On Design and Implementation of Generic Fuzzy Logic Controllers

Pallab Maji

Dept. of Electronics & Communication Engineering
National Institute of Technology Rourkela
Rourkela, Odisha, India-769 008

On Design and Implementation of Generic Fuzzy Logic Controllers

Thesis submitted for partial fulfilment of the requirements for the degree of

Doctor of Philosophy

in

Electronics and Communication Engineering

by

Pallab Maji

(Roll No.: 511EC604)

Under the Supervision of

Prof. Sarat Kumar Patra

and

Prof. Kamalakanta Mahapatra

Dept. Electronics and Communication Engineering

National Institute of Technology, Rourkela
Orissa-769008, India

June, 2015

Dept. of Electronics & Communication Engineering
National Institute of Technology Rourkela
Rourkela, Odisha, India-769 008

June 24, 2015

Certificate

This is to certify that the work in the thesis entitled ***"On Design and Implementation of Generic Fuzzy Logic Controllers"*** by **Pallab Maji** is a record of an original research work carried out under our supervision and guidance in partial fulfillment of the requirements for the award of the degree of Doctor of Philosophy in Electronics and Communication Engineering. Neither this thesis nor any part of it has been submitted for any degree or academic award elsewhere.

Dr. Kamalakanta Mahapatra
(Co-Supervisor)
Professor, Dept. of ECE
NIT Rourkela, Odisha

Dr. Sarat Kumar Patra
(Supervisor)
Professor, Dept. of ECE
NIT Rourkela, Odisha

Dedicated
to
my family....

Declaration

I certify that

(a) The work contained in this thesis is original and has been done by me under the guidance of my supervisors.

(b) The work has not been submitted to any other Institute for any degree or diploma.

(c) I have followed the guidelines provided by the Institute in preparing the thesis.

(d) I have conformed to the norms and guidelines given in the Ethical Code of Conduct of the Institute.

(e) Whenever I have used materials (data, theoretical analysis, figures, and text) from other sources, I have given due credit to them by citing them in the text of the thesis and giving their details in the references. Further, I have taken permission from the copyright owners of the sources, whenever necessary.

Pallab Maji

Acknowledgements

This dissertation would not have been possible without the guidance and the help of several individuals who in one way or another contributed and extended their valuable assistance in course of this study.

The author would like to thank his supervisor, Dr. Sarat Kumar Patra, for his guidance and support on this research work. The author would also like to acknowledge his co-supervisor, Dr. Kamalakanta Mahapatra for his kind advice and inspiration.

The author would like to thank Board of Research for Fusion Science and Technology (BRFST) and Institute of Plasma Research, Gandhinagar for funding major part of this research. The author would like to extend his gratitude towards Dr. Govindarajan, Mr. J. J. Patel, Mrs. Rachana Rajpal and Mr. Hitesh Patel of Institute of Plasma Research, Gandhinagar, for their contributions to this project.

The author would also like to thank Prof. B. Subudhi, Prof. D. P. Acharya, Prof. P. M. Khillar and Prof. S. Meher for their innovative ideas and review during the entire duration of the project.

The author would also like to acknowledge and appreciate the help from the faculty members especially Prof. S. Hiremath, Prof. A. K. Swain, Prof. M. Okhade and Prof. S. Deshmukh of the department for their continuous encouragement to delve into analytical aspects

of this work. The author would be grateful to the support extended by the staff members of the department, specially Mr. P. Oram and Mr. Ishwar during the entire journey.

The author would also like to thank his friends, especially Deepak, Umakant, Bodhisattwa, Bhaskar, Chithra, Manas, Varun, Goutam, Sankat, Bibhu, Deepak, Asis and Pramit for their accompaniment. Their enduring effort in proof reading of the thesis is unforgettable. The author would also like to extend his gratitude to his seniors Prashant, Sudheendra, Jagannath, Yogesh, Manab and Dipak. The author would also like to thank his lab partner Satyendra for his extensive support and assistance during the entire period of his doctoral studies.

And finally the author would like to thank his family and friends, whose faith and patience had always been the inspiring factor.

Pallab Maji
pallab.vsp@gmail.com

Abstract

Soft computing techniques, unlike traditional deterministic logic based computing techniques, sometimes also called as hard computing, are tolerant of imprecision, uncertainty, and approximation. The primary inspiration for soft computing is the human mind and its ability to address day-to-day problems. The primary constituents of soft computing techniques are Artificial Neural Network, Fuzzy Logic Systems, and Evolutionary Computing.

This thesis presents design and implementation of a generic hardware architecture based Type-I Mamdani fuzzy logic controller (FLC) implemented on a programmable device, which can be remotely configured in real-time over Ethernet. This reconfigurability is added as a feature to existing FLCs in literature. It enables users to change parameters (those drive the FLC systems) in real-time and eliminate repeated hardware programming whenever there is a need. Realization of these systems in real-time is difficult as the computational complexity increases exponentially with an increase in the number of inputs. Hence challenge lies in reducing the Rulebase significantly such that the inference time and the throughput time is perceivable for real-time applications.

To achieve these objectives, a modified thresholded fired rules hypercube (MT-FRHC) algorithm for Rulebase reduction is proposed and implemented. MT-FRHC reduces the useful rules without compromising system accuracy and improves the cycle time in terms of fuzzy logic operations per second (FzLOPS). It is imperative to understand that there are over sixty reconfigurable parameters, and it becomes an arduous task for a user to manage them. Therefore,

a genetic algorithm based parameter extraction technique is proposed. This will help to develop a course tuning and provide default parameters that can be later fine tuned by the users remotely through the Web-based User Interface. A hardware software co-design architecture for FLC is developed on TI C6748 DSP hardware with Sys/BIOS RTOS and seamlessly integrated with a web-based user interface (WebUI) for reconfigurability.

Fuzzy systems employ defuzzifier to convert the fuzzy output into the real world crisp output. Centroid of Area (CoA) method is most widely used defuzzification method for control applications. However, the prevalent method of CoA computation is based on the principle of Riemann sum which is computationally complex. A vertices based CoA (VBCoA) defuzzification method is introduced. It has been observed that the proposed VBCoA method for COA computation is faster than the Riemann sum based CoA computation.

A code optimization technique, exclusive to TI DSPs, is implemented to achieve memory and machine cycle optimization. The WebUI is developed in accordance to a client–server model using ASP.NET. It acquires fuzzy parameters from users, and a server application is dedicated to handling data communication between the hardware and the server. Testing and analysis of this hardware G-FLCS has been carried out by using hardware-in-loop test to control various system models in Simulink environment which includes water level control in a two tank system, intelligent cruise control system, speed control of an armature controlled DC motor and anti-windup control. The performance of the proposed G-FLCS is compared to Fuzzy Inference System of Matlab Fuzzy Logic Toolbox and PID controller in terms of settling time, transient time and steady state error. This proposed MT-FRHC based G-FLCS with VBCoA defuzzification implemented on C6748 DSP was finally deployed to control the radial position of plasma in Aditya Tokamak fusion reactor. The proposed G-FLCS is observed to deliver a smooth and fast system

response.

Contents

List of Figures

List of Tables

List of Symbols

Δ	Fuzzy operator
Δ^{-1}	Inverse fuzzy operator
Λ_c	Digital to Analog Converter
$'$	Matrix or vector transpose
$\bigcap$	T-norm operator. Basic operation includes Minimum, Product, Lukasiewicz, etc. Operated on a vector
$\bigcup$	T-conorm operator. Basic operation includes Maximum, Product, Lukasiewicz, etc
$\mathcal{O}$	*Big O* notation for complexity
$\overrightarrow{X(i)}$	implies a vector formed from i^{th} row of matrix X
$\rightarrow$	implies in between
μ	Membership function
$\mu(x)$	Value of Membership function for real value x
$\mathbf{B}$	Magnetic flux density
B_ϕ, B_θ, B_ρ	Toroidal, poloidal and radial components of the magnetic field
E	Electric field intensity
J	Plasma current density
R	Major radial coordinate
a	Minor radial coordinate
Γ	Shafranov parameter

List of Acronyms

AC	Alternating Current
ACDC	Armature Control Direct Current
ADC	Analog to Digital Converter
ASIC	Application Specific Integrated Circuit
ASIP	Application Specific Integrated Processor
ASP	Active Server Pages
BOA	Bisector of Area
CAN	Controller Area Network
CMOS	Complementary Metal Oxide Semiconductor
COA	Centroid of Area
COG	Center of Gravity
DAC	Digital to Analog Converter
DC	Direct Current
DSP	Digital Signal Processor
FIS	Fuzzy Inference System
FLC	Fuzzy Logic Controller
FCP	Fuzzy Control Parameters
FLIPS	Fuzzy Logic Inferences Per Second
FPGA	Field Programmable Gate Array
FzLOPS	Fuzzy Logic Operations Per Second
IIS	Internet Information Server
IC	Integrated Circuits
LAN	Local Area Network
LOM	Largest of Maximum
MF	Membership Function
MFG	Membership Function Generator
MFRPS	Mega Fuzzy Rules Per Second
MOM	Middle of Maximum
OMP	Overlapping Membership Function

PID	Proportional Integral Derivative
SOM	Smallest of Maximum
SPI	Serial Peripheral Interface
SIMD	Single Instruction Multiple Data
TCP	Transmission Control Protocol
TI	Texas Instruments
UART	Universal Asynchronous Receiver and Transmitter
VLIW	Very Long Instruction Word

List of Code Snippets

Chapter 1

Introduction

Overview

This chapter presents the fundamental concepts of a fuzzy system. It reviews few of the existing fuzzy logic controller designs reported in the literature and analyzes their implementation techniques. This chapter also address the issues of reconfigurability and generality of the existing fuzzy system designs, thereby lay the foundation of this research work and its contribution.

"We human beings, live in a very imprecise world. A world where our perception of reality are far more important than actual reality."

Daniel Keys Moran

It is well known that a human mind efficiently utilizes the modes of imprecision and uncertainty to solve everyday problems. It is this tolerance of imprecision, uncertainty, and approximation that helps human being make informative decisions and enforce reasoning with ease and in a short time. It is true that precision and certainty has ensured that mankind is able to fire precision laser over a long distance, build powerful processors with trillions of transistors, developed terra-pixel imaging devices, focus microscopic beams of electron to capture minute details in nanometer scale, and accomplish many more unimaginable achievement in science and technology. However, requiring precision in engineering problems incurs a high cost and long lead time in development. Prof. Lotfi Askar Zadeh described the power of uncertainty and approximate reasoning over hard computing by illustrating how a human mind work while *parking a vehicle*[206]. T. Ross took the instance of *traveling salesman* problem to exemplify similar point [153]. It is, therefore, important for any scientist or engineer to contemplate the requirement for approximate reasoning and imprecision while considering fuzzy logic to solve a problem. The prime desideratum is "how much imprecision can the system tolerate".

1.1 Introduction to Fuzzy Logic Systems

Lotfi Askar Zadeh in 1965 [205] proposed fuzzy sets and described it as "a class of objects with a continuum of grades of membership. Such a set is characterized by a membership function that assigns to each object a grade of membership ranging from zero to one." All fuzzy logic systems operate on this principle to mathematically represent linguistic variables and heuristic knowledge. Fuzzy logic systems provide an alternative to the predominant conventional binary and deterministic logic based crisp data processing systems. Fuzzy set

theory provides the mathematical tool to carry out approximate reasoning and to handle imprecision or vagueness of information. The concept of degrees of membership is employed to provide a mathematical definition of fuzzy sets. This enables various circumstances encountered in human reasoning to objectify into scientific form.

1.2 Fuzzy Sets

Conventional bivalent set theory, often known as conventional set theory, can be limiting in describing a 'humanistic' problem mathematically. For example, Fig. 1.1 below illustrates bivalent sets to model room temperature. It is obvious that the limiting feature of conventional sets is that they are mutually exclusive.

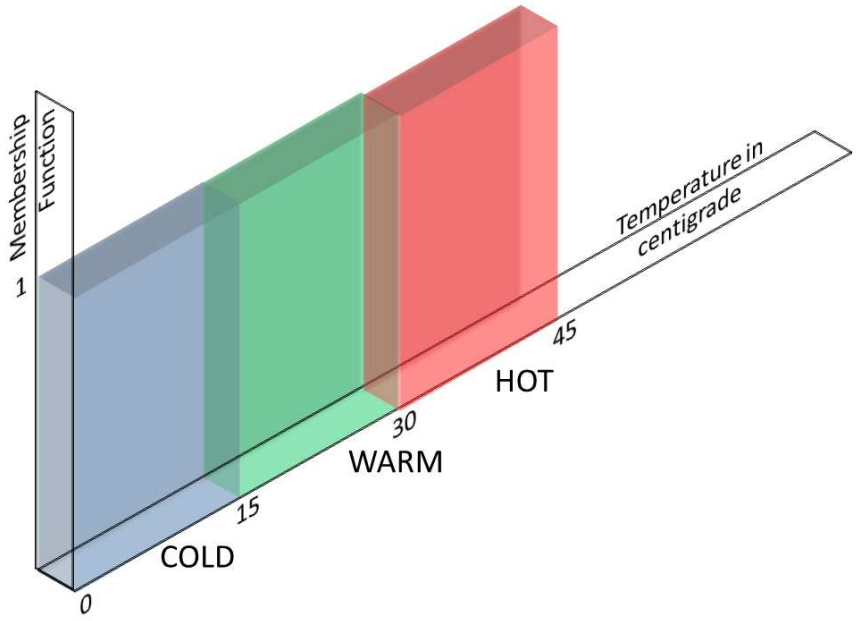


Figure 1.1: Bivalent sets to model room temperature.

It becomes impossible to generate association of a variable to more than one set. Based on how human achieves perception, it is inaccurate to model transition from quantity '**cool**' to '**warm**' when one degree centigrade of heat is added to the system. The actual modeling in real life however, occurs with a

smooth transition or drift from '**cool**' to '**warm**'. This transition can be captured if the association itself can be modeled using some functions as depicted in Fig. 1.2. Here, the association is modeled as a triangular function. In fuzzy logic theory, the function which defines the association is called as membership function. Thereby, in fuzzy set theory, apart from the value of the variable, the degree of association of the variable to the set is also captured.

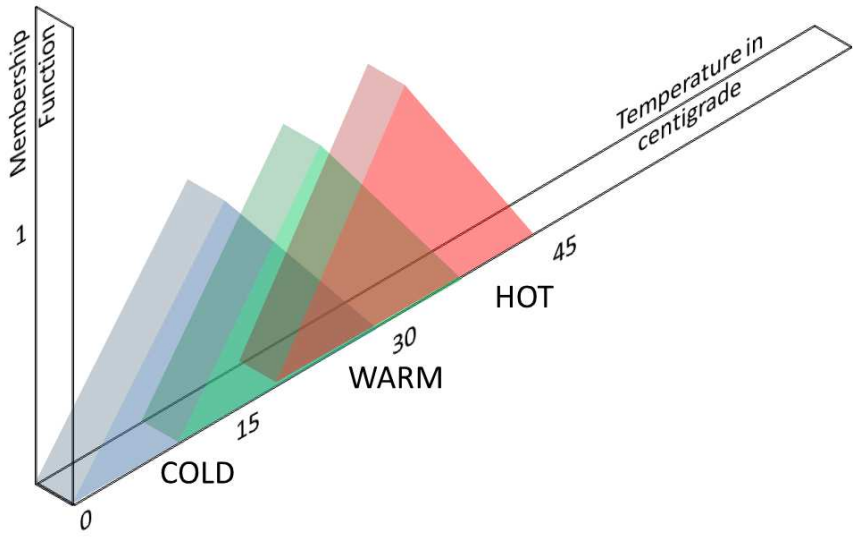


Figure 1.2: Fuzzy sets to model room temperature

Mathematically, a fuzzy set is a pair (ξ, μ), where ξ is a set, also known as universe of discourse, and $\mu : \xi \rightarrow [0,1]$. This implies that μ presents the degree of association of an element to the set ξ which lies in between $[0,1]$. Therefore, $\forall x \in \xi, \mu(x) \rightarrow [0,1]$, where $\mu(x)$ is called the grade of membership for x, x being a fuzzy number in fuzzy set ξ.

A set $\{x \in \xi | \mu(x) > 0\}$ is called support and set $\{x \in \xi | \mu(x) = 1\}$ is called core or kernel [32, 153]. The function μ is called membership function (MF) of fuzzy set (ξ, μ)[32, 153].

A fuzzy system operates on various fuzzy sets to provide a suitable output. It is often required that these fuzzy sets are combined meaningfully. Various combination operators exist in ordinary set theory, and it is imperative that there exist a commonality of operators between regular and fuzzy sets. These

operators are termed as *aggregators* [131].

1.3 Fuzzy Operators

Ordinary sets are combined or negated by using operators like intersection (AND), union (OR), and complement (NOT) [131]. Similarly in fuzzy sets, minimum, maximum and negation operators approximate AND, OR and NOT operations. However, on many occasion, **AND** operations can be achieved by functions other than the minimum operation. These functions are collectively referred to as triangular norms or simply as t-norms. Similarly, **OR** operations are referred to as triangular co-norms or t-conorms or s-norms[32, 153].

There are four basic t-norms and almost all t-norms used in a fuzzy system are derived from these basic operations [32]. Consider μ_x and μ_y as membership grade of two fuzzy numbers x and y, in a fuzzy set. Then the following equations represents the various t-norm operations in a fuzzy system.

1.Zadeh Intersection: $T(\mu_x, \mu_y) = \min(\mu_x, \mu_y)$
2.Product Intersection: $T(\mu_x, \mu_y) = (\mu_x \cdot \mu_y)$
3.Lukasiewicz Intersection: $T(\mu_x, \mu_y) = \max(0, \{\mu_x + \mu_y - 1\})$
4.Basic Intersection: $T(\mu_x, \mu_y) = \begin{cases} \mu_x, & if\ \mu_y = 1 \\ \mu_y, & if\ \mu_x = 1 \\ 0, & if\ \mu_x, \mu_y < 1 \end{cases}$

Similarly, the t-conorm or s-norm operators can be represented as following equations in a fuzzy system [32].

1.Zadeh Union: $S(\mu_x, \mu_y) = \max(\mu_x, \mu_y)$
2.Product Union: $S(\mu_x, \mu_y) = \mu_x + \mu_y - (\mu_x \cdot \mu_y)$
3.Lukasiewicz Union: $S(\mu_x, \mu_y) = \min(0, \{\mu_x + \mu_y - 1\})$

There three major fuzzy complement operators which have been widely used

in the literature [32]. These operators are;

$$\begin{aligned}
&\text{1.Standard Complement:} \quad N(\mu_x) = (1-\mu_x) \\
&\text{2.Sugeno's Complement:} \quad N_s(\mu_x) = \tfrac{1-\mu_x}{1+s\mu_x} \\
&\text{3.Yager's Complement:} \quad N(\mu_x) = (\mu_x \cdot \mu_y)
\end{aligned}$$

where s is Sugeno's constant and w is Yager's constant.

1.4 Fuzzy Rules

Words rather than numbers define linguistic variables. Fuzzy rules use these linguistic variables instead of numbers to quantify variables. These linguistic variables are represented as fuzzy sets with a certain function. This provides the mathematical background of the fuzzy systems. All fuzzy rules are divided into an antecedent part (starting with "$IF\ldots$") and a consequent part (ending with "$THEN\ldots$"). The antecedent parts describe the causes while the consequent parts describe effects relevant to desired control action. A typical form of a fuzzy rule with Rulebase index $R_b(k)$ is as shown below.

$$R_b(k): \quad \text{If } i_1 \text{ is } a_1 \text{ and } i_2 \text{ is } a_2 \text{ and } \cdots \text{and } i_N \text{ is } a_N, \text{ then output is } c_j$$

where, $i_1, i_2 \cdots i_N$ represents inputs, $a_1, a_2 \cdots a_N$ are called antecedent and c_j is the resultant consequent. All antecedents and consequents are represented by valid linguistic variables in a fuzzy system.

1.5 Fuzzy Logic Control System

Traditionally, the industrial process control was dominated by binary logic based reasoning. Heuristic knowledge hardly plays any role in these systems [24, 119]. This forced the systems to be represented by a collection of complex mathematical equations. The drive for precise and accurate control was expensive and often proved to respond sluggishly. However, in recent past, the power of human reasoning started to get acknowledged with the advent of FLCS in

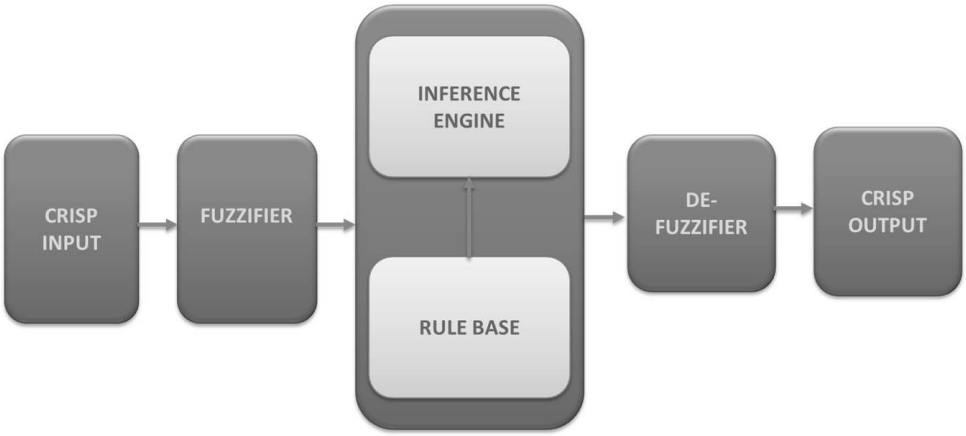


Figure 1.3: Black diagram of a FLCS

nonlinear process control [115, 116, 176, 206]. The most appreciable feature of FLCS is its ability to manage complex control problems through human knowledge and numerical models provided by fuzzy set theory, instead of using complex differential equations to derive mathematical models of a process plant.

The computing framework of a fuzzy logic control system (FLCS) rely on three conceptual components:

i. Rulebase: This contains a database of rules that represent human knowledge and heuristics. They define I/O relationship of the system in terms of linguistic variables.

ii. Database: This defines the MFs that are used to define the linguistic variables in the fuzzy rules.

iii. Inference Mechanism: This performs the reasoning procedure based on the Rulebase and Database.

These three components together constitute the Fuzzy Control Parameters (FCP). At this juncture, an operator's experience and knowledge is appropriately formulated and configured into FCP. This FCP represents the "intelligence" in any fuzzy control algorithm. Hence, it can be asserted that the information of the FLCS is dependent on the knowledge of the designer or the operator.

FLCS has four important modules which interact with the input to generate

a meaningful output[115, 119, 195]. The block diagram of the modules in a FLC is represented in Figure 1.3. These modules are;

i. Fuzzifier: The input variables in a fuzzy control system that are real world variables and also known as *crisp input data*, are in general mapped by sets of membership functions, known as "fuzzy sets". The fuzzifier generates a set of output in between 0 to 1, which are dimensionless. This process of converting a crisp input value to a set of fuzzy values is called "fuzzification".

ii. Rulebase: It stores the heuristic rules that govern a typical fuzzy system. These rules describe the output dependence of the inputs, and they are mentioned in terms of the MFs representing the inputs and outputs of the process plant.

iii. Inference Engine It accepts fuzzified inputs and applies reasoning described by the Rulebase to work out fuzzy outputs.

iv. Defuzzifier: It translates fuzzy outputs from the Inference Engine into substantive crisp output applicable to a real system.

1.6 Learning of FCP from Data

Learning of FCP from data essentially means designing or extracting parameters that governs the fuzzy controllers from a given sampled data or model of a system. According to Wang et. al. [196] there are five steps in this process.

Step 1 The input and output spaces are recognized and divided into fuzzy regions.

Step 2 The dataset is observed and fuzzy rules are generated from it.

Step 3 Each generated rule is assigned a degree to resolve any conflict that arises amongst the generated rules.

Step 4 Based on visualization paradigm and human expert knowledge, some rules can be directly inferred about the system. These rules are combined with the generated rules.

Step 5 Once the rulebase is formed, the ouput in fuzzy domain can be obtained. This output is mapped to real world output using a suitable defuzzification procedure.

In the context of learning FCP, Step 1 to 3 can be manual or it can be formulated as a learning problem itself. G. D. Finn proposed 'Fuzzex' for generating fuzzy rules in general from data [54]. In this popular technique, they proposed a five step method where cell occupancy is used. But this method does not suit nonlinear system approximation. In 2009, SÃąnchez et. al., presented a genetic algorithm based learning method which is robust to system noise [159]. Tuning fuzzy rules using genetic algorithm has been widely investigated and is often the most popular choice [161, 181, 189]. However, learning the entire FCP is a difficult task at hand. To counter this issue, a new strategy for sophisticated learning of FCP with Genetic Algorithm optimization technique is proposed in Section 3.5.

1.7 Motivation of This work

PID controllers are widely used in industries even though they are inherently linear and provides a sluggish response. They are generally not suited to control nonlinear process plants [13, 14, 25]. Their modeling requires a thorough knowledge of system dynamics and the tuning process is quite difficult too. Fuzzy Logic Controllers (FLC) provide an imprecision based control approach which is fast and reliable. However, they are driven by the human knowledge which is prone to be erroneous. Some designs provide techniques to derive the FCP from a dataset or model. However, this eradicates human knowledge completely. Therefore, it is highly desirable to have a FLCS with two level tuning.

- Coarse Tuning: Achieved by using suitable algorithm and FCP are extracted,

- Fine Tuning: Using an interface, system is fine tuned by an operator.

1.8 Objective of this work

Fuzzy logic systems have found applications in a variety of fields namely industrial process control, power systems, robotics, water resources, structural design, metallurgy and material science, business and finance, and many more. It is challenging for scientists and engineers to implement an efficient FLCS design that can be integrated into their system of interest, especially if they are not well equipped with knowledge of programming and system development. Therefore, the primary objective of this research is to design and realize a generic fuzzy logic controller system (G-FLCS) on a programmable hardware that can be used for a variety of applications without reprogramming. Some of the characteristic features of this proposed control device will include

- Plug and play framework: Essentially process plant operators are not good programmers. It is, therefore, mandatory for a controller device to possess a plug and play framework for ease of installation and operation. Users can input the fuzzy parameters through an interactive user interface.

- Runtime tunability: Most control device lacks this feature. The major advantage of run-time tunability is that it provides an opportunity to include the concept of two level tuning, one of the prime motivating factor for this research. This will offer the leverage to fine-tune the FLCS while it is in operation so that the parameters can be reset to default under critical conditions.

- Standalone operation: The prime requirement of any real-time system or controller is to be able to operate in a standalone mode.

- Flexibility in operation order: A G-FLCS system primarily implies that it can be implemented on any system with appropriate parameter tuning; be it a single-input-single-output system or multiple-input-multiple-

output system. Thereby, it is implicit that the G-FLCS should have the provision to accommodate a different number of inputs and outputs.

It becomes a challenging task when run-time tunability, flexibility in system order and plug and play framework is combined with the standalone mode of operation. Therefore, the architecture of traditional FLCS is required to be altered in a way such that, the data integrity and operational methodology remains consistent even after incorporating the above-mentioned features.

1.9 Literature Survey on Design and Implementations for FLCS on various Hardware Platforms

In recent times fuzzy logic is addressing complex problems of control, forecasting and prediction with imprecision in fields of robotics [1, 45, 45, 63, 102, 103], chemical [10, 55, 86, 98, 140, 166] and manufacturing processes [59, 150, 207], automobiles [16, 21, 31, 99, 194], business and finance[49, 88, 187, 202], power electronics [3, 28, 104], and many others[35, 49, 79, 89, 90, 178] in a better way than conventional control techniques. Wide spread application of the FLCS and its extensively high effectiveness to a larger extend is driven by formalizing necessary human knowledge and sometimes behavior in the controller as an imprecise and approximate representation. These factors impel engineers to design and implement fuzzy based controllers for wide array of applications.

The hardware designs of FLCS can be classified into three broad categories based on their circuit architecture.

- Analog,
- Digital, and
- Mixed Signal

Each of these can be further classified based on the aspect of system design and platform for implementation.

Table 1.1: Taxonomy for Hardware Implementation of FLCS

	Classification	Platform	Devices
Types of Implementation	Analog	Dedicated IC	Analog ASIC
		Programmable IC	FPAA
		Commercial Processor	–
	Digital	Dedicated IC	Digital ASIC
		Programmable IC	FPGA, CPLD
		Commercial Processor	
	Mixed	Dedicated IC	Mixed Signal ASIC
		Programmable IC	–
		Commercial Processor	–

Dedicated Integrated Circuits: These FLCS are designed primarily to target a single control application. These devices are mostly built over application specific integrated circuits (ASIC) and implement full custom analog, digital or mixed-signal designs [43, 60, 72, 100, 109, 114].

Programmable Integrated Circuits: Programmable Integrated Circuit based FLCS devices are commercially developed in integrated circuits (ICs) that can be reconfigured by the user. These tools provide attractive options to designers and engineers. They have the ability to be reprogrammed, and their high integration density is a fascinating feature. Commercially available field programmable gate arrays (FPGAs) and field programmable analog arrays (FPAAs[1]) have been widely used in designing of these systems [40, 65, 77, 125, 139, 146, 179, 186].

Commercial Processors: A software application defining the system is developed and deployed on these devices. Microprocessors (μP), Microcontrollers (μC) and Digital Signal Processors (DSPs) based systems are defined under this category [81, 91, 138, 160, 164, 203, 204].

Different forms of FLCS implementation is presented in Table 1.1. The forms of FLCS include;

[1]This is an integrated device comprising of configurable analog blocks (CAB) and in between interconnects. Lattice and Anadigm are prime manufacturers of this device.

1.9.1 Analog Implementation of FLCS Design

A Large number of FLCS in literature have been developed on analog devices. The major reasons that drive a FLCS design engineer to choose these platforms are high parallelism, high speed, low area and low power consumption [23, 144, 171, 172]. Different forms of Analog IC based implementation includes;

1.9.1.1 Dedicated IC based FLCS

There are three modes in which these devices are implemented namely,

Current Mode implementation uses fewer transistor and hence they consume low power. However, these devices can only connect to one output since they work in current mirror mode [33, 209]. Some of the work those were developed using these techniques are

- Tokmakci et. al.[185] designed current-mode CMOS FLC. The membership function circuit (MFC) implemented trapezoidal, triangle, Z-shape and S-shape MFs. which were tunable by two voltages through switches. The system developed was a two-inputs-one-output with 9 tunable rules only. The operation speed of this FLCS was reported to be 6.25 Mega Fuzzy Logic Inferences per second (MFLIPS)[185].
- Gheysari et. al.[58] proposed a Flexible Structured Fuzzy Logic Controller Chip (FS-FLC) on $0.35\mu m$ process. They implemented Ordered Weighted Averaging operator to aggregate multiple-input single-output (MISO) system. However, the FLCS design used singleton rules at output.

Voltage Mode implementation of FLCS can serve more than one outputs unlike current-mode. Some of the noted designs include

- Mokarram et. al.[124]: developed a two-input, single-output Takagi-Sugeno-Kang (TSK) FLC on $0.35\mu m$ standard CMOS process. The system supports triangular and trapezoidal membership functions. Membership function generator (MFG) provides generation and tun-

ing of the MFs but the design do not permit changing or tuning of rules.

- Aminifar et. al.[8]: The designed a FLCS on 0.35μm CMOS process with inference speed of 14.83 MFLIPS. They designed a 2-input one-output system with mere 9 rules with support to only singleton MFs at output. Moreover, the design do not allow any tunability.

1.9.1.2 Programmable IC based FLCS

There has been very limited research reported on FLCS implementationAnalog Programmable ICs. Some of the most significant works include;

- Amaral et. al[7] designs and developed in PAMA-NG, a FPAA platform, with I/O board connected to the PCI bus of the PC. The system used Genetic Algorithm (GA) to reconfigure the FPAAs.

- Ionita et al.[73] also used evolutionary algorithm to tune MFs. They developed a Mamdani type FLCS on FPAAs.

It has been observed that designs implemented on Analog Programmable ICs are extremely sensitive to problems of fanout and presence of switches on the signal path [145]. It is also known that analog circuits are more vulnerable to be affected by noise in comparison to Digital circuits.

1.9.2 Digital Implementation of FLCS Design

Fuzzy systems and control are making fast advancement in past decade and two. Owing to its pragmatic achievements in consumer electronics and industrial process control, implementation of FLCS has been rigorously researched and developed. However, increase in process complexity of the industrial plants is accelerating demand for controllers with high computational speed, low complexity, easy deployment, comfortable handling and less development time in terms of design. In order to conform to the demand-supply chain of the industry, FLCS have to be designed accordingly. A noteworthy solution to fulfill this growing market demand is to move to a digital platform. It is well known that

digital systems have high resistance to noise, temperature and voltage variations. There is a vast array of digital platforms available to an engineer for design implementation that reduce turnaround time. Although, systems designed in digital hardware platforms are not as fast as analog designs; still a good system cycle time can be achieved which provide sufficient throughput speed for the majority of the control problems. Table 1.1 shows the various digital implementation platforms for realization of different FLCS designs.

1.9.2.1 Dedicated IC based FLCS

These implementations concentrate on structuring the fuzzy rules in a FLCS and its functionality is defined by, whether these rules are evaluate sequentially or in parallel. Some of the notable designs include

- Eichfeld et. al.[50] reported a four-input single-output FLCS with 4096 rules with eight MFs for each input. However, the system operated only on two overlapping MFs and used singleton type MFs for output.
- Jacomet et. al.[78] described an architecture of a VLSI fuzzy processor fabricated in the $0.7\mu m$ digital CMOS process. High performance was achieved due to its parallel architecture. The FLCS evaluates 64 rules but this design too uses two overlapping MFs scheme. Moreover, if four inputs are used, the rules are limited to 64 and thereby, only 3 MFs per input were allowed.
- Huang et. al.[70] developed a FLCS in $0.35\mu m$ CMOS process. However, this design used trapezoidal MFs only with a fixed rulebase.
- Hamzeh et. al.[67] designed one of the most flexible structure for FLCS in the literature. This device however do not discuss the speed of performance.
- Javadi et. al.[66]'s design provides a new fuzzification method for hardware on $0.13\mu m$ but it is only applicable to piece-wise linear MFs.

1.9.2.2 Programmable IC based FLCS

1.9.2.2.1 CPLD based FLCS Design It has been shown in Table 1.1 that there are two preferred devices which can be categorized in this segment, namely FPGA and complex programmable logic device (CPLD). There are very few CPLD based designs reported in literature. Some of the important designs are,

- Hongguo Sun et. al. [177] presented a Fuzzy PID design on CPLD for PWM trigger pulse generation to a full bridge inverter and a chopper circuit. It implemented a two-input one-output FLCS with fixed rulebase and rigid MFs.

- Jingyan Xue et. al. [199] presented a novel methodology to design a fuzzy reasoning based expert system on CPLD for fault diagnosis. Similar to previous design, this too implemented a FLCS with fixed rulebase and rigid MFs.

There are not many CPLD based FLCS designs that are reported in literature. The decisive reasons are that CPLDs are cost and power intensive platform. Moreover, there are platforms which are easy to configure than CPLDs.

1.9.2.2.2 FPGA based FLCS Design

- Adhavan et. al. [2] countered the problem of non-uniform variance of the torque developed in a vector controlled permanent magnet synchronous motor by introducing a FIS implemented on an FPGA. Author have reported that the heuristic knowledge based FLCS (Fuzzy Logic Control System) has reduced the torque ripple to 1.81%.

- Ben, Zekeri et al. [26] reported PD approximated FLCS developed on cyclone II FPGA to control a dual axis sum tracking systems. The simple rules developed with human knowledge have been found to be successful in reducing chip count, cost and development time of the controller significantly.

- Santo and Ferreira [48] implemented a multi-state FLCS on virtex II FPGA and NI compact $R10-9002$ to control servo- pneumatic actuation

systems. They showed significant performance gain in term of steady state error, overshoot and settling time.

- Messai et. al. [118] reported a FLC to seek maximum power point deliverable by a photovoltaic (pr) module using measures of PV voltage and current.

- Schriber et. al. [163]presented an interval type- II FLCS implemented on a xiling spartan 6 FPGA utilizing DSP48AI slices for different linear and non-linear modules.

- Tamukoh et. al. [182] reported a new technique of bit shift based fuzzy inference method for an efficient digital hardware implemented. They implemented the proposed design on a virtex II FPGA for a self-organization relationship network.

These designs depicts that the realization of FLCS on FPGA development platform is fast and efficient. However most of these designs are application specific. It is important to realize that even the speed achieved by these designs cannot to achieved by any generic FLCS design as these appears a large amount of branching in the G-FLCS algorithm.

1.9.2.2.3 Digital ASIC Design based FLCS Design

- Martinez-rodriguez [110] has presented a FLCS on ASIC platform where the number of input to and output MFs can be varied on run time. However, in this design, the number of configurable parameters are very few with a rigid rulebase.

- Evmorfopoulos et. al. [53] reported a G-FLCS structure on digital ASIC which can had a maximum of five MFs at the input. All input MFs were Gaussian type and output MFS are singleton type.

- D'Amore [44] reported a bit scalable fuzzy processors with three input and MF generators. However like the previous design, this system do not equate with output MFs other than singleton type too.

Digital ASIC design is generally quite time intensive in development and the process incurs extreme cost. Without large production, these designs are not cost-effective solution.

1.9.2.3 Commercial Processors based FLCS

1.9.2.3.1 FLCS Implementation with TMS320F Series DSP

- EL Khatib et. al. [51] presented a FLCS based SEPIC converter on TMS32F28335 DSP device to successfully track the reference signal using MPPT to transfer power around 4.8% than conventional PI based system.
- Eskandarin et.al [52] proposed a fuzzy instantaneous power theory to improve conventional p-q theory dynamic performance and implemented it on a TMS320F28335 DSP device.
- Okumus et. al. [135] has reported on FLCS design implementation of TMS320F2812 DSP device to control a brushless DC motor and compared the result with HB current controller. The heuristic knowledge based FLCS is found to perform extensively well.

1.9.2.3.2 FLCS Implementation with TMS320C Series DSP

- Uddin et. al. [190] showed a cost effective FLCS based controller designs on TMS320C31 DSP to control an interior permanent magnet synchronous motor for high performance industrial applications.
- S. Gai et al. [57] used a TMS320C6713 DSP device to implement a fuzzy based Haar wavelet feature extraction technique to successfully classify and detect a counterfeit banknote.

1.9.2.3.3 FLCS Implementation with dSpace DSP

- Butt et. al. [39] implemented FLCS based MTPA speed control of a IPMSM drive on a DS1102 DSP.

- Like many FLCS designs reported on tracking of maximum power point using MPPT algorithm Noman et. al. [134] also proposed a FLCS design for similar application using a DS1104 DSP.
- Rafa et. al. [148] implemented a new FLCS design on DS1104 DSP to solve coupling problem in vector control of induction motor.
- Rubaai et. al. [155] used DS1104 to implement a FLCS control structure with adaptive la based Lyapunov synthesis for trajectory tracking control of a brushless servo drive systems.

There are many more DSP based FLCS designs that have been successfully implemented in various control applications. It can be readily inferred that the development of DSP based FLCS is easy compared to FPGA [92, 141, 162]. However, since the parallel architecture can be implemented on FPGA, DSP likes sequential processors are preferred less while developing application specific FLCS [6]. Moreover, the number of branching infractions are extensively reduced in application specific FLCS design as the fuzzy parameters are fixed. This makes FPGA platform more preferable. However for GFLCS design, the situation is quite inverse. In next section, it will be explained why DSP is preferred over FPGA for this particular research work.

1.10 Inference from the Literature Survey

The previous section have laid down the various FLCS designs and some essential features in their implementation. It would be beneficial to summarize the prime aspects of these designs. In this section, a brief inference of the literature survey is portrayed.

