The Islamic University of Gaza
Deanery of Graduate Studies
Faculty of Engineering
Electrical Engineering Department

الجامعة الإسلامية – غزة
عمادة الدراسات العليا
كلية الهندسة
قسم الهندسة الكهربائية

Design of GA-Fuzzy Controller For Magnetic Levitation Using FPGA

A Thesis Submitted To The Faculty Of Engineering.
In Partial Fulfillment of the Requirements For The
Degree of Master of Science in Electrical Engineering

Prepared By:
Hosam Abu Elreesh

Advisor
Dr. Basil Hamed

June 2011

صفحة نتيجة الحكم على البحث (نتيجة الحكم من قبل لجنة المناقشة)

DEDICATION

I dedicate this thesis to my father and my mother in recognition of their endless help and support,I also dedicate this work to my wife and to my children, Mohamed, Abd Alrahman and Laian who are my most precious thing in this life.

ACKNOWLEDGEMENT

At the beginning, I thank ALLAH for giving me the strength and health to let this work see the light. I thank my supervisor Dr. Basil Hamed for his time, consideration, suggestions, ideas and advice during this thesis.

Special thanks go Dr. Hatem A. Elaydi and Dr. Assad Abu-Jasser -thesis examiners- for their patience, guidance, and generous supports during this research.

Many thanks go to my best friends for their help, specially Eng Said Abu Alros.

I also thank my wife for her infinite support and patience. Words will not be enough to thank my family for their patience and encouragement during my thesis. Finally, thanks for everyone who has raised his hands and prayed ALLAH for my success.

ABSTRACT

by

Hosam Abu Alreesh

The Islamic University of Gaza, 2011

Gaza, Palestine

Dr. Basil Hamed,

The design of controllers for non- linear systems in industry is a complex and difficult task. The development of non-linear control techniques has been approached in many different ways with varied results. One approach, which has shown promise for solving nonlinear control problems, is the use of fuzzy logic control. This Thesis will discuss the Magnetic Levitation (Maglev) models as an example of non linear systems. It will also show the design of fuzzy logic controllers for this model to prove that the fuzzy controller can work properly with nonlinear system. Genetic Algorithm (GA) is used in this thesis as optimization method that optimize the membership, output gain and inputs gain of the fuzzy controllers. The result of fuzzy controller with GA optimization will be compared with H_2 controller which is one of optimal control techniques, and will try to prove that fuzzy controller with GA optimization can gives better performance over H_2 controller. Finally fuzzy controller will be implemented using FPGA chip. The design will use a high-level programming language HDL for implementing the fuzzy logic controller using the Xfuzzy CAD tools to implement the fuzzy logic controller into HDL code.

ملخص

تصميم وحدات تحكم للأنظمة غير الخطية في الصناعة هو مهمة معقدة وصعبة. وهناك طرق عديدة طورت واستخدمت للتعامل مع التحكم في الأنظمة الغير خطية. احد هذه الطرق الواعدة هي التحكم الضبابي. هذه الأطروحة سوف تناقش نموذج نظام التعليق المغناطيسي كمثال للأنظمة الغير خطية لسببين. الأول أن هناك تطبيقات كثيرة تعتمد على نظرية هذا النموذج، والثاني انه نظام غير خطي. أيضا سوف نعرض كيفية تصميم متحكم ضبابي لهذا النموذج لكي نبرهن ان المتحكم الضبابي يستطيع أن يعمل بدقة عالية مع الأنظمة الغير خطية. هذه الأطروحة أيضا سوف تستخدم الخوارزمية الجينية (GA) للتحكم في شكل الأعضاء ومعاملات التكبير للمداخل والمخرج للمتحكم الضبابي لإيجاد الشكل والقيم المثالية لها. وسوف يتم مقارنة هذا المتحكم مع نوع أخر من المتحكمات المثالية وهو اتش تو وسوف نحاول أن نثبت أن المتحكم الضبابي مع الخوارزمية الجينية يعطي أداء أحسن من المتحكم اتش تو. في النهاية سوف يتم تنفيذ هذا المتحكم باستخدام (FPGA) وسوف يتم برمجة الجهاز باستخدام لغة برمجة رفيعة المستوى(HDL) باستخدام برنامج (Xfuzzy).

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABBREVIATIONS

analog to digital converter	ADC
ant colony optimization	ACO
application-specific integrated circuits	ASIC
application-specific standard parts	ASSP
Center of Area	CoA
Center of Maximum	CoM
complex programmable logic devices	CPLD
computer-aided design	CAD
configurable logic blocks	CLB
data acquisition card	DAQ
data base	DB
digital signal processor	DSP
digital to analog converter	DAC
electrically erasable programmable read-only memory	EEPROM
erasable programmable read-only memory	EPROM
Evolutionary Algorithms	EA
field programmable gat arrays	FPGA
fuzzy associative memory	FAM
fuzzy logic controller	FLC
fuzzy logic system	FLS
fuzzy rule-based system	FRBS
generalized modus pones	GMP
generalized modus tollens	GMT
generic array logic	GAL
genetic algorithm	GA
genetic fuzzy rule-based system	GFRBS
genetic Fuzzy Systems	GFS
graphic user interface	GUI
high-level programming language	HDL

input/output blocks	IOB
integrated circuits	ICs
Knowledge Base	KB
lookup-Table	LUT
Magnetic Levitation	MAGLEV
mean of maximum	MoM
multi-input multi-output	MIMO
one time programmable	OTP
programmable array logic	PAL
programmable logic arrays	PLA
programmable logic devices	PLD
programmable read only memory	PROM
proportional- derivative	PD
proportional-integral	PI
proportional-integral-derivative control	PID
rule base	RB
simple programmable logic devices	SPLD
single-input single- output	SISO
Static Ram	SRAM
Takagi- Sugeno-Kang	TSK
traveling sell man	TSM
very high speed integrated circuit hardware description language	VHDL
Very-large-scale integration	VLSI

