_value is negative then
        gate_factor = small
```

```
            END

      RULE Rule5
         If gate_value is low and control_value is positive then
            gate_factor = medium
            END
      RULE Rule6
         If gate_value is low and control_value is negative then
            gate_factor = small
            END

      RULE Rule7
         If gate_value is nominal then
            gate_factor = small
            END

      RULE Rule8
         If gate_value is high and control_value is negative then
            gate_factor = medium
            END
      RULE Rule9
         If gate_value is high and control_value is positive then
            gate_factor = small
            END

      RULE Rule10
         If gate_value is very_high and control_value is negative then
            gate_factor = large
            END
      RULE Rule11
         If gate_value is very_high and control_value is positive then
            gate_factor = small
            END

      RULE Rule12
         If gate_value is extremely_high and control_value is negative then
            gate_factor = very_large
            END
      RULE Rule13
         If gate_value is extremely_high and control_value is positive then
            gate_factor = small
            END
   END

   CONNECT from gate_value to factor_2 END
   CONNECT from control_value to factor_2 END
   CONNECT from factor_2 to gate_factor END

END
```

Program FACTOR3.TIL

```
PROJECT FACTOR3

  VAR gate_value
     TYPE  float
     MIN    0.
     MAX    255.

     MEMBER extremely_low
       POINTS 0. 1 95. 1 115. 0
         END

     MEMBER very_low
       POINTS 95. 0 115. 1 135. 0
         END

     MEMBER low
       POINTS 115. 0 135. 1 155. 0
         END

     MEMBER nominal
       POINTS 135. 0 155. 1 175. 0
         END

     MEMBER high
       POINTS 155. 0 175. 1 195. 0
         END

     MEMBER very_high
       POINTS 175. 0 195. 1 215. 0
         END

     MEMBER extremely_high
       POINTS 195. 0 215. 1 255. 1
         END
  END

  VAR control_value
     TYPE float
     MIN  -30.0
     MAX   30.0

     MEMBER negative
       POINTS -30. 1 0.0 1 .1 0
```

```
        END

   MEMBER positive
      POINTS -.1 0 0.0 1 30. 1
         END
END

VAR gate_factor
   TYPE float
   MIN  0.0
   MAX  .302

   MEMBER small
      POINTS 0. 1 .060 1 .121 0
         END

   MEMBER medium
      POINTS .060 0 .121 1 .182 0
         END

   MEMBER large
      POINTS .121 0 .182 1 .242 0
         END
   MEMBER very_large
      POINTS .182 0 .242 1 .302 1
         END
END

FUZZY factor_3
   OPTIONS
      OUTPUTSCOPE="PUBLIC"
         END

   RULE Rule1
      If gate_value is extremely_low and control_value is positive then
         gate_factor = very_large
            END
   RULE Rule2
      If gate_value is extremely_low and control_value is negative then
         gate_factor = small
            END

   RULE Rule3
      If gate_value is very_low and control_value is positive then
         gate_factor = large
            END
   RULE Rule4
      If gate_value is very_low and control_value is negative then
         gate_factor = small
            END
```

```
    RULE Rule5
      If gate_value is low and control_value is positive then
        gate_factor = medium
          END
    RULE Rule6
      If gate_value is low and control_value is negative then
        gate_factor = small
          END

    RULE Rule7
      If gate_value is nominal then
        gate_factor = small
          END

    RULE Rule8
      If gate_value is high and control_value is negative then
        gate_factor = medium
          END
    RULE Rule9
      If gate_value is high and control_value is positive then
        gate_factor = small
          END

    RULE Rule10
      If gate_value is very_high and control_value is negative then
        gate_factor = large
          END
    RULE Rule11
      If gate_value is very_high and control_value is positive then
        gate_factor = small
          END

    RULE Rule12
      If gate_value is extremely_high and control_value is negative then
        gate_factor = very_large
          END
    RULE Rule13
      If gate_value is extremely_high and control_value is positive then
        gate_factor = small
          END
  END

  CONNECT from gate_value to factor_3 END
  CONNECT from control_value to factor_3 END
  CONNECT from factor_3 to gate_factor END

END
```

Program FACTOR4.TIL

```
PROJECT FACTOR4

  VAR gate_value
     TYPE  float
     MIN    0.
     MAX    255.