1.10.1 FLCS Implementation in FPGA and DSP Platforms

FPGAs and DSPs are most widely used programmable devices for digital implementations of various algorithms [29, 38, 83]. Traditionally, manufacturers like Texas Instruments(TI) and Analog Devices developed processors for specifically for signal processing applications and called them DSP. These are most

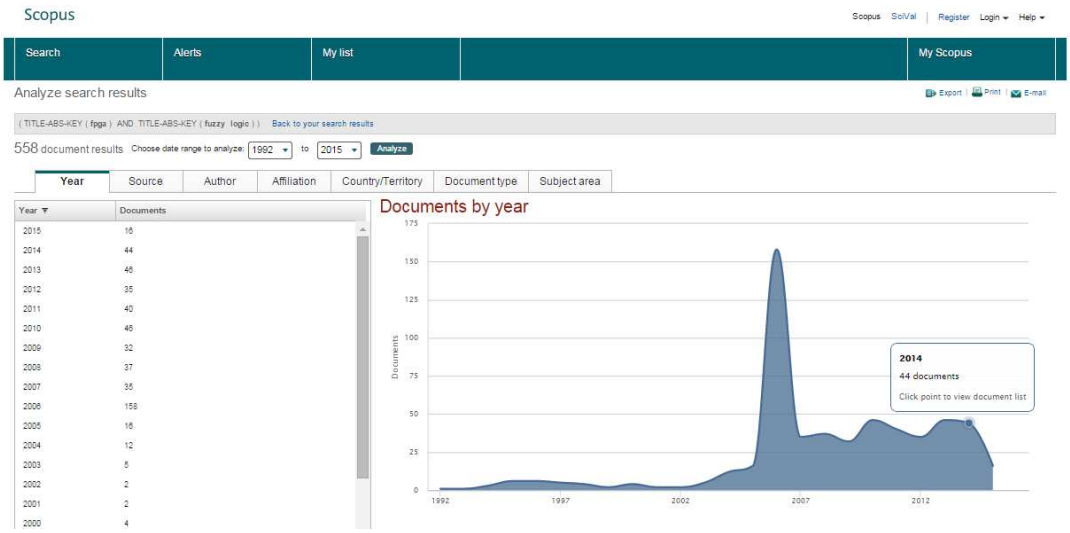


Figure 1.4: Literature on FPGA implementation of FLCS as reported by scopus as on May 2015

still most preferred platform for signal processing applications [92]. However, the unsatisfiable need for gadgets that require higher performance and address complex algorithms is driving growth that is hard to keep up with. FPGAs have emerged as a solution to the much needed reconfigurable platform and are capable of addressing the design challenges of large and complex algorithms. However, algorithms that do not confront very high algorithmic complexities are still realized in DSPs [92, 97]. TI reported that car manufacturers like BMW, Audi, and Toyota were using DSPs for driver-less cars. DSPs have also been reported to manage the real-time processing of visual data for advanced driver assistance systems (ADAS) applications in the main automotive products [162].

Many researchers have implemented FLCS on FPGA and DSP platforms. Figure 1.4 and Figure 1.5 clearly shows that these are two of the most preferred platform for FLCS implementation.

1.10.2 Comparison between various Digital Platform for FLCS Implementation

It can be seen from Figure 1.4 that FPGA have been most preferred FLCS design platform in literature [2, 26, 37, 48, 76, 106, 118, 139, 163, 182]. Some

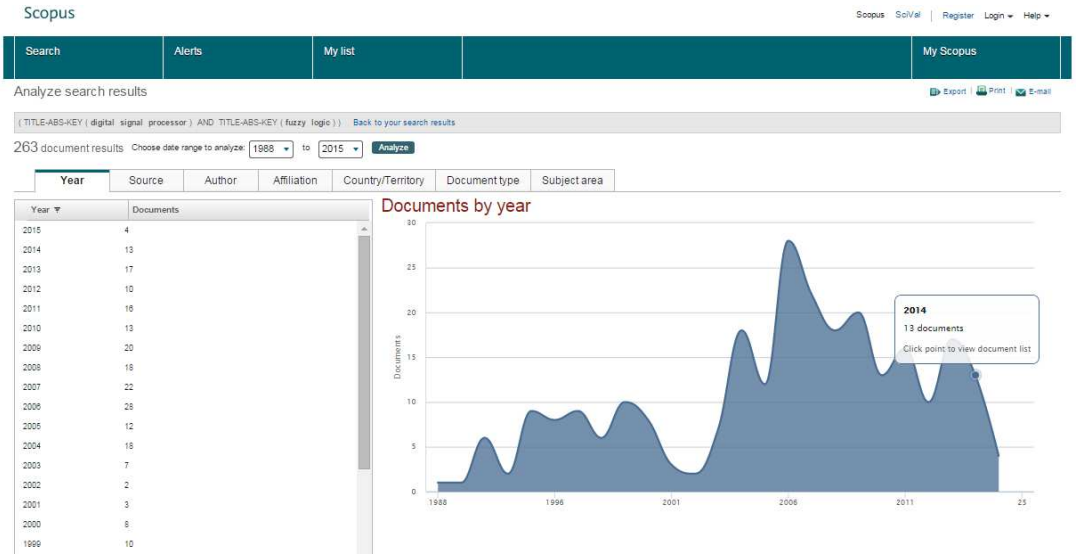


Figure 1.5: Literature on DSP implementation of FLCS as reported by scopus as on May 2015

FLCS architectures have also been developed over an application specific integrated circuit (ASIC) [37, 87, 110, 129, 154] attaining a speed of more than 50 MFLIPS [33, 129, 209]. However, the design of these controllers does not make them truly reconfigurable in nature. The major disadvantage is their unavailability for field or on-line tuning [33, 142]. It has also been observed that many of the ASIC based designs have been developed using membership function generators (MFGs). MFG circuitry can tune the membership functions (MFs) by setting some voltages on IC pins [124, 172] but the Rulebase remains static. To ensure field tunability, it is necessary to impart the FLCS with features that would enable users to change control parameters in run-time. In some of the FLCS designs, the fuzzy parameters are stored in digital memory as weights [33, 84, 209] to impart features of tunability. This technique permits a look-up-table based approach for fuzzy computations which reduces the computation time. However, the online reconfigurability of such system becomes difficult. This technique is well suited for application specific FLCS design where the rulebase and the parameters remain unchanged during the majority of the operation.

Though most of these designs are based on ASIC or FPGA platform, many

of the designs are developed on DSP platform for its ease of implementation and reconfigurability [3, 57, 84, 105, 107, 123, 130, 155, 190, 193]. From Figure 1.5 it is evident that DSP have been also been a preferred platform for implementation of FLCS designs. However, these designs suffer from limitations on the flexibility of weight update. Recently, few RISC processors have included dedicated fuzzy instructions and have been seen to deliver a very high performance in terms of FLIPS [36, 62, 157, 197]. These designs are summarized in Table 1.2 where the number of inputs and outputs supported by these systems and maximum number of rules that can be evaluated by these systems is listed. However, the common limitation of many of these designs is that the control parameters can be updated only by removing these controllers from the system, rendering the plant off-line. Some of the works have reported to update parameters on-line through complex learning processes [4, 20, 34, 46, 127]. In the current decade, DSP based FLCS designs have been widely used in control applications[39, 68, 175, 191]. Over 250 DSP based fuzzy system designs have been reported in last decade according to www.scopus.com. It has been found that some of the most preferred DSP platforms for this type of design are Texas Instruments' TMS320F series DSPs[51, 52, 71, 135, 147, 149], TMS320C series DSPs[57, 130, 173, 190, 191] and dSpace DSPs[39, 111, 134, 148, 155]. The DSP platform is also found to be a preferred platform for FLCS design.

Even though FPGA is preferred platform for implementation of FLCs compared to a programmable DSP, in this work implementation is done using a TI C6748 DSP processor. The primary reasons for selection of the mentioned hardware include:

- DSP provides efficient implementation of multiplication and accumulation (MAC) and this helps COA implementation.
- File handling and socket programming is an integral part of this design. These are achieved easily since the development is done using C language.
- This design supports high level of branching and decision making.

Collectively, for this architecture, DSPs were judicially selected over conven-

tional FPGAs.

1.11 Generic Fuzzy Logic Controller

Generic fuzzy logic controller systems (G-FLCS) are standalone and remotely tunable fuzzy logic control devices. These devices are developed on suitable hardware platforms such that they can be easily interfaced with various process plants. The major characteristic of these type of devices is that they do not require reprogramming. These devices accept fuzzy parameters from the users externally through some user interfaces or programmable pins.

G-FLCS designs are mostly crippled by their operational speed and hence they are generally forced to operate under reduced functionalities. Some of the prime G-FLCS designs on various platforms have been surveyed and tabulated in Table 1.2. The table also lists the fuzzy parameters reported in these designs. The following observations have been summed up after analyzing these designs.

- It can be observed, that majority of these designs use singleton MFs at the output to reduce computational complexity. Centroid of area (COA) method when applied to singleton, which is commonly known as weighted average defuzzification method yields far low computational complexity. However, unlike COA, weighted average does not compute the area under the curve produced from the fuzzy outputs [153]. It can be observed that COA presented in eq. (1.1)

$$Y^* = \frac{\int \mu_c(y) y dy}{\int \mu_c(y) dy} \tag{1.1}$$

  can reduced to weighted average as depicted in eq. (1.2).

$$Y^* = \frac{\sum \mu_c(y) \cdot y}{\sum \mu_c(y)} \tag{1.2}$$

  where Y^* represents the crisp output computed from output fuzzy set $\mu_c(y)$ and output support membership function value y.

- These designs uses a stringent rule reduction technique where only two

overlapping memberships have been considered.

- These systems evaluate very few rules to improve computational speed. The reduction in the number of rules with only two overlapping membership functions does not provide desired performance in terms of accuracy of the system for many applications.

- In general, these systems cannot be remotely tuned. Some of these devices have MFGs for tuning MFs but the Rulebase remains static and the performance is limited to two inputs.

These limitations motivate research in soft-core generic FLC devices on programmable hardware where multifarious control over the system can be obtained by varying different control parameters with modest computational complexity.

1.12 Problem Statement

The limitations of the existing G-FLCS designs, as defined in Table 1.2, motivated the research in developing a soft-core G-FLCS device on programmable DSP hardware with two level tuning where multifarious control over the system can be obtained with modest computational complexity. However, this design is extremely challenging owing to following conditions.

- Since the proposed design is expected to command a large number of fuzzy parameters; it is imperative to develop an interactive interface for guiding users to input fuzzy parameters.

- It is also a known fact that human beings are prone to make errors while handling large data. Therefore, an automated system has to be deployed which will extract coarse fuzzy parameters from a large input-output dataset.

- The significant challenges in designing such a system lies in managing an exponentially growing rulebase. Therefore, development of a suitable

rule reduction technique is required which will generate a desired output consuming minimum cycle time.

- It has been discussed previously that defuzzification module in FLCS using COA technique is computationally quite expensive. A desirable COA scheme with low computation time is essential to achieving a dependable system cycle time that is relevant to the majority of control applications.

1.13 Outline of Thesis

The thesis is presented in 6 chapters. Following this chapter on introduction, the remaining thesis is organized as under: This thesis is organized as follows:

- Chapter 2 presents a mathematical model of the generic fuzzy logic controller and describes the proposed MT-FRHC rule reduction technique. A vertices based centroid of area computation algorithm for defuzzification is also proposed.

- Chapter 3 explains the design architecture and develops the backbone of the proposed the G-FLCS. The proposed architecture includes a web based user interface (WebUI) for users to program fuzzy parameters in the GFLCS, a genetic algorithm based fuzzy parameter extraction scheme and a fuzzy framework.

- Chapter 4 implements the concepts developed in Chapter 2 and 3 in optimized C code and realizes the proposed G-FLCS on a C6748 DSP processor. Applicability and performance analysis of the G-FLCS is analyzed.

- In Chapter 5, the proposed G-FLCS is used to control the radial position of plasma in Aditya Tokamak Fusion Test Reactor (TFTR). The GA based fuzzy parameter extraction process is used to obtain FCP for the control problem. This FCP is used to control the plasma position in the Aditya TFTR Simulink model and the same is compared to some of the previously developed systems.

- Finally the research work is concluded in chapter 6 and the scope of future work is explained briefly. The limitations and the scope for future work of this research are also elaborated in this chapter.

Table 1.2: Important works on G-FLCS

Year	Speed (in FLIPS)	Platform	Features
1995 [121]	0.63M	BiCMOS 2μm	Output MFs: Singleton (7) I/O: 5 bit Input MFs: 11 Overlaps: 2 Rules Evaluated: 4 per IC
1996 [197]	48-122	R3000A RISC Fuzzy Processor	Output MFs: - I/O: - Input MFs: Overlaps: 2 Rules Evaluated: 51
2005 [9]	15.87M	CMOS 0.35 μm	Output MFs: Singleton (7) I/O: 2-1 Input MFs: 3 Overlaps: 2 Rules Evaluated: 9
2007 [201]	16.6M	CMOS 0.35 μ m	Output MFs: Singleton (7) I/O: 2-1 Input MFs: 4 Overlaps: - Rules Evaluated: 16
2007-2008 [61] [136]	5.5K	FPGA	Output MFs: Singleton (5) I/O: 2-1 Input MFs: 8 Overlaps: - Rules Evaluated: 64
2010 [56]	11K	FPGA	Output MFs: - (5) I/O: 2-1 Input MFs: 5 Overlaps: 2 Rules Evaluated: 25
2011 [200]	16.6M	CMOS	Output MFs: Singleton (7) I/O: 2-1 Input MFs: 4 Overlaps: 2 Rules Evaluated:16
2014 [172]	15M	CMOS 0.35 μm	Output MFs: Singleton(7) I/O: 2-1 Input MFs: 5 Overlaps: 2 Rules Evaluated:25
2015 [124]	NA	CMOS 0.35 μm	Output MFs: Singleton I/O: 2-1 Input MFs: 4 Overlaps: 2 Rules Evaluated:16

Chapter 2

Generic Fuzzy Logic Controllers

Preview

This chapter presents an introduction to mathematical deduction of G-FLCS architecture. A rule reduction scheme compatible with this architecture is also incorporated to reduce the complexity of the system and assure it is realizable in real-time. The proposed scheme is called modified and thresholded fired rules hyper cube (MT-FRHC), and it is based on rule reduction technique using overlapping membership functions. In MT-FRHC, control designers can dynamically assign the number of overlaps to be considered within the G-FLCS system. Further, a thresholded fuzzifier optimizes the system with discourse to computational complexity and throughput accuracy. In the proposed system architecture, the number of t-norm and t-conorm operations per inferences can also be dynamically varied between discrete values. This fuzzy model is analyzed with the context of its output performance and computational complexity. A formative observation in favour of MT-FRHC has been inferred from this analysis.

2.1 Introduction to Generic Fuzzy Logic Controller System

G-FLCS are essentially hardware based general purpose control devices operating on fuzzy logic principles. Characteristic features of these control devices include

- Plug and play framework,
- Runtime tunability, and
- Standalone operation.

In this chapter, a G-FLCS is presented that provides a runtime reconfigurable framework. This fuzzy system can be employed to control any system that is in tandem with the controller I/O specifications. In most G-FLCS designs, fuzzy control parameters (FCP) are updated as weights. FCP defines the parameters like number of Inputs and Outputs, number of MFs in each Input and Output, details of each MF like, their function type and their co-ordinates, the Rulebase of the FLC defined by the index numbers of the inputs and outputs. These parameters are put into a FLC framework that drives the operation of the fuzzy system. However, it can be noted that not all FCPs in a G-FLCS are updated at run-time. The parameters that can be updated at run-time depends on the architecture of G-FLCS, and can be termed as *flexible parameters* of G-FLCS. Yosefi et. al. [200, 201] implemented G-FLCS on 0.35 μm CMOS technology to achieve 16.6 MFLIPS but used singleton type membership functions at the output with a rigid Rulebase. Moreover, only two overlapping membership functions are considered in the design. Vasantha Rani et. al. [151] introduced multicycle architecture for G-FLCS and implemented it on FPGA to achieve an operational speed of 31K FLIPS. This system used singleton membership functions at the output. It also used reduction in the number of rules that can be accommodated. Most FLC designs in literature, are developed on reconfigurable hardware namely FPGAs [2, 37, 117]. Some of these have also been developed on application specific integrated circuit (ASIC) and have attained a speed of more than 50 MFLIPS [33, 209]. But it can be observed among these G-FLCS

designs that, the majority of FCPs are not recognized as flexible parameters and programmed directly into FLC core to achieve high speed. This leads to conclusion that, the computational complexity of a G-FLCS system depends on,

- Number of FCPs considered as flexible parameters in the G-FLCS design;
- Maximum number of rules that the G-FLCS can accommodate,
- The type of defuzzification method employed for the conversion of the output from fuzzy domain to crisp real value.

Some of the commonly used defuzzification methods include, centroid of area (COA), bisector of area (BOA), mean of maxima (MOM), largest of maxima (LOM) and smallest of maxima [153]. COA is one of the most commonly used defuzzification method in control applications and implemented as [64].

$$Y^* = \frac{\int \mu_c(y) y dy}{\int \mu_c(y) dy} \tag{2.1}$$

where Y^* represents the crisp output computed from output fuzzy set $\mu_c(y)$ and output support membership function value y. However, COA is highly resource intensive when fuzzy set associated with the output consists of fuzzy numbers of type other than singleton. For singleton type output fuzzy set the continuous representation in (2.1) is reduced to discrete representation leading to [153]

$$Y^* = \frac{\sum \mu_c(y) \cdot y}{\sum \mu_c(y)}$$

This chapter introduces a novel rule reduction technique named as MT-FRHC. It improves the accuracy of existing overlap based rule reduction technique. Fast inference time is the most important feature of the overlapping MFs based rule reduction technique. The proposed MT-FRHC system achieves to increase the accuracy without increasing the inference time. The FRHC rule reduction algorithms which is widely accepted and used because, it is the only algorithm which performs effective rule selection without pruning or forking into the Rulebase provided from the user [4, 127, 152, 161]. Pruning and forking of rules in original rulebase is an application specific process [133, 165, 196].

These type of rule reduction techniques are not suitable for generic fuzzy systems. Pruning and forking of rules can be applied as a wrapper on top of the proposed G-FLCS but these methods cannot be programmed into the core of the G-FLCS. There are also layer-structured machine learning algorithms (Neural Network, Genetic Algorithm) to learn the effective rules [74, 75, 167]. However, these methods for rule extraction are computationally quite expensive. Thus when an application is known, the fuzzy control parameters (which include rulebase) can be extracted using these techniques. These techniques provide good default parameters and this is described in Chapter 5 elaborately. For implementation of G-FLCS in radial position control of plasma in Aditya TFTR, a genetic algorithm based rule extraction technique is used which reduces the rules in rulebase and thereby reducing effective rules. Therefore, it was required to derive an effective rule reduction scheme that do not fork and prune the original rulebase like FRHC. Therefore, in this work, the FRHC algorithm has been mathematically analyzed and suitable assumptions have been introduced in it which can dynamically control the computational complexity of the G-FLCS based on user requirement.

2.1.1 Rule Reduction using Overlapping Membership Functions

The concept of overlapping membership functions has been widely used in reducing computational time of fuzzy systems, specially in hardware development of FLC as it eradicates the non-linear dependency between number of inputs and computational complexity in the G-FLCS. It was originally proposed by Eichfled et. al. in 1992 [50] and it was improvised in 1999 by I. Kalaykov and named as Fired-Rules-Hyper-Cube (FRHC) [84]. However, this rule reduction method is constrained by anticipating fixed number of overlaps that can affect controller performance. It is characterized by a layered parallel architecture of the fuzzy inference. Moreover, it reduces the dependency of processing time on the number of inputs to the fuzzy system while dependency on the number of rules and fuzzy partitioning of all variables are completely eradicated. This concept has been adopted in various implementations of FLCS resulting in en-

hancement of speed [4, 127, 152, 161]. However, this stipulation of segregation of input space with maximum of two overlapping membership functions, causes major bottleneck in accuracy and tuning of the FLC, specially because the MFs cannot be unevenly distributed over the input spaces, which is seen to be circumstantial in majority of non-linear FLC system design. However, if employed in control of a non-linear system where the MFs are unevenly distributed over the input space, this technique of rule reduction would fail to provide expected accuracy.

2.1.2 Motivation for Modified FRHC (M-FRHC)

FRHC rule selection method constraints the Rulebase design by limiting the number of MFs operating over any part of the input region to two. But when this algorithm is applied to higher order non-linear systems with uncertainties, the system performance gets affected and tuning becomes extremely difficult [4, 5]. Fuzzy controllers are important in situations where large uncertainties or unknown variations are predominant in parameters and structures of a plant. By introducing FRHC, this essence of Fuzzy Control is lost to certain extent because of the assumption that only two overlapping membership operates over the solution space. It is not sufficient to create a fuzzy hypothesis which can incorporate large uncertainty and unknown variations. But, at same instance, FRHC based G-FLCS is computationally cheap and easy to implement.

The most important module in the tuning of G-FLCS is the learning of FCP, that can be achieved by many algorithms, namely Genetic Algorithm (GA), Univariate Marginal Distribution Algorithm (UMDA) and others. The convergence of these optimization algorithms depend on the flexibility of the membership functions and the fuzzy partitioning of the input and the solution space. Since FRHC is a constraint based rule reduction scheme, the convergence of the optimization algorithms are subject to the complexity of the system. This implies that the convergence may or may not occur for a fairly complex system. Therefore, it is the modality of the rule reduction scheme that needs to refined in accordance to the proposed G-FLCS.

Thus to keep the simplicity of the FRHC intact while countering its limi-

tations of flexibility and modality, a modified fired-rules-hyper-cube (M-FRHC) rule reduction technique is proposed and incorporated.

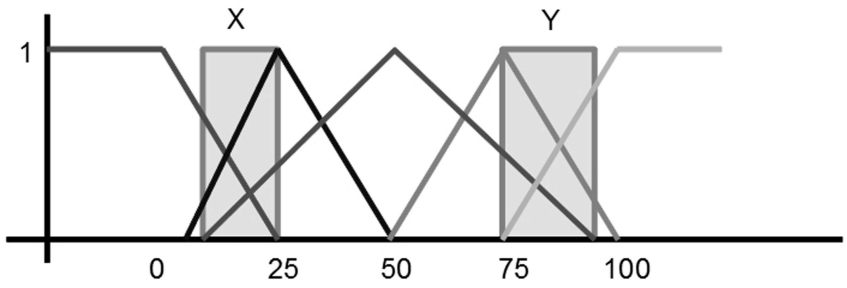


Figure 2.1: More than two fuzzy logic antecedent membership functions overlapping at once

Advantages of the proposed M-FRHC over its conventional counterpart FRHC, can be understood considering a case where fuzzy logic antecedent MFs are distributed in the input space as shown in Figure 2.1. In the input space marked as X and Y, any crisp input shall be fuzzified to more than two non-zero membership grade since there are more than two overlapping MFs. It is observed that the number of non-zero membership grade in a fuzzified input cannot exceed more than the maximum number of overlaps between the MFs distributed in the input space. In systems where optimization of FCPs are implemented, it is more often than not that the MFs in the input, as well as in the output space, will be distributed unevenly, and their overlaps will be inconsistent. When FRHC is applied on these set of input MFs, useful data is lost if crisp input lies within X or Y region of the input space. FRHC algorithm will fire the first two rules related to the non-zero fuzzy values instead of three rules which is evident. This process is potentially erroneous where the error in the control output is proportional to the weights of the membership function(s) discarded and its associative implication on the Rulebase. Implementation of M-FRHC will allow the number of overlaps to vary suitable with conjunction to the complexity of the control system.

M-FRHC operates on a platform of dynamical assertion of overlaps in membership functions as per the Table 2.1. Based on the number of inputs, M-FRHC

will dynamically controls the number of overlaps to be considered. Introduction of this feature allows fuzzy model to be flexible enough to accommodate the uncertainties and modality of the system. However, with increase in input, when there is an exponential growth in computational complexity, it reduces the overlaps that are considered to reduce the computational complexity. This enforces a trade off between accuracy and speed. However, in truly generic systems, the operating point of this trade off should be defined by user. M-FRHC introduces a sophisticate approach to achieve this through the concept of N_{Cells} which defines the operating point on the accuracy and speed trade off. It is to be noted that in Table 2.1, the bold and underlined values of N_{Cells} are used for all experiments appearing in later part of this thesis. Although, it remains in the discretion of the user to use any values of N_Cells for a given O and n.

For example, consider the following rules,

$$If\ input\ I\ is\ MF_1\ then\ Output\ O\ is\ P_1$$
$$If\ input\ I\ is\ MF_2\ then\ Output\ O\ is\ P_2$$
$$If\ input\ I\ is\ MF_3\ then\ Output\ O\ is\ P_3$$

and the membership distribution over input space be as in Figure 2.1. For an input value assuming to be 20, the actual fuzzication output will be approximately,

$$MF_1 = 0.4,\ MF_2 = 0.5,\ MF_3 = 0.2$$

FRHC under the assumption of two overlapping membership function will return,

$$MF_1 = 0.4,\ MF_2 = 0.5$$

which implies that rule number three does not fire. However, the system expects rule three to fire since it has a weight of 0.2 associated with membership function P_3 in the output or solution space. In retrospective, if N_{Cells} is configured to 5 with with number of inputs being one, M-FRHC will consider maximum number of five overlaps in the membership function. Hence, it will generate

$$MF_1 = 0.4,\ MF_2 = 0.5,\ MF_3 = 0.2$$

ass the system would expect and thus firing rule number three in the Rulebase.

2.1.3 Analytical Differences between Conventional Overlapping Membership Function (OMF) method and M-FRHC

FRHC based rule reduction scheme uses the concept of two overlapping membership functions to reduce the computation. This method is valid when the maximum number of overlapping MFs distributed over the input space is limited to two. However, to counter uncertainty and variations in a nonlinear plant, adaptive control is imperative. The basic objective of an adaptive control strategy is to maintain smooth performance of a system in the presence of these uncertainties. Therefore, introduction of online and offline adaptive fuzzy control is important feature of a G-FLC which essentially means updating and tuning FCP at various time intervals. The implementation of adaptive fuzzy control strategy in proposed G-FLCS is explained in Chapter 3. So, with M-FRHC in operation, the uncertainties are modeled better compared to conventional overlapping membership methods of active rule reduction.

In FRHC technique, [50, 84]

$$y = y_{aggr}.\left(y_w^{-1}\right) \tag{2.2}$$

where (2.3)

$$y_{aggr} = \sum_{i=1}^{N_{cells}} w_i.y_i, y_w \qquad = \sum_{i=1}^{N_{cells}} w_i$$

where w_i represents the non-zero weights of the fuzzified input and y_i represents the *Kernel* of a output fuzzy set fired by the Rulebase.

Definition 1 (Fuzzy Set) Let X be a nonempty set. A *fuzzy set* A in X is characterized by its MFs.

$$\mu_A : X \rightarrow [0,1]$$

and $\mu_A(x)$ is interpreted as the degree of membership of element x in fuzzy set A for all $x \in X$.

Definition 2 (Kernel or Core) Let A be a fuzzy subset of X; the support of A, denoted $supp(A)$, is the crisp subset of X whose elements all have nonzero membership grades in A.

$$core(A) = \{x \in X | A(x) = 1\}$$

Definition 3 (Support) Let A be a fuzzy subset of X; the support of A, denoted $supp(A)$, is the crisp subset of X whose elements all have nonzero membership grades in A.

$$supp(A) = \{x \in X | A(x) > 0\}$$

Table 2.1: Computed N_{cells} with varying n and O

	$n=1$	$n=2$	$n=3$	$n=4$
$O=2$	2	4	8	**16**
$O=3$	3	9	**27**	81
$O=4$	4	**16**	64	256
$O=5$	**5**	25	125	625
$O=6$	6	36	216	1296
M-FRHC	**5**	**16**	**27**	**16**

Most G-FLCS are designed on the principles described in (2.2). For all such FLCs, it can be observed that the computational complexity depend on N_{cells} which is non-linearly related to number of overlapping membership functions and number of system inputs. The number of system inputs can be presented as:

$$N_{cells} = O^n \tag{2.4}$$

where, O represents the number of overlaps considered and n represents the number of inputs. The values of N_{cells} for different O and n is computed and tabulated in Table 2.2. This shows that N_{cells} increases exponentially with respect to the n. This dependency of N_{cells} on O and n can be analyzed for the Table 2.2. For system implementation FRHC is a popular choice, where $O = 2$ as [84] states that, "..uncertainty has to be on boundary between two fuzzy

sets". Hereby, FRHC will take values from first column of Table 2.1 depending on the in the system inputs. Now, if O predefined, the generic FLC system assumes value of N_{cells} based on n and this remains static. Additionally, the computational complexity increases exponentially with increase in n.

Consider a fuzzy logic system with n inputs where each input have x numbers of membership functions. This implies that the number of rules that can result from this combination is x^n. Now, for any input value $I = [i_1, i_2, i_3 ... i_n]$, the resultant vectors $\{\Psi(i_p), \forall p \in [1, n]\}$ (fuzzified inputs) will have maximum of O numbers of non-zero values, where O is the maximum number of overlapping membership functions distributed over each of the input space. The number of effective or active rules that can be fired will only depend on number of non-zero values in the resultant fuzzified input. The number of non-zero values are directly proportional to the number of overlaps as shown in region X and Y in Figure 2.1. It can be observed that, O^n out of a set of x^n possible rules will be effective rules. N_{cells} represents the number of active rules in FRHC rule reduction scheme.

In the proposed MT-FRHC system, the maximum number overlapping membership functions considered at inference can be dynamically varied to provide computational advantage. To strike a balance between accuracy and computational complexity, it is important to control the maximum number of overlapping membership functions dynamically. In this G-FLCS implementation, the maximum number of overlaps to be considered has been systematically reduced with increase in the number of inputs. With increase in the number of inputs, number of rules in an FLC increases exponentially leading to increased computational complexity. However, this exponential growth in computational complexity can be curtailed by decreasing the number of overlaps.

It can be noted that, in MT-FRHC the values of N_{cells} can be assumed from Table 2.1 randomly. However, the proposed MT-FRHC scheme allow us to dynamically choose the maximum number of overlapping membership functions which are underlined in Table 2.1. It can be observed that the assumed values appear as diagonal elements of Table 2.1. This is a naive approach to implement the concept that as the number of inputs increases, the maximum number of overlaps to be considered decreases. As, $N_{cells} = O^n$ implies that

$O = \sqrt[n]{N_{cells}}$. Again, N_{cells} is directly proportional to computational complexity as shown in (2.2). To achieve a constant N_{cells}, O should be varied in accordance to the change in number of input n. Thus in this G-FLCS implementation, $\forall n = \{1,2,3,4\}$, $O = 5,4,3,2$. Using these values in (2.4), in the proposed M-FRHC based G-FLCS design, N_{cells} are assumed as following

$$N_{cells} = \begin{cases} 5, \ \forall n = 1 \\ 16, \ \forall n = 2 \\ 27, \ \forall n = 3 \\ 16, \ \forall n = 4 \end{cases}$$

These values of N_{cells} are assumed such that the number of effective rules for every inference could be kept low without affecting the accuracy of the system. This combination is seen to be optimal for this system since, increasing the number of inputs will have minimum effect on overall computation time and complexity.

In summary, for any system where MFs are distributed randomly over the input space (as shown in an example Figure 2.1), FRHC will fail to produce desired control output. However, configuring N_{Cells} effectively, M-FRHC can tackle the evenness in the distribution of the fuzzy memberships in the input and the solution space. The flexibility and modality of the M-FRHC algorithm is completely tunable and hence it provides more inclusive environment for tuning and optimization of FCP using learning algorithms. Due to the flexibility in the structure, M-FRHC can generate complex hypothesis for FCP to incorporate large uncertainties and variations. Thus in these applications, FRHC can be replaced by proposed M-FRHC as the rule reduction technique and tuned accordingly for improved performance and accuracy.

2.2 Mathematical Modeling of G-FLCS

Lemma 1 (Vector Combination) *Let A and B be two vector of dimension d. $\Lambda_c(A,B)$ returns a matrix M of dimension $d^2 \times 2$.*

$$\boldsymbol{m}_{i,j} = \left(a_i, b_j\right) \forall i,j \rightarrow [1,d]$$

where,
$\boldsymbol{m}_{i,j}$ represents individual row in matrix M and consists of two elements a_i and b_j each from set A and B respectively.

A multiple-input single-output (MISO) FLCS with N number of inputs is considered. Each input space is segregated in z_j fuzzy numbers and M represents the maximum number of membership functions that the system can accommodate. The set of fuzzy numbers spread over each input space can be presented as

$$Z = \left\{z_j | z_j \in [1,M], \forall j \rightarrow [1,N]\right\}$$

The FLCS transform the crisp inputs to the fuzzy domain using a *fuzzifier* module. If input i is introduced to the *fuzzifier* with M membership functions, then *fuzzifier* module returns a set of values corresponding to degree of each membership when input i is mapped on them. This can be represented as

$$\Delta(i) = \psi_i = \left\{\mu_1(i), \mu_2(i), \ldots \mu_M(i)\right\} \tag{2.5}$$

where $\left\{\mu_j(i) \in [0,1], \forall j \rightarrow [1,M]\right\}$

Input vector $I = \{i_1, i_2 \cdots i_N\}$ is a set of scalar inputs, also known as crisp input, to the FLCS system. When I is introduced to *fuzzifier* module it is transformed to fuzzy domain as shown in (2.6).

$$\Delta(I) = \begin{bmatrix} \psi_1 \\ \psi_2 \\ \vdots \\ \psi_N \end{bmatrix} = \begin{bmatrix} \mu_1(i_1) & \ldots & \mu_M(i_1) \\ \mu_1(i_2) & \ldots & \mu_M(i_2) \\ \vdots & \vdots & \vdots \\ \mu_1(i_N) & \ldots & \mu_M(i_N) \end{bmatrix} \tag{2.6}$$

Λ_c is vector combination function operated on $\Delta(I)$ to generate C_R, a matrix of size $\left(M^N \times N\right)$.

$$C_R = \Lambda_c\left(\psi_1, \psi_2, \cdots \psi_N\right)'$$

If a Rulebase matrix R_b with N_R rules is sorted based on antecedent, then each set of antecedents corresponds to the index of consequent. A generalized structure of a rule in Mamdani FLCS can be represented as,

k^{th} Rule:

$$R_b(k): \quad \text{If } i_1 \text{ is } a_{1,k} \text{ and } i_2 \text{ is } a_{2,k} \text{ and } \cdots \text{and } i_N \text{ is } a_{N,k}, \text{ then } c_j \tag{2.7}$$

where, antecedents $a_1, a_2 \cdots a_N$ can be represent a unique index k, and k is represented by

$$k = (a_N a_{N-1} \cdots a_0)_M = \sum_{j=0}^{N} a_j M^j \tag{2.8}$$

where, M represents the maximum number of membership functions and N repesents the number of inputs with $j \rightarrow [1, N]$. Therefore, the k^{th} index in Rulebase matrix points to corresponding consequent.

$$R_b(k) = c_j \tag{2.9}$$

where $c_j \in [1, M] \ \forall j = \{1, 2, \ldots N_R\}$

The fuzzy output inferred from Rulebase is a fuzzy set θ_f which is essentially a vector of M elements.

$$\theta_f(c_j) = \bigcap_{k=0}^{N_R - 1} \left(\theta_f(R_b(k)), \left(\bigcup_{l=0}^{N-1} \overrightarrow{C_R(l)} \right) \right) \tag{2.10}$$

where, θ_f is fuzzy output vector, index k varying from 0 to number of rules N_R, R_b is the rule base matrix, C_R represents the vector combination of fuzzified input, l varying from 0 to $N-1$ and $\bigcap$ and $\bigcup$ representing t-norm and s-norm operations respectively. The *defuzzifier* in G-FLCS converts fuzzy output vector

θ_f into scalar output commonly known as crisp output.

$$\begin{aligned} \theta_f &= \{\mu_1(\theta), \mu_2(\theta), \ldots \mu_M(\theta)\} \\ \theta &= \Delta^{-1}(\theta_f) \end{aligned} \tag{2.11}$$

where, Δ^{-1} denotes inverse fuzzy operator operating on fuzzy output vector θ_f.

2.2.1 Overlapping Membership based Rule Reduction

FRHC can be realized by considering $M = 2$ in (2.6). To implement this, the index of corresponding non-zero membership functions needs to be pursued and used during inference mechanism. It is recorded in a matrix P_i. Thus, for this condition (2.5) can be represented as

$$\Delta(i) = \psi_i = \begin{cases} \{\mu_1(i), \mu_2(i), \ldots \mu_M(i)\} \\ \{p_1(i), p_2(i), \ldots p_M(i)\} \end{cases}$$

Now, replacing $M = 2$, (2.6) becomes,

$$\begin{aligned} \psi_I &= \begin{bmatrix} \mu_1(i_1) & \mu_2(i_1) \\ \vdots & \\ \mu_1(i_N) & \mu_2(i_N) \end{bmatrix} \\ P_I &= \begin{bmatrix} p_1(i_1) & p_2(i_1) \\ \vdots & \\ p_1(i_N) & p_2(i_N) \end{bmatrix} \end{aligned} \tag{2.12}$$

where ψ_I represents fuzzified input matrix with non-zero membership degree and P_I represents the index of these non-zero membership degree.

$$\begin{aligned} C_{R_k} &= \Lambda_c(P_1, P_2, \ldots P_N) \\ C_{R_V} &= \Lambda_c(\psi_1, \psi_2, \ldots \psi_N) \end{aligned}$$

Thus, the vector combination operation is to be applied on both matrices, P_I and ψ_I to obtain resultant matrices C_{R_k} and C_{R_V}. With this the output relationship

(2.10) can be represented as

$$\theta_f(c_j) = \bigcap_{k_x=0}^{(N^2)-1} \left(\theta_f(R_b(k_x)), \left(\bigcup_{l=0}^{N-1} \overrightarrow{C_{R_V}(l)} \right) \right) \tag{2.13}$$

where θ_f is fuzzy output vector, index k_x varying from 0 to number of rules $(N^{O_l} - 1)$, R_b is the rule base matrix, C_{R_V} represents the vector combination of non-zero fuzzified input values, l varying from 0 to $N-1$ and $\bigcap$ and $\bigcup$ representing t-norm and s-norm operations respectively.

2.2.2 Modified Fired Rulebase Hyper Cube (M-FRHC)

The G-FLCS described earlier in this section, has a complexity of $\mathcal{O}(M^N)$. For FRHC rule reduction technique $M = 2$, the computational complexity reduces to $\mathcal{O}(2^N)$. Table 2.2 presents the relationship between and number of overlaps considered, with reference to number of Inputs. The complexity $\mathcal{O}$ is proportional to number of operations per fuzzy inference n_{op}. Therefore these notions can be cumulatively formulated as

$$n_{op} = {O_l}^N \tag{2.14}$$

where O_l is the number of overlaps considered.

In G-FLCS, implementation of M-FRHC will allow the number of overlaps to vary suitable with conjunction to the complexity of the control system. Here, n_{op} per fuzzy inference becomes a programmable parameter in this proposed M-FRHC rule reduction technique. The effect of this complexity reduction is presented in Figure 2.2. It can be observed that the number of fuzzy inferences or operations varies exponentially against the number of inputs considering a different number of overlapping membership functions except the plot corresponding to M-FRHC. In this G-FLCS implementation, $\forall n = \{1,2,3,4\}$, $O = 5,4,3,2$. Using these values in (2.4), in the proposed M-FRHC based G-FLCS design, $N_{cells} = \{5,16,27,16\}$ This causes a slight drop in the green line in Figure 2.2 specially when value of n transits from 3 to 4. It is observed from Figure 2.2, that the plot corresponding to M-FRHC is considerably parallel to

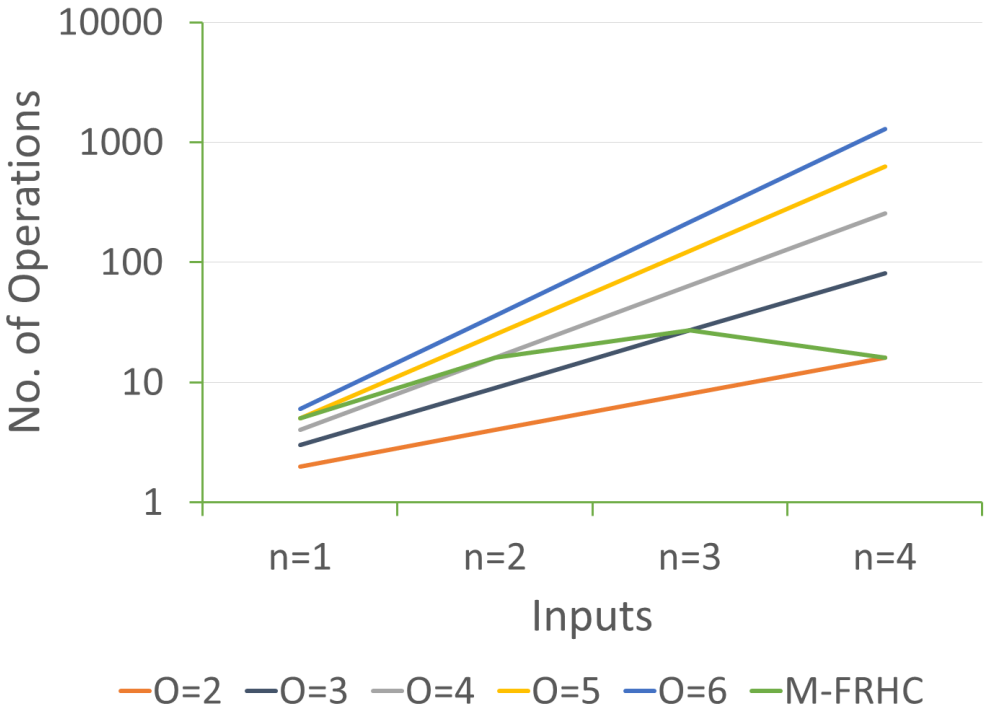


Figure 2.2: Inputs vs No. of Operations with constant overlaps

x-axis on a logarithmic scale where all others are linearly increasing. This implies that, M-FRHC generates a constant number of operations per inference for all inputs $N \rightarrow [1,4]$ whereas, the complexity of the G-FLCS increases exponentially with the increase in the number of inputs. It also caters a flexibility of adjusting the uncertainties on the boundary between the fuzzy sets. Hereby, (2.12) can translated to (2.15).