CHAPTER 1 INTRODUCTION

1.1 Overview

Control systems have been in use for decades, and it is one of important science, which contribute in development and advancement of modern civilization and technology. Control systems are found in many types of applications in industry, such as power systems, computer control, space technology, robotics, weapon systems and many others. Due to development of civilization and progress in the computer science, the control systems are developed, multiplied and play a very important role in today's world. Because of the quick development in control domain, different types of controller appeared. Each type has its advantages and disadvantages, and during designing of control system, the engineer select one of these techniques according to many factors such as nature of plant (size, complexity, nonlinearity, time varying), control objective, specifications and cost considerations. Control systems can be divided to two mains types; conventional control techniques and intelligent control techniques. Next Some examples of conventional control techniques will be shown

1- Classical Control

In this type the methods are based on frequency response measurement and the system is described by transfer function model. Classical control refers mostly to single-input single-output (SISO) systems and it is mainly graphical method. Examples of classical control systems are root-locus, bode plot, nyquist and proportional-integral-derivative control (PID). The main technique used in this category is PID, which is widely used in industrial process control application such as petroleum refining and metal forming. PID controller response is based on three parameters: Proportional gain Kp, integral time Ki, derivative time Kd.

2- Modern Control

Contrary to classical control method, in this type the methods are based on time response measurement and the system under control in this type is described in stat space model. Modern control methods cover the multi-input multi-output (MIMO) systems and are mainly analytical method. Examples of this types are pole-placement, deadbeat and sliding mode.

3- Optimal Control

This type is like the modern control type, but it has one advantage that is can be used with linear time varying systems. Examples of this type are LQR, H_2 and H_∞ which are stable and robust.

Intelligent control techniques can works with systems which can not be described by differential/ difference equations and can be applied on systems whose complexity defies conventional control methods. Intelligent control has some advantages over conventional control, but its two main advantage are.

1- The algorithms of intelligent control system is software, that can be changed if the plant is changed; contrary to the conventional control algorithms.
2- It is faster than conventional controllers.

There are two basic approaches to intelligent control systems, knowledge-base-expert systems (adaptive fuzzy logic) and neural networks..

1.2 Linear and Nonlinear Systems

Any system can be divide into linear or nonlinear system. There are different techniques to control the two types of systems, some of these techniques work well with linear and non linear systems. Such as fuzzy logic.

1.2.1 Linear System

A linear system is a system to which the principle of supper position applies [1]. Supper position means that if $y_1(t)$ is the response of input $x_1(t)$ and $y_2(t)$ is the response of $x_2(t)$. Then if

$$y_1(t) = ax_1(t) \tag{1.1}$$
$$y_2(t) = bx_2(t) \tag{1.2}$$
$$x(t) = x_1(t) + x_2(t) \tag{1.3}$$

then

$$y(t) = y_1(t) + y_2(t) = ax_1(t) + bx_2(t) \tag{1.4}$$

1.2.2 Nonlinear System

A nonlinear system is a system to which the supperposition dose not apply. In fact most systems are nonlinear. Some of them are nonlinear over all the range of the operation, and others are linear only over certain ranges and nonlinear for remaining rang of operation. In this cases an approximation is used to deal with them as linear system via linearization, but this method is used only in certain ranges. Magnetic levitation is one of the most common applications in use in the field of control engineering because it is complex, nonlinear, and unstable system. Magnetic levitation systems have practical importance in many engineering systems such as high-speed Magnetic Levitation trains, and frictionless bearings. In recent years, much works have been reported for controlling magnetic levitation systems. Feedback linearization, (but this control technique does not guarantee robustness in the presence of modeling error and disturbances), and sliding mode, robust linear controller methods such as $H\infty$ optimal control, μ-synthesis, and Q-parameterization have also been applied to control magnetic levitation systems.

1.3 Fuzzy Logic Implementation

Fuzzy controller is simple to be designed using simulation software such as Matlab software or Labview; however, in the real world the use of such software is not practical. Due to practical consideration, applications need to be standalone. So, the control designer uses processors or microcontrollers. To implement the algorithm of fuzzy controller, there are two kinds processors and microcontroller.

1- General-purpose fuzzy processor

The general-purpose fuzzy processor has been implemented on various platforms, such as computers; processors (μP, μC, digital signal processor (DSP)**).** The advantages of this processor is it can be implemented quickly and be applied flexibly, but it has lower performance.

2- Dedicated fuzzy hardware

The dedicated fuzzy hardware has been implemented by different technologies since 1986. After that date many industries developed various FLC chips. The disadvantage of dedicated fuzzy hardware is requiring long development time, but it offers high performance

3- FPGA

Recently FPGA attracted a lot of research because it offers a compromise between special-purpose hardware and general-purpose processors. Different from the dedicated hardware**,** FPGA is flexible because the end user can change the program easily, and use the FPGA for another system. Another advantage of FPGA, it offers high performance and speed.

1.4 Previous Studies

- McKenna and Wilamowski (2001:189-194) [2] have investigated a method to implement fuzzy logic controller (FLC) on an field FPGA and obtained very smooth control surfaces. Their investigation showed that if design changes are needed, the FPGA chip can be reprogrammed for the new design in a matter of seconds. Their method of implementation was based on a Lookup-Table (LUT) scheme. This methode need large size of memory specially if the number of inputs are increased.

- Patyra and Grantner (1998:19-54) [3] presented a paper investigating design issues for digital fuzzy logic system (FLS) circuits. In their study, comparisons between the current trends were conducted and they proposed a new methodology whereby a fully parallel architecture is employed to achieve high performance in hardware implementation of digital FLSs. They presented ways to translate an FLS into hardware, and discussed methods for testing the FLS hardware performance. Both SISO and MIMO FLC hardware implementations were presented. Their proposed methodology provides an improved solution for high-speed, real-time applications. this way take very long time to design fuzzy controller.