     MEMBER extremely_low
        POINTS 0. 1 170. 1 180. 0
           END

     MEMBER very_low
        POINTS 170. 0 180. 1 190. 0
           END

     MEMBER low
        POINTS 180. 0 190. 1 200. 0
           END
     MEMBER nominal
        POINTS 190. 0 200. 1 210. 0
           END

     MEMBER high
        POINTS 200. 0 210. 1 220. 0
           END

     MEMBER very_high
        POINTS 210. 0 220. 1 230. 0
           END

     MEMBER extremely_high
        POINTS 220. 0 230. 1 255. 1
           END
  END

  VAR control_value
     TYPE float
     MIN  -30.0
     MAX   30.0

     MEMBER negative
        POINTS -30. 1 0.0 1 .1 0
           END
```

```
   MEMBER positive
      POINTS -.1 0 0.0 1 30. 1
         END
END

VAR gate_factor
   TYPE float
   MIN  0.0
   MAX  .077

   MEMBER small
      POINTS 0. 1 .015 1 .031 0
         END

   MEMBER medium
      POINTS .015 0 .031 1 .046 0
         END

   MEMBER large
      POINTS .031 0 .046 1 .062 0
         END

   MEMBER very_large
      POINTS .046 0 .062 1 .077 1
         END
END

FUZZY factor_4
   OPTIONS
      OUTPUTSCOPE="PUBLIC"
         END

   RULE Rule1
      If gate_value is extremely_low and control_value is positive then
         gate_factor = very_large
            END
   RULE Rule2
      If gate_value is extremely_low and control_value is negative then
         gate_factor = small
            END

   RULE Rule3
      If gate_value is very_low and control_value is positive then
         gate_factor = large
            END
   RULE Rule4
      If gate_value is very_low and control_value is negative then
         gate_factor = small
            END
```

```
    RULE Rule5
      If gate_value is low and control_value is positive then
        gate_factor = medium
          END
    RULE Rule6
      If gate_value is low and control_value is negative then
        gate_factor = small
          END

    RULE Rule7
      If gate_value is nominal then
        gate_factor = small
          END

    RULE Rule8
      If gate_value is high and control_value is negative then
        gate_factor = medium
          END
    RULE Rule9
      If gate_value is high and control_value is positive then
        gate_factor = small
          END

    RULE Rule10
      If gate_value is very_high and control_value is negative then
        gate_factor = large
          END
    RULE Rule11
      If gate_value is very_high and control_value is positive then
        gate_factor = small
          END

    RULE Rule12
      If gate_value is extremely_high and control_value is negative then
        gate_factor = very_large
          END
    RULE Rule13
      If gate_value is extremely_high and control_value is positive then
        gate_factor = small
          END
  END

  CONNECT from gate_value to factor_4 END
  CONNECT from control_value to factor_4 END
  CONNECT from factor_4 to gate_factor END

END
```

Program FACTOR5.TIL

```
PROJECT FACTOR5

  VAR gate_value
     TYPE  float
     MIN    0.
     MAX    255.

     MEMBER extremely_low
        POINTS 0. 1 170. 1 180. 0
           END

     MEMBER very_low
        POINTS 170. 0 180. 1 190. 0
           END

     MEMBER low
        POINTS 180. 0 190. 1 200. 0
           END

     MEMBER nominal
        POINTS 190. 0 200. 1 210. 0
           END

     MEMBER high
        POINTS 200. 0 210. 1 220. 0
           END

     MEMBER very_high
        POINTS 210. 0 220. 1 230. 0
           END

     MEMBER extremely_high
        POINTS 220. 0 230. 1 255. 1
           END
  END

  VAR control_value
     TYPE float
     MIN  -30.0
     MAX   30.0

     MEMBER negative
        POINTS -30. 1 0.0 1 .1 0
```

```
        END

   MEMBER positive
      POINTS -.1 0 0.0 1 30. 1
         END
END

VAR gate_factor
   TYPE float
   MIN  0.0
   MAX  .210

   MEMBER small
      POINTS 0. 1 .042 1 .084 0
         END

   MEMBER medium
      POINTS .042 0 .084 1 .126 0
         END

   MEMBER large
      POINTS .084 0 .126 1 .168 0
         END

   MEMBER very_large
      POINTS .126 0 .168 1 .210 1
         END
END

FUZZY factor_5
   OPTIONS
      OUTPUTSCOPE="PUBLIC"
         END

   RULE Rule1
      If gate_value is extremely_low and control_value is positive then
         gate_factor = very_large
            END
   RULE Rule2
      If gate_value is extremely_low and control_value is negative then
         gate_factor = small
            END