Table 2.2: Computed n_{op} with varying Inputs and Overlaps

N \ O_l	2	3	4	5	6
1	2	3	4	5	6
2	4	9	16	25	36
3	8	27	64	125	216
4	16	81	256	625	1296

$$\psi_I = \begin{bmatrix} \mu_1(i_1) & \cdots & \mu_{o_l}(i_1) \\ \vdots & \ddots & \vdots \\ \mu_1(i_N) & \cdots & \mu_{o_l}(i_N) \end{bmatrix}$$
$$P_I = \begin{bmatrix} p_1(i_1) & \cdots & p_{o_l}(i_1) \\ \vdots & \cdots & \vdots \\ p_1(i_N) & \cdots & p_{o_l}(i_N) \end{bmatrix} \tag{2.15}$$

where, there are O_l number of elements in ψ_I. O_l represents the number of overlaps considered.

It is required to track the index of membership function and link it to the Rulebase matrix appropriately. Hereby, in this scheme it is required to implement the vector combination operation on P_I along with ψ_I to generate C_{R_k} and C_{R_v} respectively. C_{R_k} and C_{R_v} can be represented as

$$C_{R_k} = \Lambda_c(P_1, P_2, \ldots P_N)$$
$$C_{R_v} = \Lambda_c(\psi_1, \psi_2, \ldots \psi_N)$$

where Λ_c denotes the vector combination as defined in Lemma 1.

C_{R_k} is required to derive the index of the Rulebase matrix R_b. The index k_x of Rulebase matrix R_b can be derived by

$$k_x = \left\{ \sum_{j=0}^{N-1} C_{R_k}(x, j), \forall x | x \rightarrow \left[1, N^{O_l}\right] \right\} \tag{2.16}$$

$$R_b(k_x) = c_j$$
$$c_j \rightarrow [1, M] \forall j = \{1, 2, \ldots N_R\} \tag{2.17}$$

Finally, fuzzy output is derived from the following relationship.

$$\theta_f(c_j) = \bigcap_{k_x=0}^{n_{op}-1} \left(\theta_f(R_b(k_x)), \left(\bigcup_{l=0}^{N-1} \overrightarrow{C_{R_v}(l)} \right) \right) \tag{2.18}$$

where, θ_f is fuzzy output vector, index k_x varying from 0 to number of fuzzy

operations ($n_{op}-1$), R_b is the rule base matrix, C_{R_V} represents the vector combination of non-zero fuzzified input values, l varying from 0 to $N-1$ and $\bigcap$ and $\bigcup$ representing t-norm and s-norm operations respectively. It is important to analyze the data path of the system architecture to take advantage of the device architecture on which it will be deployed. The target processor is a VLIW based DSP device. The system architecture can be modified at later stage however, there should be scope for data and instruction level parallelism.

2.2.3 Modified and Thresholded Fired Rulebase Hyper Cube (MT-FRHC)

In this section, MT-FRHC is proposed to tackle the challenges of removing unwanted firing of rules by insignificantly close to zero fuzzy values. The removal of these rules significantly increases computational speed without affecting the output accuracy. This can be achieved by introducing a threshold in (2.15). Consider an element in ψ_I that assume a very low value and eventually may fire one or more rules. Based on T-Norm operators, weights of all these fired rules are likely to be close to the value of the element. This value will produce a minuscule λ cut-set at the fuzzy output set if there exist no value greater than the current weight assigned to the corresponding member of the fuzzy set. The effect of this λ cut-set is likely to result in a very fine change in output after defuzzification. Thus analytically, it is beneficial to remove these values from ψ_I based on a suitable threshold.

Considering all elements of ψ_I is greater than threshold τ, ${\psi_I}^T$ can be computed as

$$\begin{aligned} {\psi_I}^T = \{d_{jq} | (d_{jq} \in \psi_I, \forall j \rightarrow [1, O_l], q \rightarrow [1, N]) \\ \text{and } d_{jq} > \tau\} \end{aligned} \tag{2.19}$$

This implies,

$$d_{jq} \notin \psi_I \forall d_{jq} < \tau$$

where, d_{jq} represents individual elements of matrix ${\psi_I}^T$. As the elements in ψ_I decreases, the row vectors in C_{R_v} and C_{R_k} decreases. The number row vectors

is equal to n_{op} and thus it can be inferred that τ is inversely proportional to the computational complexity.

Consider the following example where the rules are

$$If\ input\ I\ is\ MF_1\ then\ Output\ O\ is\ P_1$$
$$If\ input\ I\ is\ MF_2\ then\ Output\ O\ is\ P_2$$
$$If\ input\ I\ is\ MF_3\ then\ Output\ O\ is\ P_3$$

and the input space is fuzzy partitioned as in Figure 2.3.

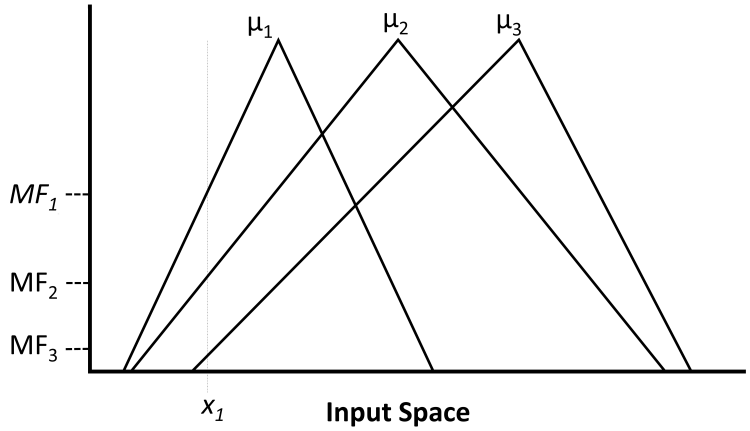


Figure 2.3: An Example: Input Membership Function

Assuming, $N_Cells = 5$ with one input and the input value being x_1, the M-FRHC output is,

$$[MF_1,\ MF_2,\ MF_3]$$

It is obvious from Figure 2.3 that, $MF_1 > MF_2 > MF_3$. Now if MF_3 is very very small, then its effect on the output is small. However, due to its firing, an additional rule is active and it gets evaluated. This evaluated rule is extremely weak as the weight it carries (from MF_3) is very small. Thus, if this rule is excluded from the computation, the accuracy of the system does not change much, but the reduction in computation is reduced by one-third(instead of 3 rules, only 2 rules are evaluated) in above example. Thus in MT-FRHC, a threshold τ is expended to discard rules which carry extremely small weight.

2.3 Defuzzification

As discussed in the earlier section, to generate a quantifiable output using fuzzy logic that can be implied in a real system, defuzzification process is obligatory. Inference engine in a FLCS will have a number of rules that transform variables into a fuzzy result described in terms of membership in fuzzy sets.

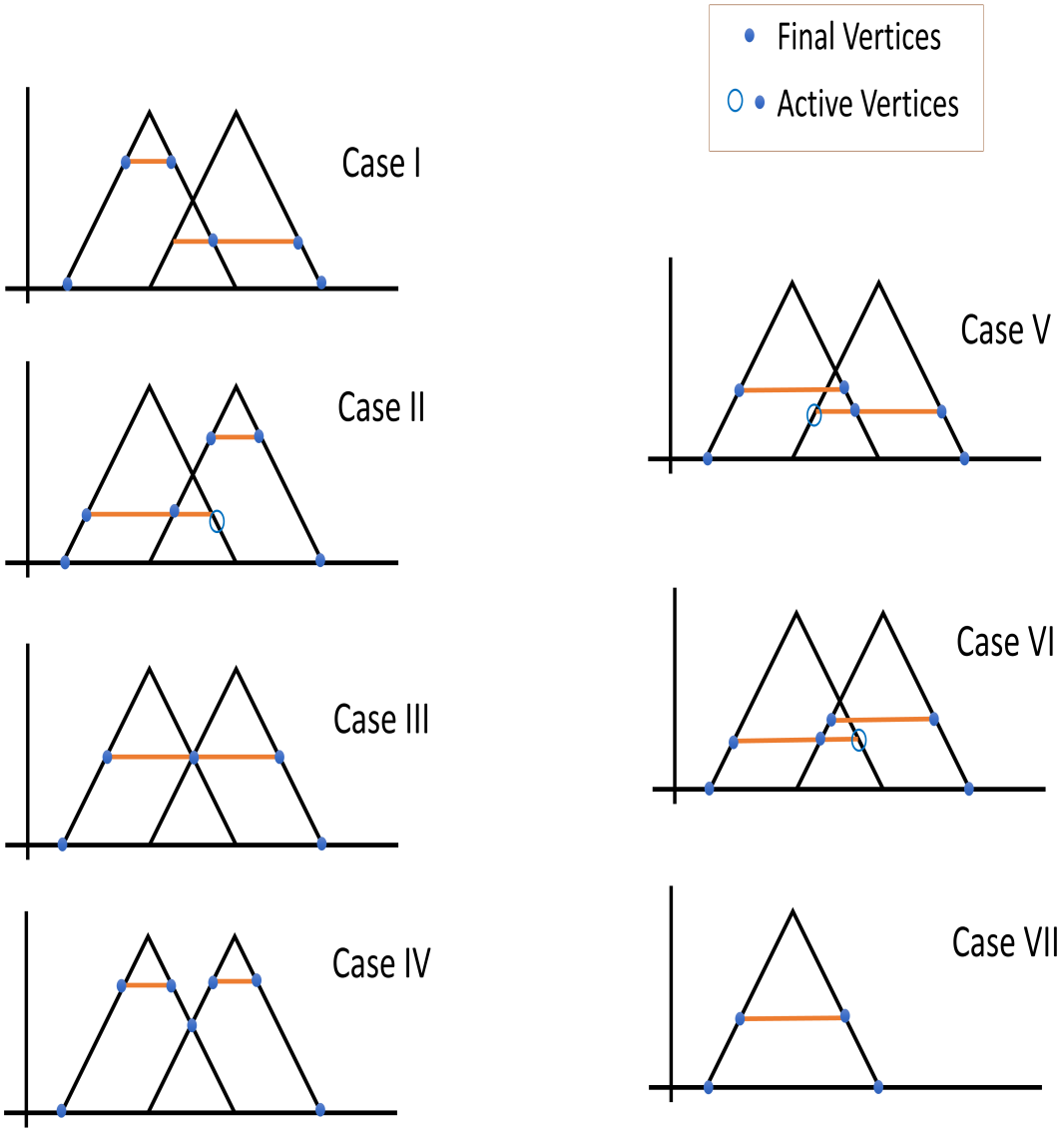


Figure 2.4: Various cases for vertices computation for Centroid of Area (COA) Defuzzification

The defuzzification process converts output expressed in fuzzy sets to crisp output using MFs. It employs certain mathematical operations to interpret the membership degrees of the fuzzy sets into a particular decision or real value. There are number of defuzzification algorithms in literature [94, 95, 153, 158]. The varied processes of defuzzification yields different crisp output. The most widely used defuzzification methods are discussed further.

2.3.1 Defuzzification Algorithms

Leekwijck et. al. [96] classified defuzzification methods broadly in to

1. Maxima methods and its derivatives,
2. Distribution methods and its derivatives,
3. Area methods, and
4. Miscellaneous methods.

Leekwijck et. al. claim that the *Maxima* methods are suitable candidates for fuzzy reasoning systems whereas the *Area* methods display the attribute of continuity that makes them appropriate for FLCs. Zavala et. al.[210] states that among numerous different defuzzification methods available in literature, most used for hardware purposes are Center of Area (CoA) which is classified under Area methods and Mean Of Maxima which is classified under Maxima methods. WA defuzzification method is the also among most frequently used defuzzification technique which is classified under Area methods. Therefore, since our target applications are fuzzy control, CoA and weighted average (WA) is ultimately chosen as the defuzzification method for proposed hardware G-FLCS.

2.3.1.1 Weighted Average Defuzzification Technique

WA defuzzification method is the most frequently used defuzzification technique for fuzzy controllers owing to its low computational complexity. It can even be easily implemented on slow processors like microcontrollers for real-time applications. But this technique can be applied to symmetrical output

MFs. The WA method is computed by weighting each output MF by its respected maximum MF value and accumulating them. This can be represented as,

$$X_{WA} = \frac{\sum \mu_{\widetilde{C}}(\overline{x}) \cdot \overline{x}}{\sum \mu_{\widetilde{C}}(\overline{x})}$$

where $\overline{x}$ represents centroid of each symmetric output MF $\widetilde{C}$. It is generally not used for G-FLCS with asymmetric output MFs [153]. Although there are many instances where this method is used for FLCs with asymmetric output MFs [176].

2.3.1.2 CoA Defuzzification Technique

CoA method is commonly known as center of gravity (CoG) defuzzification. It was developed by Sugeno in 1985[176]. CoA is most commonly used technique in fuzzy control and provides good accuracy [143, 153]. Mathematically this technique is represented as

$$X_{CoA} = \frac{\int \mu_c(x) x dx}{\int \mu_c(x) dz}$$

where $\mu_c(x)$ represents membership degree of each output MF. Continuity and computational efficiency are of utmost importance for hardware G-FLCS. In most realization of CoA, calculating the whole area and determining where its weighted midpoint is essential. As it uses all elements from input universe, it requires $k = 2^n - 1$ iterations according to number of bits (n) used for input universe. These techniques aim at reducing resource consumption and computational time without loss of accuracy. Some hardware implementation for CoA uses Center of Slice Area Average (COSAA)defuzzification technique proposed by Zavala et. al. [69, 208, 210]. COSAA uses the summation of midpoints for all $\alpha-$ levels instead of integrating the area under a curve.

2.3.2 Vertices based Center of Area (VBCoA) Computation

The existing techniques for defuzzification using CoA have been seen to be computationally time-consuming. One of the most widely used technique is the

Riemann sum based CoA computation. Riemann integral is used for deriving the centroid of area in actual method [96, 143]. Centroid of a polygon can also be computed using their vertices and has been widely used in geospatial applications [174]. This feature is used in the proposed VBCoA defuzzification method. To reduce the defuzzification time, a new vertices based CoA (VBCoA) computation method is proposed. The proposed method can be implemented in following steps.

Step 1 Generate cut set matrix from all cut points with non-zero fuzzy output values.

Step 2 Use cut set to segregate into any one of cases as presented in Figure 2.4.

Step 3 Generate intersecting matrix. Intersecting matrix includes intersecting points of output MF.

Step 4 Generate individual set of vertices from intersecting matrix and cut set matrix for various case structures as in Figure 2.4.

Step 5 Centroid on X-axis can be calculated as

$$COA = \frac{1}{6A}\sum_{i=0}^{n-1}(y_i + y_{i+1})(x_i y_{i+1} - x_{i+1} y_i)$$
$$A = \frac{1}{2}\sum_{i=0}^{n-1}(x_i y_{i+1} - x_{i+1} y_i)$$

where $n \rightarrow$number of vertices, $x_i \rightarrow x$ co-ordinate of i^{th} vertex, $y_i \rightarrow y$ co-ordinate of i^{th} vertex

The steps 1 through 5 can be used to defuzzify fuzzy output θ_f using COA process in a fast and efficient manner[1]. The proposed VBCoA defuzzification algorithm and the traditional Riemann sum based defuzzification algorithm were implemented on a C6748 DSP processor with 300 MHz operating frequency. To analyze the efficacy of the proposed algorithm, computation time using the

[1]Download Code Here: https://goo.gl/83bVna

VBCoA were compared to existing Riemann sum based CoA computation technique. A random set of fuzzy output was generated and defuzzified using these two methods. The process was repeated for five times and he observed cycle time is tabulated in Table 2.3. The randomly generated fuzzy output set appears in the first row of the table. The next two rows shows the consumed cycles and the cycle time (in μs) respectively for traditional Riemann sum based CoA computation. The final two rows shows the consumed cycles and the cycle time (in μs) respectively for the proposed VBCoA computation. Table 2.3 reflects that the proposed VBCoA technique provides a slightly better performance in terms of computational time. VBCoA shows approximately 30% improvement in the cycle time.

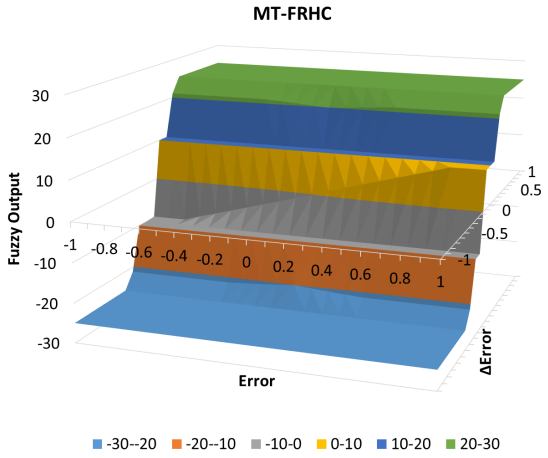
Table 2.3: Centroid computation on C6748 DSP Hardware

	Riemann sum		**VBCoA** Method	
	Cycles	Time(μs)	Cycles	Time(μs)
[0.2,0.5,0.3]	7662	25.54	5316	17.72
[0.4,0.3,0.1]	7681	25.60	5320	17.73
[0.8,0.8,0.6]	7528	25.54	5305	17.68
[0.1,0.7,0.1]	7650	25.50	5316	17.72
[0.3,0.2,0.9]	7677	25.59	5309	17.70

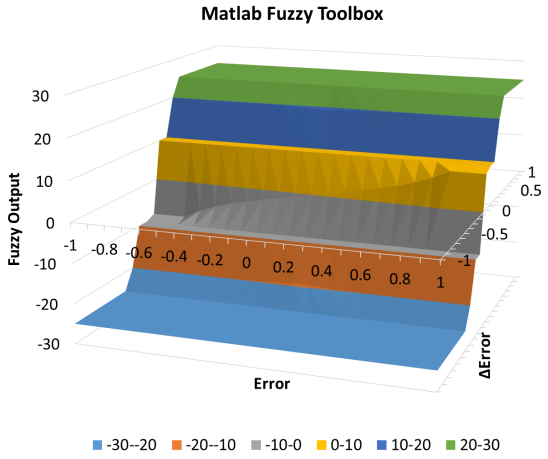
2.4 Performance Analysis

In this section, the proposed MT-FRHC based G-FLCS analyzed. It is important to analyze the designed methodology on existing FLCS designs. Matlab Fuzzy Logic Toolbox (FLT) have been widely used to develop many fuzzy control applications. Every Matlab FLT uses a fuzzy parameter file called as *fis* file which stores the FCP. In this analysis, the performance of the proposed MT-FRHC based G-FLCS is compared to Matlab FLT.

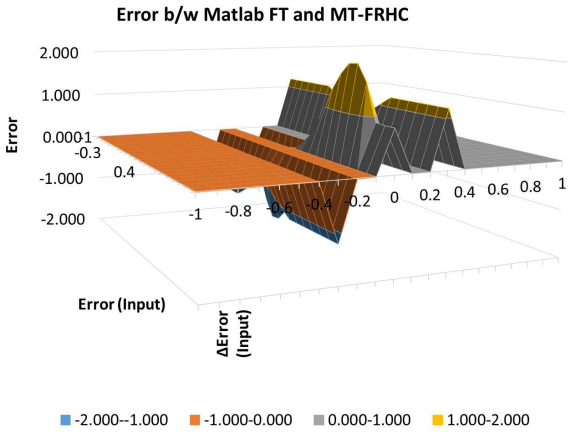
To analyze the performance of the proposed system, it was implemented on a test problem. Shiva Malla [170] used Matlab FLT to design a Fuzzy PI approximate controller for speed control of an Armature Controlled Direct Current



(a) Fuzzy PI Approximation FIS: MT-FRHC

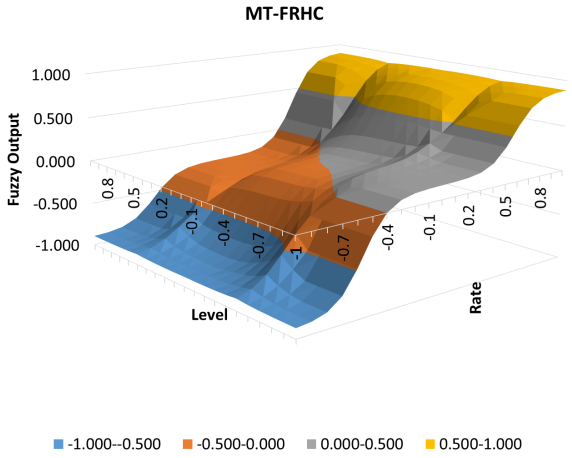


(b) Fuzzy PI Approximation FIS: Matlab Fuzzy Toolbox

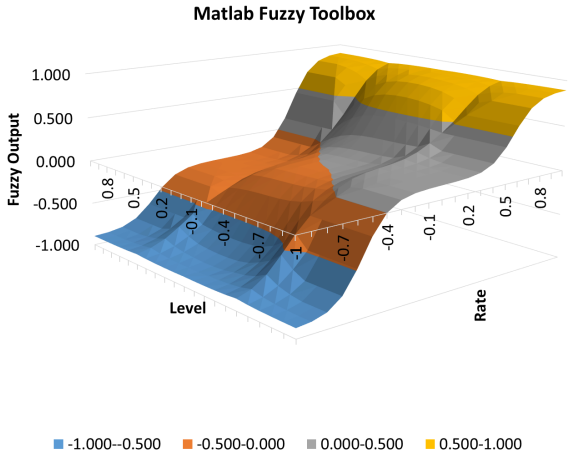


(c) Fuzzy PI Approximation FIS: Error Plot

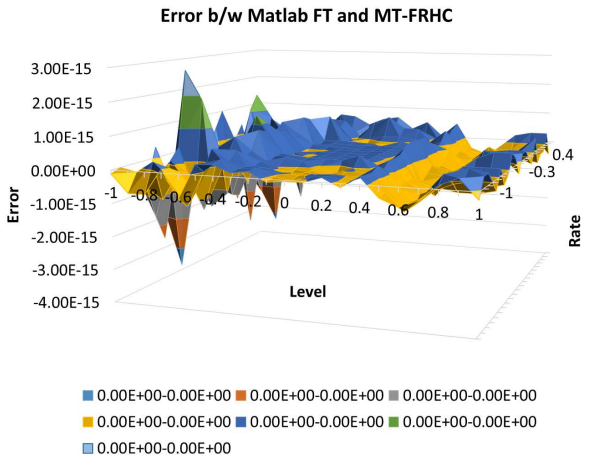
Figure 2.5: Surface Plot to test Fuzzy Inference Parameter for Fuzzy Inference Structure (FIS) used in Fuzzy PI approximation controller for ACDC motor control [170]



(a) Two Tank FIS: MT-FRHC

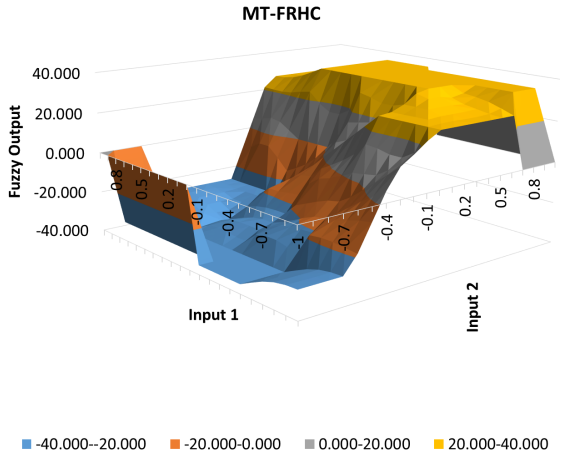


(b) Two Tank FIS: Matlab Fuzzy Toolbox

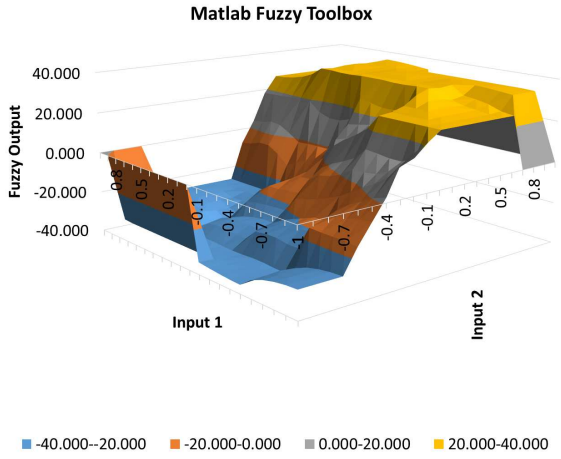


(c) Two Tank FIS: Error Plot

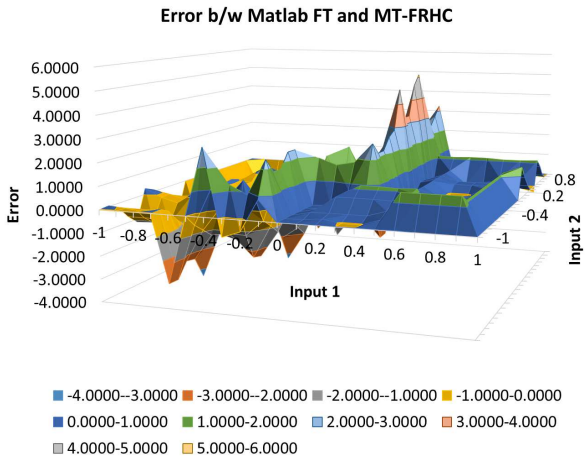
Figure 2.6: Surface Plot to test Fuzzy Inference Parameter for Fuzzy Inference Structure (FIS) used in Fuzzy PI approximation controller for Two Tank System [112]



(a) Truck Backer FIS: MT-FRHC



(b) Truck Backer FIS: Matlab Fuzzy Toolbox



(c) Truck Backer FIS: Error Plot

Figure 2.7: Surface Plot to test Fuzzy Inference Parameter for Fuzzy Inference Structure (FIS) used in Fuzzy PI approximation controller for Truck Backer Control [142]

(ACDC) motor. A web link to a FIS structure file used by Shiva Malla [170] for Fuzzy PI approximation to control ACDC motor is provided in Appendix-A. The Fuzzy PI Controller is a two-input one-output system with error and change in error ($\delta Error$) as the input and control signal as the output. The range of both the input space varies from -1 to 1. he output ranges from -30 to 30. All possible combinations inputs with a step of 0.001 are sequentially introduced into the MT-FRHC based G-FLCS, and the correspond results are noted for threshold $\tau = 0.2$. A surface plot is generated from these observed results as shown in Figure 2.5a. The x-axis represents the error ranging from 0 to 1 with a step size of 0.001, while the y-axis shows the change in error($\delta Error$) also ranging from 0 to 1 with a step size of 0.001. The output of the G-FLCS is plotted in the z-axis. Similarly, a surface plot is generated using Matlab FLT using the same FCP. This is presented in Figure 2.5b. The individual error between these two system output is calculated and plotted in Figure 2.5c. The x and the y- axes, which represents the inputs remains same as in the other plots. However, the z-axis represents the absolute error between the system output generated from the proposed MT-FRHC based G-FLCS system and Matlab FLT. Considering Matlab MLT as the true value, it can be observed MT-FRHC based G-FLCS produces a peak error of ± 2 over an output range of ± 30. At this moment, the proposed system produces a maximum of 6.67% error.

The generality of a G-FLCS can only be established once it is implemented on different FLCS designs. The proposed MT-FRHC based G-FLCS controller is tested similarly with two other benchmark control problems, namely, a water level control of a two tank system [112] and a truck backer control system[142]. Figure 2.6a shows the surface plot for output (on z-axis) of the MT-FRHC based G-FLCS system with respect to the water level (in x-axis) and input flow rate (y-axis). Figure 2.6b represents the surface plot for output (on z-axis) of the Matlab FLT system with respect to the water level (in x-axis) and input flow rate (y-axis). The surface plot of the error between these two systems is presented in Figure 2.6c. It can be seen that for this particular example, MT-FRHC based G-FLCS produces a peak error of $\pm 3 \times 10^{-15}$ over an output range of ± 1. Now, the proposed system produces a negligibly small error that is very close to 0.

In Figure 2.7a, the surface plot for fuzzy output (on the z-axis) of the MT-FRHC based G-FLCS system is plotted for a truck backer system. Figure 2.7b represents the surface plot for fuzzy output (on the z-axis) of the Matlab FLT system for the same system. The surface plot of the error between these two systems is presented in Figure 2.7c. It can be seen in error surface plot that, MT-FRHC based G-FLCS produces a peak error of ± 5 over an output range of ± 40. At this moment, the proposed system produces an error of 1.25%.

Following the previous analysis, the same benchmark control problems were employed to draw dependency between the threshold τ and mean square error. It is analytically evident that MSE is proportional to the threshold τ. However, the computational complexity of G-FLCS is inversely proportional to τ. In an ideal scenario, a G-FLCS encounters following optimization problem.

- Lower value of τ to reduce MSE.
- Higher value of τ to reduce computational complexity.

For MT-FRHC based G-FLCS, MSE is calculated with respect to Matlab FLT against different threshold values. In Figure 2.8, the trend of MSE with increase in τ is presented for Fuzzy PI Control of ACDC motor [170], a water level control of a two tank system [112] and a truck backer control system[142]. It can be observed that MSE of a system is variedly dependent on different FIS structure file at the constant threshold. Thereby, it can be safely inferred that τ has to be dynamically assigned to a G-FLCS depending on the system it is employed, and this factor cannot be generalized. As a result of this, selection of τ should be based on

- accuracy demand from the system
- throughput time of the system

2.5 Proposed MT-FRHC based G-FLCS Implementation and its Validation

At this juncture, it is necessary to implement the proposed algorithm on a programmable device and validate the results. To achieve this, the proposed sys-

tem is implemented on a TI TMS320C6748 DSP processor operating on 300 MHz with $\tau = 0.2$. For comparative analysis of the MT-FRHC based G-FLCS technique, the Overlapping Membership Function (OMF) based G-FLCS is also implemented on the same platform. A 4-input/1-output, 9 rule, COA defuzzification FLC was used for testing. A download link to this FCP is provided in Appendix-A. The timing analysis of this experiment conducted is presented at Table 2.4. Timing analysis of this experiment was executed using Code Composer Studio Timing Profiler. This tool presented the machine cycles consumed during execution of the method.

Table 2.4: Hardware Implementation: Timing Analysis

Sl.	Cycles	Time (ms)	FLIPS
1.	43370	0.1445	6920.42
2.	38546	0.1285	7782.10
3.	38451	0.1282	7800.31
4.	38182	0.1273	7853.6
5.	37567	0.1282	7800.31
6.	30944	0.1032	9689.5
7.	38995	0.13	7692.3
8.	38513	0.1286	7788.2
9.	38595	0.1287	7770.00
10.	38578	0.1286	7776.05

From the Table 2.5 it can be observed that the execution time for MT-FRHC is 0.0259 ms per inferences compared to Overlapping Membership Function (OMF) with 0.0346 ms per inference. This implies that the number of fuzzy logic inferences per second (FLIPS) completed by the proposed technique is quite higher. It has been found that the MT-FRHC based G-FLCS achieved 27 % higher performance in terms of speed compared to the OMF based G-FLCS. Table 2.5 presents the timing analysis of the individual modules in the G-FLCS. It can be seen that fuzzifier consumes significant machine cycles in comparison to other modules. This is mostly because the code has been written in a sequential manner, and there is a single thread of fuzzifier that fuzzifies crisp data from all four input channels to fuzzy data sequentially. The cycle time for fuzzifier can be significantly improved by invoking multiple threads

using TI Sys/Bios[1] .

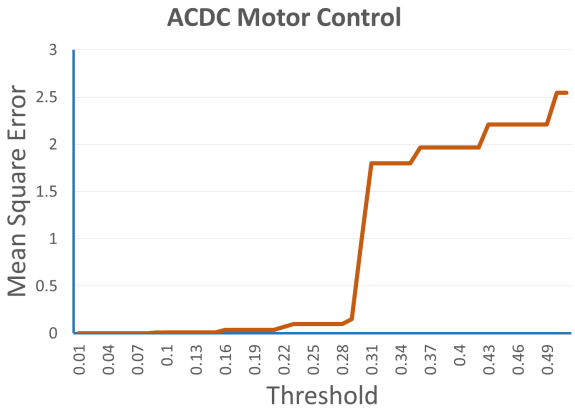
Table 2.5: Hardware Implementation: Average Time Response

	Fuzzifier	Inference	Defuzzifier	Total	Time	FLIPS
	(Cycles)				(ms)	(Kilo)
MT-FRHC	29393	1823	6332	38548	0.1285	7.8
OMF	41843	2198	8565	52606	0.1754	5.7

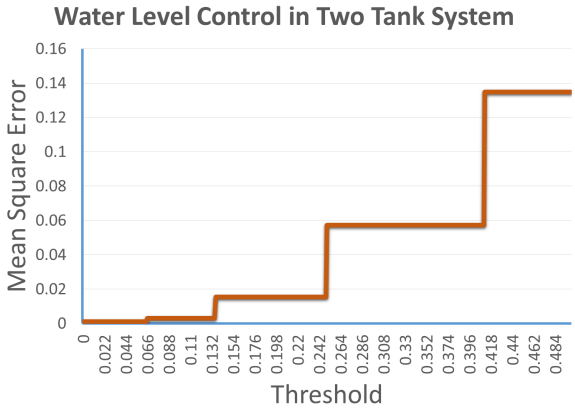
2.6 Summary

In this chapter, a theoretical analysis of M-FRHC and MT-FRHC has been elaborated which was aimed at countering the significant limitations of predominant FRHC technique. Simulation results indicate that the proposed scheme can be implemented on hardware G-FLCS and replace the prevailing FRHC as rule reduction technique. This algorithm has managed to improve controller accuracy with a significant decrease in the computational complexity. This analysis portrays promising results in favour of MT-FRHC in both fronts and its utility in G-FLC design is dominantly inferred. This method also provides a platform where users can vary the number of overlaps and threshold value for fuzzifier of a G-FLCS in real-time. It has also been shown that with varying number of overlapping membership functions, the number of operations in the inference engine can be controlled. The introduction of threshold in fuzzifier will eradicate insignificant computations in the system. These programmable features in MT-FRHC provide added control on the performance of the controller itself and a user can steer the MT-FRHC based G-FLCS to the performance with better efficiency.

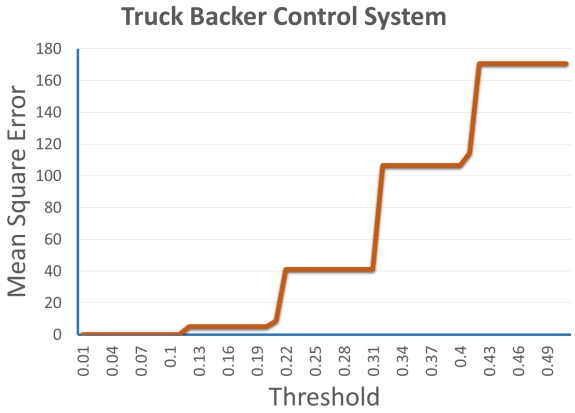
[1]TI Sys/Bios is a real-time operating system primarily developed for TI manufactured DSPs, ARM and other programmable devices.



(a) Fuzzy PI Approximation to Control ACDC Motor



(b) Water Level Control of Two Tank System



(c) Truck Backer Control System

Figure 2.8: Dependency of MSE on threshold introduced in MT-FRHC Rule reduction technique for FIS structure file employed in FLC to control various systems. These systems considered are two input one output systems, no. of operations per inference is 16.

Chapter 3

System Architecture for MT-FRHC based G-FLCS

Preview

In this chapter we present a G-FLCS architecture and its necessary modules and submodules. The chapter also provides a detailed explanation of a web based user interface for the proposed MT-FRHC based G-FLCS for reconfigurability. This hardware based fuzzy framework provides online reconfigurability of the system from remote location. However, it is important to note that there are over sixty flexible parameters that needs to be configured and it becomes an arduous task for an user to manage and configure them. Therefore, in this chapter we also introduce a genetic algorithm based parameter extraction technique. This technique helps to develop a course tuning and provide startup parameters which can be later fine tuned by the users remotely through the Web based User Interface.

3.1 Introduction

In previous chapter, a MT-FRHC based Type-I Mamdani G-FLCS system is proposed and described. As discussed briefly in Chapter 2, to counter uncertainty and variations in a nonlinear plant, adaptive control is extremely important. The adaptive control strategy maintains smooth performance of a system in the presence of these uncertainties by updating and tuning FCP at various time intervals. This is implemented in the proposed G-FLCS architecture by introduction of online and offline tuning methods. However, a theoretical analysis of the proposed MT-FRHC rule reduction technique is essential before implementation of the tuning methods because the tuning parameters are strictly driven by the active rule reduction mechanism. Subsequently, these modules and submodules gets integrated with G-FLCS and analyzed after implementation on a DSP hardware. Earlier in section 1.8 it was discussed that *Plug and Play Framework* and *Runtime Tunability* is the essence of G-FLCS design. To achieve these key features, it is important to integrate an interactive user interface with the G-FLCS design. It is also imperative to formulate the system parameters in accordance to which the G-FLCS is to be designed. There are large number of parameters in a Fuzzy Logic Controller. However, a truly generic fuzzy system is impractical because of the fixed resources for implementation. Therefore it is necessary to put an upper bound to each flexible parameter in the Fuzzy Control Parameters (FCP). Nevertheless, unlike most existing G-FLCS designs, in this design the number of flexible parameters are not compromised to gain a higher speed. Rather other areas like optimization of code and algorithm is exploited to attain a desired speed and performance. It is also important to note that the FCP driving a G-FLCS are generally programmed by users. There are large number of fuzzy parameters for an user to input in the G-FLCS. This makes the process quite cumbersome. Hereby, a technique for automatic fuzzy parameter extraction from input-output relationship of a process plant will grant the necessary ease of operation for an user. Therefore to start with the design of G-FLCS, the system parameters are specified first.

3.2 G-FLCS Parameters

To start with the design of G-FLCS, the system parameters are specified first. The FCP of proposed G-FLCS is segregated into two segments namely configurable parameters and fixed parameters. Table 3.1 presents the features of the proposed G-FLCS. This table presents configurable and non-configurable (fixed) parameters along with order and variable values for a four-input and one-output system. Characteristics features of this architecture is as follows:

- A 32 bit precision Input and Output is considered.
- A web based user interface (WebUI) to remotely acquire fuzzy parameters.
- Wide range of output MFs tuning consisting of singleton, triangular, trapezoidal, Gaussian and GBell type membership functions.
- MIN-MAX Inference with full set rules.

This controller was implemented on a TI C6748 DSP processor. The major reasons for using DSP are:

- DSP provides an effective implementation of multiplication and accumulation (MAC) and this helps in efficient COA implementation.
- File handling and socket programming is an integral part of this design. These were achieved easily since the development is in C language.