- Vuong *et al* (2006:1-8) [4] described a methodology of implementing FLS using very high speed integrated circuit hardware description language (VHDL). The main advantages of using HDL are rapid prototyping and allowing usage of powerful synthesis tools such as Xilinx ISE, Synopsys, Mentor Graphic, or Cadence to be targeted easily and efficiently. Their work was motivated by the need for inexpensive hardware implementation of a generic FLS for use in industrial and commercial applications.

- Bonilla, J.E.; Grisales, V.H.; Melgarejo, M.A (2001: 1084 – 1087) [5] This paper presented the development of an FPGA-based proportional-differential (PD) fuzzy LUT controller. The fuzzy inference used a 256-value LUT. This method was used due to its reduced computation time cost. The controller architecture focused on the treatment of errors and changes in errors with tuning gains in order to regulate the control system dynamics using a traditional method in industrial processes. The controller was probed with several nonlinear plants, like an inverted pendulum and magnetic levitation. GA was used as a tuning tool to obtain a particular overshoot in the transient response of the control system. This way take large size of memory of the FPGA.

- Chia-Feng Juang; Chun-Ming Lu (2009: 597-475) [6] This paper proposed the design of fuzzy controllers by ant colony optimization (ACO) incorporated with fuzzy-Q learning, called ACO-FQ, with reinforcements. For a fuzzy inference system, the antecedent part was partitioned *a priori* and then all candidate consequent actions of the rules were listed. In ACO-FQ, the tour of an ant was regarded as a combination of consequent actions selected from every rule. Searching for the best one among all combinations was partially based on pheromone trail. Each candidate was assigned in the consequent part of the rule, a corresponding *Q*-value. Update of the *Q*-value was based on fuzzy-Q learning. The best combination of consequent values of a fuzzy inference system is searched according to pheromone levels and *Q*-values. ACO-FQ is applied to three reinforcement fuzzy control problems: (1) water bath temperature control; (2) magnetic levitation control; and (3) truck backup control. Comparisons with other reinforcement fuzzy system design methods verified the performance of ACO-FQ. The ACO-FQ is difficult and not general as GA.

1.5 Thesis Contribution

This thesis presents the design of fuzzy controller for the magnetic levitation model CE152, It also presents a Matlab program for applying the GA to optimize the memberships of the fuzzy controller. Finally this thesis realize fuzzy controller through FPGA using Xilinx Spartan 3e FPGA.

1.6 Outlines of the Thesis

This thesis consists of six chapters. Chapter 2 presents a review and introduction to fuzzy logic and its application, fuzzy sets operations, the main concepts in fuzzy sets such as membership functions, and linguistic variable. Chapter 3 presents a review and introduction to genetic algorithm, its using and main concepts in genetic algorithm such as cross over, mutation, reproduction. Chapter 4 presents a review and introduction of FPGA and VHDL language, and introduction to Xfuzzy tool. Chapter 5 presents the magnetic levitation model CE152 analyze and simulation results of the fuzzy controller without GA, with GA and VHDL fuzzy controller implementation. The final chapter concludes this thesis.

CHAPTER 2 FUZZY CONTROL

2.1 Fuzzy Logic

What is the fuzzy logic? fuzzy logic is a superset of conventional (Boolean) logic that has been extended to handle the concept of partial truth. In the (Boolean) logic we see that the results for any operation can be true or false if we refer to true by (1) and the false by (0) then the result may be (1) or (0).

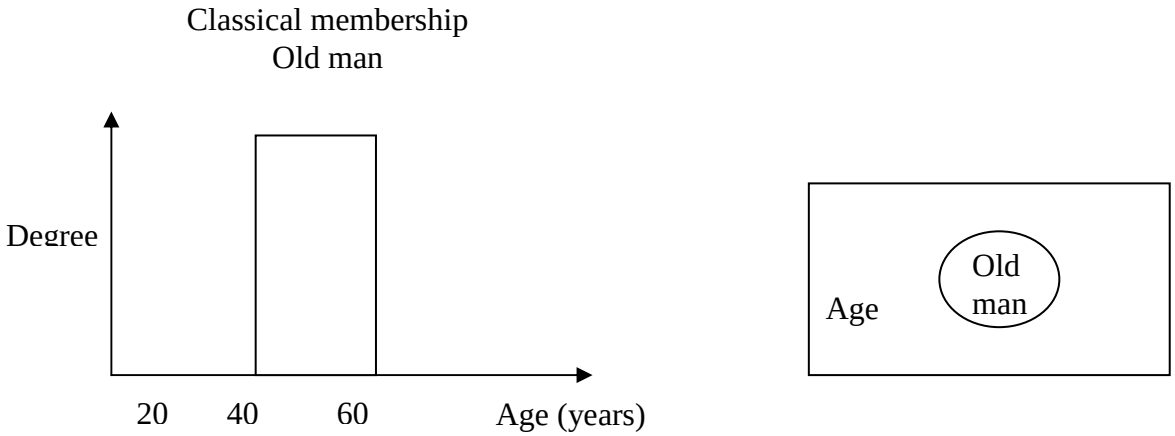


Figure (2.1): Classical Sets

Figure 2.1a shows an example for classical set, that has two values true or false. We see that the classical set have crisp boundary. And this example shows an age example: the man is old if he between 50 years and 60 years in that interval all age has the same degree (1). And outside of this interval, it has (0) degree. But there is problem, what about 49 years and 11 months, is the man young! No he old but has degree less than the 50 years, but in the Classical sets there are not degrees there are two values 1 or 0. So what is the solution, fuzzy sets give the solution.

2.2 Fuzzy Sets

Fuzzy sets are the sets, which do not have crisp [7]. There are not two values (true or false) but there are two limit is (1) completely true and the lower limit is (0) completely false and the result can have different degree between these limits.