   RULE Rule3
      If gate_value is very_low and control_value is positive then
         gate_factor = large
            END
   RULE Rule4
      If gate_value is very_low and control_value is negative then
         gate_factor = small
```

```
         END

      RULE Rule5
         If gate_value is low and control_value is positive then
            gate_factor = medium
               END
      RULE Rule6
         If gate_value is low and control_value is negative then
            gate_factor = small
               END

      RULE Rule7
         If gate_value is nominal then
            gate_factor = small
               END

      RULE Rule8
         If gate_value is high and control_value is negative then
            gate_factor = medium
               END
      RULE Rule9
         If gate_value is high and control_value is positive then
            gate_factor = small
               END

      RULE Rule10
         If gate_value is very_high and control_value is negative then
            gate_factor = large
               END
      RULE Rule11
         If gate_value is very_high and control_value is positive then
            gate_factor = small
               END

      RULE Rule12
         If gate_value is extremely_high and control_value is negative then
            gate_factor = very_large
               END
      RULE Rule13
         If gate_value is extremely_high and control_value is positive then
            gate_factor = small
               END
   END

   CONNECT from gate_value to factor_5 END
   CONNECT from control_value to factor_5 END
   CONNECT from factor_5 to gate_factor END

END
```

APPENDIX D

PROGRAM LISTING

BIAS.TIL

This appendix contains the listing of the program BIAS.TIL. This program is written in the TIL Fuzzy C language. It establishes the main membership functions used to control the RF slice.

```
PROJECT bias

   VAR gain
      TYPE  float
      MIN   -1.25
      MAX    1.25
      MEMBER extremely_low
         POINTS -1.25 1 -.3 1 -.2 0
            END
      MEMBER very_low
         POINTS -.3 0 -.2 1 -.1 0
            END
      MEMBER low
         POINTS -.2 0 -.1 1 0.0 0
            END
      MEMBER ok
         POINTS -.1 0 0.0 1 .1 0
            END
      MEMBER high
         POINTS 0.0 0 .1 1 .2 0
            END
      MEMBER very_high
         POINTS .1 0 .2 1 .3 0
            END
      MEMBER extremely_high
         POINTS .2 0 .3 1 1.25 1
            END
   END

   VAR dgain
      TYPE float
      MIN -2.5
      MAX  2.5
      MEMBER decreasing
         POINTS -2.5 1 -.05 1 0.0 0
            END
      MEMBER small
         POINTS -.05 0 0.0 1 0.05 0
            END

      MEMBER increasing
         POINTS 0.0 0 .05 1 2.5 1
            END
   END

   VAR controller
      TYPE float
      MIN -5.0
      MAX  5.0
      MEMBER max_decrease
```

```
    POINTS -5.0 1 -2.0 1 -1.0 0
      END
  MEMBER decrease
    POINTS -2.0 0 -1.0 1 0.0 0
      END
  MEMBER idle
    POINTS -1.0 0 0.0 1 1.0 0
      END
  MEMBER increase
    POINTS 0.0 0 1.0 1 2.0 0
      END
  MEMBER max_increase
    POINTS 1.0 0 2.0 1 5.0 1
      END
END

FUZZY gain_control
  OPTIONS
    OUTPUTSCOPE="PUBLIC"
      END
  RULE Rule1
    If gain is extremely_low then
      controller = max_increase
        END

  RULE Rule2
    If gain is very_low and dgain is decreasing then
      controller = max_increase
        END
  RULE Rule3
    If gain is very_low and dgain is increasing then
      controller = increase
        END
  RULE Rule4
    If gain is very_low and dgain is small then
      controller = max_increase
        END

  RULE Rule5
    If gain is low and dgain is decreasing then
      controller = increase
        END
  RULE Rule6
    If gain is low and dgain is increasing then
      controller = idle
        END
  RULE Rule7
    If gain is low and dgain is small then
      controller = increase
        END
```