Table 3.1: System Parameters

Parameters		Values
Configurable	System Input	1 to 4 (32 bits)
	number of MFs per I/O	2 to 7
	Shape of MFs	Triangular, GBell, Trapezoidal, Gaussian
	Defuzzification Method	Centroid of Area
	Rulebase	2401 (7^4)
Fixed	System Output	1
	Method	MIN-MAX
	I/O Signal Range	-5 to +5 V

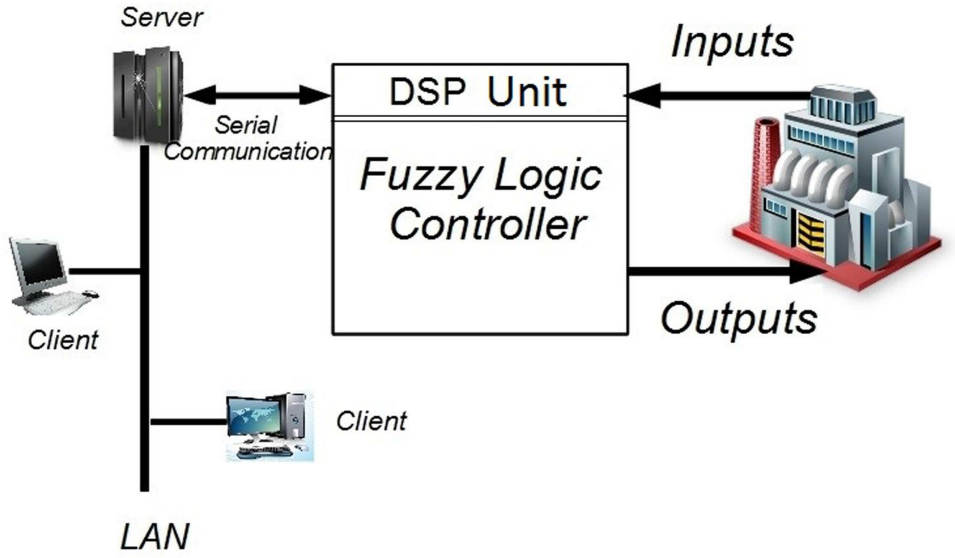
- This design supports high level of branching and decision making.

3.3 System Architecture of Proposed G-FLCS

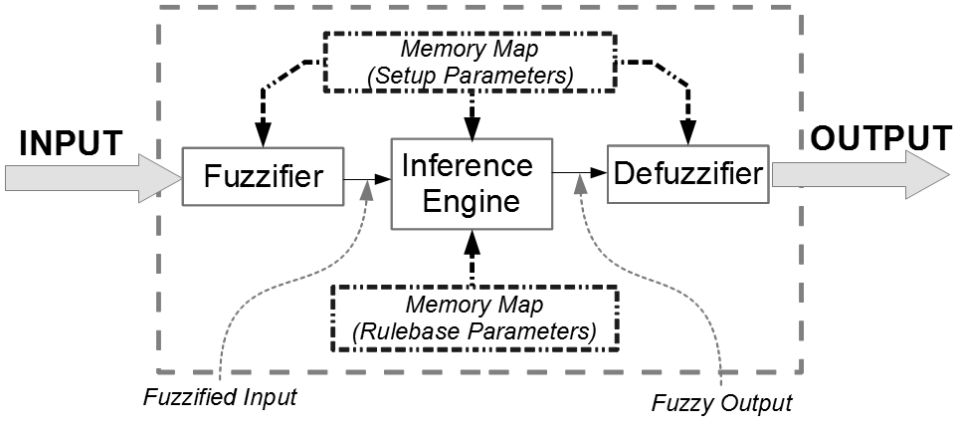
The proposed system architecture involves hardware-software co-design to present a complete reconfigurable G-FLCS as shown in Figure 3.1a and Figure 3.1b. In Figure 3.1a, a graphical abstract of the overall system is represented. It can be observed that the controller is interfaced on one side in a feedback loop with the plant and on other side with the users through a client-server based interface. The client-server model allows reconfigurability of the system from remote locations. Figure 3.1b presents the internal architecture of the G-FLCS design.

The WebUI in client server model represents the software and the driver layer to interface the hardware G-FLCS through serial port to the server. The DSP hardware receives FCP data serially and stores them in predefined memory locations. These parameters are segregated in two categories namely *Setup* parameters and *Rulebase* data. The driver layer in the hardware G-FLCS receives and acknowledges the data transmission. A WebUI drives the proposed hardware G-FLCS storing FCP data in a file in specific format. The file is shown in Appendix-A. It may be noted that the FCP file is generated deliberately in accordance with Matlab Fuzzy Inference System (FIS) file format to provide liberty to integrate a parameter data file generated using Matlab Fuzzy Logic Toolbox as well. Submitting the parameters triggers a desktop application that is native to the server. Objective of this program is to transmit the FCP from database to the hardware G-FLCS through serial communication. The FCP data is stored in the DSP board based according to a predefined memory map. G-FLCS designed with MT-FRHC rule reduction scheme, operates on the inputs with these FCP data to provide desired control action. Fuzzifier, inference engine and defuzzifier is programmed with information about the data and its corresponding memory location. With these information, fuzzifier transforms crisp inputs in fuzzy domain and a Mamdani inference engine, coupled with the Rulebase in the system memory produces fuzzy output. Defuzzifier transforms the fuzzy output compiled by the inference engine to provide crisp output. There are various types of defuzzifier but in this work centroid of area (COA) is

used considering its popularity and effectiveness.



(a) System Architecture



(b) Hardware Design

Figure 3.1: Proposed G-FLCS Design Architecture

3.4 Development of a Client-Server Model User Interface

A client-server model is widely used for data transfer over large distance. Computer applications like are Email, network printing, and the World Wide Web use clientâĂŞserver model for data exchange. It is reliable, fast and secured distributed application structure. In the proposed design, it is cardinal to have a distribution structure over which users from large distance can operate the controller.

3.4.1 client-server Model

The client-server model, also known as server-client model, is a computing model developed on distributed computing structure. This structure segregates workload between a centrally connected service provider (*Server*) and the various service users (*Clients*) [169]. Its is a standard model for developing network applications. The *Server* process starts the computing module by initializing itself. Once the initialization protocol successfully completes, it goes to sleep and waits for incoming client requests. *Client* processes can initiate from any system in the network including the host system running the *Server* process. Once a service request is attended, this *Server* application goes back to sleep and awaits incoming requests. Server applications or processes are of two types.

- **Iterative:** In these type of applications, a single copy of the server application executes to service only a single user at any time. While a user request is attended, other users have to wait. These applications are developed when prior knowledge about the time to service individual request is present.

- **Concurrent:** These *Server* applications provides service to multiple users at any time. When a user requests for a service, the *Server* application replicates a copy of itself which dedicatedly serves that user. This process repeats for every incoming service requests from the *Clients*. These applications are developed when prior knowledge about the time to service

Table 3.2: TCP/IP Communication Layers and their Protocols

SL. No.	Layer	Protocol
1.	Data Link Layer	Ethernet
2.	Network Layer	IP
3.	Transport Layer	TCP
4.	Application Layer	HTTP

individual request is absent.

The client-server model of communication developed for hardware G-FLCS uses TCP. TCP is an important protocol of TCP/IP networks where it helps to establish connection and exchange data. Establishment of a connection between two hosts depends upon five components

- Protocol used
- Source IP address
- Source port number
- Destination IP address
- Destination port number

With proper set of information, a connection between two hosts is established.

3.4.2 ASP.NET and development of WebUI

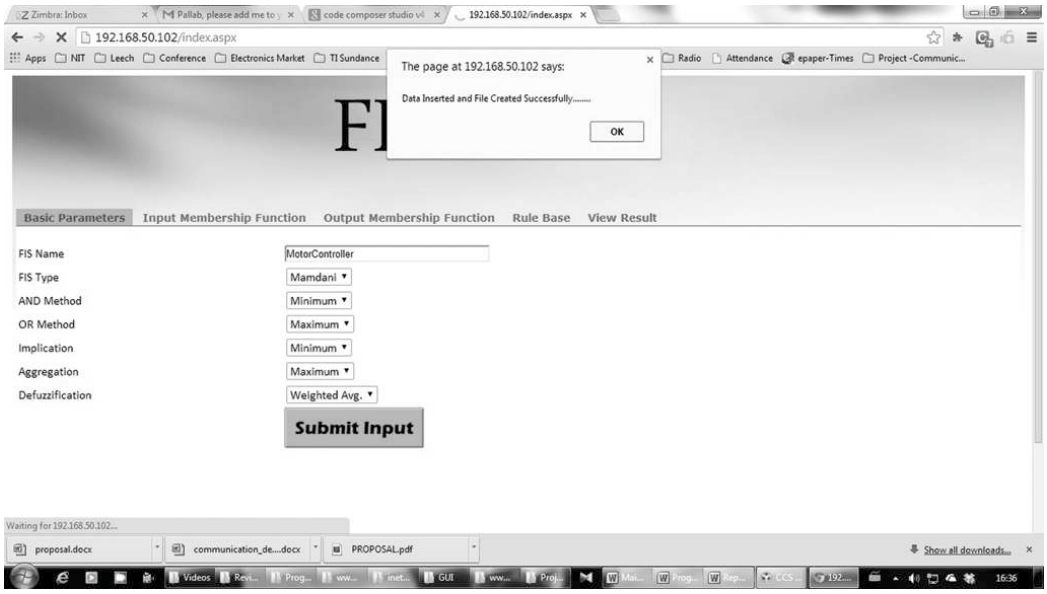
In this work, ASP.NET has been extensively used to develop the WebUI for hardware G-FLCS. This web application is designed with an intention to develop an user interface which, once installed on a central server, can be accessed over Internet to provide remote reconfigurability to the proposed hardware G-FLCS. The framework behind this application is demonstrated pictorially in Figure 3.2. The figure shows that Microsoft IIS7 hosts and web application and interfaces it to the internet. Users from different location can access this application. Standard IP protocols are used for the WebUI for hardware G-FLCS as stated in Table 3.2 Here the WEBUI is hosted on a PC with IP address

192.168.50.102. Using HTTP, users can access the WebUI over Internet upon request and authentication. Since this application is time critical and sensitive to FCP data, it is important to develop it as an iterative server application which automatically limits user to one. Microsoft IIS7 has been used for webserver and it can host web applications developed on ASP.NET framework. This was a major motivation of using ASP.NET for developing this application. A user request is processed by IIS and it passes the request to the web application WebUI, which generates response in HTML and returns it to IIS. Thereafter, IIS returns this HTML response to the user initialing the request.

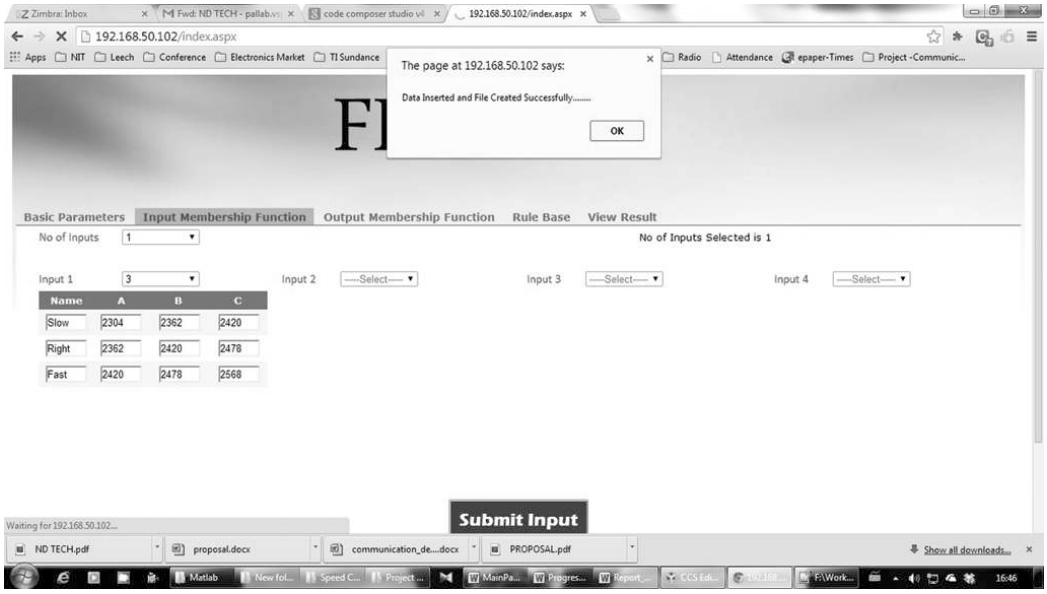
Figure 3.2: Framework behind WebUI for hardware G-FLCS

3.4.3 WebUI for Hardware G-FLCS

WebUI is developed in ASP.NET with C# and deployed using Microsoft IIS-7. The Web Pages that serves to collect various information pertaining to parameters related to FLC from users are shown in Figure 3.3 and Figure 3.4. Web page in Figure 3.3a presents the WebUI where, parameters like *name*, *type*, *implication*, *aggregation*, *and* method, *or* method and *defuzzification* type are defined. Web page in Figure 3.3b accepts details of inputs and their MFs whereas Web page in Figure 3.4a accepts details about outputs and their corresponding MFs. Web page in Figure 3.4b accepts the Rulebase and stores all data in FCP file as mentioned in Appendix. On submission, these information data are val-

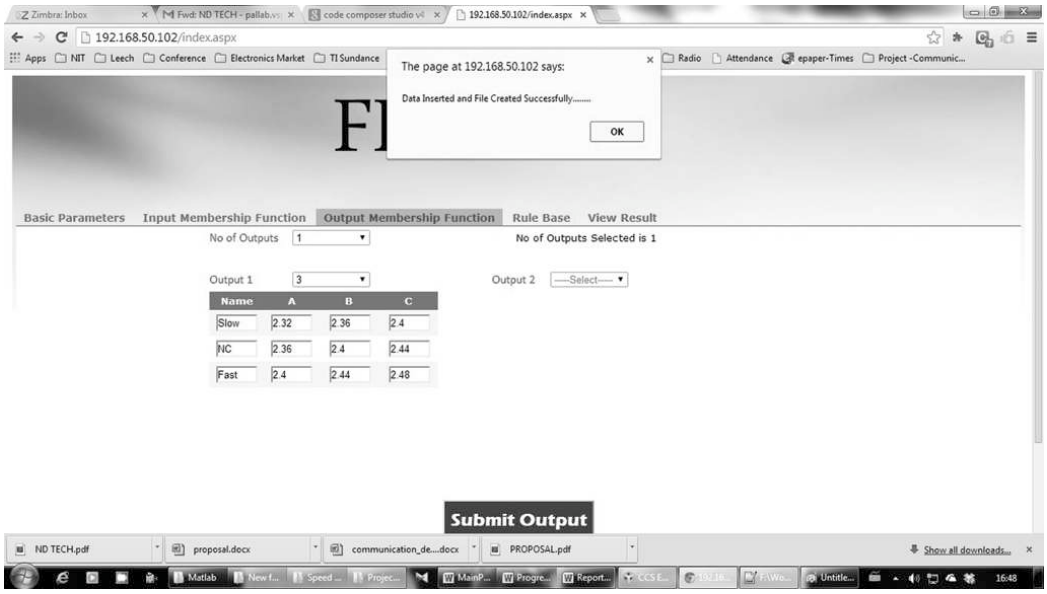


(a) Basic Parameters

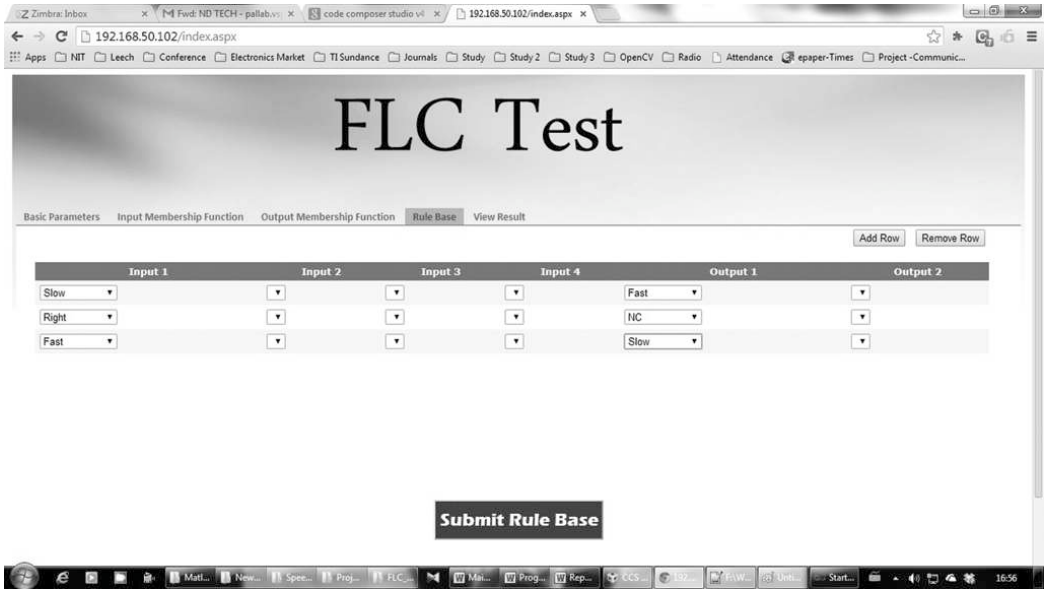


(b) Input Membership

Figure 3.3: WebUI for Hardware G-FLCS developed using ASP.NET with C# and hosted using Microsoft IIS7



(a) Output Membership



(b) Rulebase

Figure 3.4: WebUI for Hardware G-FLCS developed using ASP.NET with C# and hosted using Microsoft IIS7

Table 3.3: Genetic Algorithm Parameters

Parameters	Value/Function
Population	120
Generation	200
Fitness Scaling	Rank
Selection	Stochastic
Crossover Probability	0.8
Elite Count	6
Mutation Function	Adaptive feasible
Crossover Function	Adaptive feasible
Stopping Criteria	Fitness Limit

idated and a server application for communication with the board is invoked. This WebUI can be accessed over Ethernet from a client system situated far away from the controller and plant (simulated model).

3.5 Genetic Algorithm based Fuzzy Parameter Extraction

A genetic algorithm (GA) is a search heuristic that mimics the process of natural selection and has been widely used evolutionary method for optimization of FLCs, both type I and II[5, 20, 75, 108, 161, 168]. This technique has been widely used in the literature for fuzzy optimization. Other popular methods for fuzzy optimization are Univariate Marginal Distribution Algorithm (UMDA)[24], stochastic Hill Climbing(SHC) [42, 120], Baysian Optimization Algorithm (BOA) [82].

Response from a system with maximum of four inputs and one output is recorded. Once the dataset is derived, it is passed on to the system for FCP extraction. The proposed method for FCP extraction is based on Genetic Algorithm. Details of the parameters used in Genetic Algorithm is displayed in Table 3.3. There are 60 system parameters that are optimized using genetic algorithm to achieve two objectives, namely

- Quick settling, and

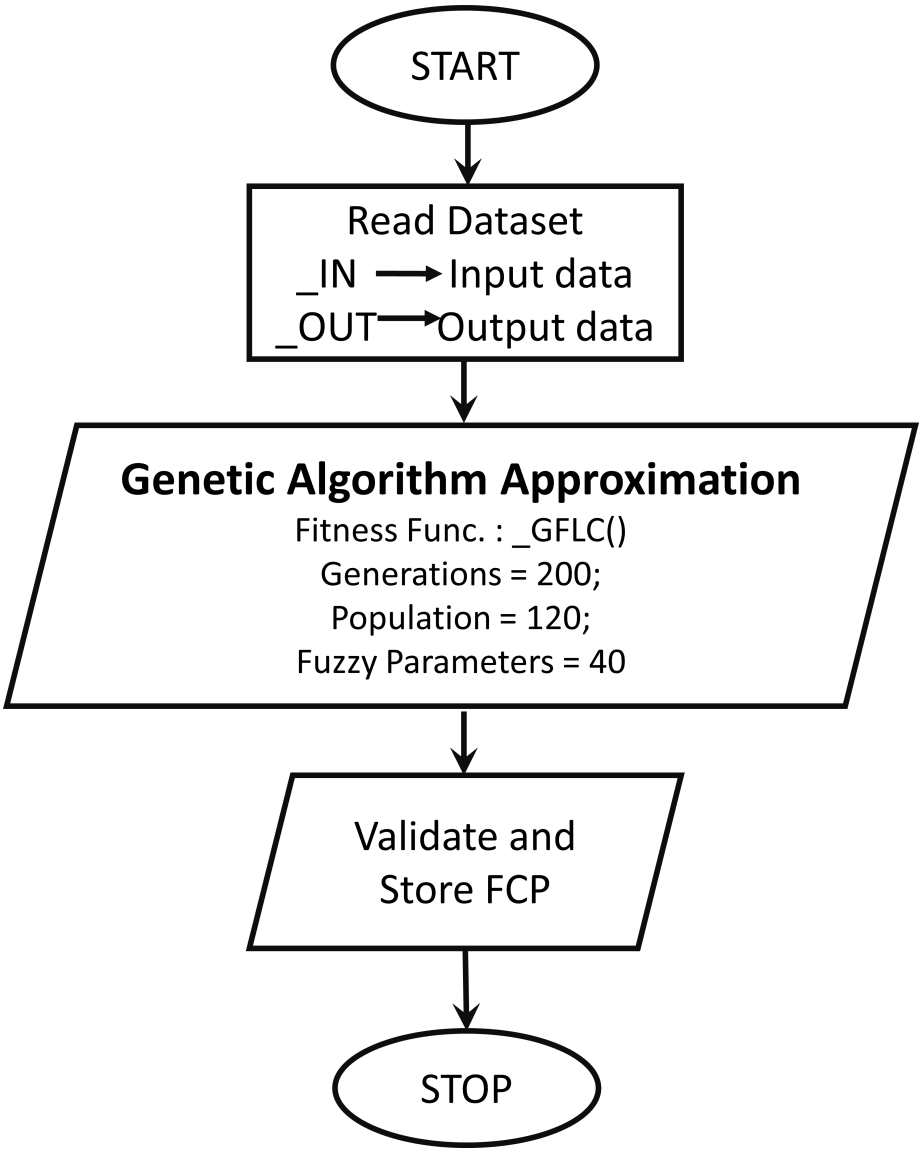


Figure 3.5: Fuzzy Control Parameter (FCP) extraction using Genetic Algorithm

- Tracking reference.

Figure 3.5 portrays basic blocks of a Genetic Algorithm based FCP extraction technique. An initial set of random parameters are input to the Genetic Algorithm system. Every optimization technique operates on a cost function. The cost function can be represented by the process plant simulation model or its I/O dataset. The FCP parameters are manipulated such that a set of optimized

FCP is obtained which provide quick settling and fast reference tracking. The Simulink model with initial FCP is used as an objective function which returns the transient and settling time. The co-ordinates of the membership functions are treated as the nonlinear inequality constraints. Thus varying these FCP will provide with an absolute tracking of reference[1].

3.6 Data flow of the proposed system

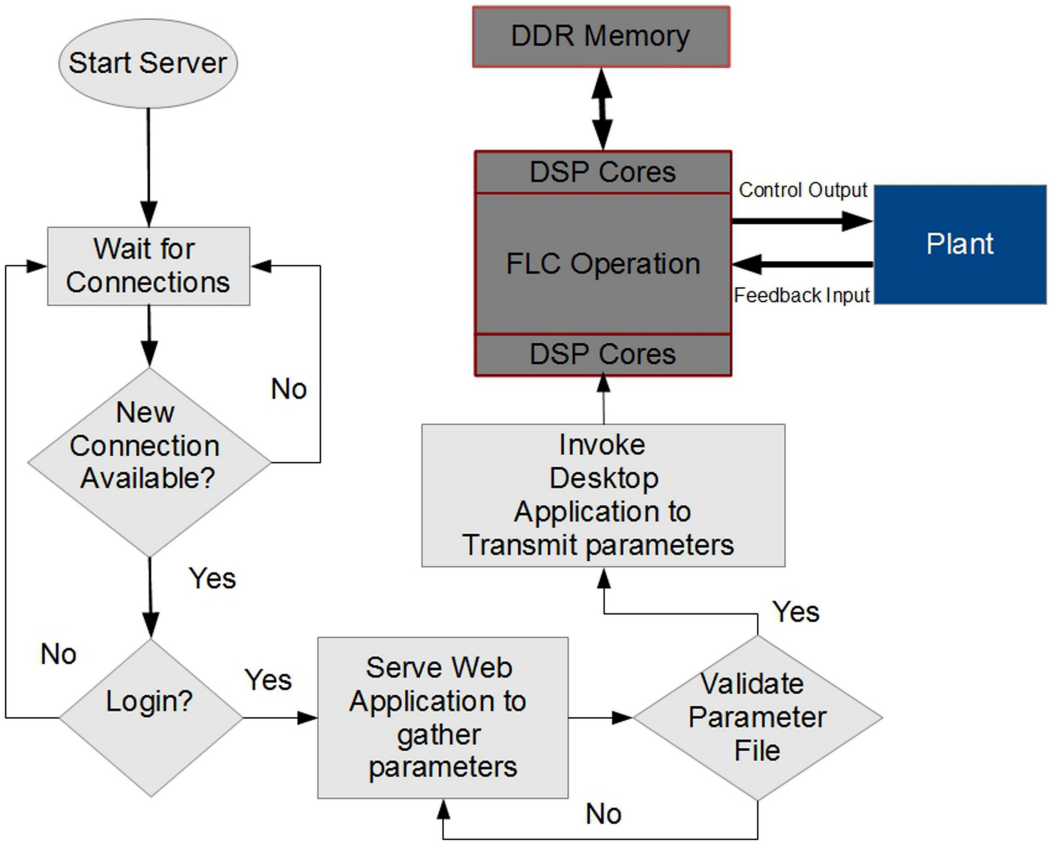


Figure 3.6: Dataflow of the proposed G-FLCS system

[1]The codes can be downloaded from https://goo.gl/Zb17fZ

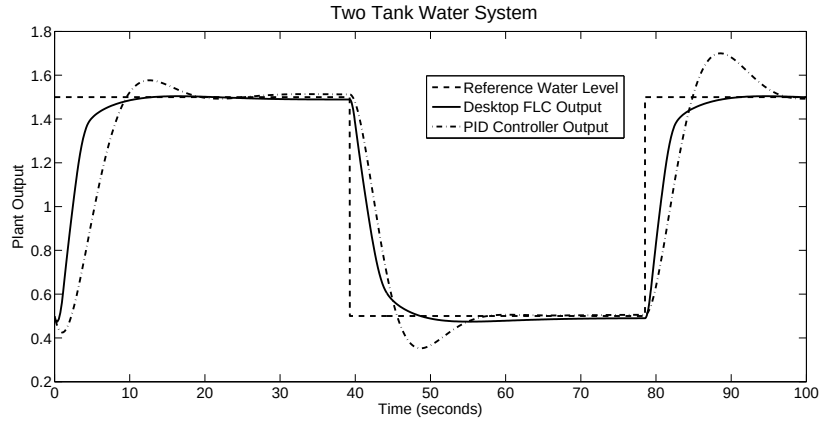
In this implementation process, data synchronization and communication between user and the hardware G-FLCS is critical. The process is presented in Figure 3.6. This figure depicts how data communication is achieved with the help of various control signals between, client-server and server-GFLCS. The web application provides a systematic user interface which collects data from authenticated users. The application waits for a new connection request on startup. On successful login, the user is presented with a web page containing four different tab-windows, each for basic parameters, information about inputs, outputs and Rulebase. These windows are preloaded with extracted parameters as presented in Figure 3.5 and provide a way for fine tuning the control parameters by an operator. In Figure 3.6, operations like connections, login, parameter collection and communication with hardware G-FLCS through a server program are performed by the web application and are shaded in *Light Grey*. When a user provides the fuzzy control parameters (FCP) data, the web application validates the entered data based on following protocols.

- All MFs have properly defined co-ordinates within specified range of operation.
- Number of Input(s) and Outputs are correctly defined.
- Rules are validated according to Mamdani model.

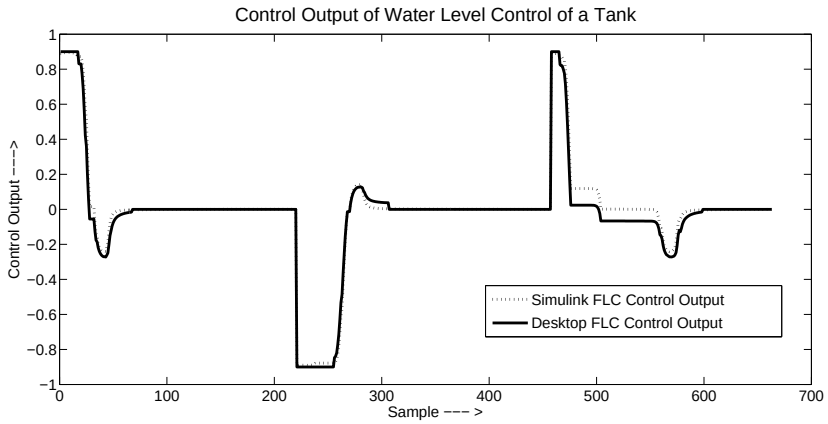
After validation of new parameters, the server application connects serially to the hardware G-FLCS. Serial communication protocol has been used here for the ease of implementation. This data communication can be easily extended to industry standard controller area network (CAN) protocol. G-FLCS completes current execution and generates control signal to the system. Thereafter, it acknowledges any incoming serial communication request and starts receiving and storing fuzzy parameters in the data memory. Sys/BIOS is widely used real time operating system for TI DSPs and has been used in this work to take care of the multitasking of fuzzy processes.

3.7 System Integrity Test

To provide proof of concept for this proposed design, an experiment was carried out with an Intel Corei5-2400 3.1 GHz PC with 4GB memory operating as a server with the WebUI. It is available to all the clients in the local network over same gateway. Authenticated user loads FCP data in a text file located in the server.

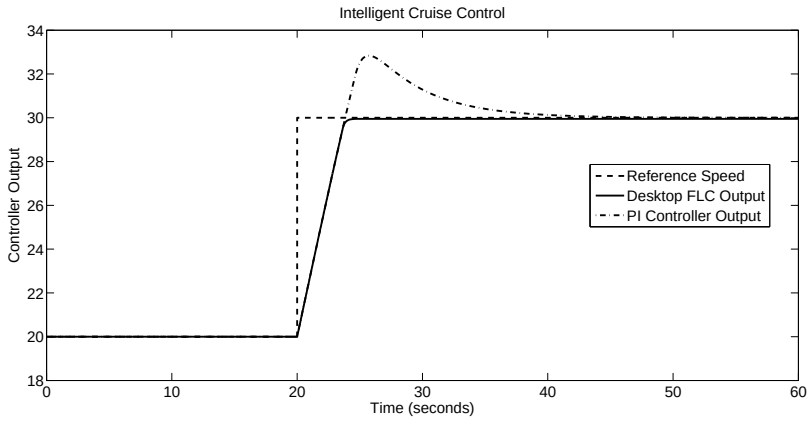


(a) Plant Output: Two Tank Water Level System

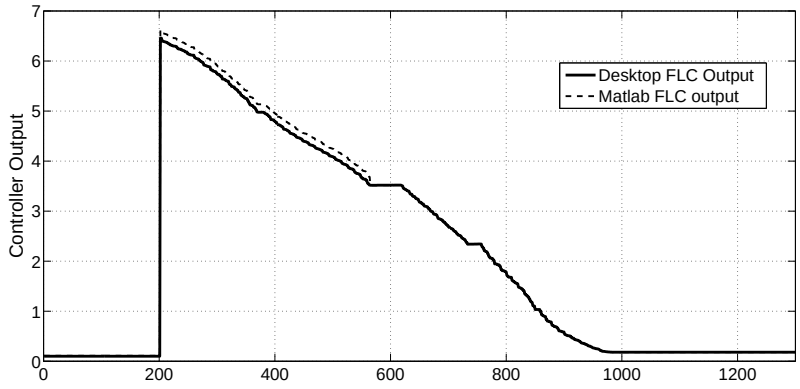


(b) Controller Output: Two Tank Water Level System

Figure 3.7: Plant output and Controller output of various test models. The controller output is a comparison between output from Matlab Fuzzy Logic Toolbox and proposed hardware G-FLCS. Plant output shows performance of the proposed FLC structure with PID controllers conducted using through HIL testing environment



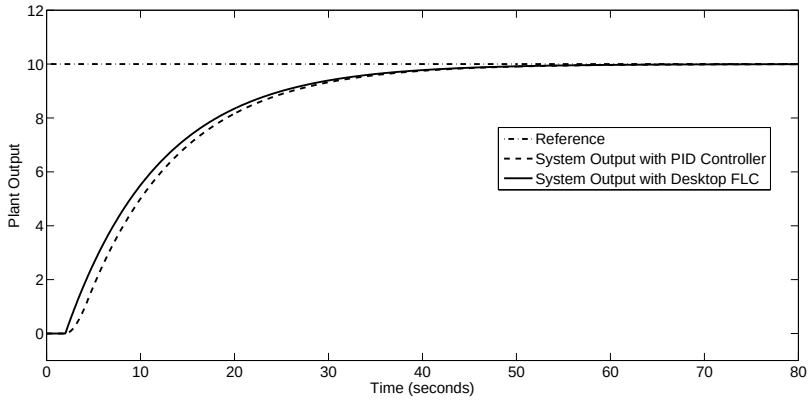
(a) Plant Output: Intelligent Cruise Control System



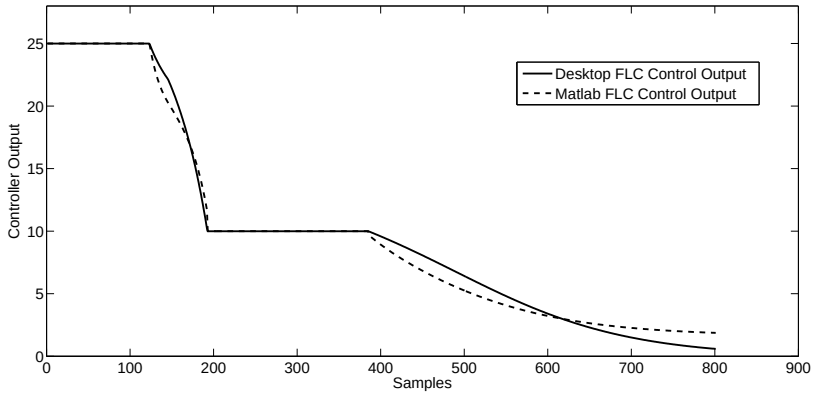
(b) Controller Output: Intelligent Cruise Control System

Figure 3.8: Plant output and Controller output of various test models. The controller output is a comparison between output from Matlab Fuzzy Logic Toolbox and proposed hardware G-FLCS. Plant output shows performance of the proposed FLC structure with PID controllers conducted using through HIL testing environment

The code developed for hardware G-FLCS is compiled using MS Visual Studio to generate a desktop application which runs on the server and mimics the hardware G-FLCS. This FCP data is used by desktop FLC to generate control signal. Plant models are developed in Matlab and communicates with the desktop FLC using *system* command. The functionality is executed in following steps:



(a) Plant Output: First-order system with dead time for anti-windup scheme



(b) Controller Output: First-order system with dead time for anti-windup scheme

Figure 3.9: Plant output and Controller output of various test models. The controller output is a comparison between output from Matlab Fuzzy Logic Toolbox and proposed hardware G-FLCS. Plant output shows performance of the proposed G-FLCS structure with PID controllers conducted through HIL test environment

Step 1 Generation of I/O dataset from Simulink model using PID controller.

Step 2 Apply GA based FCP extraction algorithm described in section 3.5, to extracted FCP from the I/O dataset generated in Step 1.

Step 3 Appropriate FCP data is programmed through WebUI and submitted to the server.

Step 4 The plant model is executed with the stored FCP. For controlling the plant, Matlab uses *system* command to invoke the desktop FLC program and passes the FCP and input data _IN as arguments. Desktop FLC computes and returns the output to Matlab. This forms the controller output.

Step 5 The Simulink plant output is computed with controller output. It is stored and plotted with respect to plant output using PID controllers for comparative analysis.

Step 6 The dataset is used to compare the control signal from desktop FLC to control signal from Matlab Fuzzy Inference System for performance analysis.

Some benchmark control problems were used to test the applicability and generality of the proposed architecture, namely Two Tank Water Level Controller [93, 112], Intelligent Cruise Control [192], first order system with dead time for anti-windup scheme [47]. All these system models were implemented on Simulink and they are available in Mathworks File Exchange repository. The integrity of the proposed architecture is tested with these models according to the algorithm described above. In Figure 3.7, Figure 3.8 and Figure 3.9 the observed results from these simulated tests are displayed. The proposed MT-FRHC based Web Configurable G-FLCS is implemented on the desktop server using C code interface by an application program to the WebUI. This provides a platform for evaluation of the proposed technique before it is realized on actual hardware platform. In Figure 3.7a, the plant output of this proposed desktop G-FLCS is compared to a tuned PID controller when applied to control a Two Tank Water Level Controller [93, 112]. The figure plots the plant response under different controllers with respect to time in seconds. It can be observed that the proposed G-FLCS provides a smooth and fast control compared to the PID controller. Similar results are observed when the proposed GFLCS and a tuned PID controller is employed to control an Intelligent Cruise Control System[192] and first order system with dead time for anti-windup scheme [47]. These figures plots the plant response under desktop G-FLCS and PID controller with

respect to time in seconds. Figure 3.8a and 3.9a shows that the G-FLCS performs better than the PID controller. Figure 3.7b, Figure 3.8b and Figure 3.9b compares the control output from the desktop G-FLCS and Matlab FLT for every sample while controlling Two Tank Water Level Controller [93, 112], Intelligent Cruise Control [192] and first order system with dead time for anti-windup scheme [47] process plants respectively. These tests indicates that the objective of generality and remote reconfigurability is achieved for proposed MT-FRHC based G-FLCS design. The results also convincingly reflects that the proposed system architecture performs satisfactorily and can be implemented on real-time. However, its real-time nature can be concluded only after conducting the timing analysis and profile. This is presented in the next chapter along with the aspects of hardware implementation.

3.8 Summary

This chapter presents a background framework for MT-FRHC based remotely tunable G-FLCS. The proposed controller can suitably replace existing controllers in a process plant which confirms the generic nature of the designed G-FLCS. The algorithm for achieving this is described in section 3.5. In this work, process control applications based on PID controller were chosen specifically. The major reason is their wide usage and acceptability in industries. By following protocols mentioned in section 3.5, other variants of industrial controllers like sliding mode and model predictive controllers can also be suitable approximated by the proposed G-FLCS. This chapter elaborates the proposed MT-FRHC based remotely tunable G-FLCS architecture with Genetic Algorithm based FCP extraction technique. The generality and applicability of the design is also tested by applying it to various benchmark control problems in a simulation environment. The results portrays a proof-of-concept for the objectives that were set in chapter 1. However, the major aspect of its implementation is yet to be evaluated. The next chapter deals with the prospect of implementation of the proposed G-FLCS and explain various schemes adopted to make the design feasible on a DSP platform.

Chapter 4

Implementation of Remotely Tunable MT-FRHC based G-FLCS with VBCoA on Programmable DSP

Preview

This chapter present a MT-FRHC based remotely tunable G-FLC implemented on a programmable DSP platform. The algorithm is ported on a single core fixed point DSP, which can be remotely configured in real-time over Ethernet. This feature of reconfigurability enables a user to change fuzzy parameters in real-time, eliminating repeated hardware programming. The scheme also eliminates the requirement of removing the controller from the process plant for configuration. A hardware software co-design architecture for the proposed generic FLC is developed on TI C6748 DSP with Sys/BIOS RTOS and seamlessly integrated with a web based user interface (WebUI) for reconfigurability. The WebUI acquires the fuzzy parameters from users and a server application is dedicated to data communication between the hardware and the server. Analysis of this design is carried out by using hardware-in-loop (HIL) test to control various plant models in Simulink/Matlab environment. Performance of the proposed system is compared to Fuzzy Toolbox of Matlab and PID controllers.