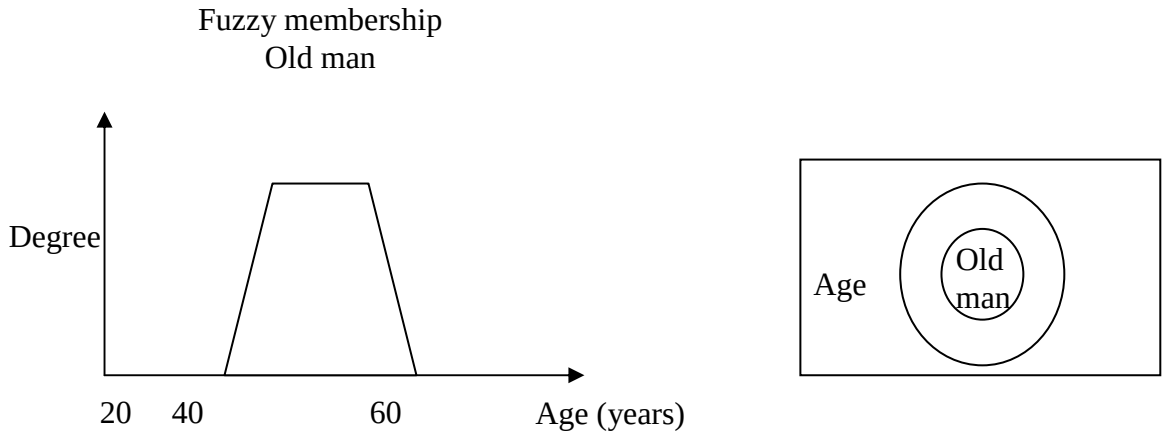


Figure (2.2): Fuzzy Sets

As shown in Figure 2.2a. the ages between the interval (50, 60) have the same degree. But out this interval, the man is old but has different degree this set solve the problem of 49 years and 11 month. In Figure 2.2b we see two circles the small circle is (old man) set. Every element inside it has the same degree (1), and between the small circle and the big circle there are different degree of (old man), and every element out side the big cycle has the same degree (0). The next definition is the mathematically representation of fuzzy sets.

Definition 2.1 (Membership function and fuzzy set) Let F be a set from the domain X. A membership function μF(x) of set F is a function that assigns value, or membership degree, to every x $\in$ F, μ: X → [0, 1]. Then set F is called a fuzzy set [8].

2.3 Fuzzy Sets History

Fuzzy logic idea was born in July 1964.Lotfi A. Zadeh [7] Found that the traditional system analysis techniques were very precise for complex systems but these traditional techniques took long time to solve the system and very difficult to make it by hardware component. So he proposed different kind of mathematics to solve any system without making complex calculations. In July 1964 the idea of fuzzy logic was beginning and in 1965 was the birth of fuzzy logic techniques. From 1965 to 1979 fuzzy logic was rules on the papers, but there were not any applications of it. The first use of fuzzy logic was in control systems, in 1979 was the first application of the fuzzy logic in industrial world by Blue Circle Cement and SIRA in Denmark (or Mamdani) [7]. The system was cement kiln controller.

This system began to operate in 1982. After that system, Japanese attended to fuzzy logic and started to use it in control systems, and they designed the first automatic subway train controller in 1987. This controller was not the first Japanese applications on fuzzy logic [7]. But in 1985 Japanese starts to make the first general-purpose fuzzy logic controller which converted the logic from mathematics form into practical model. After the subway train, Japanese developed the water-treatment system by using fuzzy logic.

2.4 Why Use Fuzzy Logic?

There are many reasons.

1- fuzzy logic is used to control the complex, and nonlinear systems without make analysis for these systems.
2- Fuzzy control enables engineers to implement the control technique by human operators to make ease of describing the systems [7], because of the man want to invent controller which like him (controller dose the same things which man can do it). So the best way is transfer the operators which can the man do to the controller, and that transfer is easy in fuzzy control.
3- Fuzzy logic is flexible with any given systems [9]. If any changes are happening in the system we do not need to start from the first step, but we can add some functions on top of it.
4- Fuzzy logic can be blended with conventional technique [9] to simplify their implementation.

2.5 Applications of Fuzzy Logic

The Japanese used fuzzy logic in many of applications, such as (subway train and water-treatment control), but in these years there are many more applications of fuzzy logic. For example in the control field**:**

1- Fuzzy logic is used to control the Camcorder to make stabilization in image if there are any rock [7].
2- In washing machine there is a soft and bad manner clothes, and there are different quantities of laundry. Control of washing cycle is based on these date..
3- Robotics controls, Refrigerators for temperature control.

4- Engine Control in the modern cars.

Other usages of fuzzy logic, in image processing, such as image identification, representation, and description. Now a day's fuzzy logic is used in Internet researching.

2.6 The Operations of Fuzzy Sets

In the classical sets there are basic operations that are found, such as intersection and union between sets, complement of set. If we have two sets (**A** and **B**) and these sets are subsets of universe (**U**), as we see in Figure 2.3 then

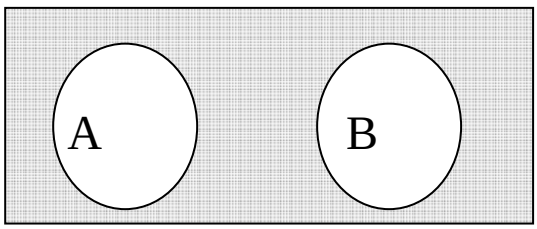


Figure (2.3): Two Classical Sets

2.6.1 AND Operation

The intersection between A and B = A ∩ B can turn out to be mathematically equivalent Boolean operation (AND). For example if there are element (x) and there are two variables C and D.

If x is found in (A), we will say c is true.

If x is found in (B), we will say d is true.