```
    RULE Rule8
      If gain is ok and dgain is increasing then
        controller = decrease
          END
    RULE Rule9
      If gain is ok and dgain is decreasing then
        controller = increase
          END
    RULE Rule10
      If gain is ok then
        controller = idle
          END

    RULE Rule11
      If gain is high and dgain is decreasing then
        controller = idle
          END
    RULE Rule12
      If gain is high and dgain is increasing then
        controller = decrease
          END
    RULE Rule13
      If gain is high and dgain is small then
        controller = decrease
          END

    RULE Rule14
      If gain is very_high and dgain is decreasing then
        controller = decrease
          END
    RULE Rule15
      If gain is very_high and dgain is increasing then
        controller = max_decrease
          END
    RULE Rule16
      If gain is very_high and dgain is small then
        controller = max_decrease
          END

    RULE Rule17
      If gain is extremely_high then
        controller = max_decrease
          END
  END

  CONNECT from gain to gain_control END
  CONNECT from dgain to gain_control END
  CONNECT from gain_control to controller END
END
```

APPENDIX E

PROGRAM LISTING

BIASDRV.C

This Appendix contains a listing of the program BIASDRV.C. This program is the main program used to control the RF slice and calls the Fuzzy C routines and low level routines used to control the RF slice.

```
/***************************************************************************/
/*      FILE:   BIASDRV.C                                                  */
/*                                                                         */
/*      PURPOSE:        Main program to control an RF slice using the      */
/*                      subroutines generated by the Togai Fuzzy C         */
/*                      compiler.                                          */
/*                                                                         */
/*      LATEST UPDATE:  Oct. 12, 1990                                      */
/*                                                                         */
/*      LANGUAGE:       Microsoft Quick C version 2.1                      */
/*                                                                         */
/*      Copyright (c) Mark de Payrebrune, 1990                             */
/*      All Rights Reserved                                                */
/*                                                                         */
/***************************************************************************/

#include <stdio.h>
#include <sys\timeb.h>      /* Used to get time */
#include "dt2801.h"         /* Definitions for DT2801 board */

extern gain_control(double, double, double *);      /* Fuzzy C routines */
extern factor_1(double, double, double *);
extern factor_2(double, double, double *);
extern factor_3(double, double, double *);
extern factor_4(double, double, double *);
extern factor_5(double, double, double *);

double check_limits( double gate );
unsigned char update_bias( int fet, double new_gate);

struct timeb tstruct;     /* Used to get time */

main()
{
  FILE *fptr;

  double dgain, old_gain, controller;
  double gate1, factor1;
  double gate2, factor2;
  double gate3, factor3;
  double pin1, factor4;
  double pin2, factor5;
  double factor;

  float gate1_control, gate2_control, gate3_control;
  float pin1_control, pin2_control;
  float raw_gain;
  float new_gain, multiplier;
  float vi=10.0, vo, r1=100.0, r2, temperature;
  float new_temp=0, old_temp=0;
```

```
   int foo, i, j, flag;
   unsigned char gate1_byte, gate2_byte, gate3_byte, pin1_byte, pin2_byte;

   reset_board();          /* Reset DT2801 data acquisition board */

   set_bias(1,148);        /* Set initial bias value for Gate1 */
   gate1 = 148.3;

   set_bias(2,132);        /* Set initial bias value for Gate2 */
   gate2 = 132.5;

   set_bias(3,155);        /* Set initial bias value for Gate3 */
   gate3 = 145.7;

   set_bias(4,200);        /* Set initial bias value for Pin1 */
   pin1 = 200.5;

   set_bias(5,200);        /* Set initial bias value for Pin2 */
   pin2 = 200.5;

   if( (fptr = fopen( "d:\\foo.foo", "w" )) != NULL )
   {
     for( foo=1; foo<=5000; foo+=1)    /* Main Loop */
     {
       vo = 0.0;
       for( i=1; i<=50; i+=1 )                 /* Read thermal sensor    */
         vo += read_a_to_d( 1, 1 );            /* taking the average of */
       vo = vo / 50.0;                         /* 50 readings            */
       r2 = vo * r1 / ( vi - vo );
       temperature = r2 / .618 - 142.5;
       old_temp = temperature;

       ftime( &tstruct );    /* Record the present time */

       new_gain = 0.0;
       for( j=1; j<=2000; j+=1 )                /* Read the gain       */
         new_gain += read_a_to_d(0, 8);           /* taking the average */
       new_gain = new_gain / 2000.0;           /* of 2000 samples     */