4.1 Introduction

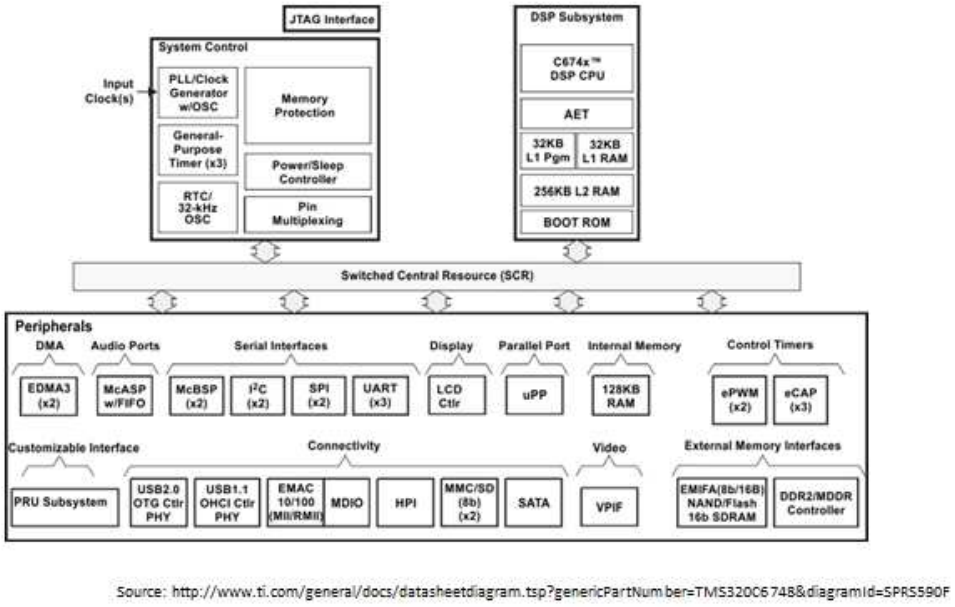
In previous chapter, the proposed MT-FRHC based remotely tunable G-FLCS architecture with Genetic Algorithm based FCP extraction technique is described. The results convincingly depicted that the objectives set in chapter 1 can be achieved in simulation environment. This chapter extrapolates the concepts introduced in last chapter to implementation it on a DSP platform. This chapter also introduces implementation and optimization of the code to realize the proposed MT-FRHC based G-FLCS with VBCoA defuzzification method. It is important that the code size is reduced to its maximum for reliable operation of the system. Large code size decreases power efficiency and computational speed. Smaller code size require less memory fetching since larger part of the on chip memory will be available for execution. This also increases the stack size. In this implementation it is imperative that the code size is limited to 60% of the on-chip SHRAM.

4.2 Hardware Device: TI LCDK C6748

To achieve an efficient implementation, it is important to understand the architecture of th hardware device before final implementation. With a sequential processing on a DSP device, the objective of the proposed MT-FRHC based G-FLCS with VBCoA defuzzification to meet 11K FLIPS achieved by Yi Fu et. al. [56] for their G-FLCS architecture on an FPGA is unachievable. Hereby, the implementation of the proposed G-FLCS architecture has to be parallelized. With this context, the hardware device at hand, TI's LCDK C6478 needs to be inspected.

LCDK C6478 incorporates a TMS320C6748 DSP which is a floating point very long instruction word (VLIW) DSP processor running at 476 MHz clock frequency. This device supports SIMD (single instruction, multiple data)and double precision VLIW operations [183]. It is reported that this is one of the speediest single precision device. This is an important feature of this device that suits the proposed G-FLCS architecture since all operations are single precision. SIMD instructions supports data parallelism as opposed to VLIW's

instruction parallelism. To take leverage of the SIMD or VLIW operations, it is imperative that the typical architecture of the G-FLCS is reconsidered to match either of these features of the device considering the functional block diagram in 4.1.



Source: http://www.ti.com/general/docs/datasheetdiagram.tsp?genericPartNumber=TMS320C6748&diagramId=SPRS590F

Figure 4.1: Functional Block Diagram of TMS320C6748 DSP

4.3 Generic FLC on DSP (TI LCDK C6748)

4.3.1 System Architecture

The system architecture of the proposed MT-FRHC based G-FLCS is represented as shown in (2.18),

$$\theta_f(c_j) = \bigcap_{k_x=0}^{n_{op}-1} \left(\theta_f(R_b(k_x)), \left(\bigcup_{l=0}^{N-1} \overrightarrow{C_{R_v}(l)} \right) \right) \tag{4.1}$$

There are two data loops in this architecture. The inner loop is set to a small constant by the number of active MFs (in this case completely controlled by the user); while the outer loops are independent of the data. Therefore it can be

concluded that the dependency of the outer loop is already parallelized in this architecture.

4.3.2 Code Optimization

Generally a code that is written in assembly (ASM) is processor-specific, whereas C code can readily be ported from one platform to another. However, optimized ASM code runs faster than C and requires less memory space. Before optimizing a code, it is required to make sure that the code is functional and yields correct results. After optimizing, the code can be reorganized and resequenced. The code becomes extremely difficult to follow and debug. It needs to be realized that if a C-coded algorithm is functional and its execution speed is satisfactory, there may not be a necessity to optimize it further. All these motivates us to stretch the optimization so that the best possible efficiency is extracted from the system [22, 184]. A rigorous code optimization strategy was devised as follows[41]:

Step 1 G-FLC is programmed in C Language without using any compiler optimization levels. With help of CCS Profiler tool, all the submodules are inspected for performance with respect to execution time and memory consumed.

Step 2 Intrinsic functions are used along with various compiler optimization levels. Functions like minimum (min), maximum (max), product (prod), division (div) are written as intrinsic functions and called as necessary by the modules and submodules of hardware G-FLCS.

Step 3 Thereafter, the CCS Profiler tool is exploited again to determine and identify the functions and submodules that may need further optimization.

Step 4 The functions that do not meet expected time and memory budget, are converted to linear ASM. The resultant is again inspected using the CCS Profiler tool to check for final efficiency.

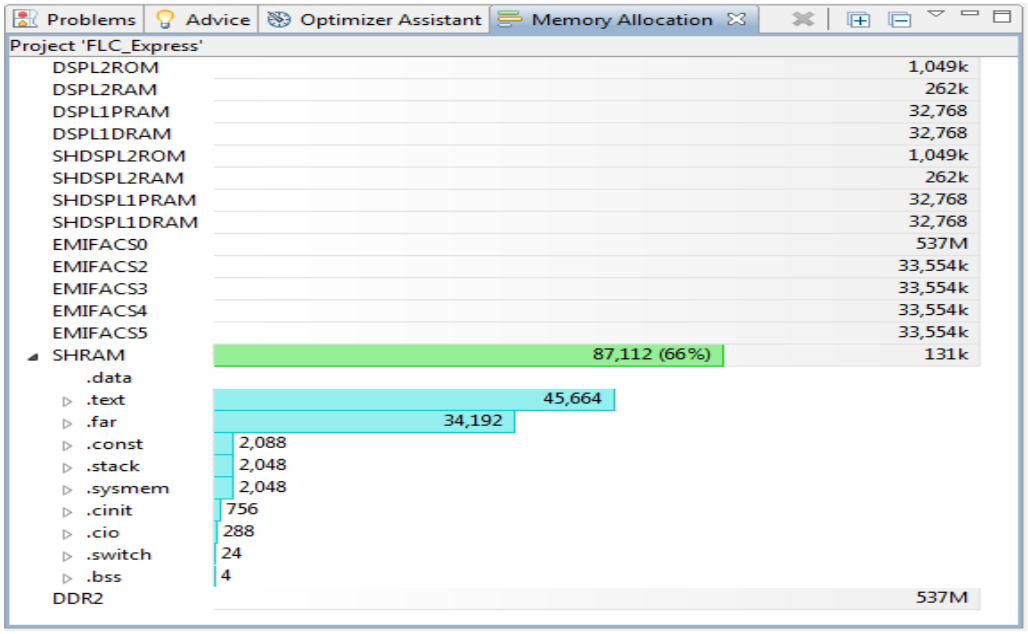
These process is realized in the system as follows:

1. '-o3' with optimization '-mt'
2. '-k' with optimization '-mw' as feedback option for compiler
3. Minimize the loop carried dependency bound.
4. MUST_ITERATE and UNROLL pragmas.
5. Operate on single precision data type
6. SIMD
7. Intrinsics from TI library.

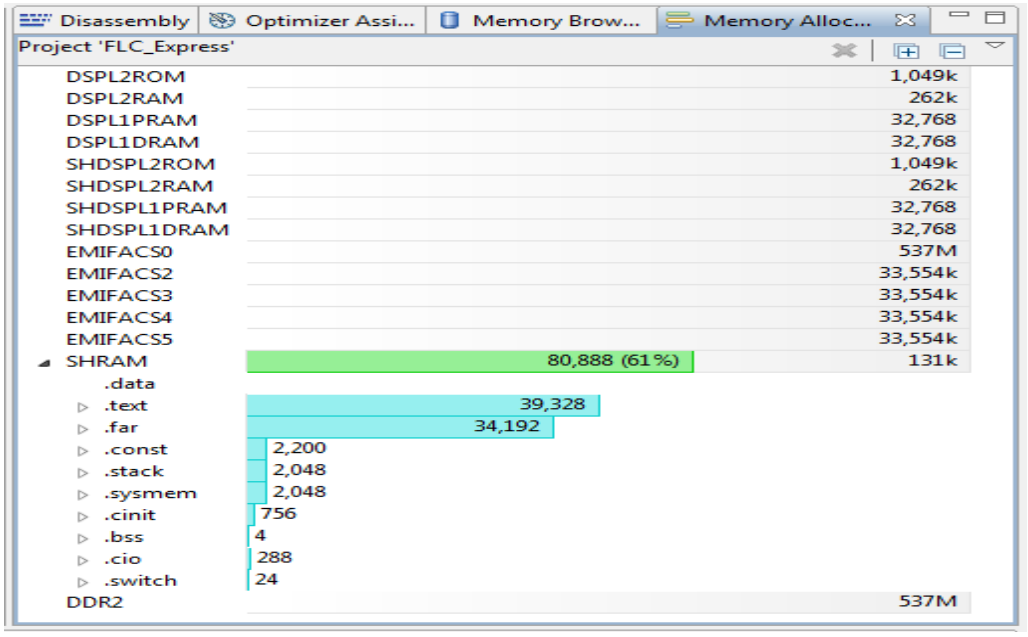
Table 4.1: Options in Compiler Level Optimization

Options	Features
-o3	This activates Compiler optimization level 3. In this setting, the compiler majorly tries to perform software pipelining.It also converts small functions to inline calls.
-mt	It explicitly mentions that the pointer-based parameters of a function will never point to the same location.
-k, -mw	These options instruct the compiler to generate feedback which can be analyzed for tuning performance.

The compiler can insert calls to special functions in the run-time support library (RTS) to support operations that are not natively supported by the ISA. For example, the compiler calls $\_\_c6xabi\_divi()$ ($\_divi()$ in COFF) function to perform 32-bit integer divide operation. Such functions are called compiler helper functions, and result in a function call within the loop body. For example in G-FLC, the compiler accomplishes the division operation by calling the compiler helper function "$\_divi$" in fuzzification and defuzzification modules.



(a) Without Optimization



(b) Optimization

Figure 4.2: Memory Utilization of Proposed System Realized on TI C6748 DSP

4.3.3 Code Implementation

Code Composer Studio (CCS) is a proprietary integrated development environment developed by TI for programming DSP and ARM processors. The design is programmed in C language and optimized as described in section 4.3.2. Thereafter it is cross compiled using CCS v5.5 compiler and implemented on TMS320C6748 as target DSP processor. This system represents the hardware G-FLCS as discussed previously. G-FLCS is connected to a server PC using onboard UART and provides a platform which is capable of accepting FCP file to operate as a standalone tunable G-FLCS.

To achieve a high performance in throughput, the following optimization technique were used inside the developed code [183].

The MUST_ITERATE Pragma The MUST_ITERATE pragma specifies the lower bound, upper bound and factors of a loop. The lower bound and upper bound specifically mentions the minimum and maximum possible iterations of the loop and the factor defines the step size between them.

```
#pragma MUST_ITERATE(lower_bound, upper_bound, factor)
```

Code Snippet 4.1: MUSTITERATE Pragma

Loop Unrolling and the UNROLL Pragma Manual unroll of loops are imbibed in code to improve code efficiency. Consider following code snippets.

```
void Loop(int * restrict output, int * restrict input1, int *
    restrict input2, int n)
{
int i;
for (i=0; i<n; i++)
{
output[i] = input1[i] + input2[i];
}
}
//An excerpt from its compiler feedback
;* Partitioned Resource Bound(*) : 2
```

```
;* Resource Partition:
;* A-side B-side
;* .L units 0 0
;* .S units 0 1
;* .D units 2* 1
;* .M units 0 0
```

Code Snippet 4.2: A Loop Code With Unbalanced Resource Partition

The .D unit on the A side of the device is used twice every iteration, and the .D unit on the B side is only used once. This indicates that the .D unit on the B side is left open for 1 cycle in each iteration of the loop. In this case, the partitioned resource bound is affected by this unbalanced partition. Ideally, it would be more efficient if both units are used 1.5 cycles per iteration or 3 cycles per 2 iterations

```
void Loop(int * restrict output, int * restrict input1, int *
    restrict input2, int n)
{
int i;
for (i=0; i<n; i+=2)
{
output[i] = input1[i] + input2[i];
output[i+1] = input1[i+1] + input2[i+1];
}
}
//An excerpt from its compiler feedback
;* Partitioned Resource Bound(*) : 3
;* Resource Partition:
;* A-side B-side
;* .L units 0 0
;* .S units 1 0
;* .D units 3* 3*
;* .M units 0 0
```

Code Snippet 4.3: Manually Unrolled Loop

The .D units are used 3 times on each side per iteration, but the number of iterations is halved. This, in theory, can lead to a 25% reduction in overall cycle count. Although effective, the manual unroll process can be tedious for most loops. Generally, it is recommended that the C6000 compiler and pragmas be used to unroll loops.

Detailed memory utilization of the built code is shown in Figure 4.2. It can be observed that the code size without optimization acquires 66% of the SHRAM and has a code size of 88K. After employing the optimization strategy, the code size is reduced to 61% of the SHRAM with a code size of 80K.

4.4 Interfacing G-FLC with WebUI

4.4.1 Data Communication between Hardware G-FLCS and Server

The proposed system architecture involves hardware-software co-design to present a complete reconfigurable FLC. The WebUI in client server model represents the software and the driver layer to interface the hardware G-FLCS through serial port as shown in Figure 3.1a. The DSP hardware receives FCP data serially and stores them in predefined memory locations as shown in Table 4.2. These parameters which are, segregated in two categories namely *Setup* and *Rulebase* data. The driver layer in the hardware G-FLCS receives and acknowledges the data transmission serially over UART.

4.4.2 WebUI and its Operation

The WebUI is a web application that drives the proposed hardware G-FLCS was presented in section 3.4 and displayed in Figure 3.3 and Figure 3.4 stores FCP data in a file[1]. It may be noted that the FCP file is generated deliberately in accordance with Matlab Fuzzy Inference System (FIS) file format to provide liberty to integrate a parameter data file generated using Matlab Fuzzy Logic

[1]File can be downloaded from https://goo.gl/YAVxez

Toolbox as well. Submitting the parameters triggers a desktop application that is native to the server. Objective of this program is to transmits the FCP from database to the hardware G-FLCS through serial communication. The FCP data is stored in the DSP board based according to the memory map provided in Table 4.2. The first column in Table 4.2 shows the offset address and the second column elaborates the number of bit individual fuzzy parameter consumes. The last two columns describes and provides remarks about the different fuzzy parameters FLC designed with M-FRHC rule reducing Inference Engine, operates on the inputs with these FCP data to provide desired control action. FCP data is segregated in two parts, *Setup* and *Rulebase*. Offset address **0000**H to **00D6**H in Table 4.2 shows the details of Setup parameters. *Rulebase* parameters appear in memory location **00D7**H to **1D0C**H which is reserved to store **2401** rules. Fuzzifier, inference engine and defuzzifier is programmed with information about the data and its memory location. With these information, fuzzifier transforms crisp inputs in fuzzy domain and a Mamdani inference engine, coupled with the Rulebase in the system memory produces fuzzy output. Defuzzifier transforms fuzzy output compiled by the inference engine back to crisp output.

The Memory Map in Table 4.2 is also segregated into two parts as FCP data. The first part, incorporates the *Setup* FCP data, specifies the Fuzzifier and Defuzzifier parameters and it includes the following information;

- Initialization of the FLC,
- Number of Inputs and Outputs,
- Number of Membership Functions in each Input and Output, and
- Details of each Membership Functions like, type of membership function and their co-ordinates.

The second part, incorporates *Rulebase* FCP data, specifies the Rulebase of the FLC defined by the index numbers of the inputs and outputs. In this work, the original user specified Fuzzy Control Parameters (FCP) (as specified in the Table 4.2 Memory Map) is not forked or pruned. Rather, at every inference the

proposed MT-FRHC rule reduction algorithm, operates on this user specified Memory Map to derive another map with reduced rules.

4.5 System Performance and Analysis

4.5.1 System Modeling of Armature Controlled DC Motor

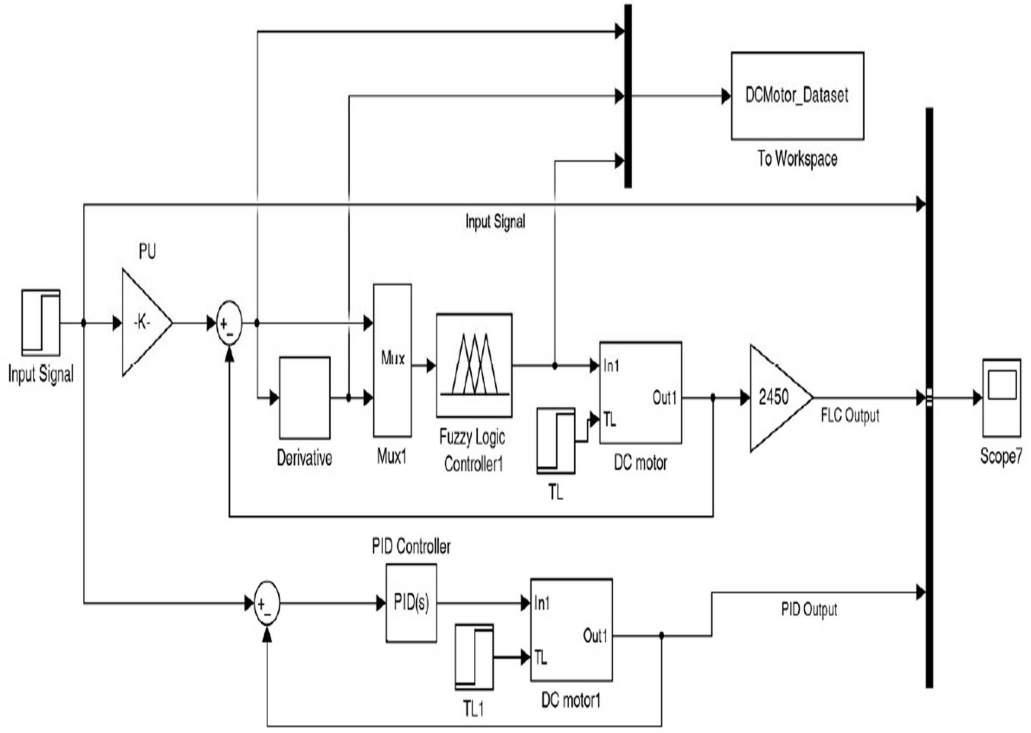


Figure 4.3: Simulink Model for Speed Control of DC Motor

A Simulink model to simulate Speed Control of a DC Motor with PID and Fuzzy Logic Controllers is tested in this experiment [112, 170]. The Simulink model of the process plant is presented in Figure 4.3. The control problem uses a DC Motor with Armature Resistance (R_A) = 1 Ω, Armature Inductance (L_A) =

0.5, Inertia (J_M) = 0.01, Damping (B_M) = 0.1, Torque Constant (K_τ) 0.01 Nm/A and Back EMF Constant (K_B) = 0.01 Vs/rad. Transfer function of the plant model is stated as:

$$\frac{\theta(s)}{V_A(s)} = \frac{K_\tau}{L_A J_M s^3 + (R_A J_M + L_A B_M) s^2} \text{ x } \frac{1}{(K_\tau K_B + R_A B_M) s} \tag{4.2}$$

The ACDC Motor is controlled by FLC and PID controllers in different datapaths. Simulation of this model produces the I/O dataset and is saved in a file which concludes first phase of this experiment. The data set consists of two inputs, error in speed and its derivative, and the control signal (voltage to the DC Motor) as an output.

4.5.2 Hardware-in-Loop Test

The Hardware-in-the-Loop (HIL) testing method is a prevailing testing method in industries. It has been exploited to test wide array of embedded system applications over past two decades. It was initially introduced in the Aviation industry. The major factor which makes this test methodology prevalent in the industry over the years are time to market and its low complexity. HIL test method caters an efficient platform to test real-time system designs by adding the plant complexity to be controlled in the test platform. The plant model and all related dynamic systems under control is mathematically presented. This algebraic representation of the entire plant model is called as the “plant simulation”. The real-time embedded system is tested by interacting with this plant simulation which executes in a computer, sometimes referred to as the Host Computer. Figure 4.4 shows the main concept of the HIL test methodology where an embedded system is developed on a real-time processor and is interacted with the process plant simulation in the host computer.

The proposed MT-FRHC based G-FLCS is developed on a TI C6748 DSP processor. This represents the embedded system under test in the HIL loop. The dynamic modeling of the ACDC Motor is developed in Simulink. This represents the “plant simulation”. The HIL test was performed as depicted in

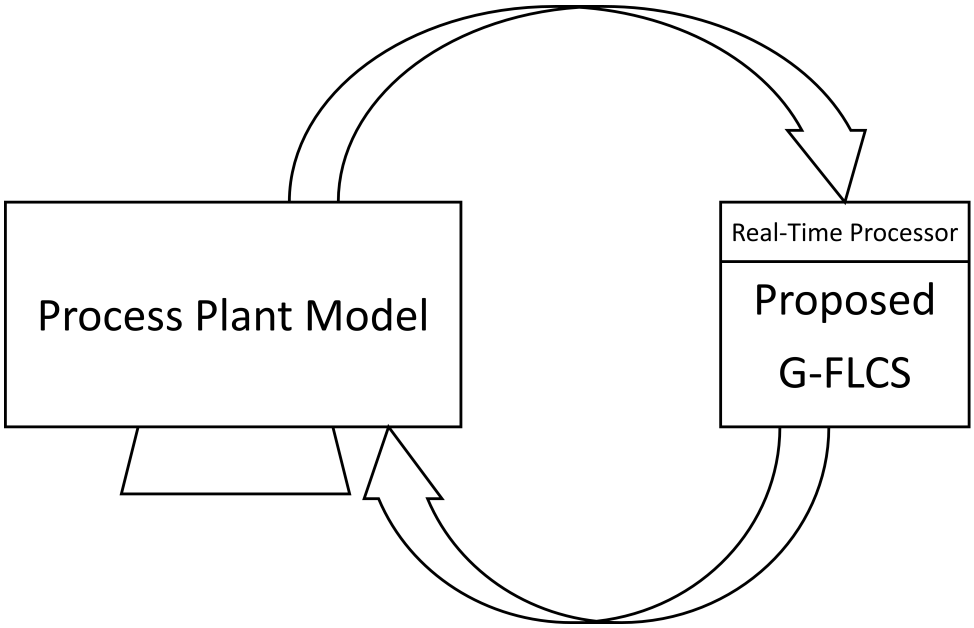
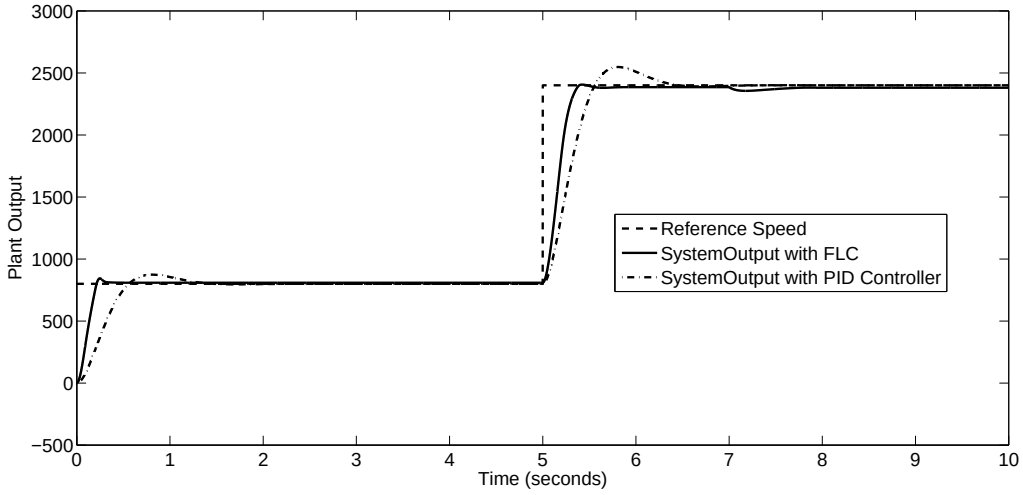


Figure 4.4: Test Setup for Hardware-in-Loop Testing of G-FLCS

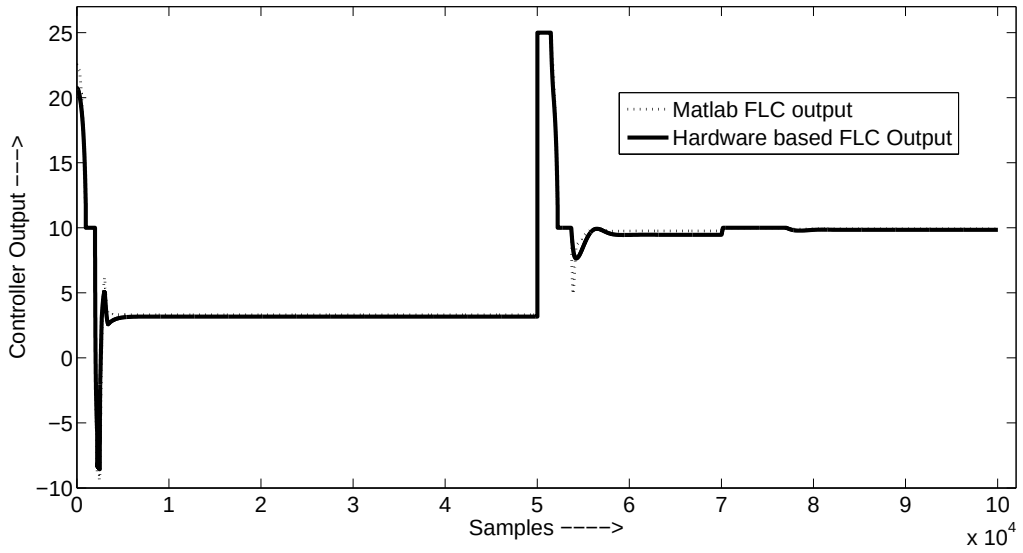
Figure 4.4. From executing the "plant simulation" with controllers developed in Simulink provides the simulated results. Figure 4.3 represents the "plant simulation" model. This model is developed based on the transfer function derived in (4.2). The results from simulation and HIL test was compared and analyzed where an error of 2% was recorded. Details of this test in provided in section 4.5.4. However, before analyzing the performance, a method for FCP generation is explained in section 4.5.3. This method was found to decrease the time consumed by GA based FCP extraction at large.

4.5.3 Fuzzy Control Parameter Generation

A genetic algorithm based fuzzy control parameter extraction technique is described in section 3.5. However, in many of the test processes, it is found the genetic algorithm based FCP extraction technique extensively time consuming. One way of reducing this extraction time is by providing initial parameters to the GA. With proper initial parameters the search space for GA can decrease drastically. This can reduce the FCP extraction time. One of the widely used techniques for FLCS design is a Fuzzy approximation of PID controller



(a) Simulation output for the Armature Controlled DC Motor



(b) Control Signal Data plot for Simulated and hardware G-FLCS

Figure 4.5: Plant output and Controller output of ACDC Motor simulated using Matlab Fuzzy Logic Toolbox and HIL test with proposed hardware G-FLCS

[101, 196]. The FCP generated from this scheme assume a FLCS structure analogous to classical PID controller [11, 12, 24, 32]. However unlike classical PI controller, Fuzzy PI controller is non-linear. This method implicitly implies that the generated FCP will consistsof fuzzyinput fuzzy variable e and Δe. The FCP used for this Initial parameters for GA based FCP is finallys [46, 156]. The

FCP used for this G-FLC syst system for verification of the controller[1].

In algorithm 1, a plant simulation model with PID controller is developed on Simulink as described in section 4.5.2. An input-output relationship between the error signals (consisting of e and Δe) and control output (u) is generated and recorded for large number of samples. Introduce input MF corresponding to maximum and minimum values of inputs e and Δe. Similarly, for each set of input membership functions introduced, a corresponding MF is introduced. The kernel of these MFs are set to the respective minimum and maximum output value from the dataset. In this algorithm, initial choice of the memberships are trapezoidal and for every other iteration, triangular membership functions are added appropriately.

procedure ERROR AND CONTROL SIGNAL GENERATION
$N \leftarrow$ Mamimum no. of membership functions allowed
top:
$i = i + 1$
Run Simulink Model with PID Controller
$e, \Delta e \leftarrow$ Error signal at each sample
$u \leftarrow$ *Control signal generated at each sample*
Note u at maximum and minimum value of e and Δe
loop:
Introduce input MF corresponding to minimum and maximum value of e and Δe
Introduce output MF corresponding to minimum and maximum value of u
Addition of triangular MFs to map rules as introduced above
Evaluate rules at minimum and maximum e to generate $\hat{u}$
if $i < N$ **then**
  if $u - \hat{u} \neq 0$ **then**
    goto *loop*.
  else
    goto *top*.
  end
else
  close;
end

Algorithm 1: A technique for Fuzzy PI Approximation

[1]File can be downloaded from https://goo.gl/83bVna

4.5.4 Performance Analysis

A HIL test was conducted for the proposed MT-FRHC based G-FLCS on TI C6478 as described in section 4.5.2 for the plant model shown in Figure 4.3. The response of the HIL test was recorded and tabulated in Table 4.3. The controller performance has been analyzed with respect to inference time and control output separately and plotted in Figure 4.5. The simulation experiment was found to work seamlessly. The hardware realization of the proposed G-FLCS application was necessary to determine if this approach is perceivable in real-time.

4.5.4.1 Performance Analysis: Control Output

The output from HIL test is conducted for the Simulink model in Figure 4.3 and the output presented in Figure 4.5a. The inputs and corresponding output was recorded and saved in file which is intended to be used as data source. The same FCP file used in this simulation was loaded in the FLC based DSP hardware through the server application as described in chapter 3. A separate Windows application was created to read the input data stored in the data source file and send to the G-FLCS externally.

The G-FLCS received the inputs and generated suitable control outputs which was sent back to the PC and recorded in a loop. The recorded data is then compared to actual simulated output for data validation. The time elapsed in this entire process has also been recorded. The inferences observed during this experiment has been tabulated in Table 4.3. There were total of 100021 input samples in this entire simulation. Control signals generated from the simulation and G-FLCS was recorded for each input samples and plotted in Figure 4.5b. This figure represents a plot of the control output corresponding to the DSP based proposed G-FLCS and Matlab FLT for all the input samples. It validates that the control data output from the G-FLCS is in synchronization with the simulated result with a MSE of 2%. There is a slight deviation observed in control output of the G-FLCS and Simulink simulation. This can be attributed to the design and implementation of MT-FRHC rule reduction technique with VBCoA defuzzification method as discussed in chapter 2.

4.5.4.2 Performance Analysis: Inference Time

The system has also been tested for its inference time. Inference time is defined as the time taken for a set of input to propagate to the output and produce a control action. Figure 4.5b provides no observation related to the execution time of the G-FLCS. Thereby, time elapsed in the entire process of testing with 100021 samples was also been recorded. It found to be 7.248949 seconds for entire process and on average 0.073 ms or 73 μs for single inference. This data provides an estimate of the inference time that can be achieved with the proposed G-FLCS. However it is should to be noted that this time does not include any instance where parameters have tuned in between the process.

4.5.5 Comparison to Existing Works

The G-FLCS design is implemented on a programmable DSP. There are few works reported in the literature which implements similar designs on FPGA platform. However, these systems provide fewer flexible FCP features in comparison to the proposed design. The performance of the proposed G-FLCS was compared to the existing FPGA based designs presented in [56, 122]. A summary of these designs are tabulated in Table 4.4. It can be observed that the proposed G-FLCS design provides variety of features with adequate speed and flexibility in comparison to the FLCS designs reported in [56, 122]. This design provides a speed of approximately 13 KFLIPS for a 2-input 1-output system with 49 rules with seven MFs for each input and output. The proposed G-FLCS is designed to support a maximum 4-input 1-output system with seven MFs for each input and output space. This adds up to a total of 2401 rule system. The code size of 61 % reported in Figure 4.2 implements 4-input 1-output system.

4.6 Summary

The realization of the remotely tunable MT-FRHC based G-FLCS with VBCoA defuzzification on programmable DSP hardware is explained in this chapter. This technology opens line of approach for implementation to several explorations. The motivation behind this research was deployment of a generic fuzzy

framework with complex features in a programmable device such that it can be remotely tuned by altering its parameters in real-time. In this chapter, a successful implementation of these objectives is described. Existing G-FLCS have been able to produce significant speed by reducing functionalities in their architecture. This architecture provides large number of functionalities to its users along with sufficient speed to drive most industrial processes. This system is standardized with MATLAB Fuzzy Logic Toolbox and has ability to incorporate FIS files generated by this toolbox. This system presents a framework for remotely tunable MT-FRHC based G-FLCS that can suitably replace other controllers by following the design protocols explained in this thesis. The proposed systems is observed to perform well within the multiple testing paradigms. Through investigations have been done using multiple applications to ascertain its generality and applicability. In summary, this chapter successfully presents a remotely tunable MT-FRHC based G-FLCS with VBCoA module developed on programmable DSP with interface to a WebUI which can operate as standalone controller with an operating speed of around 13K FLIPS.

Table 4.2: Memory Map

Offset Addr.	Memory bits	Description	Details
00H	[1:0]	00-No Operation 01-New inputs are enables 10-new Rulebase is enabled 11- new Rulebase and inputs are enabled	To start the FLC operation for the first time programmed with 0x03h then if input only changes program with 0x01H
00H	[7:2]	**Don't Care**	
01H	[1:0]	00- Number of inputs = 1 01- Number of inputs = 2 10- Number of inputs = 3 11- Number of inputs = 4	Number of Inputs and Outputs
01H	[2]	0- Number of outputs = 1 1- Number of outputs = 2	
01H	[7:3]	**Don't Care**	
02H	[2:0]	Number of MFs for Input 1	000- 0 (I/O not used), 001- 111 refers to the number 1-7
02H	[4:6]	Number of MFs for Input 2	
03H	[2:0]	Number of MFs for Input 3	
03H	[6:4]	Number of MFs for Input 4	
04H	[2:0]	Number of MFs for Output 1	
04H	[6:4]	Number of MFs for Output 2	
MFs consists of type of MF and the number of co-ordinates are based on them. Each input and output can have maximum of 7 MFs.			
MFs for Input 1			
05H	[7:0]	1st Co-ordinate of MF 1 of input 1	MF 1 of Input 1
06H	[7:0]	2nd Co-ordinate of MF 1 of input 1	
07H	[7:0]	3rd Co-ordinate of MF 1 of input 1	
08H	[7:0]	4th Co-ordinate of MF 1 of input 1	
09H	[2:0]	Type of MF number 1 of input 1	000- Trapezoidal
			001- Gbell
			010- Gaussian
			011- Curved Triangle
			100- 111- Triangle
...	...	...	MF 2-7, Input 1
MFs for Input 2			
28H	[7:0]	1st Co-ordinate of MF 1 of input 2	MF 1 Input 2
3AH	[7:0]	2nd Co-ordinate of MF 1 of input 2	
3BH	[7:0]	3rd Co-ordinate of MF 1 of input 2	
3CH	[7:0]	4th Co-ordinate of MF 1 of input 2	
3DH	[2:0]	Type of MF number 1 of input 2	000- 111 (Same as above)
...	...	...	
D6H	[2:0]	Type of MF number 7 of output 2	
Memory is reserved for 4 Inputs and 2 Outputs consequently even if, FLC does not operate with all 4 input and 2 output.			
Rule 1			
D7H	[3:0]	Index number of Input 1	000- 0 (I/O not used), 001- 111 refers to the number 1-7
D7H	[7:4]	Index number of Input 2	
D8H	[3:0]	Index number of Input 3	
D8H	[7:4]	Index number of Input 4	
D9H	[3:0]	Index number of Output 1	
D9H	[7:4]	Index number of Output 2	
Rule 2			
...	...	...	
Continues till end of rules. Max rules supported by this system is 2401 (7^4)			

Table 4.3: Results from hardware G-FLCS experiment with Simulink Model

Output Parameters	Observation
Total data Samples	100021
Total Execution Time	7.248989 s
Average Execution Time	7.247427e-005 s
FLIPS	13.8 K
Percentage Error	2% w.r.t. Matlab FIS
Mean Square Error (MSE)	0.1298

Table 4.4: Comparison between Proposed hardware G-FLCS and Similar Designs based of Reconfigurable Parameters

Year	Reference	Speed (in FLIPS)	Platform	Features
2008	Millan et. al.[122]	5.5 K	FPGA	Output MFs: Singleton (5) I/O: 2-1 Input MFs: 8 Overlaps: Dynamic Rules Evaluated : 64
2010	Yi Fu et. al.[56]	11 K	FPGA	Output MFs: (5) I/O: 2-1 Input MFs: 5 Overlaps: 2 Rules Evaluated : 25
Proposed G-FLCS		13 K	DSP	Output MFs: (7) I/O: 4-1 (Configurable) Input MFs: 7 Overlaps: Dynamic (4) Rules Evaluated : 49 (2401)

Chapter 5

Implementation of Proposed G-FLCS for Radial Plasma Position Control in Aditya Tokamak Fusion Test Reactor

Preview

In this chapter, a nonlinear process plant, Aditya Tokamak Fusion Test Reactor (TFTR) is controlled by the proposed G-FLCS. Tokamak is a torus shaped magnetic field based confinement device for the extremely hot plasma. It is one of the most widely researched device since it is key to only form of clean energy that can be industrially generated. To stabilize the plasma across radial position is crucial to the success of a tokamak reactor. Plasma is highly sensitive and nonlinear state of a matter and its position control is extremely time critical. In this chapter, the proposed hardware based G-FLCS controller is used to control Radial Plasma Position in Aditya TFTR, installed at Institute of Plasma Research (IPR), Gandhinagar, India.

5.1 Introduction

In last chapter, a remotely tunable MT-FRHC based G-FLCS with VBCoA is implemented on TI C6748 DSP. A HIL test was also conducted to see the performance of this system. In this chapter, the proposed G-FLCS is used to control the radial position of a plasma column in Aditya Tokamak Fusion Test Reactor (TFTR). Aditya TFTR is installed at installed at Institute of Plasma Research (IPR), Gandhinagar, India. It is a medium size tokamak with major radius of 0.75 m and a minor radius of 0.25 m. There are 20 toroidal field coils in the design which produces a maximum field strength of 1.2 tesla. Genetic algorithm based FCP is used to extract FCP which can drive the G-FLCS in controlling the radial plasma position in the torus. The FCP can also be extracted partly using conventional optimization methods. Gradient Descent, Back propagation and other stochastic methods are employed to derive the co-ordinates of the antecedent membership functions [131–133]. When the rules are defined the conventional optimization methods may be used provided the cost function is convex in nature. However, mostly it is not true. Moreover, the output of the FLC is not a continuous function. So optimizing the antecedents and consequents are not possible at same instance. Thus, evolutionary algorithms are preferred, specially Genetic Algorithm, owing to its ease of implementation has been used in this work [20, 80, 181].

5.1.1 Controlled Thermonuclear Fusion

Researchers observed that there was an enormous release of energy that occurred when two light nuclei (with masses lower than iron, Fe, in periodic table) fuse together. This phenomenon was termed as thermonuclear fission reaction. Then, World War II ended with an historical uncontrolled thermonuclear fusion, the bombings at Hiroshima and Nagasaki. At this time, many scientist and engineers realized that if this phenomenon can be confined and controlled, the possibility of harnessing this energy for development of mankind would be enormous. A thermonuclear fusion can be confined and controlled by

- Gravitational confinement

- Magnetic confinement
- Inertial confinement
- Electrostatic confinement

5.1.2 Tokamak Fusion Reactor

A tokamak is a magnetic field based plasma confining device in the shape of a torus. A stable plasma equilibrium can be achieved by generating magnetic field lines which can helically move around the torus. The plasma position control in a Tokamak reactor is a highly nonlinear control problem. In Tokamak reactor, magnetic field is used to confine the plasma in desired position. Plasma is highly sensitive state of matter and can be unstable under slightest trigger in the surrounding environment. It is therefore very important to design a fast but highly robust controller. A tokamak can successfully operate if the plasma is stable and confined to the geometric center of the vacuum vessel. The radial position of the confined plasma inside the torus vessel inflict on the quality of the plasma discharge. Unstable plasma when approaches too close to the wall of the vessel, it may lead to partial or complete disruption of the plasma. Hence it is of primal importance that the plasma position is controlled throughout the plasma discharge process.