(C AND D) is true if and only if c is true and d is true (Boolean logic) that is means

(C AND D) is true if and only if $x \in A \cap B$, as see in Figure 2.4

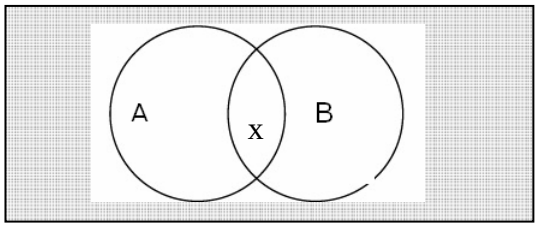


Figure (2.4): A AND B

2.6.2 OR Operation

The union between A and B= $A \cup B$ can be turn on to be equivalent Boolean operation (OR).

(C OR D) is true if and only if any of (c or d) is true that is means

(C OR D) is true if and only if $x \in A \cup B$ as we see in Figure 2.5 (a), (b)

a **b**

Figure (2.5): A OR B

2.6.3 NOT Operation

Complement of any set (A) = A^c is the elements which are in universe (U), and are not in set (A). The complement can turn out to be Boolean operation equivalent (NOT).
(NOT (A)) is true if and only if c is false
C is false it is not in set (A) $x \notin A$ as we see in Figure 2.6

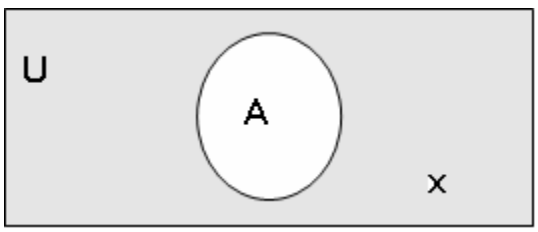


Figure (2.6): NOT A

The previous operations are some basic operations of Boolean logic. But the result was TRUE or FALSE. In fuzzy logic there are the same operations such as AND, OR, and COMPLEMENT, but the output of these operations are not only true but also result may be between these limits (true and false). For example, if there are two fuzzy sets A and B, and

there is element (x). In the fuzzy sets the element (x) can be in set A and has membership degree (0.5), and it can be in set B and has membership degree (0.7). We will make some operations such as AND, OR, and COMPLEMENT. The output of these operations will have different degree such as the input. The degree of (x) in every set is $\mu_{\text{set name}}(x)$

$$\mu_{A\cap B}(x) = \min\{\mu_A(x), \mu_B(x)\} \quad (2.1)$$

That is means the out put has degree equal to (min (0.5,0.7) = 0.5)

$$\mu_{A\cup B}(x) = \max\{\mu_A(x), \mu_B(x)\} \quad (2.2)$$

That is means the out but has degree (max(0.5,0.7) = 0.7)

$$\mu_{A^c}(x) = 1 - \mu_A(x) \quad (2.3)$$

That is means the out but has degree (1-0.5 = 0.5).

Some properties of fuzzy set operations are given in Table 2.1[10][11].

Table 2.1 Some Properties of Fuzzy Sets Operations

Law of contradiction	$A \cap A^- = \Phi$
Law of excluded middle	$A \cup A^- = I$
De Morgan's laws	$(A \cap B)^- = A^- \cup B^-$ $(A \cup B)^- = A^- \cap B^-$
Involution (Double negation)	$A^{--} = A$
Commutative	$A \cap B = B \cap A$ $A \cup B = B \cup A$
Associative	$A \cap (B \cap C) = (A \cap B) \cap C$ $A \cup (B \cup C) = (A \cup B) \cup C$
Distributive	$A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$ $A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$

2.7 Membership Function

Fuzzy set can be represented by a membership function, that gives the degree of membership [12], as shown in Figure 2.7. In that Figure $\mu_A(x)$ in Y –axis, is the symbol that refers to the degree of the membership function and $\mu(x)$ can be define as possibility function not probability function [12], and it takes value between (0,1). Any body can ask what is the different between probability function and possibility function? To answer this question let us see this example. If you and your friend went to visit another friend. And in the car your friend asks you, "Do you sure he is at the home?" and you answer "yes, I am sure but I do not know if he is in the bed room or on the roof ".You can give him another answer. You can answer him "I think 90% he is there". Look to the answer. in the first answer you are sure he is in the house but you do not know where he is in the house exactly. But in the second answer you are not sure he may be there and may be not there. That is the different between possibility and the probability in the possibility function the element is in the set by certain

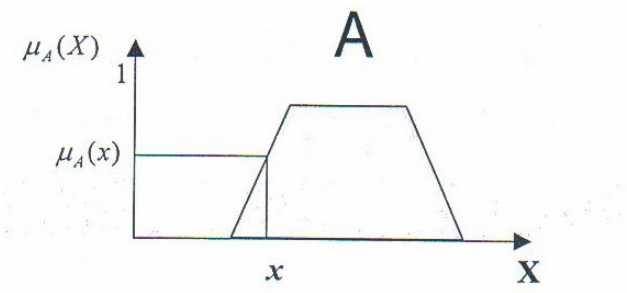


Figure (2.7): Membership Function

possibility and the probability in the possibility function the element is in the set by certain degree of, the probability function means that the element may be in the set or not . So if the probability of (x) = 0.7 that means (x) may be in the set by 70%. But in possibility if the possibility of (x) is 0.7 that means (x) is in the set and has degree 0.7. In the classical sets there is one type of membership function but in fuzzy sets there are different types of membership function, we now will show some of these types.

1- Triangular membership function: - This type of membership function is specified by three parameters (x, y, z) as we see in Figure 2.8.