/* Call Fuzzy Control routine */
       dgain = new_gain - old_gain;
       gain_control(dgain, new_gain, &controller);

/* Determine Scale Factors */
       factor_1( controller, gate1, &factor1 );
       factor_2( controller, gate2, &factor2 );
       factor_3( controller, gate3, &factor3 );
       factor_4( controller, pin1,  &factor4 );
       factor_5( controller, pin2,  &factor5 );
```

```
/* Compute composite factor */
        factor = 1/(factor1+factor2+factor3+factor4+factor5);

/* Determine scaled control outputs */
        gate1_control = controller * factor1*factor;
        gate2_control = controller * factor2*factor;
        gate3_control = controller * factor3*factor;
        pin1_control  = controller * factor4*factor;
        pin2_control  = controller * factor5*factor;

        gate1 = gate1 + gate1_control;  /* Add to present value */
        gate1 = check_limits( gate1 );  /* Check value not exceed 8 bits */
        gate1_byte = gate1;             /* Convert to a byte value */
        set_bias( 1, gate1_byte );      /* Send value to D/A converter */

        gate2 = gate2 + gate2_control;      /* Determine new Gate 2 value */
        gate2 = check_limits( gate2 );
        gate2_byte = gate2;
        set_bias( 2, gate2_byte );

        gate3 = gate3 + gate3_control;      /* Determine new Gate 3 value */
        gate3 = check_limits( gate3 );
        gate3_byte = gate3;
        set_bias( 3, gate3_byte );

        pin1 = pin1 + pin1_control;         /* Determine new Pin 1 value */
        pin1 = check_limits( pin1 );
        pin1_byte = pin1;
        set_bias( 4, pin1_byte );

          pin2 = pin2 + pin2_control;         /* Determine new Pin 2 value */
          pin2 = check_limits( pin2 );
          pin2_byte = pin2;
          set_bias( 5, pin2_byte );

          set_bias(1,148);       /* Set bias values back to original values */
          set_bias(2,132);
          set_bias(3,155);
          set_bias(4, 200);
          set_bias(5, 200);

          raw_gain = 0.0;        /* Determine what the uncompensated gain is */
          for( i=1; i<=2000; i+=1 )
            raw_gain += read_a_to_d( 0, 8 );
          raw_gain = raw_gain / 2000.0;

          set_bias(1,gate1_byte);   /* Set bias back to compensated values */
          set_bias(2,gate2_byte);
          set_bias(3,gate3_byte);
          set_bias(4, pin1_byte);
```

```
            set_bias(5, pin2_byte);

           printf( "%d, %4.1f, %6.3f, %6.3f %7.2f %d %d %d %d %d\n",
                  foo, temperature,
                  raw_gain, new_gain, controller,
                  gate1_byte, gate2_byte, gate3_byte,
                  pin1_byte, pin2_byte
                );

          fprintf( fptr, "%ld.%u  0.0 %f %f %f %f %f %d %f %f %d %f %f %d %f %f
                            %d %f %f %d %f %f\n",
                  tstruct.time, tstruct.millitm,
                  temperature,
                  raw_gain, new_gain, dgain, controller,
                  gate1_byte, gate1, factor1,
                  gate2_byte, gate2, factor2,
                  gate3_byte, gate3, factor3,
                  pin1_byte, pin1, factor4,
                  pin2_byte, pin2, factor5
                );

          old_gain = new_gain;
       }
       fclose( fptr );
    }
    else
       printf( "\n\n****** File error *******\n\n");
}

double check_limits( double gate )
{
   if(gate  255.) gate = 255.;
   if(gate  0.) gate = 0.;
   return(gate);
}

unsigned char update_bias( int fet, double gate)
{
   unsigned char gate_byte;

   gate_byte = gate;
   set_bias(fet, gate_byte);

   return( gate_byte );
}
```