To achieve a stable plasma equilibrium and to confine it inside, a fusion reactor is required to generate magnetic field lines that helically embraces the torus shaped plasma. These magnetic field lines can be generated using electromagnets positioned accordingly. Generation of helical field can be achieved by adding a magnetic field that circularly travels around the torus (toroidal field) and another field that travels orthogonally across to the toroidal field (poloidal field). These field are generated by toroidal field coils and inner and outer poloidal field coils as shown in Figure 5.1. When a current is passed to a centrally located helical inner poloidal magnetic field coil, it produces an induced current in the plasma. Direction of the coil current and induced plasma current is shown in red arrows. This plasma current generates a poloidal magnetic field. The required toroidal magnetic field is produced by the circularly

surrounded coils across the torus. The position of the plasma can be controlled by driving the electric current to these coils.

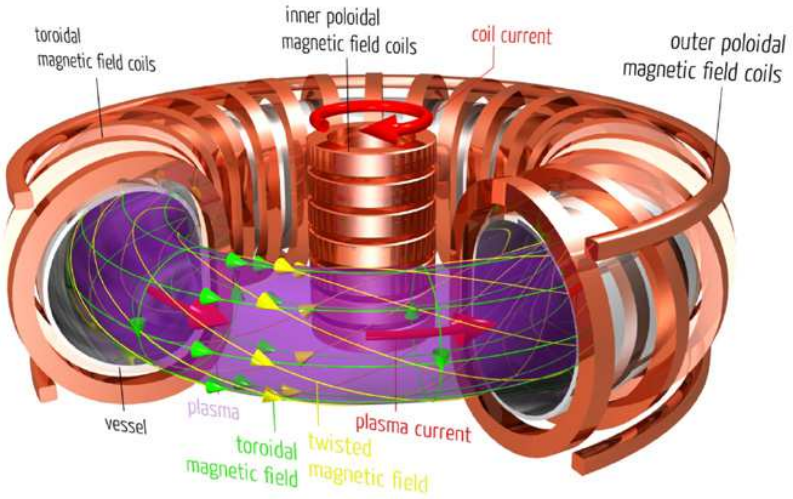


Figure 5.1: Schematic of a tokamak.
Photo credit: Abteilung Öffentlichkeitsarbeit - Max-Planck Institut für Plasmaphysik. Licensed under Creative Commons BY–SA 3.0 via Wikimedia Commons

5.1.3 Aditya Tokamak Fusion Reactor

Table 5.1: Parameters of Aditya Tokamak under different power supplies

Power Supply	Parameters	Approx. Values
Capacitor Bank	Plasma Current	30 kA
	Shot Duration	25 ms
	Central Electron Temp.	100 eV
	Core Plasma Density	$10^{19} m^{-3}$
Aditya Pulse Power Supply (APPS)	Plasma Current	100 kA
	Shot Duration	25 ms
	Central Electron Temp	100 eV
	Core Plasma Density	$3x10^{19} m^{-3}$

Aditya Tokamak Fusion Reactor (Aditya) is India's first Tokamak Fusion Test Reactor (TFTR) [27]. It is a medium sized test reactor designed, developed and stationed at Institute of Plasma Research, Gandhinagar, India. The plasma has a major and minor radius of 0.75 m and 0.25 m respectively. There are

twenty toroidal magnetic field coils symmetrically arranged across the torus. These coils produce a maximum magnetic field strength of 1.2 tesla. Table 5.1 describes the parameters of Aditya Tokamak under different power supplies.

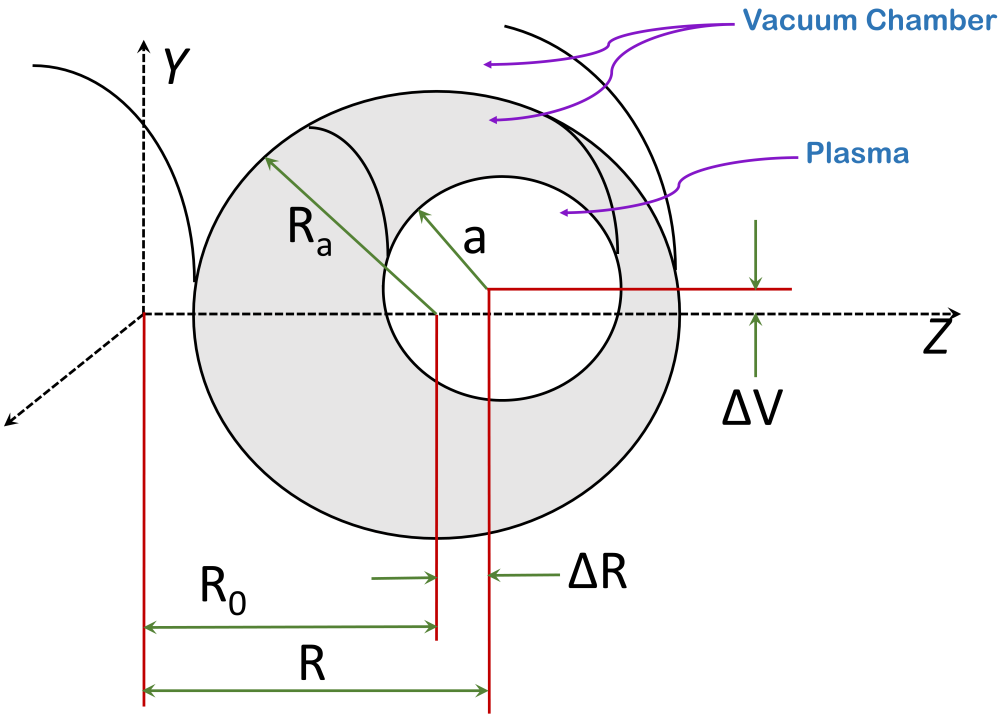


Figure 5.2: Cross-sectional view of plasma position and displacement inside the vacuum chamber

5.2 Aditya Tokamak System Modeling

In this section, the control problem of radial position of plasma in Aditya TFTR using fast feedback (FF) coil has been analysed using a RZIP model. The geometric center of vacuum vessel of Aditya TFTR is at 0.75 m and it is critical that the radial position of the plasma maintained at this point. This model is developed with the assumption that small variation in coil currents produces small change in plasma position and current.

Unlike circular cross-section plasmas, Tokamak operates on highly non-circular torus shape. Non-circular shapes are more difficult to produce and

to control accurately, since currents through several control coils must be adjusted simultaneously[30]. Due to uncertainties in the current and pressure distributions within the plasma, the desired accuracy for plasma control can only be achieved by making real-time measurements of the position and shape of the boundary, and using error feedback to adjust the currents in the control coils. The modeling of the discharge parameters like plasma current, position and shape is a challenging task, as they are highly nonlinear and time varying in nature. Hence, due to inherent complexity of the plasma position control system and its nonlinear nature, it is difficult to achieve control of plasma position using traditional controllers [180]. A similar approach was also taken by Morelli et. al. [126] in plasma position control of STOR-M Tokamak Fusion Test Reactor.

Considering the above modeling parameters taken from the work by Bandyopadhyay et. al. [18, 19],

$$\dot{X} = \mathbf{A}X + \mathbf{B}U \tag{5.1}$$

where $\mathbf{A} = \overline{\mathscr{M}}^{-1} \cdot \overline{\mathscr{R}}$, $\mathbf{B} = \overline{\mathscr{M}}^{-1}$, and

$$X = \begin{bmatrix} I_C \\ zI_P \\ RI_P \\ I_P \end{bmatrix}, U = \begin{bmatrix} V_C \\ 0 \\ -\frac{\mu_0 I_P}{2}\Gamma \\ I_P \end{bmatrix}$$

$\overline{\mathscr{M}}$ and $\overline{\mathscr{R}}$ refers to vector of mutual inductances and resistances of all circuits with plasma [17, 19, 180].

$$\overline{\mathscr{M}} = \begin{bmatrix} M_C & \left(M'_R\right)^T & M_{PC} \\ M'_Z & 0 & 0 \\ M'_R & M_{22} & M_{22} \\ M_{PC} & M_{22} & L_{P0} \end{bmatrix} \text{ and } \overline{\mathscr{R}} = \begin{bmatrix} \Omega_C & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & \Omega'_P & \Omega_P \end{bmatrix}$$

where, $M_{22} = \left(\frac{\mu_0}{2}\frac{d\Gamma}{dr} + \frac{2\pi B_{z0}}{I_{p0}} + \frac{2\pi R_0 B'_{z0}}{I_{p0}}\right)$, $M_{23} = \left(\mu_0\Gamma_0 + \frac{2\pi R_0 B_{z0}}{I_{p0}}\right)$, $M_{32} = \left(\mu_0(1+f_0) + \frac{2\pi R_0 B_{z0}}{I_{p0}}\right)$, M_C and Ω_C are mutual inductance and resistance matrices of all the circuits, M_{PC} and M_R are the vector of mu-

tual inductances of the circuits with the plasma and their radial derivatives respectively, and Γ is known as Shafranov parameter. This shows that A and B are matrices of dependent on the mutual inductances and resistances of all circuits with plasma which is highly non-linear in nature.

In Aditya TFTR, four magnetic probes are used to measure the radial position of the plasma. These probes are places close to the outer periphery of the vacuum vessel. A Rogowski coil is used to measure the plasma current (I_P).

P. Suratia et. al. [180] and I. Bandyopadhyay et. al. [17] explained the major control operatives as - "ADITYA has been provided with a primary vertical coil field with adjustable gain proportional to plasma current, to compensate the change in vertical displacement of plasma column. The shift in radial position due to minor disruptions is controlled by a separate pair of Fast Feedback coils, this fast feedback coils produces adequate magnetic field to bring the plasma column back to its geometrical center."

5.3 Control Strategy

5.3.1 Using PID Control

Traditional PID controllers are used presently in Aditya TFTR to which radial position signal is fed as input. The controller generates a suitable control signal to actuate the current in the fast feedback coils and accordingly the plasma

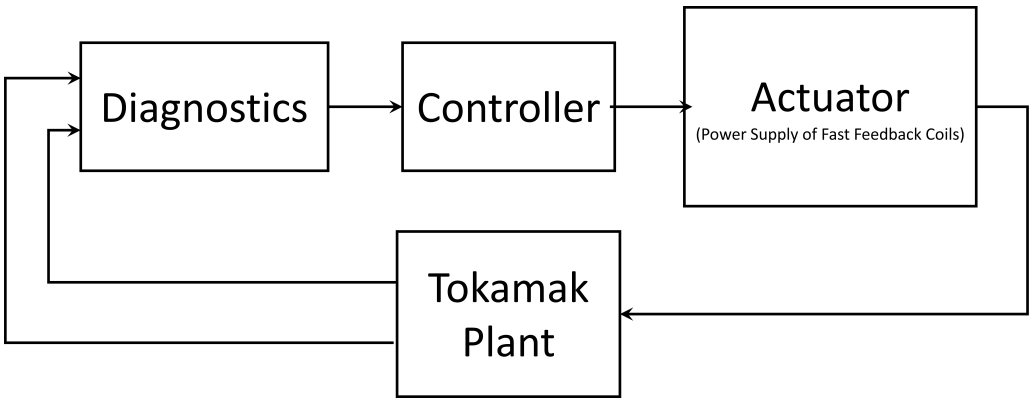


Figure 5.3: Control strategy for radial plasma position control in Aditya TFTR

is confined in radial direction [180]. Figure 5.3 shows the control strategy employed in radial position control of plasma in Aditya TFTR.

$$\frac{\Delta B_v}{B_v} = \frac{\Delta R}{R}$$

The vertical field required to maintain the radial position of plasma in Aditya TFTR can be obtained from Grad-Shafranov equation [128, 180] presented at (5.2).

$$B_v = \frac{\mu_0 I_P}{4\pi R}\left[\ln\left(\frac{8R}{a}\right) + \beta_P + \frac{l_i - 3}{2}\right] \tag{5.2}$$

where, **B** is Magnetic flux density, B_ϕ, B_θ, B_ρ is Toroidal, poloidal and radial components of the magnetic field, E is Electric field intensity, J is Plasma current density, R is Major radial coordinate and a is Minor radial coordinate. It can be observed from (5.2) that, the total vertical magnetic field for proper position control of plasma is proportional to the magnitude of

1. Internal inductance of plasma l_i,
2. Plasma current I_P, and
3. Plasma poloidal beta[1].

All these parameters are time varying and highly nonlinear in nature. The limitations of PID controllers have been already explained and therefore, a controller equipped to handle these parameters to provide a smooth, fast and robust control action is of utmost importance. However, it is important to exercise the existing knowledge gathered from the system response with PID controller. A PID control loop for radial plasma position control of Aditya TFTR is developed in Simulink as shown in Figure 5.4. It represent the mathematical model explained in (5.2). The PID controller is tuned using Ziegler-Nichols method. The Ziegler-Nichols tuning method is a heuristic method of tuning a PID controller [17, 19]. The simulation output data is observed and recorded.

[1] It is the ratio of the poloidal plasma pressure to the poloidal magnetic pressure

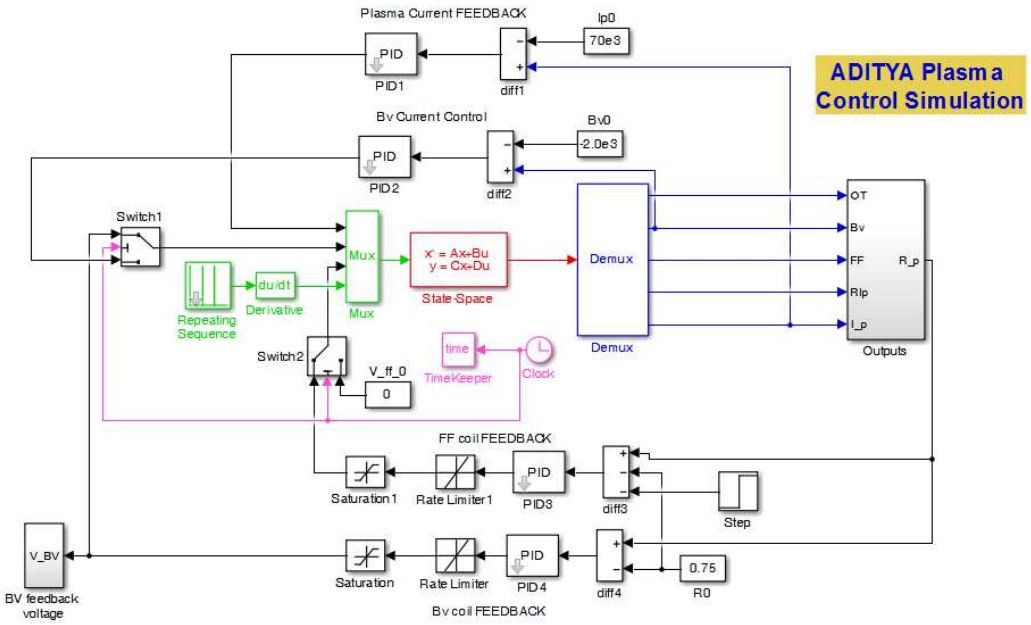


Figure 5.4: Simulink model of radial plasma position control in Aditya TFTR with PID controller

5.3.2 Plasma Position Control in Aditya using Traditional Fuzzy Logic Controller

P. Suratia et. al. proposed a fuzzy logic controller for radial plasma position control in Aditya TFTR [180]. The characteristic features of this controller and the proposed G-FLCS used in the control simulation is tabulated in Table 5.2. The Simulink model with the FLC is shown in Figure 5.5. It can be observed in this figure that the inner PID loop in the outer position control loop in Figure 5.4 is replaced by a FLC loop in Figure 5.5. The download link to the FCP used in [180] to control the radial position of plasma in Aditya TFTR is provided in Appendix-A.

Table 5.2: Characteristics of FLCs used in [180] and G-FLCS

Parameters	[180]	G-FLCS
Inputs	2	2
Output	1	1
Antecedant MFs	7 (triangular)	7 (trapezoidal, triangular)
Consequent MFs	7 (singleton)	7 (trapezoidal, triangular)
Aggregation	MIN	MIN
Implication	MAX	MAX
MF Overlapping Degree	2	Dynamic (4)
Defuzzification Method	Weighted Average	CoA

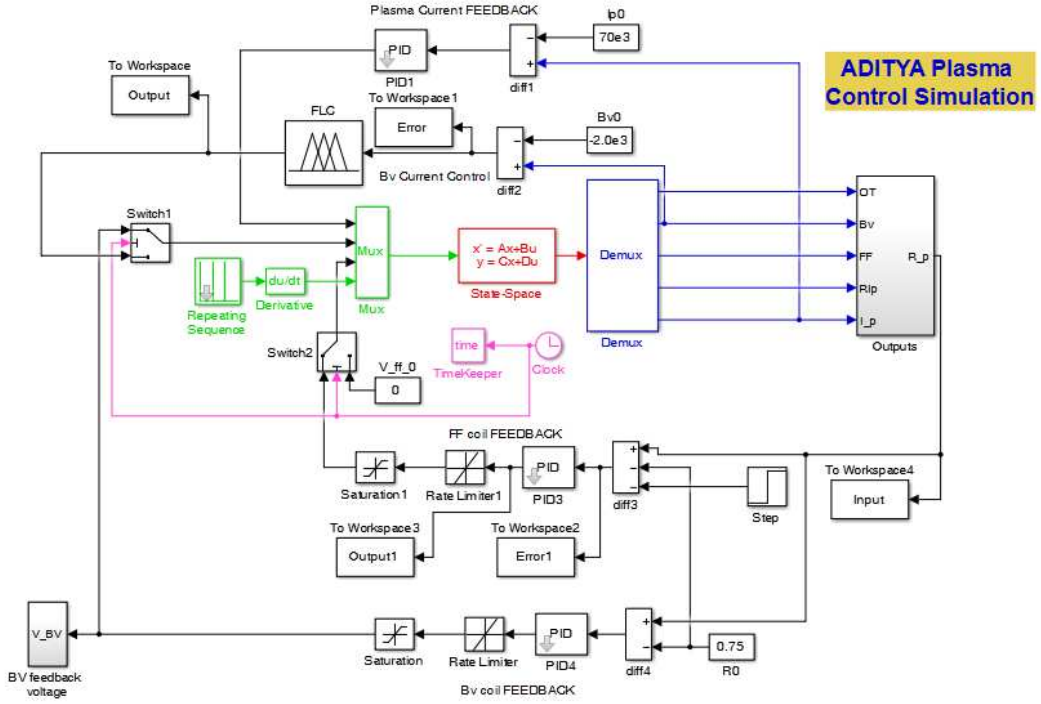


Figure 5.5: Simulink model of radial plasma position control in Aditya TFTR with FLC

5.4 Introduction to Multi Objective Genetic Algorithm

Majority of real world processes and plants operate at their best when an optimum state is achieved by balancing multiple objectives. A multi objective

optimization problem is mathematically represented as [137, 198];

$$F(x) = (f_1(x), \ldots, f_m(x))^T$$
$$x \in \Omega$$

where x is a decision vector in a Ω vector space. $F(x)$ constitutes of m number of objective functions

$$f_i : \Omega \rightarrow R, \forall i \rightarrow [1, m]$$

where R^m represents the objective space. Now in (5.4), the objective function f_i, mostly experiences trade off between each other. Thus to achieve an optimal solution is important for any engineer. The best trade off solution that can be achieved using these objective functions is called the Pareto Optimal Solutions.

Definition 4 An optimal feasible solution $x^* \in \Omega$ in (5.4) is called a pareto optimal solution

$$\text{iff } \nexists y \in \Omega$$

such that $F(y) < F(x^*)$. Set of all pareto optimal solutions

$$SPOS = \left\{x \in \Omega | \ \nexists y \in \Omega, F(y) < F\left(x^*\right)\right\}$$

Evolutionary algorithms like GA, PSO and others where population based heuristic search is conducted, can successfully approximate the whole $SPOS$ of a multi objective optimization problem. Genetic Algorithm (GA) is arguably most widely used search heuristic in the field of artificial intelligence and machine learning. It is inspired by Darwin's theory of evaluation and natural selection [198]. Figure 5.6 provides a general block diagram to the basic outline as follows.

Initial Population To generate P_0 set of valid parameters in accordance to the parameter constraints. This individual set of parameters are called chromosomes.

Fitness To evaluate fitness $F(x)$ for each chromosome x in the population P_0 and $P_k = P_0$.

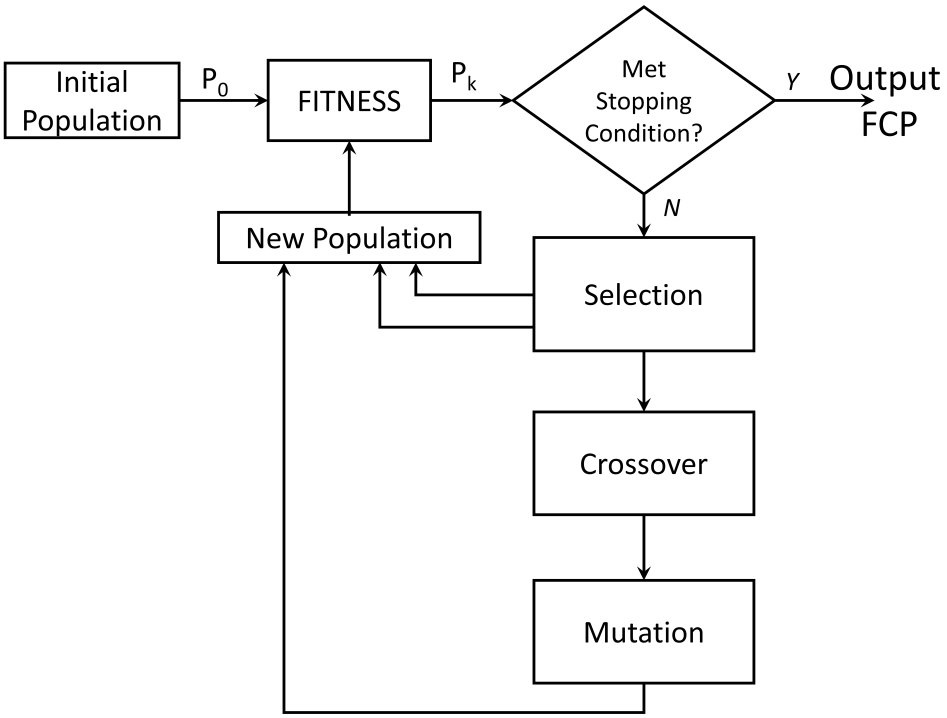


Figure 5.6: Flowchart of Genetic Algorithm

Stopping Condition Check Check if the stopping conditions for all objectives are met. If *yes* then exit and return parameter set with best fitness value.

New Population New population can be selected from existing P_k following the steps mentioned below.

a. Selection Select n best fitness chromosome from P_k. These represents the [arent chromosomes. Reject others.]

b. Crossover With the probability for crossover, generate new offspring.

c. Mutation With mutation probability, mutate the new offspring.

d. Acceptance Generate the population P_k using new population.

Loop Go to Fitness.

5.5 GA based FCP Extraction

In this section, a GSA based FCP extraction for Aditya TFTR is presented. The details of the parameters to be tuned using GS are presented in section 5.5.2. The optimization problem is as follows. Tune sixty-one FCP as mentioned in 5.5.2 such that the response of the TFTR fulfils the objectives of

1. settling time,
2. transient time, and
3. steady state error.

All these corresponds to as minimum optimization problem.

Genetic algorithm based fuzzy parameter extraction scheme, as explained in section 3.5, is implemented to derive the FCP to control radial position of plasma in Aditya TFTR model as shown in Figure 5.5. The extracted FCP is provided in Appendix B.

5.5.1 FLC I/O Identification

The parameters that is considered are
R_P Error $\rightarrow$ Input with 7 MFs (Triangular and Trapezoidal)
I_P $\rightarrow$ Input with 7 MFs (Triangular and Trapezoidal)
Control Signal $\rightarrow$ Output with 7 MFs (Triangular and Trapezoidal)

5.5.2 FLC Parameter Identification

(i) Radial Position Error R_P:
Range $\rightarrow [-0.05, 0.05]$
Parameters:

$$\left[\begin{array}{l} (R_{P_{1A}}, R_{P_{1B}}), (R_{P_{2A}}, R_{P_{2B}}, R_{P_{2C}}), (R_{P_{3A}}, R_{P_{3B}}, R_{P_{3C}}), (R_{P_{4A}}, R_{P_{4B}}, R_{P_{4C}}), \\ (R_{P_{5A}}, R_{P_{5B}}, R_{P_{5C}}), (R_{P_{6A}}, R_{P_{6B}}, R_{P_{6C}}), (R_{P_{7A}}, R_{P_{7B}}) \end{array}\right]$$

Number of parameters $n_{R_P} = 19$

(ii) Plasma Current I_P:
Range → $[5e4, 8e4]$
Parameters:

$$\begin{bmatrix} (I_{P_{1A}}, I_{P_{1B}}, I_{P_{1C}}), (I_{P_{2A}}, I_{P_{2B}}, I_{P_{2C}}), (I_{P_{3A}}, I_{P_{3B}}, I_{P_{3C}}), (I_{P_{4A}}, I_{P_{4B}}, I_{P_{4C}}), \\ (I_{P_{5A}}, I_{P_{5B}}, I_{P_{5C}}), (I_{P_{6A}}, I_{P_{6B}}, I_{P_{6C}}), (I_{P_{7A}}, I_{P_{7B}}, I_{P_{7C}}) \end{bmatrix}$$

Number of parameters $n_{I_P} = 21$
(iii) Control Signal u:
Range → $[-60, 60]$
Parameters:

$$\begin{bmatrix} (u_{1A}, u_{1B}, u_{1C}), (u_{2A}, u_{2B}, u_{2C}), (u_{3A}, u_{3B}, u_{3C}), (u_{4A}, u_{4B}, u_{4C}), \\ (u_{5A}, u_{5B}, u_{5C}), (u_{6A}, u_{6B}, u_{6C}), (u_{7A}, u_{7B}, u_{7C}) \end{bmatrix}$$

Number of parameters $n_{I_P} = 21$

5.5.3 Parameter Constraints

There are total of sixty one identified parameters in section 5.5.2. The shape of the membership functions are dependent of these parameters and therefore the relationship between these parameters are cardinal. Genetic algorithm is well equipped to handle these nonlinear equality constraints between the parameters. Listing 5.1

```
function [c, ceq] = confun_FLC(X)

% Nonlinear equality constraints
c = [X(1) + 0.001 - X(2);    % Input Rp Starts Here
X(3) + 0.001 - X(4);    X(4) + 0.001 - X(5);
X(6) + 0.001 - X(7);    X(7) + 0.001 - X(8);
X(9) + 0.001 - X(10);   X(10) + 0.001 - X(11);
X(12) + 0.001 - X(13);  X(13) + 0.001 - X(14);
X(15) + 0.001 - X(16);  X(16) + 0.001 - X(17);
X(18) + 0.001 - X(19);    % Input Rp Ends Here
X(20) + 5 - X(21);    % Control Output u starts here
```

```
X(21) + 5 - X(22);  X(23) + 5 - X(24);
X(24) + 5 - X(25);  X(26) + 5 - X(27);
X(27) + 5 - X(28);  X(29) + 5 - X(30);
X(30) + 5 - X(31);  X(32) + 5 - X(33);
X(33) + 5 - X(34);  X(35) + 5 - X(36);
X(36) + 5 - X(37);  X(38) + 5 - X(39);
X(39) + 5 - X(40)];   % Control Output u Ends Here

% Nonlinear equality constraints

ceq = [];
```

Code Snippet 5.1: Describing nonlinear equality constraints

5.5.4 Parameter Extraction

In Figure 5.11, the parameter extraction mechanism is graphically represented. Aditya TFTR Radial Plasma Position Control Simulink model is the fitness function used in GA based FCP extraction. "*sim*" command in Matlab is used to execute the Simulink model from the GA Matlab script. The operation can be seen in the Code Snippet 5.2. The *Error Calculation* block in Figure 5.11 represents the objectives computed from the cost function. These objectives are

1. settling time,
2. transient time, and
3. steady state error.

```
simOut =
    sim('aditya_fast','SimulationMode','normal','AbsTol','1e-5',...
'StopTime', '0.03', ...
'ZeroCross','on', ...
'SaveTime','on','TimeSaveName','tout', ...
'SaveState','on','StateSaveName','xoutNew',...
'SaveOutput','on','OutputSaveName','youtNew',...
```

```
'SignalLogging','on','SignalLoggingName','logsout');
results = simOut.get( 'youtNew' );
t = simOut.get( 'tout' );
```

Code Snippet 5.2: Fitness Function

```
%Steady State Error Calculation
errOut = ((0.75 - results(end))^2)^0.5;

% Calculation of Settling Time and Rise Time
results = simOut.get( 'youtNew' );
t = simOut.get( 'tout' );
sTime = stepinfo(results,t,0.749721);
```

Code Snippet 5.3: Fitness Computation

5.6 FLC Design and Implementation

The extracted parameters for the radial position control provides the default FCP for the proposed G-FLCS as discussed in the earlier chapters. The extracted FCP is described in Appendix - B. G-FLCS is connected serially through UART to the server. An optimized code for FLC with MT-FRHC ispreloaded in the SHRAM of the C6748 DSP development kit. This system is prepared for a HIL test to control the plasma position. In the Simulink model presented in Figure 5.12, the plasma position control is achieved using three different controller mechanisms, namely,

- Tradition PID controller tuned by Ziegler–Nichols method
- FLC as described by P. Suratia et. al. to control radial position of plasma in Aditya TFTR
- G-FLCS with GA based FCP extraction scheme

5.6.1 HIL Testing

The G-FLCS is connected to a PC with Simulink model of radial plasma position control of Aditya TFTR. Using UART, data can be exchanged between the two system. The hardware G-FLCS polls for any input at the UART. Once it receives the input, it completes the FLC with MT-FRHC execution process to return a suitable control signal to the power supply of the feedback output. This completes a HIL loop and it is continued for the entire simulation. Simulation for other controllers are carried out sequentially by chnaging the manual switches as shown in Figure 5.12. These simulation data are recorded for all control schemes and analyzed for the performance of the controllers.

5.7 Performance Analysis

The recorded data from the simulations explained in previous section is plotted as depicted in Figure 5.13. This plot clearly displays the difference in the control action. A significant improvement in rise time and settling time is observed in accordance to the PID controller and existing FLC. The hardware G-FLCS is observed to provide a smooth and fast response. It caters a robust control scheme for the radial position control in Aditya TFTR. A comparative analysis of the control parameters are drawn and tabulated in Table. 5.3. It can be observed that the G-FLCS 59% faster rise time and 87% speedy settling time in comparison to existing control schemes. However, a slight overshoot of 0.0009 m is reported by the proposed G-FLCS. This overshoot implies a plasma displacement of 9 mm in a vacuum chamber of 750 mm which is effectively 1.2 % of the radius.

5.8 Summary

This chapter implements the proposed MT-FRHC based G-FLCS with VBCoA on C6748 DSP in a critical and highly nonlinear control problem. In Aditya TFTR, the confinement of plasma within the vacuum chamber is crucial. Therefore, this system requires a fast and robust control algorithm. The GA based

Table 5.3: Comparison of performance parameters of PID, FLC[180] and G-FLCS

Parameters	PID	FLC[180]	GFLCS
Rise Time	0.0062	0.0062	0.0025
Settling Time	0.1255	NaN	0.0160
Settling Min	0.7483	0.7474	0.7483
Settling Max	0.7497	0.7486	0.7509
Overshoot	0	0	0.0009
Undershoot	0	0	0
Peak	0.7497	0.7486	0.7509
Peak Time	0.2	0.0235	0.0154

FCP extraction algorithms generates the parameters to drive the hardware G-FLCS. The proposed controller is then serially interfaced to the Simulink model through UART and tested with other control applications. The observation obtained from this system was exciting as it provided 59% faster rise time and 87% speedy settling time in comparison to existing control schemes. These results are extremely positive and encouraging.

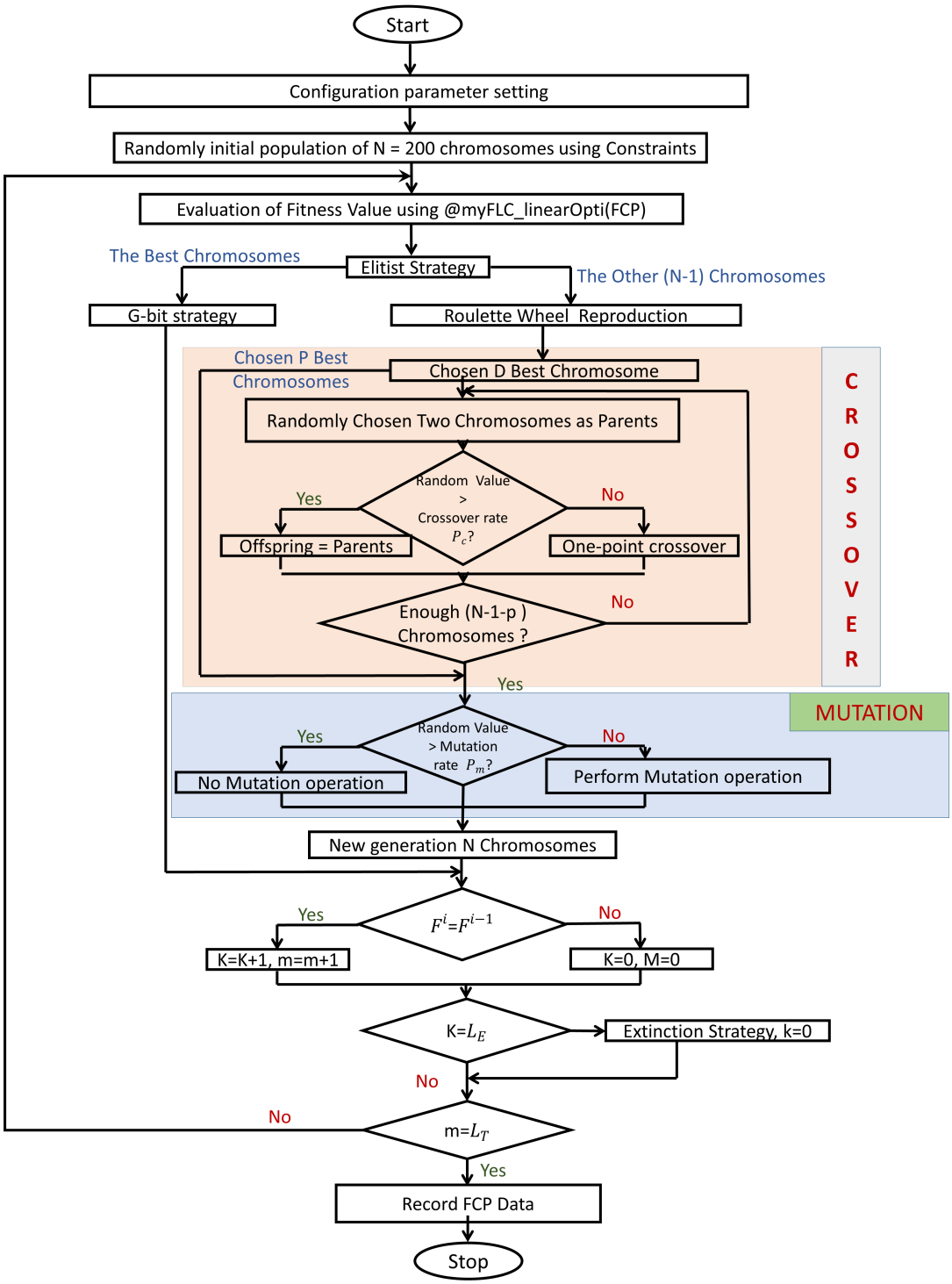


Figure 5.7: Flowchart of Genetic Algorithm

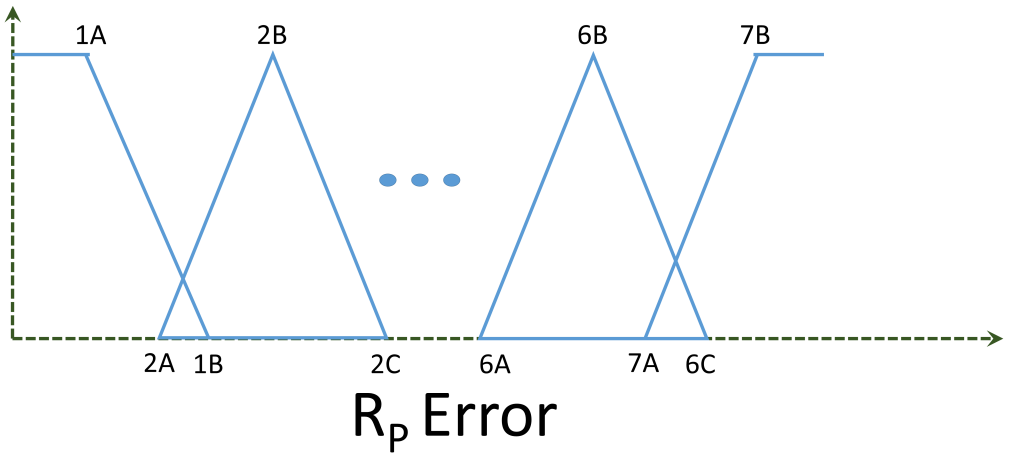


Figure 5.8: MF co-ordinates for Parameter Extraction: Radial Position Error

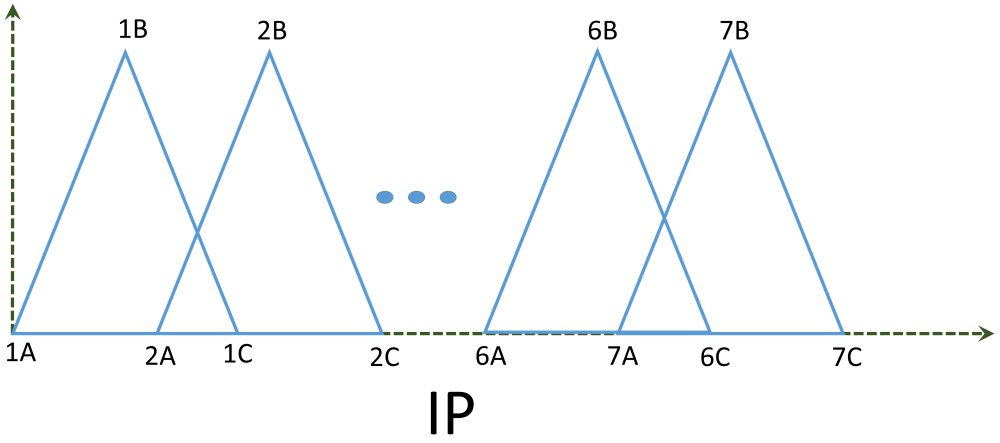


Figure 5.9: MF co-ordinates for Parameter Extraction: Plasma Current

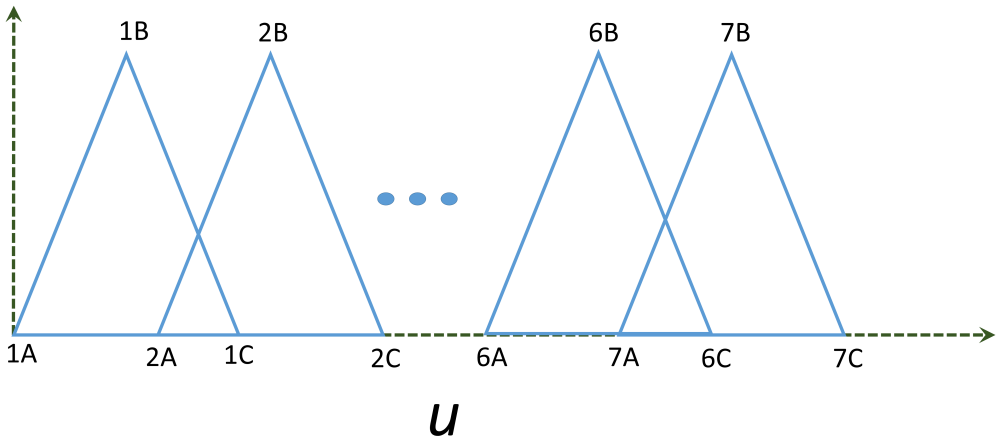


Figure 5.10: MF co-ordinates for Parameter Extraction: Control Signal

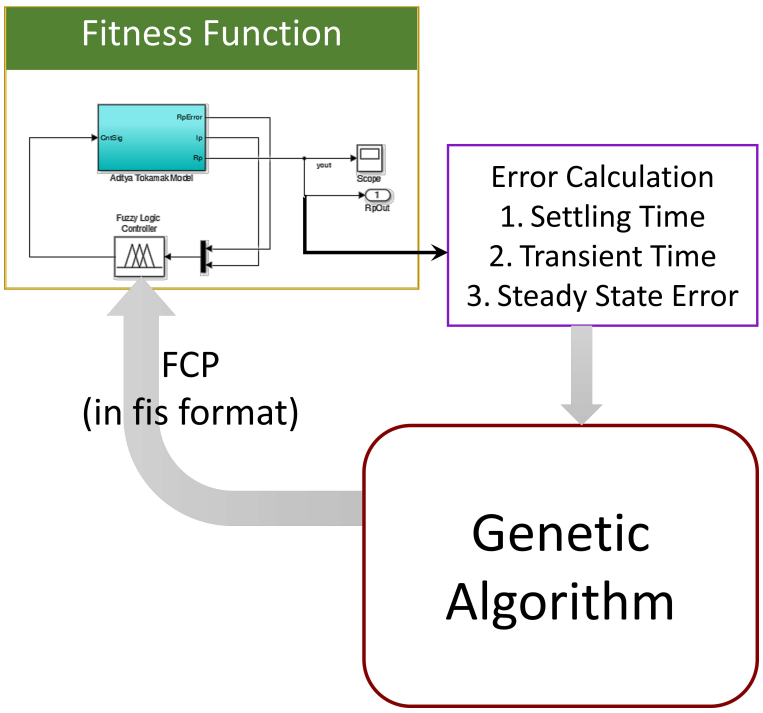


Figure 5.11: Block Diagram for FCP Extraction for Radial Position Control in Aditya TFTR