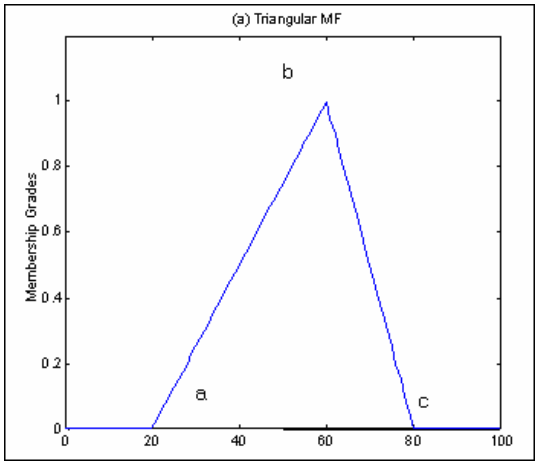


Figure (2.8): Triangular Membership Function

In this Figure a, c have same degree (the smallest degree of the membership), and b is the largest degree of the membership. the degree of any element (x) inside of this membership can be found by the following equation [7].

$$\mu \text{ Triangular Membership } (x) = \begin{cases} 0 & x < a \\ \dfrac{x-a}{b-a} & a \le x \le b \\ \dfrac{c-x}{c-b} & b \le x \le c \\ 0 & x > c \end{cases} \quad \ldots\ldots\ldots \quad (2.4)$$

2- **Trapezoidal membership function:-** This type has four specified parameters (a,b, c and d) as shown in Figure 2.9. The degree of any element **(x)** inside this membership can be found the following equation [7].

$$\mu_{\text{ Trapezoidal Membership}}(x) = \begin{cases} 0 & x < a \\ \dfrac{x-a}{b-a} & a \le x \le b \\ 1 & b \le x < c \\ \dfrac{d-x}{d-c} & c \le x < d \\ 0 & x \ge d \end{cases} \quad \ldots\ldots\ldots \quad (2.5)$$

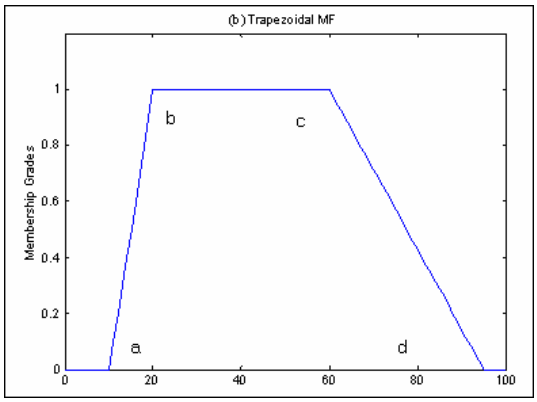


Figure (2.9): Membership Function

3- Gaussian membership function:- Figure 2.10 shows example of this type, there are two specified parameters **(c, w),** where c is the center of the membership function, and **w** is the width of it, that means any change in **w,** the width of membership will change.

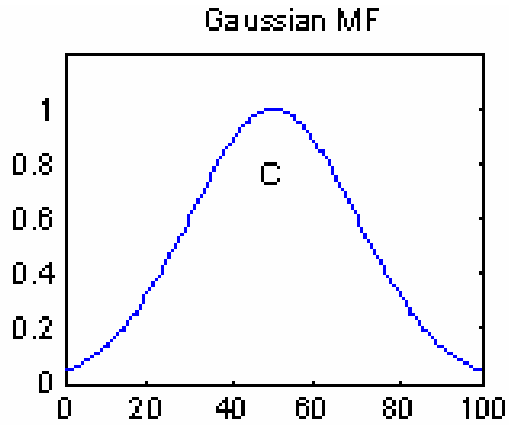


Figure (2.10): Gaussian Membership Function

The degree of any element (**x**) inside this membership can be determined by the following equation [7].

$$\mu_{\text{Gaussian Membership Function}}(x) = e^{\left(\frac{-(x-c)^2}{w^2}\right)} \qquad (2.6)$$

4- Bell membership function:- Figure.2.11 shows example of this type. There are three specified parameters (p, c, w), p is usually positive number [7], c is the center of the membership function, and w is the width of it.

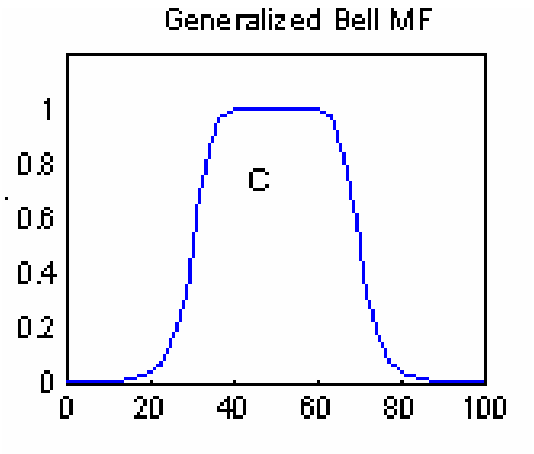


Figure (2.11): Bell Membership Function

The degree of any element **(x)** inside this membership can be determined by the following equation [7].

$$\mu_{\text{Bell Membership Function}}(x) = \frac{1}{1+\left|\frac{x-c}{w}\right|^{2p}} \tag{2.7}$$

The previous membership function are the most commonly used in practice, and can use any type of these memberships for solving the system that we want. But, there are other different types that we will not talk about in this thesis.

2.8 Linguistic Variables

Linguistic variable is an important concept in fuzzy logic [7].When fuzzy sets are used to solve the problem without analyzing the system; but the expression of the concepts and the knowledge of it in human communication are needed [7]. Human usually do not use mathematical expression but use the linguistic expression. For example, if you see heavy box and you want to move it, you will say, "I want strong motor to move this box" we see that, we use strong expression to describe the force that we need to move the box. In fuzzy sets we do the same thing we use linguistic variables to describe the fuzzy sets. These variable are words or sentence in natural or synthetic language [13]. For example if we take the universe U refer to the human age, we can take the interval between (50,60) years and give it name "old man"

this name is called linguistic variable. We can use this name to refer to the set that contains the interval (50,60) years.

3.9 IF – THEN Rules

The previous section showed that fuzzy sets can be represented by linguistic variables, and it is an important concept in the fuzzy logic. But another important concept, this concept is fuzzy rules:

IF (input1 is MFa) AND (input2 is MFb) AND…AND (input n is MFn) THEN (output is MFc)

Where MFa, MFb, MFn, and MFc are the linguistic variables of the fuzzy membership functions that are in input 1,input 2…input n, and output. For example, in a system where the inputs of the system are Serves and Food and the output is the Tip. Food may be (good, ok, bad), and serves can be (good, ok, bad), the output tip can be (generous, average, cheap), where good, ok, bad, generous, average, and cheap are the linguistic variables of fuzzy membership function of the inputs (food, and serves) and the output (tip). We can write the rules such as.

IF (**Food** is **bad**) and (**Serves** is **bad**) **THEN** (**Tip** is **cheap**)
——————— antecedent———————— ——— consequent———————

The maximum number of rules of any system can be found by the next equation(if all inputs have same number of memberships).

max *num* of rules = M^N.

Where (M) is number of the membership function in the input and (N) is the number of the input. The previous example shows the way to write the rules of fuzzy systems. There are two main types of fuzzy inference rules in fuzzy logic reasoning namely, generalized modus pones (GMP) and generalized modus tollens (GMT).

Table 2.2 Comparison between GMP and GMT

GMP Direct reasoning	**GMT Indirect reasoning**
Premise 1: If x is A Then y is B Premise 2: x is A Consequence: y is B	Premise 1: If x is A Then y is B Premise 2: y is B Consequence: x is A
Forward goal-driven inference Eg: Fuzzy logic control	Backward goal-driven inference Eg: Expert System (AI)

But how can the designer write the rules of fuzzy systems; there are four ways to derivation the fuzzy system rules.

1- Expert Experience and control engineering knowledge.
2- Based on fuzzy modeling of human operators central action.
3- Based on learning
4- Based on fuzzy model of a process.

2.10 Implication Functions

The IF-THEN rules can be interpreted in classical logic by the implication operators. Suppose there is a statement such as "IF *a* THEN *b*", then the classical set represents this by $a \Rightarrow b$. The truth table for this rule can be given as. For the first case a is false so by the rule b will be false too; second case a is false and b is true this is true case because $a \Rightarrow b$ not $b \Rightarrow a$ so it can happens b is activated when a is false; the third case a is true and by using the rule b must be true, but b is false so it is false statement; finally the fourth case is true because a is true so it implicate b is true.

Table 2.3 Classical Set Truth Table

a	**b**	$a \Rightarrow b$
F	**F**	**T**
F	**T**	**T**
T	**F**	**F**
T	**T**	**T**

The implication operator can also be written as

$$\bar{a} \vee b \quad \text{or} \quad (a \wedge b) \vee \bar{a}$$

Each rule in the fuzzy knowledge base corresponds to a fuzzy relation. Various approaches can be taken in determining the relation corresponding to a particular fuzzy rule. The common implication functions are Mini rule (Mamdani), Product rule (Larsen), Max-min rule (Zadeh), Arithmetic rule (Zadeh) and Boolean and others [10].

Common implication operators are:

.

1- Mamdani $\mu R(x,y) = \min[\mu A(x), \mu B(y)]$. (2.8)

2- Zadeh $\mu R(x,y) = \max\{\min[\mu A(x), \mu B(y)], 1 - \mu A(x)\}$. (2.9)

3- Larsen $\mu R(x,y) = \mu A(x) \bullet \mu B(y)$. (2.10)

4- Lukasiewicz $\mu R(x,y) = \min\{ 1, [1 - \mu A(x) + \mu B(y)]\}$ (2.11)

The two most important fuzzy implications are Mamdani and Larson**.** This thesis will use the Mamdani implication

2.11 Mamdani Implication

Mamdani proposed a fuzzy implication rule for fuzzy control in 1977. It is a simplified version of Zadeh implication operator. The Mamdani fuzzy logic operator is given in equation (2.8). Next example shows how can to calculate the output of each rule, the system is SISO. The input is temperature and the output is the number of customers. Figure 2.12 a and b shows the membership of the input and output respectively. Suppose the next rule is given.

IF the temperature is hot OR temperature is moderately hot, THEN the ice cream parlor is crowded.

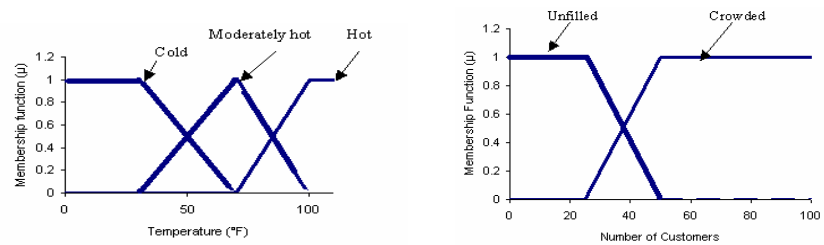


Figure (2.12): a) The Input Membership b) The Output Membership

Suppose the temperature is (75 F), so it is found in two sets Moderately hot and hot; with two degrees $\mu_{m.hot}$=0.8 and μ_{hot}=0.2 respectively. Any rule contains OR operation can be converted to more than one rule. So the previous rule can be separate to two rules.

1- **IF the temperature is hot THEN the ice cream parlor is crowded.**

2- **IF the temperature is moderately hot THEN the ice cream parlor is crowded.**

The output will be crowded in the two cases but by different degrees.