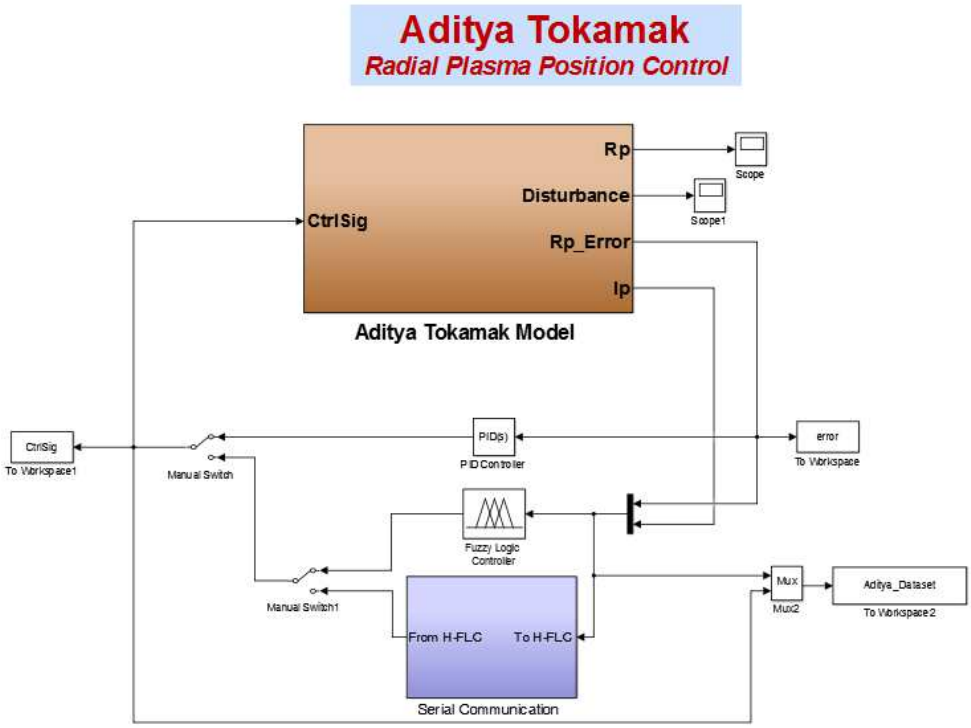
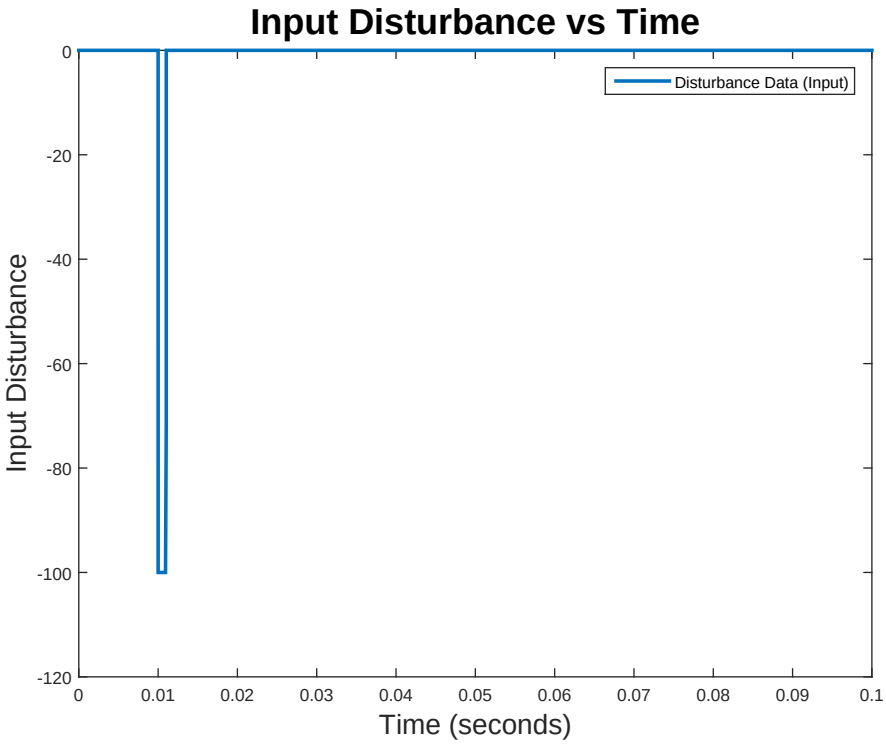
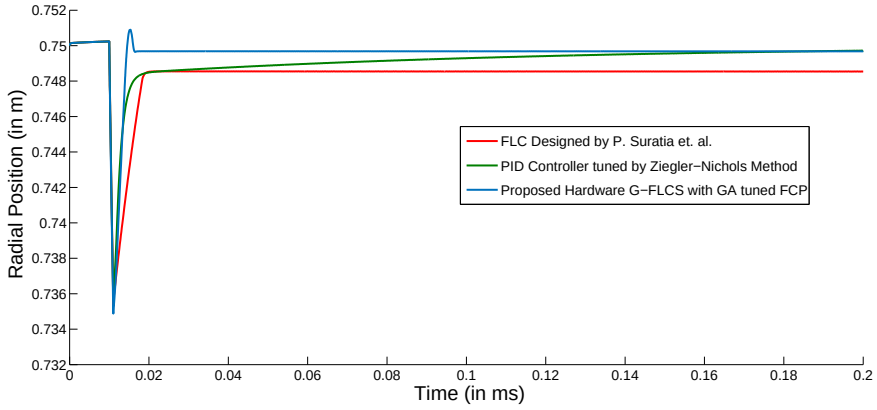


Figure 5.12: Radial Plasma Position Control of Aditya TFTR: HIL Simulation with PID, FLC[180] and G-FLCS



(a) Input Disturbance Signal



(b) System Response

Figure 5.13: Performance of various controllers in presence of disturbances in plasma position

Chapter 6

Conclusion

Preview

This chapter concludes the thesis and summarizes the findings during this research work. It also provides the limitations of this work and points to new dimensions in which the current work can be enhanced.

The development of the generic remotely tunable MT-FRHC based FLCS with VBCoA defuzzification in programmable hardware opens a line of approach to several explorations. This architecture provides a large number of functionalities to its users along with sufficient speed to drive most industrial processes like plasma position control in TFTR, water level control in coupled tank, etc. This system is standardized with MATLAB Fuzzy Logic Toolbox and has the ability to incorporate FIS files generated by this toolbox. The proposed systems are observed to perform well within the multiple testing paradigms mentioned in this work. Thorough investigations are done using multiple applications to ascertain generality and applicability of G-FLCS to various control applications.

6.1 Summarized Results

A brief overview of the various work done in the entire duration of theÂăproject is necessary to conclude this thesis. In chapter 2, MT-FRHC rule reduction technique is mathematically established and analyzed in comparison to the predominant FRHC technique. Both of the technique was implemented on a DSP hardware for timing analysis. MT-FRHC based G-FLCS achieved 27 % higher performance in terms of speed compared to the OMF based G-FLCS. Without a code optimization based on memory and speed, the MT-FRHC based G-FLCS on DSP achieved a speed of around 10 KFLIPS. This chapter also introduced vertices based CoA computation technique and compared to Riemann sum based CoA computation. It was observed that on average, the proposed VBCoA consumed 5314 machine cycles in compared to 7640 machine cycles consumed by Riemann sum based CoA computation.

The system architecture for MT-FRHC based G-FLCS is presented in chapter 3. A WebUI is developed on ASP.NET and hosted using MS IIS7 for remote reconfigurability and tunability. A Genetic Algorithm based FCP extraction scheme was also described. The modules and the submodules supporting the proposed G-FLCS were explained in this chapter. This chapter does not provide any result ; however, it lays the essential ground work required for the actual implementation of the proposed G-FLCS.

In chapter 4, the proposed MT-FRHC based G-FLCS with VBCoA defuzzification was implemented on a DSP hardware. The code was developed in C language and further optimized using linear ASM and intrinsic functions to achieve a 5 % improvement on the code size. The proposed design was compared to existing designs that closely matches to the objective of this work. It was observed that proposed DSP based G-FLCS provided a speed of more than 13 KFLIPS in comparison to 5.5 KFLIPS [122] and 11 KFLIPS [56].

Table 6.1: Comparison between Proposed hardware G-FLCS and Similar Designs based of Reconfigurable Parameters

Year	Reference	Speed (in FLIPS)	Platform
2008	Millan et. al.[122]	5.5 K	FPGA
2010	Yi Fu et. al.[56]	11 K	FPGA
Proposed G-FLCS		13 K	DSP

Chapter 5 implements the proposed MT-FRHC based G-FLCS with VBCoA on C6748 DSP to control radial plasma position of Aditya TFTR. It was compared to an existing FLCS designed exclusively for this application [180]. Compared to the presently deployed control techniques, the proposed system achieved 59% faster rise time and 87% speedy settling time. However, a slight overshoot of 0.0009 m is reported by the proposed G-FLCS. This overshoot implies a plasma displacement of 9 mm in a vacuum chamber of 750 mm which is effectively 1.2 % of the radius.

6.2 Contribution of this Thesis

In summary, this research successfully contributes

- A G-FLCS module with WebUI has been proposed that can operate as standalone remotely tunable controller. All existing G-FLCS, had no user interactions [56, 122] and hence even though they were developed on field programmable hardware, the system architecture do not allow field programmability of the G-FLCS. The proposed MT-FRHC based GFLCS sys-

tem can be programmed through the web interface. The novelty of this application lies in its system architecture which is elaborated in Chapter 3 and 4.

- A code optimization process is implemented to develop memory consumption and speed optimized G-FLCS controller on C6748 DSP processor. A 5% memory saving was observed after this optimization process. Industry standard code optimization techniques are used in this work and they are explained in section 4.2.1. However, the standard HIL testing process involves high end debuggers and emulation devices. They also operate on various copyright protocols. However, in section 4.2.2, a naive UART based HIL testing process was described that provided performance and timing analysis of the proposed G-FLCS.

- MT-FRHC rule reduction technique ensures that the proposed G-FLCS achieves an operating speed of around 10K FzLOPS.

- A GA based FCP extraction algorithm in conjunction with a Fuzzy PID approximation based initial FCP generation technique was proposed. Section 4.4.3 shows GA based FCP extraction technique to drive the G-FLCS for various control problems. The proposed method extracts minimal rules to reduce complexity. However, it does not guarantee a minimal number of overlaps among the input membership function. But, MT-FRHC algorithm reduces the complexity by dynamically controlling the overlaps. Thereby, together MT-FRHC and GA based G-FLCS provides a balanced performance that can be observed from section 5.7.

- A vertices based centroid computations for polygons were extensive used in geospatial applications [174]. In section 2.3.2, VBCoA defuzzification scheme is proposed. A novel algorithm for computation of vertices and its co-ordinates was proposed. This was observed to speed up the defuzzification process significantly compared to the widely used Reimann Integral Sum based CoA computation.

- Finally the G-FLCS is implemented to control radial plasma position of

Aditya TFTR model. The observation obtained from this system was exciting as it provided 59% faster rise time and 87% speedy settling time in comparison to existing control schemes.

6.3 Limitations of this Work

The major limitations of the proposed MT-FRHC based G-FLCS with VBCoA design can be summarized as follows:

- G-FLCS is developed on a programmable device. The methodology of G-FLCS is implemented using a DSP which works with a sequential code. A parallel architecture developed on FPGA or hybrid computing platform would unleash the complete power of the proposed method.
- G-FLCS is tested in HIL environment with Simulink models and not with practical systems. Although the HIL test results are promising, but a real-time testing is will assure that the system performs as expected.
- G-FLCS is connected to a server PC using standard protocol. In this implementation the security of the data communication network is not stressed upon. It is indispensable to evaluate the network security and analyze the threats.
- The proposed methods of GA based FCP extraction and Fuzzy PID approximation based initial FCP generation requires knowledge about the dynamics of a process plant. The actual essence of an FLCS is that it requires no knowledge about the dynamics of the plant. Although the basic MT-FRHC based G-FLCS with VBCoA operates on the ideology of a typical fuzzy system, the GA based FCP extraction and Fuzzy PID approximation based initial FCP generation is not applicable to process plants where the dynamics of the system is unknown.

6.4 Few Scope for Future Work

The work presented in this thesis elaborates the design and implementation of a MT-FRHC based G-FLCS with VBCoA defuzzification method. This design has potential for wide explorations. Some of the significant area of future work includes the following.

- The proposed G-FLCS architecture is implemented on the type-I Mamdani fuzzy logic control system. This architecture has been implemented using modular design methodology. The modules in proposed G-FLCS can be integrated with neural network to achieve a DSP based generic neuro-fuzzy system. There are various applications based on neuro-fuzzy systems [85, 188]. These designs have been implemented on FPGA platform. As it has been already established that the proposed G-FLCS architecture can perform better than similar designs on FPGA. It will be interesting to analyse the performance of DSP based generic neuro-fuzzy system.

- An ASIP implementation of MT-FRHC based G-FLCS with VBCoA can be achieved. The proposed G-FLCS architecture is developed on a general purpose DSP processor. The deign of G-FLCS includes many modules which have been implemented using linear ASM. There are many RISC based fuzzy processor reported in the literature [33, 164]. The proposed MT-FRHC based G-FLCS with VBCoA can be introduced in the instruction set of these RISC based fuzzy processors. To realize this concept, MT-FRHC rule-reduction process and VBCoA defuzzification method have been implemented using linear ASM. However, the conversion of these linear ASM functions into individual instructions and integrating these instructions into the existing instruction set can be a challenging aspect.

- This concept of G-FLCS can be extended to include Type-II Fuzzy sets. The Type-II Fuzzy sets have included uncertainty in membership functions that makes it more complex and challenging area of research. There are various industrial application based on type-II FLCS [102, 163]. These designs can also be implemented in proposed G-FLCS architecture; by extrapolating the design reported in this thesis from type-I fuzzy sets to

type-II fuzzy set. The resultant design can achieve a speed better than reported in these designs [93, 148]. It has already been shown in the thesis that the present G-FLCS architecture achieves the speed higher than similar designs implemented on FPGAs.

Appendix A

A.1 Fuzzy Parameter Files

Download Links to FIS Structure File

- Fuzzy PI approximation to control ACDC motor
  http://goo.gl/zhanJN
- Water level control in a two tank system
  http://goo.gl/9CWP8e
- Truck backer control system
  http://goo.gl/HX2Db5
- Test Parameter File (Used in hardware realization)
  http://goo.gl/LtqBl4
- FCP File (Used in [180])
  https://goo.gl/A2MLP8

Appendix B

B.1 GA based Extracted FCP for Radial Position Control

[$System$]
Name='GA_Tuned_FCP'
Type='mamdani'
Version=2.0
NumInputs=2
NumOutputs=1
NumRules=49
AndMethod='min'
OrMethod='max'
ImpMethod='min'
AggMethod='max'
DefuzzMethod='centroid'

[$Input1$]
Name='R_PError'
Range=[-0.05 0.05]
NumMFs=7
MF1='fin':'trapmf',[-0.065 -0.05167 -0.0420000000000024 -0.0280000000000052]
MF2='in':'trimf',[-0.033 -0.0250000000000001 -0.018]

MF3='jin':'trimf',[-0.0219999999999999 -0.011 0]
MF4='g':'trimf',[-0.002 0 0.002]
MF5='jout':'trimf',[0 0.012 0.022]
MF6='out':'trimf',[0.018 0.0265482226169141 0.032]
MF7='fout':'trapmf',[0.028 0.039 0.05167 0.06503]

[$Input2$]
Name='I_P'
Range=[50000 80000]
NumMFs=7
MF1='ln':'trimf',[50000 55000 60000]
MF2='sn':'trimf',[66920 68920 70920]
MF3='mn':'trimf',[58250 63250 68250]
MF4='tiny':'trimf',[70500 71500 72500]
MF5='sp':'trimf',[72000 73500 75000]
MF6='mp':'trimf',[74500 75000 76000]
MF7='lp':'trimf',[75800 77500 79000]

[$Output1$]
Name='CtrlSig'
Range=[-60 60]
NumMFs=7
MF1='LN':'trimf',[-59.7629179528005 -43.2499999974027 -23.2500000025973]
MF2='MN':'trimf',[-59.7880292411719 -32.5 -7]
MF3='SN':'trimf',[-35.3483494741782 -15.4275784230086 -0.052578424943468]
MF4='Tiny':'trimf',[-5 2.69288883575541e-16 5]
MF5='SP':'trimf',[1.93970084438888e-09 16.3124999926697 34.1249999942683]
MF6='MP':'trimf',[0.430775859048123 22.8750000025973 51.8749999974027]
MF7='LP':'trimf',[25.0000000059503 47.0000000036437 59.999999995951]

Rulebase

[*Rules*]
7 1, 7 (1) : 1
7 3, 7 (1) : 1
7 2, 7 (1) : 1
7 4, 4 (1) : 1
7 5, 7 (1) : 1
7 6, 7 (1) : 1
7 7, 7 (1) : 1
6 1, 6 (1) : 1
6 2, 6 (1) : 1
6 3, 6 (1) : 1
6 4, 4 (1) : 1
6 5, 6 (1) : 1
6 6, 6 (1) : 1
6 7, 6 (1) : 1
5 1, 5 (1) : 1
5 2, 5 (1) : 1
5 3, 5 (1) : 1
5 4, 4 (1) : 1
5 5, 5 (1) : 1
5 6, 5 (1) : 1
5 7, 5 (1) : 1
4 1, 4 (1) : 1
4 2, 4 (1) : 1
4 3, 4 (1) : 1
4 4, 4 (1) : 1
4 5, 4 (1) : 1
4 6, 4 (1) : 1
4 7, 4 (1) : 1
3 1, 3 (1) : 1
3 2, 3 (1) : 1
3 3, 3 (1) : 1
3 4, 4 (1) : 1
3 5, 3 (1) : 1
3 6, 3 (1) : 1
3 7, 3 (1) : 1
2 1, 2 (1) : 1
2 2, 2 (1) : 1
2 3, 2 (1) : 1
2 4, 4 (1) : 1
2 5, 2 (1) : 1
2 6, 2 (1) : 1
2 7, 2 (1) : 1
1 1, 1 (1) : 1
1 2, 1 (1) : 1
1 3, 1 (1) : 1
1 4, 4 (1) : 1
1 5, 1 (1) : 1
1 6, 1 (1) : 1
1 7, 1 (1) : 1

Appendix C

C.1 Experiment 1: Automatic Cruise Control System for Cars[12]

C.1.1 Aim

To design and develop a cruise control system of a car.

C.1.2 System Modeling

Consider a vehicle of mass m moving at a velocity v. A force F is generated from the engine while a disturbance force F_d is resisting motion of the vehicle [12, 15]. Therefore, the equation of motion of the vehicle is given by

$$m\frac{dv}{dt} = F - F_D \tag{C.1}$$

The vehicle engine generates force F which is proportional to the rate of injected fuel in the engine. This phenomenon in turn controls the throttle of the vehicle. Torque produced T at engine speed ω can be mathematically represented as

$$T(\omega) = T_m\left(1 - \beta\left(\frac{\omega}{\omega_m} - 1\right)^2\right) \tag{C.2}$$

where T_m is maximum torque generated by the engine at full throttle to attain ω_m, the maximum engine speed with torque coefficient β. For a gear ratio n

and wheel radius r, current velocity can be related to the engine speed by (4.2)

$$\omega = \frac{n}{r} v = \alpha_n v \tag{C.3}$$

Therefore the driving force can be computed as

$$F = \frac{nu}{r} T(\omega) = \alpha_n u T(\alpha_n v) \tag{C.4}$$

Basically there are three major disturbance forces working on the vehicle namely, gravitational force (F_G), rolling friction of the road and vehicle tires (F_R), and aerodynamic drag due to the body of the vehicle (F_A).

$$F_D = F_G + F_R + F_A \tag{C.5}$$

The gravitational force acting on the vehicle can be modeled based on the slope of the roads.

$$F_G = mg\sin\theta \tag{C.6}$$

$$F_R = mgC_r sgn(v) \tag{C.7}$$

$$F_A = \frac{1}{2}\rho C_d A v^2 \tag{C.8}$$

Combining (C.5) and (C.6), (C.1) becomes

$$m\frac{dv}{dt} = \alpha_n u T(\alpha_n v) - mg\sin\theta - mgC_r \mathrm{sgn}(v) - \frac{1}{2}\rho C_d A v^2 \tag{C.9}$$

C.1.3 Controller Design and Tuning

A PI controller was used to benchmark the performance of this control problem. The designed PI controller was tuned using Ziegler-Nichols method with help of Matlab Control System Toolbox. The controller gains are specified in Table C.1.

Table C.1: Proportional and Integral Gains in ACC System

Gain	Value
K_P	0.1
K_I	0.5

C.2 Experiment 2: Two Tank Water Level Control [93]

C.2.1 Aim

To control water level in a coupled tank.

C.2.2 System Modeling

Consider the coupled tank system in Figure C.1. The system comes from two flow balances and the non-linear equations for flow appear through the valves. When the valves are assumed to have ideal orifice, the system nonlinearity is described by square root law. The flow balance equations are,

$$Q_i - C_{db}a_b\sqrt{2g(H_1-H_2)} = A\frac{dH_1}{dt}$$
$$C_{db}a_b\sqrt{2g(H_1-H_2)} - C_{db}a_b\sqrt{2gH_2} = A\frac{dH_2}{dt}$$

C.2.3 Controller Design and Tuning

For the controller design, the above equations are linearized. This is done assuming small variations in q_i in Q_i, h_1 in H_1, h_2 in H_2. After linearizing of the above equations can be presented as,

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} k_{11} & k_{12} \\ k_{21} & k_{22} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} A^{-1} \\ 0 \end{bmatrix} q_i$$
$$\begin{bmatrix} h_1 \\ h_2 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

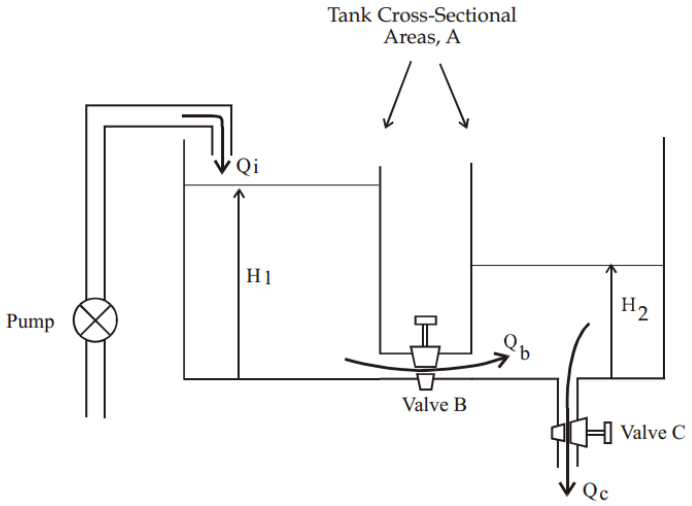


Figure C.1: Coupled Tanks System [93]

The transfer function model is as follows

$$\frac{h_2(s)}{q_i(s)} = \frac{G}{(T_1 s+1)(T_2 s+1)}$$

Implementing a PI controller,

$$y(s) = \frac{g(k_i + k_p s)r(s)}{Ts^2 + s(1+gk_p) + gk_i} + \frac{(gs)d(s)}{Ts^2 + s(1+gk_p) + gk_i}$$

With natural frequency of 0.01 Hz and a damping factor of 1, the control system parameters were set up to be $k_i = 0.1$ and $k_p = 2.7$

C.3 Experiment 3: Armature Controlled DC Motor[170]

C.3.1 Aim

To control speed of a DC motor.

C.3.2 System Modeling

Consider the Figure C.2, which shows the operation of an Armature controlled DC motor [113, 170].

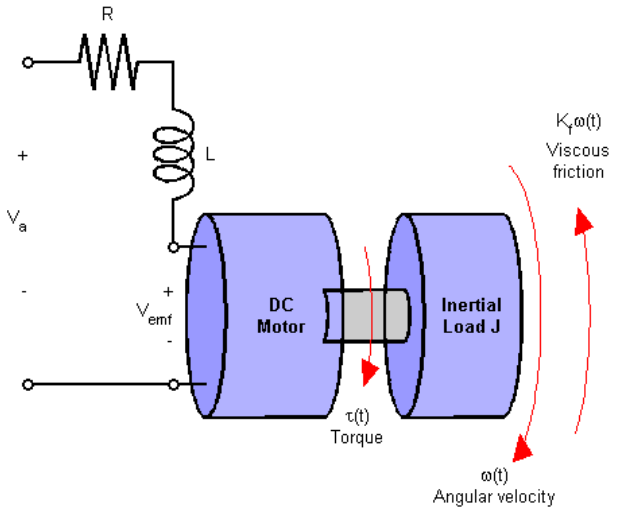


Figure C.2: Armature Controlled DC Motor

The physical constants of the control problem are Armature Resistance (R_A) = 1 Ω, Armature Inductance (L_A) = 0.5, Inertia (J_M) = 0.01, Damping (B_M) = 0.1, Torque Constant (K_τ) 0.01 Nm/A and Back EMF Constant (K_B) = 0.01 Vs/rad.

Transfer function of the plant model is stated as,

$$\frac{\theta(s)}{V_A(s)} = \frac{K_\tau}{L_A J_M s^3 + (R_A J_M + L_A B_M) s^2} \text{ x } \frac{1}{(K_\tau K_B + R_A B_M) s} \tag{C.10}$$

C.3.3 Controller Design and Tuning

The above model is loaded with Torque T_D. A PID controller is employed to obtain a smooth control of the Armature Controlled DC motor. The controller is tuned using Mathworks Systune PID Tuner. The controller gains are tabulated in Table C.2.

Table C.2: Controller Gains in Speed Control of Armature Controlled DC Motor

Gain	Value
K_P	17.94
K_I	43.45
K_D	-0.78

Bibliography

[1] J. J. Acevedo, B. n. C. Arrue, J. M. Diaz-Bañez, I. Ventura, I. Maza, and A. Ollero, "One-to-one coordination algorithm for decentralized area partition in surveillance missions with a team of aerial robots," *Journal of Intelligent and Robotic Systems: Theory and Applications*, vol. 74, no. 1-2, pp. 269–285, Oct. 2014.

[2] B. Adhavan and C. S. Ravichandran, "FPGA implementation to minimize torque ripples in permanent magnet synchronous motor driven by field oriented control using fuzzy logic controller," *Journal of Theoretical and Applied Information Technology*, vol. 61, no. 2, pp. 369–377, 2014.

[3] A. Al Nabulsi and R. Dhaouadi, "Efficiency optimization of a DSP-based standalone PV system using fuzzy logic and dual-MPPT control," *IEEE Transactions on Industrial Informatics*, vol. 8, no. 3, pp. 573–584, Aug. 2012.

[4] R. Alcalá, J. Alcalá-Fdez, M. J. Gacto, and F. Herrera, "Fuzzy rule reduction and tuning of fuzzy logic controllers for a HVAC system," in *Studies in Fuzziness and Soft Computing*. Springer Berlin Heidelberg, 2006, vol. 201, pp. 89–117.

[5] ——, "Rule base reduction and genetic tuning of fuzzy systems based on the linguistic 3-tuples representation," *Soft Computing*, vol. 11, no. 5, pp. 401–419, Jun. 2007.

[6] Altera Corporation, "White Paper FPGA vs . DSP Design Reliability and Maintenance," Altera Corporation, Tech. Rep., 2007.

[7] J. F. M. Amaral, J. L. M. Amaral, C. C. Santini, M. A. C. Pacheco, R. Tanscheit, and M. H. Szwarcman, "Intrinsic evolution of analog circuits on a programmable analog multiplexer array," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 3038, pp. 1273–1280, 2004.

[8] S. Aminifar and A. B. Marzuki, "Voltage-mode fuzzy logic controller," *Indian Journal of Science and Technology*, vol. 5, no. 11, pp. 3630–3633, 2012.

[9] R. Amirkhanzadeh, A. Khoei, and K. Hadidi, "A mixed-signal current-mode fuzzy logic controller," *AEU - International Journal of Electronics and Communications*, vol. 59, no. 3, pp. 177–184, Jun. 2005.

[10] F. Aqlan and E. Mustafa Ali, "Integrating lean principles and fuzzy bow-tie analysis for risk assessment in chemical industry," *Journal of Loss Prevention in the Process Industries*, vol. 29, no. 1, pp. 39–48, May 2014.

[11] N. K. Arun and B. M. Mohan, "Mathematical models and computational aspects of the simplest fuzzy two-term controllers," in *IFAC Proceedings Volumes (IFAC-PapersOnline)*. IFAC Secretariat, 2014, pp. 882–889.

[12] K. J. Aström and R. M. Murray, *Feedback Systems: An Introduction for Scientists and Engineers*, 2nd ed. Princeton University Press, 2009.

[13] D. Atherton, "PID controller tuning," *Computing & Control Engineering Journal*, vol. 10, no. 2, p. 44, 1999.

[14] D. Atherton and S. Majhi, "Limitations of PID controllers," in *Proceedings of the 1999 American Control Conference (Cat. No. 99CH36251)*, vol. 6. IEEE, 1999, pp. 3843–3847.

[15] J. Awrejcewicz and Z. Koruba, *Classical mechanics. Applied mechanics and mechatronics*. Springer New York LLC, 2012.

[16] M. D. Baldania, D. A. Sawant, and A. B. Patki, "Fuel saving of an automobile using fuzzy logic based embedded controller," in *2014 IEEE International Conference on Advanced Communications, Control and Computing Technologies*. IEEE, May 2014, pp. 136–140.

[17] I. Bandyopadhyay, S. M. Ahmed, P. K. Atrey, S. B. Bhatt, R. Bhattacharya, M. B. Chaudhury, S. P. Deshpande, C. N. Gupta, R. Jha, Y. S. Joisa, V. Kumar, R. Manchanda, D. Raju, C. V. S. Rao, P. Vasu, and t. A. Team, "Modelling of Ohmic discharges in ADITYA tokamak using the Tokamak Simulation Code," *Plasma Physics and Controlled Fusion*, vol. 46, no. 9, pp. 1443–1453, 2004.

[18] I. Bandyopadhyay, S. P. Deshpande, and S. Chaturvedi, "Design analysis of plasma position control in SST1," *Fusion Engineering and Design*, vol. 54, no. 2, pp. 151–166, 2001.

[19] I. Bandyopadhyay, D. Raju, and S. Deshpande, "Modelling of advanced plasma configurations in SST-1 tokamak," *Nuclear Fusion*, vol. 46, no. 3, pp. S62–S71, 2006.

[20] R. Bandyopadhyay, U. K. Chakraborty, and D. Patranabis, "Autotuning a PID controller: A fuzzy-genetic approach," *Journal of Systems Architecture*, vol. 47, no. 6, pp. 663–673, 2001.

[21] N. C. Basjaruddin, Kuspriyanto, D. Saefudin, E. Rakhman, and A. M. Ramadlan, "Overtaking assistant system based on fuzzy logic," *Telkomnika (Telecommunication Computing Electronics and Control)*, vol. 13, no. 1, pp. 76–84, 2015.

[22] D. Batten, S. Jinturkar, J. Glossner, M. Schulte, and P. D'Arcy, "New approach to DSP intrinsic functions," in *Proceedings of the Hawaii International Conference on System Sciences*. IEEE, 2000, p. 214.

[23] I. Baturone, S. Sanchez-Solano, A. Barriga, and J. L. Huertas, "CMOS fuzzy controllers implemented as mixed-signal ICs," in *Proceedings - IEEE International Symposium on Circuits and Systems*, vol. 3. IEEE, 1996, pp. 422–425.

[24] L. Behera and I. Kar, *Intelligent Systems and Control: Principles and Application*, 1st ed. Oxford University Press, 2009.

[25] S. Bennett, "A brief history of automatic control," *Control Systems, IEEE*, vol. 16, no. 3, pp. 17–25, Jun. 1996.

[26] A. Benzekri and A. Azrar, "FPGA-Based Design Process of a Fuzzy Logic Controller for a Dual-Axis Sun Tracking System," *Arabian Journal for Science and Engineering*, vol. 39, no. 8, pp. 6109–6123, 2014.

[27] S. B. Bhatt, D. Bora, and B. N. Buch, "ADITYA: The first Indian tokamak," *Indian Journal of Pure and Applied Physics*, vol. 7, no. 9, pp. 710–742, 1989.

[28] C. Bhende, S. Mishra, and S. Jain, "TS-Fuzzy-Controlled Active Power Filter for Load Compensation," *IEEE Transactions on Power Delivery*, vol. 21, no. 3, pp. 1459–1465, Jul. 2006.

[29] D. C. M. Bilsby, R. L. Walke, and R. W. M. Smith, "Comparison of a programmable DSP and a FPGA for real-time multiscale convolution," 1998.

[30] C. M. Bishop, P. S. Haynes, M. E. U. Smith, T. N. Todd, D. L. Trotman, and C. G. Windsor, "Real-time control of a Tokamak plasma using neural networks," in *Neural Computation*, 1995, vol. 7, no. 1, pp. 1007–1013.

[31] S. Bogdan, B. Birgmajer, and Z. Kovačić, "Model predictive and fuzzy control of a road tunnel ventilation system," *Transportation Research Part C: Emerging Technologies*, vol. 16, no. 5, pp. 574–592, Oct. 2008.

[32] S. Bogdan and Z. Kovacic, *Fuzzy Controller Design: Theory and Applications*, 1st ed. CRC press, Taylor and Francis, 2006.

[33] G. Bosque, I. Del Campo, and J. Echanobe, "Fuzzy systems, neural networks and neuro-fuzzy systems: A vision on their hardware implementation and platforms over two decades," *Engineering Applications of Artificial Intelligence*, vol. 32, pp. 283–331, Jun. 2014.

[34] H. Boubertakh, M. Tadjine, and P. Y. Glorennec, "A new mobile robot navigation method using fuzzy logic and a modified Q-learning algorithm," *Journal of Intelligent and Fuzzy Systems*, vol. 21, no. 1-2, pp. 113–119, 2010.

[35] H. Boumaaraf, A. Talha, and O. Bouhali, "A three-phase NPC grid-connected inverter for photovoltaic applications using neural network MPPT," *Renewable and Sustainable Energy Reviews*, vol. 49, pp. 1171–1179, Sep. 2015.

[36] M. Brox and S. Sanchez-Solano, "Development of IP Modules of Fuzzy Controllers for the Design of Embedded Systems on FPGAs," in *2006 International Conference on Field Programmable Logic and Applications*. IEEE, 2006, pp. 1–2.

[37] M. Brox, S. Sanchez-Solano, E. Del Toro, P. Brox, and F. J. Moreno-Velo, "CAD tools for hardware implementation of embedded fuzzy systems on FPGAs," *IEEE Transactions on Industrial Informatics*, vol. 9, no. 3, pp. 1635–1644, Aug. 2013.

[38] C. Buccella, C. Cecati, and H. Latafat, "Digital Control of Power Converters-A Survey," *IEEE Transactions on Industrial Informatics*, vol. 8, no. 3, pp. 437–447, Aug. 2012.

[39] C. B. Butt, M. A. Hoque, and M. A. Rahman, "Simplified fuzzy-logic-based MTPA speed control of IPMSM drive," *IEEE Transactions on Industry Applications*, vol. 40, no. 6, pp. 1529–1535, 2004.

[40] C. Cecati, F. Ciancetta, and P. Siano, "A Multilevel Inverter for Photovoltaic Systems With Fuzzy Logic Control," *IEEE Transactions on Industrial Electronics*, vol. 57, no. 12, pp. 4115–4125, 2010.

[41] R. Chassaing, *Digital Signal Processing and Applications with the C6713 and C6416 DSK*, 1st ed. John Wiley & Sons, Ltd, 2005.

[42] B. S. Chen, B. K. Lee, and L. B. Guo, "Optimal Tracking Design for Stochastic Fuzzy Systems," *IEEE Transactions on Fuzzy Systems*, vol. 11, no. 6, pp. 796–813, 2003.

[43] A. Costa, A. De Gloria, P. Faraboschi, A. Pagni, and G. Rizzotto, "Hardware solutions for fuzzy control," *Proceedings of the IEEE*, vol. 83, no. 3, pp. 422–434, Mar. 1995.

[44] R. D'Amore, O. Saotome, and K. Kienitz, "A two-input, one-output bit-scalable architecture for fuzzy processors," *IEEE Design & Test of Computers*, vol. 18, no. 4, pp. 56–64, 2001.

[45] T. Das and I. Kar, "Design and implementation of an adaptive fuzzy logic-based controller for wheeled mobile robots," *IEEE Transactions on Control Systems Technology*, vol. 14, no. 3, pp. 501–510, May 2006.

[46] O. Demir, I. Keskin, and S. Cetin, "Modeling and control of a nonlinear half-vehicle suspension system: A hybrid fuzzy logic approach," *Nonlinear Dynamics*, vol. 67, no. 3, pp. 2139–2151, Jul. 2012.

[47] M. Description and P. W. U. Anti-windup, "Anti-Windup Control Using a PID Controller," pp. 1–9, 2014.

[48] M. P. S. Dos Santos and J. a. F. Ferreira, "Novel intelligent real-time position tracking system using FPGA and fuzzy logic," *ISA Transactions*, vol. 53, no. 2, pp. 402–414, 2014.

[49] P. Dostál, *Nostradamus 2013: Prediction, Modeling and Analysis of Complex Systems*, ser. Advances in Intelligent Systems and Computing, I. Zelinka, G. Chen, O. E. Rössler, V. Snasel, and A. Abraham, Eds. Heidelberg: Springer International Publishing, 2013, vol. 210.

[50] H. Eichfeld, M. Lohner, and M. Muller, "Architecture of a CMOS fuzzy logic controller with optimized memory organisation and operator

design," in *[1992 Proceedings] IEEE International Conference on Fuzzy Systems*. San Diego, CA: Publ by IEEE, 1992, pp. 1317–1323.

[51] A. El Khateb, N. A. Rahim, J. Selvaraj, and M. N. Uddin, "Fuzzy-Logic-Controller-Based SEPIC Converter for Maximum Power Point Tracking," *IEEE Transactions on Industry Applications*, vol. 50, no. 4, pp. 2349–2358, Jul. 2014.

[52] N. Eskandarian, Y. A. Beromi, and S. Farhangi, "Improvement of Dynamic Behavior of Shunt Active Power Filter Using Fuzzy Instantaneous Power Theory," *Journal of Power Electronics*, vol. 14, no. 6, pp. 1303–1313, Nov. 2014.

[53] N. E. Evmorfopoulos and J. N. Avaritsiotis, "An Adaptive Digital Fuzzy Architecture for Application-Specific Integrated Circuits," *Active and Passive Electronic Components*, vol. 25, no. 4, pp. 289–306, 2002.

[54] G. D. Finn, "Learning fuzzy rules from data," *Neural computing & applications*, vol. 8, no. 1, pp. 9–24, 1999.

[55] M. Foerster, K. Lam, E. Sorensen, and A. Gavriilidis, "In situ monitoring of microfluidic distillation," *Chemical Engineering Journal*, vol. 227, pp. 13–21, Jul. 2013.

[56] Y. Fu, H. Li, and M. E. Kaye, "Hardware/software codesign for a fuzzy autonomous road-following system," *IEEE Transactions on Systems, Man and Cybernetics Part C: Applications and Reviews*, vol. 40, no. 6, pp. 690–696, Nov. 2010.

[57] S. Gai, P. Liu, J. Liu, and X. Tang, "A method for banknote feature extraction based on Haar wavelet and fuzzy logic," *Gaojishu Tongxin / Chinese High Technology Letters*, vol. 20, no. 11, pp. 1149–1155, 2010.

[58] K. Gheysari and B. Mashoufi, "Implementation of CMOS flexible fuzzy logic controller chip in current mode," *Fuzzy Sets and Systems*, vol. 185, no. 1, pp. 125–137, Dec. 2011.

[59] J. Gokulachandran and K. Mohandas, "Prediction of cutting tool life based on Taguchi approach with fuzzy logic and support vector regression techniques," *International Journal of Quality & Reliability Management*, vol. 32, no. 3, pp. 270–290, Mar. 2015.

[60] S. Gomariz, E. Alarcon, J. A. Martinez, A. Poveda, J. Madrenas, and F. Guinjoan, "Minimum time control of a buck converter by means of fuzzy logic approximation," pp. 1060–1065, 1998.

[61] J. L. González, O. Castillo, and L. T. Aguilar, "FPGA as a tool for implementing non-fixed structure fuzzy logic controllers," in *Proceedings of the 2007 IEEE Symposium on Foundations of Computational Intelligence, FOCI 2007*. IEEE, Apr. 2007, pp. 523–530.

[62] G. Goos, J. Hartmanis, and J. V. Leeuwen, *Real-Time and Embedded Computing Systems and Applications: 9th International Conference, RTCSA 2003*, G. Goos, J. Hartmanis, and J. V. Leeuwen, Eds. Lecture Notes in Computer Science, Springer, 2003.

[63] S. Gopinath, I. Kar, and R. Bhatt, "Experience inclusion in iterative learning controllers: Fuzzy model based approaches," *Engineering Applications of Artificial Intelligence*, vol. 21, no. 4, pp. 578–590, Jun. 2008.

[64] T. L. Grigorie, *Fuzzy Controllers, Theory and Applications*. Intech, 2011.

[65] J. Guajardo, S. S. Kumar, G. J. Schrijen, and P. Tuyls, "FPGA intrinsic PUFs and their use for IP protection," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 4727 LNCS, 2007, pp. 63–80.

[66] M. Haji Seyed Javadi, H. R. Mahdiani, and E. Zeinali Kh, "A hardware oriented fuzzification algorithm and its VLSI implementation," *Soft Computing*, vol. 17, no. 4, pp. 683–690, Oct. 2012.

[67] M. Hamzeh, H. R. Mahdiani, A. Saghafi, S. M. Fakhraie, and C. Lucas, "Computationally efficient active rule detection method: Algorithm and

architecture," *Fuzzy Sets and Systems*, vol. 160, no. 4, pp. 554–568, Feb. 2009.

[68] B. Heber and Y. Tang, "Fuzzy logic enhanced speed control of an indirect field-oriented induction machine drive," *IEEE Transactions on Power Electronics*, vol. 12, no. 5, pp. 772–778, 1997.

[69] A. Hernández Zavala, J. A. Huerta Ruelas, and O. Camacho Nieto, "Centre of slice area average defuzzifier for digital fuzzy systems and its hardware implementation," *Journal of Multiple-Valued Logic and Soft Computing*, vol. 21, no. 1-2, pp. 25–52, 2013.

[70] S. H. Huang and J. Y. Lai, "A high-speed VLSI fuzzy inference processor for trapezoid-shaped membership functions," *Journal of Information Science and Engineering*, vol. 21, no. 3, pp. 607–626, 2005.

[71] Y.-C. Hung, F.-J. Lin, J.-C. Hwang, J.-K. Chang, and K.-C. Ruan, "Wavelet Fuzzy Neural Network With Asymmetric Membership Function Controller for Electric Power Steering System via Improved Differential Evolution," *IEEE Transactions on Power Electronics*, vol. 30, no. 4, pp. 2350–2362, Apr. 2015.

[72] M. Idros, S. Ali, and M. S. Islam, "Condition based engine oil degradation monitoring system, synthesis and realization on ASIC," in *2014 IEEE International Conference on Semiconductor Electronics (ICSE2014)*. IEEE, Aug. 2014, pp. 84–87.

[73] S. Ionita and E. Sofron, "Field-programmable analog filters array with applications for fuzzy inference systems," in *Proceedings - HIS'04: 4th International Conference on Hybrid Intelligent Systems*, 2005, pp. 470–471.

[74] H. Ishibuchi, T. Murata, and I. Turksen, "Selecting linguistic classification rules by two-objective genetic algorithms," in *1995 IEEE International Conference on Systems, Man and Cybernetics. Intelligent Systems for the 21st Century*, vol. 2. IEEE, 1995, pp. 1410–1415.

[75] H. Ishibuchi, K. Nozaki, N. Yamamoto, and H. Tanaka, "Selecting fuzzy if-then rules for classification problems using genetic algorithms," *IEEE Transactions on Fuzzy Systems*, vol. 3, no. 3, pp. 260–270, 1995.

[76] M. S. Islam, M. S. Bhuyan, and S. H. M. Ali, "FPGA realization of a fuzzy based wheelchair controller," *Research Journal of Applied Sciences*, vol. 8, no. 9, pp. 442–448, 2013.

[77] A. F. Jabeen and M. Y. Sanavullah, "FPAA Based Bandwidth Recovery in Satellite Subsystems via Intrinsic Evolution," in *2008 Fifth International Conference on Fuzzy Systems and Knowledge Discovery*, vol. 3. IEEE, Oct. 2008, pp. 555–559.

[78] M. Jacomet and R. Walti, "VLSI fuzzy processor with parallel rule execution," in *IEEE International Conference on Fuzzy Systems*, vol. 1. IEEE, 1996, pp. 554–558.

[79] V. K. Jadoun, N. Gupta, K. Niazi, and A. Swarnkar, "Modulated particle swarm optimization for economic emission dispatch," *International Journal of Electrical Power & Energy Systems*, vol. 73, pp. 80–88, Dec. 2015.

[80] K. K. Jajulwar, "Genetic Optimization of Fuzzy Logic controllers," *International Journal of Simulation: Systems, Science and Technology*, vol. 10, no. 4, pp. 19–24, 2009.

[81] X. Jia and G. Fettweis, "Integration of code optimization and hardware exploration for a VLIW architecture by using fuzzy control system," in *2011 IEEE International SOC Conference*. IEEE, Sep. 2011, pp. 36–41.

[82] H. Jiang and J. R. Eastman, "Application of Fuzzy Measures in Multi-Criteria Evaluation in GIS," *International Journal of Geographical Information Science*, vol. 14, no. 2, pp. 173–184, 2000.

[83] D. Jinghong, D. Yaling, and L. Kun, "Development of Image Processing System Based on DSP and FPGA," in *2007 8th International Conference*

on Electronic Measurement and Instruments. IEEE, Aug. 2007, pp. 2–791–2–794.

[84] I. Kalaykov, "New speed limits of the fuzzy controller hardware," in *42nd Midwest Symposium on Circuits and Systems (Cat. No.99CH36356)*, vol. 2. IEEE, 1999, pp. 918–921.

[85] P. Karuppusamy, A. M. Natarajan, and K. N. Vijeyakumar, "An Adaptive Neuro-Fuzzy Model to Multilevel Inverter for Grid Connected Photovoltaic System," *Journal of Circuits, Systems and Computers*, vol. 24, no. 05, p. 1550066, Jun. 2015.

[86] B. Khoshnevisan, M. A. Rajaeifar, S. Clark, S. Shamahirband, N. B. Anuar, N. L. Mohd Shuib, and A. Gani, "Evaluation of traditional and consolidated rice farms in Guilan Province, Iran, using life cycle assessment and fuzzy modeling." *The Science of the total environment*, vol. 481, no. 1, pp. 242–51, May 2014.

[87] D. Kim, "An Implementation of fuzzy logic controller on the reconfigurable fpga system," *IEEE Transactions on Industrial Electronics*, vol. 47, no. 3, pp. 703–715, 2000.

[88] T. Korol, "A fuzzy logic model for forecasting exchange rates," *Knowledge-Based Systems*, vol. 67, pp. 49–60, Sep. 2014.

[89] A. Kumar, A. Khosla, J. S. Saini, and S. S. Sidhu, "Range-free 3D node localization in anisotropic wireless sensor networks," *Applied Soft Computing*, vol. 34, pp. 438–448, Sep. 2015.

[90] K. Kumar, S. Deep, S. Suthar, M. Dastidar, and T. Sreekrishnan, "Application of fuzzy inference system (FIS) coupled with Mamdani's method in modelling and optimization of process parameters for biotreatment of real textile wastewater," *Desalination and Water Treatment*, pp. 1–8, Jun. 2015.

[91] K. A. Kurniawan, D. Utomo, and S. Nugroho, *Intelligence in the Era of Big Data*, ser. Communications in Computer and Information Science,

R. Intan, C.-H. Chi, H. N. Palit, and L. W. Santoso, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2015, vol. 516.

[92] N. Lall, Xilinx, "FPGAs and DSPs - What Makes Sense for Your Design?" *RTC Magazine*, p. 1, 2005.

[93] E. Laubwald, "Coupled Tank System," Control Systems Principles, UK, Tech. Rep., 2006.

[94] C. C. Lee, "Fuzzy logic in control systems: Fuzzy logic controller - Part I," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 20, no. 2, pp. 404–418, 1990.

[95] ——, "Fuzzy logic in control systems: Fuzzy logic controller - Part II," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 20, no. 2, pp. 419–435, 1990.

[96] W. V. Leekwijck and E. E. Kerre, "Defuzzification: criteria and classification," *Fuzzy Sets and Systems*, vol. 108, no. 2, pp. 159–178, Dec. 1999.

[97] Leon Adams, "Choosing the Right Architecture for Realtime Signal Processing Designs," Texas Instruments Incorporated, Tech. Rep., 2002.

[98] N. Lerkkasemsan and L. E. Achenie, "Pyrolysis of biomass âĂŞ fuzzy modeling," *Renewable Energy*, vol. 66, pp. 747–758, Jun. 2014.

[99] H. Li, X. Jing, H.-K. Lam, and P. Shi, "Fuzzy sampled-data control for uncertain vehicle suspension systems." *IEEE transactions on cybernetics*, vol. 44, no. 7, pp. 1111–26, Jul. 2014.

[100] S. Li, "Fuzzy logic control ASIC chip," *Journal of Computer Science and Technology*, vol. 12, no. 3, 1997.

[101] Y. Y. Lin and M. Y. Liao, "Image processor and fuzzy PID controller design for robot-car intercept mission," *Journal of the Chinese Society of Mechanical Engineers, Transactions of the Chinese Institute of Engineers,*

Series C / Chung-Kuo Chi Hsueh Kung Ch'eng Hsuebo Pao, vol. 30, no. 5, pp. 401–408, 2009.

[102] O. Linda and M. Manic, "Uncertainty-robust design of interval type-2 fuzzy logic controller for delta parallel robot," *IEEE Transactions on Industrial Informatics*, vol. 7, no. 4, pp. 661–670, Nov. 2011.

[103] K. Lochan and B. K. Roy, *Control of two-link 2-DOF robot manipulator using fuzzy logic techniques: A review*, ser. Advances in Intelligent Systems and Computing, K. N. Das, K. Deep, M. Pant, J. C. Bansal, and A. Nagar, Eds. New Delhi: Springer India, 2015, vol. 335.

[104] C. W. Lou and M. C. Dong, "A novel random fuzzy neural networks for tackling uncertainties of electric load forecasting," *International Journal of Electrical Power & Energy Systems*, vol. 73, pp. 34–44, Dec. 2015.

[105] P. Maji, S. K. Patra, K. K. Mahapatra, J. Govindarajan, and J. J. Patel, "Realization of reconfigurable FLC on ADSP-BF537 processor," in *2013 4th International Conference on Computing, Communications and Networking Technologies, ICCCNT 2013, IEEE*. IEEE, Jul. 2013, pp. 1–4.

[106] P. Maji, B. R. Jammu, S. K. Patra, and K. Mahapatra, "Design and implementation of online fuzzy logic controller on FPGA," in *2014 Annual IEEE India Conference (INDICON)*. IEEE, Dec. 2014, pp. 1–5.

[107] P. Maji, S. K. Patra, and K. K. Mahapatra, "Design of Fuzzy Logic Controller based on TMS320C6713 DSP," *12th International Conference on Intelligent Systems Design and Application, IEEE, Kochi*, pp. 635–639, 2012.

[108] Y. Maldonado, O. Castillo, and P. Melin, "A multi-objective optimization of type-2 fuzzy control speed in FPGAs," *Applied Soft Computing Journal*, vol. 24, pp. 1164–1174, Nov. 2014.

[109] J. Malek, A. Sebri, S. Mabrouk, K. Torki, and R. Tourki, "Automated Breast Cancer Diagnosis Based on GVF-Snake Segmentation, Wavelet

Features Extraction and Fuzzy Classification," *Journal of Signal Processing Systems*, vol. 55, no. 1-3, pp. 49–66, Jun. 2008.

[110] M. C. Martinez-Rodriguez, P. Brox, and I. Baturone, "Digital VLSI Implementation of Piecewise-Affine Controllers Based on Lattice Approach," *IEEE Transactions on Control Systems Technology*, vol. 23, no. 3, pp. 842–854, May 2015.

[111] S. N. Mat Isa, Z. Ibrahim, J. M. Lazi, M. H. N. Talib, and A. S. Abu Hasim, "dSPACE DSP based implementation of simplified fuzzy logic speed controller for vector controlled PMSM drives," in *2012 IEEE International Conference on Power and Energy (PECon)*. IEEE, Dec. 2012, pp. 898–903.

[112] Mathworks Inc., "Simulate Fuzzy Inference Systems in Simulink," 2010.

[113] ——, "DC Motor Control," 2014.

[114] Y. Matsumoto, T. Tanaka, K. Sonoda, K. Kanda, T. Fujita, and K. Maenaka, "Low Power ECG Processing ASIC," *IEEJ Transactions on Sensors and Micromachines*, vol. 134, no. 5, pp. 108–113, 2014.

[115] J. M. Mendel, "Fuzzy logic systems for engineering: a tutorial," *Proceedings of the IEEE*, vol. 83, no. 3, pp. 345–377, Mar. 1995.

[116] ——, "Uncertainty, fuzzy logic, and signal processing," *Signal Processing*, vol. 80, no. 6, pp. 913–933, Jun. 2000.

[117] A. Messai, A. Mellit, P. A. Massi, A. Guessoum, and H. Mekki, "FPGA-based implementation of a fuzzy controller (MPPT) for photovoltaic module," *Energy Conversion and Management*, vol. 52, no. 7, pp. 2695–2704, 2011.

[118] A. Messai, A. Mellit, A. Massi Pavan, A. Guessoum, and H. Mekki, "FPGA-based implementation of a fuzzy controller (MPPT) for photovoltaic module," *Energy Conversion and Management*, vol. 52, no. 7, pp. 2695–2704, Jul. 2011.

[119] K. Michels, F. Klawonn, R. Kruse, and A. Nürnberger, *Fuzzy Control: Fundamentals, Stability and Design of Fuzzy Controllers*, J. Kacprzyk, Ed. Springer, 2006.

[120] M. Michta, "On set-valued stochastic integrals and fuzzy stochastic equations," *Fuzzy Sets and Systems*, vol. 177, no. 1, pp. 1–19, 2011.

[121] T. Miki and T. Yamakawa, "Fuzzy inference on an analog fuzzy chip," *IEEE Micro*, vol. 15, no. 4, pp. 8–18, 1995.

[122] I. Millán, O. Montiel, R. Sepúlveda, and O. Castillo, *Soft Computing for Hybrid Intelligent Systems*, ser. Studies in Computational Intelligence, O. Castillo, P. Melin, J. Kacprzyk, and W. Pedrycz, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, vol. 154.

[123] K. Mohanasundaram, N. Rajasekar, J. B. Edward, and G. Saravanailango, "A Fuzzy Logic Approach for Speed Controller Design of A.C. Voltage Controller Fed Induction Motor Drive," *2009 Fifth International Conference on MEMS NANO, and Smart Systems*, vol. 49, no. 1, pp. 133–136, 2009.

[124] M. Mokarram, A. Khoei, and K. Hadidi, "CMOS fuzzy logic controller supporting fractional polynomial membership functions," *Fuzzy Sets and Systems*, vol. 263, no. -, pp. 112–126, Mar. 2014.

[125] E. Monmasson and M. Cirstea, "FPGA Design Methodology for Industrial Control Systems A Review," *IEEE Transactions on Industrial Electronics*, vol. 54, no. 4, pp. 1824–1842, Aug. 2007.

[126] J. Morelli, A. Hirose, and H. Wood, "Fuzzy-logic-based plasma-position controller for STOR-M," *IEEE Transactions on Control Systems Technology*, vol. 13, no. 2, pp. 328–337, Mar. 2005.

[127] R. Muñoz Salinas, E. Aguirre, O. Cordón, and M. García-Silvente, "Automatic tuning of a fuzzy visual system using evolutionary algorithms: Single-objective versus multiobjective approaches," *IEEE Transactions on Fuzzy Systems*, vol. 16, no. 2, pp. 485–501, Apr. 2008.

[128] V. S. Mukhovatov and V. D. Shafranov, "Plasma equilibrium in a Tokamak," *Nuclear Fusion*, vol. 11, no. 6, p. 605, 1971.

[129] A. M. Murshid, S. A. Loan, S. A. Abbasi, and A. R. M. Alamoud, "VLSI architecture of fuzzy logic hardware implementation: A review," pp. 74–88, 2011.

[130] N. B. A. Mustafa, S. Gandi, Z. A. M. Sharrif, and S. K. Ahmed, "Real-time implementation of a fuzzy inference system for banana grading using DSP TMS320C6713 platform," in *Proceeding, 2010 IEEE Student Conference on Research and Development - Engineering: Innovation and Beyond, SCOReD 2010*, 2010, pp. 324–328.

[131] H. Nguyen, N. Prasad, C. Walker, and E. Walker, *A First Course in Fuzzy and Neural Control*. Chapman and Hall, CRC press, 2003.

[132] H. T. Nguyen, V. Kreinovich, and O. Sirisaengtaksin, "Fuzzy control as a universal control tool," *Fuzzy Sets and Systems*, vol. 80, no. 1, pp. 71–86, May 1996.

[133] N. N. Nguyen, W. J. Zhou, and C. Quek, "GSETSK: a generic self-evolving TSK fuzzy neural network with a novel Hebbian-based rule reduction approach," *Applied Soft Computing*, vol. 35, pp. 29–42, Oct. 2015.

[134] A. M. Noman, K. E. Addoweesh, and H. M. Mashaly, "DSPACE Real-Time Implementation of MPPT-Based FLC Method," *International Journal of Photoenergy*, vol. 2013, pp. 1–11, 2013.

[135] H. Okumus, H. Kahveci, and M. Ekici, "Improved brushless DC motor speed controller with digital signal processor," *Electronics Letters*, vol. 50, no. 12, pp. 864–866, Jun. 2014.

[136] J. A. Olivas, R. Sepúlveda, O. Montiel, and O. Castillo, *Soft Computing for Hybrid Intelligent Systems*, ser. Studies in Computational Intelligence, O. Castillo, P. Melin, J. Kacprzyk, and W. Pedrycz, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, vol. 154.

[137] U.-M. O'Reilly, "Genetic programming," in *Proceedings of the fourteenth international conference on Genetic and evolutionary computation conference companion - GECCO Companion '12*. New York, New York, USA: ACM Press, 2012, p. 693.

[138] D. Oswald, W. Li, L. Niu, J. Zhang, Y. Li, J. Yu, and F. Sun, "Implementation of fuzzy color extractor on NI myRIO embedded device," in *2014 International Conference on Multisensor Fusion and Information Integration for Intelligent Systems (MFI)*. IEEE, Sep. 2014, pp. 1–6.

[139] R. Palakeerthi and P. Subbaiah, "High speed charging and discharging current controller circuit to reduce back EMF by neuro fuzzy logic," *International Journal of Applied Engineering Research*, vol. 9, no. 22, pp. 12 949–12 960, 2014.

[140] H. Pan, H. Wong, V. Kapila, and M. S. de Queiroz, "Experimental validation of a nonlinear backstepping liquid level controller for a state coupled two tank system," *Control Engineering Practice*, vol. 13, no. 1, pp. 27–40, Jan. 2005.

[141] M. Parker, "FPGA versus DSP design reliability and maintenance," in *Design*, no. May, 2010, pp. 1–4.

[142] K. Passino, *The Control Systems Handbook, Second Edition*, 1st ed. Addison-Wesley Longman, Dec. 2010, vol. 20103237.

[143] A. Patel and B. Mohan, "Some numerical aspects of center of area defuzzification method," *Fuzzy Sets and Systems*, vol. 132, no. 3, pp. 401–409, Dec. 2002.

[144] H. Peyravi, A. Khoei, and K. Hadidi, "Design of an analog CMOS fuzzy logic controller chip," *Fuzzy Sets and Systems*, vol. 132, no. 2, pp. 245–260, Dec. 2002.

[145] E. Pierzchala, G. Gulak, L. Chua, and A. Rodríguez-Vázquez, *Field-Programmable Analog Arrays*. Springer Science & Business Media, 2013.

[146] E. Pierzchala and M. A. Perkowski, "A High-Frequency Field-Programmable Analog Array (FPAA) Part 2: Applications," *Analog Integrated Circuits and Signal Processing*, vol. 17, no. 1-2, pp. 157–169, 1998.

[147] Y. Pu, J. Gu, U. Farooq, and Y. Xu, "Fuzzy logic based control of vibration energy harvesting device for automotive suspension system," in *2014 IEEE International Conference on Information and Automation (ICIA)*. IEEE, Jul. 2014, pp. 592–597.

[148] S. Rafa, A. Larabi, L. Barazane, M. Manceur, N. Essounbouli, and A. Hamzaoui, "Implementation of a new fuzzy vector control of induction motor." *ISA transactions*, vol. 53, no. 3, pp. 744–54, May 2014.

[149] R. Rahmani, M. Seyedmahmoudian, S. Mekhilef, and R. Yusof, "Implementation of Fuzzy Logic Maximum Power Point Tracking Controller for Photovoltaic System," *American Journal of Applied Sciences*, vol. 10, no. 3, pp. 209–218, Mar. 2013.

[150] S. Rajak and S. Vinodh, "Application of fuzzy logic for social sustainability performance evaluation: a case study of an Indian automotive component manufacturing organization," *Journal of Cleaner Production*, Jun. 2015.

[151] S. V. Rani, P. Kanagasabapathy, and A. S. Kumar, "Digital Fuzzy Logic Controller using VHDL," in *2005 Annual IEEE India Conference - Indicon*, vol. 2005. IEEE, 2005, pp. 463–466.

[152] A. Razib, S. Dick, and V. Gaudet, "Log-domain arithmetic for high-speed fuzzy control on a field-programmable gate array," *Studies in Fuzziness and Soft Computing*, vol. 291, pp. 269–287, 2013.

[153] T. J. Ross, *Fuzzy logic with engineering applications*, 3rd ed. John Wiley & Sons, Ltd, 2010.

[154] S. Roy Chowdhury, A. Roy, and H. Saha, "ASIC design of a digital fuzzy system on chip for medical diagnostic applications." *Journal of medical systems*, vol. 35, no. 2, pp. 221–35, Apr. 2011.

[155] A. Rubaai, A. R. Ofoli, and D. Cobbinah, "DSP-based real-time implementation of a hybrid H-infinity adaptive fuzzy tracking controller for servo-motor drives," *IEEE Transactions on Industry Applications*, vol. 43, no. 2, pp. 476–484, 2007.

[156] B. K. Sahu, S. Pati, P. K. Mohanty, and S. Panda, "Teaching-learning based optimization algorithm based fuzzy-PID controller for automatic generation control of multi-area power system," *Applied Soft Computing*, vol. 27, pp. 240–249, Feb. 2015.

[157] V. Salapura and M. Gschwind, "Hardware/software co-design of a fuzzy RISC processor," in *Proceedings -Design, Automation and Test in Europe, DATE*. IEEE Comput. Soc, 1998, pp. 875–882.

[158] D. Z. Saletic and U. Popovic, "On Possible Constraints in Applications of Basic Defuzzification Techniques," in *2006 8th Seminar on Neural Network Applications in Electrical Engineering*. IEEE, 2006, pp. 225–230.

[159] L. Sánchez, I. Couso, and J. Casillas, "Genetic learning of fuzzy rules based on low quality data," *Fuzzy Sets and Systems*, vol. 160, no. 17, pp. 2524–2552, 2009.

[160] I. Santoso, T. S. Widodo, A. Susanto, and M. Tjokronagoro, "Application of fuzzy logic for temperature control in microcontroller based 2.45 GHZ microwave hyperthermia device," *International Journal of Applied Engineering Research*, vol. 9, no. 6, pp. 665–673, 2014.

[161] J. Sanz, A. Fernndez, H. Bustince, and F. Herrera, "A genetic tuning to improve the performance of fuzzy rule-based classification systems with

interval-valued fuzzy sets: Degree of ignorance and lateral position," *International Journal of Approximate Reasoning*, vol. 52, no. 6, pp. 751–766, Sep. 2011.

[162] R. Schneiderman, "Automotive Industry Is a Key Component to the Success of the DSP Sector [Special Reports]," *IEEE Signal Processing Magazine*, vol. 31, no. 1, pp. 18–21, Jan. 2014.

[163] M. D. Schrieber and M. Biglarbegian, "Hardware implementation and performance comparison of interval type-2 fuzzy logic controllers for real-time applications," *Applied Soft Computing*, vol. 32, pp. 175–188, Jul. 2015.

[164] S. Selvaperumal and C. C. A. Rajan, "Investigation of fuzzy control based LCL resonant converter in RTOS environment," *Journal of Intelligent and Fuzzy Systems*, vol. 26, no. 2, pp. 913–924, 2014.

[165] M. Setnes, R. Babuska, U. Kaymak, and H. R. van Nauta Lemke, "Similarity measures in fuzzy rule base simplification." *IEEE transactions on systems, man, and cybernetics. Part B, Cybernetics : a publication of the IEEE Systems, Man, and Cybernetics Society*, vol. 28, no. 3, pp. 376–386, Jan. 1998.

[166] A. Shamiri, S. W. Wong, M. F. Zanil, M. A. Hussain, and N. Mostoufi, "Modified two-phase model with hybrid control for gas phase propylene copolymerization in fluidized bed reactors," *Chemical Engineering Journal*, vol. 264, pp. 706–719, Mar. 2015.

[167] J. Shann and H. Fu, "A fuzzy neural network for rule acquiring on fuzzy control systems," *Fuzzy Sets and Systems*, vol. 71, no. 3, pp. 345–357, May 1995.

[168] E. G. Shopova and N. G. Vaklieva-Bancheva, "BASIC-A genetic algorithm for engineering problems solution," *Computers & Chemical Engineering*, vol. 30, no. 8, pp. 1293–1309, Jun. 2006.

[169] A. Simon, M. Adjouadi, and M. Ayala, "A .NET solution for distributed computing applications," *IEEE Potentials*, vol. 25, no. 2, pp. 24–28, Mar. 2006.

[170] Siva Malla, "Fuzzy controller based Speed Control of DC Motor," 2012.

[171] M. Soleimani, A. Khoei, and K. Hadidi, "Current-mode analog CMOS Fuzzy Logic Controller," in *2010 IEEE Asia Pacific Conference on Circuits and Systems*. IEEE, Dec. 2010, pp. 224–227.

[172] ——, "Design of current-mode modular programmable analog CMOS FLC," *Journal of Intelligent and Fuzzy Systems*, vol. 26, no. 1, pp. 63–76, 2014.

[173] G. Sousa, B. Bose, and J. Cleland, "Fuzzy logic based on-line efficiency optimization control of an indirect vector-controlled induction motor drive," *IEEE Transactions on Industrial Electronics*, vol. 42, no. 2, pp. 192–198, Apr. 1995.

[174] S. Stankute and H. Asche, "Geometrical DCC-algorithm for merging polygonal geospatial data," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 6016 LNCS, no. PART 1, 2010, pp. 515–527.

[175] M. Suetake, I. N. da Silva, and A. Goedtel, "Embedded DSP-Based Compact Fuzzy System and Its Application for Induction-Motor V/f Speed Control," *IEEE Transactions on Industrial Electronics*, vol. 58, no. 3, pp. 750–760, Mar. 2011.

[176] M. Sugeno, "An introductory survey of fuzzy control," *Information Sciences*, vol. 36, no. 1-2, pp. 59–83, Jul. 1985.

[177] H. Sun and J. Wu, "Plating pulse switching power based on a CPLD," *Electrochimica Acta*, vol. 105, pp. 342–346, Aug. 2013.

[178] X. Sun, S. Chung, and F. T. Chan, "Integrated scheduling of a multi-product multi-factory manufacturing system with maritime transport limits," *Transportation Research Part E: Logistics and Transportation Review*, vol. 79, pp. 110–127, Jul. 2015.

[179] Y. Sun, S. Tang, Z. Meng, Y. Zhao, and Y. Yang, "A scalable accuracy fuzzy logic controller on FPGA," *Expert Systems with Applications*, vol. 42, no. 19, pp. 6658–6673, Nov. 2015.

[180] P. Suratia, J. Patel, R. Rajpal, S. Kotia, and J. Govindarajan, "FPGA based Fuzzy Logic Controller for plasma position control in ADITYA Tokamak," *Fusion Engineering and Design*, vol. 87, no. 11, pp. 1866–1871, Nov. 2012.

[181] H. Surmann and M. Maniadakis, "Learning feed-forward and recurrent fuzzy systems: A genetic approach," *Journal of Systems Architecture*, vol. 47, no. 6, pp. 649–662, 2001.

[182] H. Tamukoh, K. Horio, and T. Yamakawa, "A bit-shifting-based fuzzy inference for self-organizing relationship (SOR) network," *IEICE Electronics Express*, vol. 4, no. 2, pp. 60–65, 2007.

[183] Texas Instruments, "TMS320C6748 Fixed and Floating Point DSP," Tech. Rep., 2013.

[184] ——, "TMS320C6000 Optimizing Compiler v7.6," Texas Instruments, Tech. Rep., 2014.

[185] M. Tokmakci and M. Alci, "A current-mode CMOS fuzzy logic controller," *International Journal of Automation and Control*, vol. 2, no. 4, p. 487, 2008.

[186] H. L. Tran, V. N. Pham, and D. T. Nguyen, "A hardware implementation of intelligent ECG classifier," *COMPEL - The international journal for computation and mathematics in electrical and electronic engineering*, vol. 34, no. 3, pp. 905–919, May 2015.

[187] W. L. Tung, C. Quek, and P. Cheng, "GenSo-EWS: a novel neural-fuzzy based early warning system for predicting bank failures." *Neural networks : the official journal of the International Neural Network Society*, vol. 17, no. 4, pp. 567–87, May 2004.

[188] M. N. Uddin and A. B. M. S. Hossain, "Development and Implementation of a Simplified Self-Tuned Neuro-Fuzzy based IM Drive," *IEEE Transactions on Industry Applications*, vol. 50, no. 1, pp. 51–59, Jan. 2014.

[189] M. N. Uddin and M. A. Rahman, "High-Speed Control of IPMSM Drives Using Improved Fuzzy Logic Algorithms," *IEEE Transactions on Industrial Electronics*, vol. 54, no. 1, pp. 190–199, Feb. 2007.

[190] ——, "High-speed control of IPMSM drives using improved fuzzy logic algorithms," *IEEE Transactions on Industrial Electronics*, vol. 54, no. 1, pp. 190–199, 2007.

[191] M. Uddin, T. Radwan, and M. Rahman, "Fuzzy-logic-controller-based cost-effective four-switch three-phase inverter-fed IPM synchronous motor drive system," *IEEE Transactions on Industry Applications*, vol. 42, no. 1, pp. 21–30, Jan. 2006.

[192] A. Vahidi and A. Eskandarian, "Research advances in intelligent collision avoidance and adaptive cruise control," *IEEE Transactions on Intelligent Transportation Systems*, vol. 4, no. 3, pp. 132–153, Sep. 2003.

[193] C. H. Wang and D. Y. Huang, "A new intelligent fuzzy controller for nonlinear hysteretic electronic throttle in modern intelligent automobiles," *IEEE Transactions on Industrial Electronics*, vol. 60, no. 6, pp. 2332–2345, 2013.

[194] C.-H. Wang and C.-C. Wang, "Finding the Real Surge Boundaries of Turbo-charged Automobiles Using Intelligent Fuzzy Reasoning Technique," *International Journal of Fuzzy Systems*, vol. 17, no. 2, pp. 224–235, May 2015.

[195] L.-X. Wang, *A Course in Fuzzy Systems and Control*, 1st ed. Prentice Hall International Inc., 1997.

[196] L. X. Wang and J. M. Mendel, "Generating fuzzy rules by learning from examples," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 22, no. 6, pp. 1414–1427, 1992.

[197] H. Watanabe, D. Chen, and S. Konuri, "Evaluation of min/max instructions for fuzzy information processing," *IEEE Transactions on Fuzzy Systems*, vol. 4, no. 3, pp. 369–374, 1996.

[198] D. Whitley, "A genetic algorithm tutorial," *Statistics and Computing*, vol. 4, no. 2, pp. 65–85, Jun. 1994.

[199] J. Xue, L. Sun, C. Qiao, and H. Qian, "Research on high-speed fuzzy reasoning with CPLD for fault diagnosis expert system," in *2009 9th International Conference on Electronic Measurement & Instruments*. IEEE, Aug. 2009, pp. 564–568.

[200] G. Yosefi, S. Aminifar, S. Neda, and M. A. Daneshwar, "Design of a mixed-signal digital CMOS fuzzy logic controller (FLC) chip using new current mode circuits," *AEU - International Journal of Electronics and Communications*, vol. 65, no. 3, pp. 173–181, Mar. 2011.

[201] G. Yosefi, A. Khoei, and K. Hadidi, "Design of a new CMOS controllable mixed-signal current mode fuzzy logic controller (FLC) chip," in *Proceedings of the IEEE International Conference on Electronics, Circuits, and Systems*. IEEE, Dec. 2007, pp. 951–954.

[202] Y. Yoshida, "The valuation of European options in uncertain environment," *European Journal of Operational Research*, vol. 145, no. 1, pp. 221–229, Feb. 2003.

[203] H. Youness, M. Moness, and M. Khaled, "MPSoCs and Multicore Microcontrollers for Embedded PID Control: A Detailed Study," *IEEE Transactions on Industrial Informatics*, vol. 10, no. 4, pp. 2122–2134, Nov. 2014.

[204] H. A. Yousef, "Fuzzy-logic obstacle avoidance control: Software simulation and hardware implementation," in *European Control Conference, ECC 1999 - Conference Proceedings*. Institute of Electrical and Electronics Engineers Inc., 2015, pp. 1458–1463.

[205] L. A. Zadeh, "Fuzzy sets," *Information and control*, vol. 8, no. 3, pp. 338–353, 1965.

[206] ——, "Fuzzy logic, neural networks and soft computing," *Communications of the ACM*, vol. 37, no. 3, pp. 77–84, 1994.

[207] I. A. Zammar, I. Mantegh, M. S. Huq, A. Yousefpour, and M. Ahmadi, "Intelligent Thermal Control of Resistance Welding of Fiberglass Laminates for Automated Manufacturing," *IEEE/ASME Transactions on Mechatronics*, vol. 20, no. 3, pp. 1069–1078, Jun. 2015.

[208] A. H. Zavala, O. C. Meto, and I. Batyrshin, "Center of slice area average defuzzifier for digital implementations of fuzzy systems," in *Proceedings of the 2010 International Conference on Artificial Intelligence, ICAI 2010*, vol. 1, 2010, pp. 97–102.

[209] A. H. Zavala and O. C. Nieto, "Fuzzy hardware: A retrospective and analysis," *IEEE Transactions on Fuzzy Systems*, vol. 20, no. 4, pp. 623–635, Aug. 2012.

[210] A. H. Zavala, O. C. Nieto, and C. Y. Marquez, "Soft-core implementation for centre of Slice Area Average defuzzifier," in *2011 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE 2011)*. IEEE, Jun. 2011, pp. 910–916.

Dissemination of Work

Journals

1. P. Maji, S. K. Patra, K. Mahapatra, “Design of Real-time Reconfigurable Fuzzy Logic Controller with M-FRHC Rule Reduction Technique”, Journal of Intelligence and Fuzzy Systems, vol. 30, no. 4, pp. 1973-1986, 2016.

2. P. Maji, S. K. Patra, K. Mahapatra, “Design and Implementation of Fuzzy Approximation PI Controller with Genetic Algorithm based Fuzzy Parameter Extraction” Advances in Artificial Intelligence, vol. 2015, Article ID 624638, 7 pages, 2015. doi:10.1155/2015/624638.

Conference

1. **P. Maji**, S. K. Patra, and K. Mahapatra, “Implementation of FPGA based fuzzy PI approximate control for automatic cruise control system,” in International Conference on Circuits, Communication, Control and Computing, I4C-2014, IEEE, Bangalore, 2014, pp. 203-206..

2. **P. Maji**, B. R. Jammu, S. K. Patra, and K. Mahapatra, “Design and implementation of online fuzzy logic controller on FPGA,” 2014 Annual IEEE India Conference (INDICON), Pune, 2014, pp. 1-5.

3. **P. Maji**, S. K. Patra, K. Mahapatra, J. Govindarajan, and J. J. Patel, “Realization of reconfigurable FLC on ADSP-BF537 processor,” 4th Interna-

tional Conference on Computing, Communications and Networking Technologies, ICCCNT-2013, IEEE, 2013, pp. 1-4.

4. **P. Maji**, S. K. Patra, and K. Mahapatra, "Design of Fuzzy Logic Controller based on TMS320C6713 DSP," 12th International Conference on Intelligent System Design and Application, ISDA-2012, IEEE, Kochi, 2012, pp. 635-639.

Journals: Under Review

1. P. Maji, S. K. Patra, K. Mahapatra, "Rule Reduction Technique for Generic Fuzzy Logic Systems" to the journal of Fuzzy Information and Engineering (Elsevier).

2. P. Maji, S. K. Patra, K. Mahapatra, "Generic Fuzzy Logic Controller: Design and Implementation with Vertices based Centroid of Area Defuzzification" to the International Journal of Computational Intelligence Systems (World Scientific).

Author's Biodata

Name of the Candidate	:	Pallab Maji
Father's Name	:	Keshab Chandra Maji
Date Of Birth	:	22 – 10 – 1986
Present Address	:	Adv. Communication Lab. Dept. of Electronics & Communication Engg. National Institute of Technology Rourkela-769 008 Odisha (India)
Permanent Address	:	Searsole Babupara, P.O.-Rajbari Ranganj, Dist - Burdwan PIN - 713358 West Bengal (India)

Academic Qualifications :

(i) B.Tech. in Electronics & Instrumentation Engg., Bengal College of Engineering and Technology, Durgapur, West Bengal

(ii) M.Tech. in Electronics & Communication Engg., National Institute of Technology, Rourkela, Odisha

Publications :

(i) Communicated 04 papers in International Journals

(ii) Published 04 papers in International Conferences