$\mu_{R1}=\min(\mu_{hot})$

$\mu_{R2}=\min(\mu_{m.hot})$.

the $\mu_{output}=\max(\mu_{R1}, \mu_{R2})=\max(0.8,0.2)=0.8$

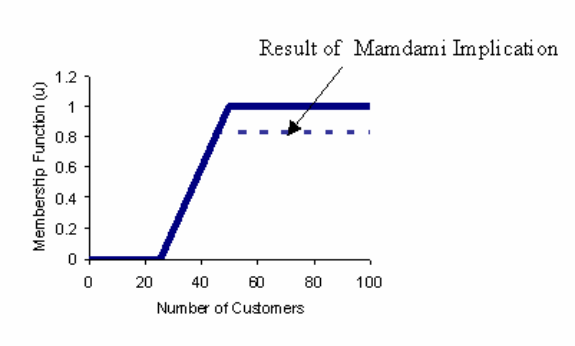


Figure (2.13): Membership Function of Customer After Mamdani Implication Rule.

2.12 Fuzzy Logic Control (FLC)

Fuzzy Control applies fuzzy logic to the control of processes by utilizing different categories, usually 'error' and 'change of error', for the process state and applying rules to decide a level of output, i.e. a suitable control action. The linguistic variables used for both input and output variables are often of the form 'negative large', 'positive small', 'zero' etc. A rule base of the FLC can be written as matrix to or as list of rules as shown before atypical rules matrix for a two-input, single-output system with three membership functions per variable is shown in Table 2.4

Table 2.4 Simple Rule Base

E de	N	Z	P
N	N	N	Z
Z	N	Z	P
P	Z	P	P

There are many models of FLC, but the most famous are the Mamdani model, Takagi-Sugeno-Kang (TSK) model and Kosko's additive model (SAM) [7]. But this Thesis will use Mamdani model only.

2.13 Mamdani Model:

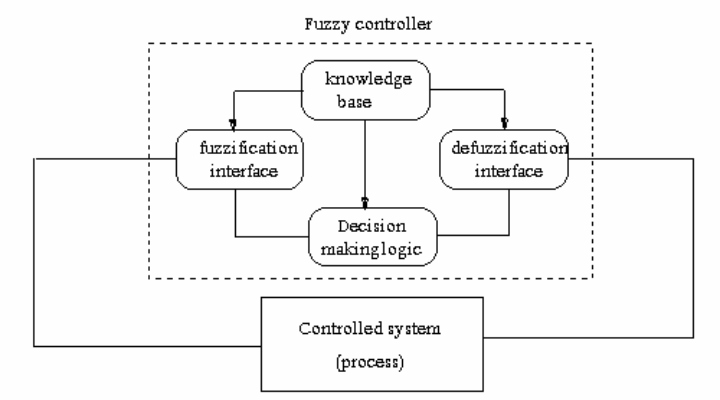


Figure (2.14): Mamdani Model

Figure 2.14 shows the block diagram of Mamdani fuzzy system model, the rule base of this model is in the next form.

IF (X is A) and (Y is B) … THEN (Z is C)

Where C is membership of the output as shown in Figure 2.15. Mamdani model block consist of three stage.

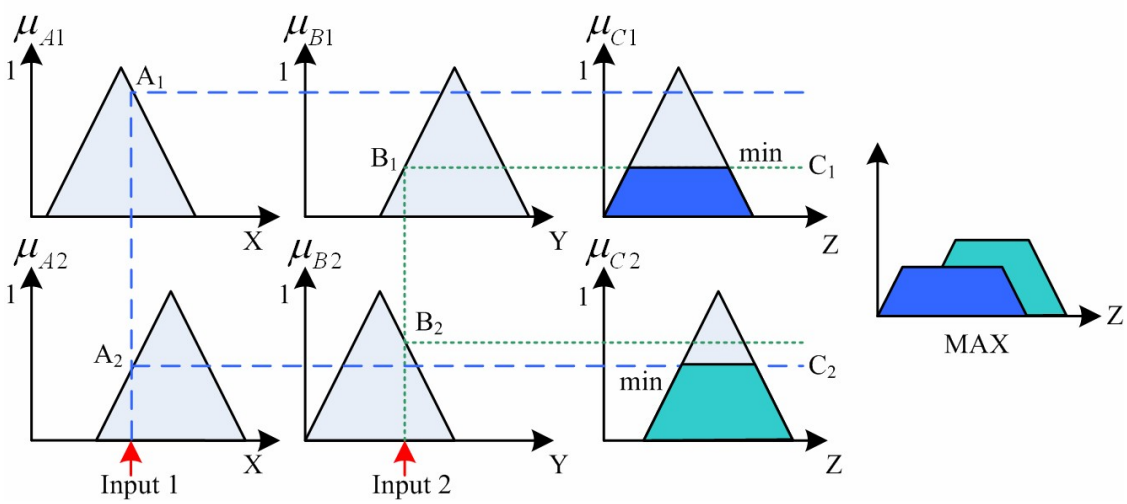


Figure (2.15): Mamdani Model Example

- **Fuzzification: -** Fuzzification means converting a crisp value of process variable into a fuzzy set. In order to make it compatible with the fuzzy set representation of the process state variable.

- **Fuzzy Associative Memory (FAM): -** FAM is a set of fuzzy associations between the input and the output [14]. This stage consists of two parts:

1. **Knowledge base: -** Knowledge base contains a data base and rule base. Data base provides necessary definitions for linguistic rules, and the rules base consist of the IF-THEN rules, which can be derived by using four ways as shown in (section 3.10)

2. **Decision-Making: -** Decision-Making means choosing the most appropriate action from several possible actions.

- **Defuzzification: -** Defuzzification strategy is aimed at producing a non-fuzzy control action, or we can say defuzzification means the conversion of the fuzzy output values into crisp values. For example, if we say "the output force must be large" and large variable takes the values between (70, 90) N, then what is the force will be needed 75 or 80 or …N, we can know what is the force we want by using defuzzification method. There are different types of defuzzification methods.

1- Mean of Maximum method (MoM): In the Mean of Maximum (MoM) defuzzification method, the fuzzy logic controller first identifies the scaled membership function with the

greatest degree of membership. The fuzzy logic controller then determines the typical numerical value for that membership function. The typical numerical value is the mean of the numerical values corresponding to the degree of membership at which the membership function was scaled.

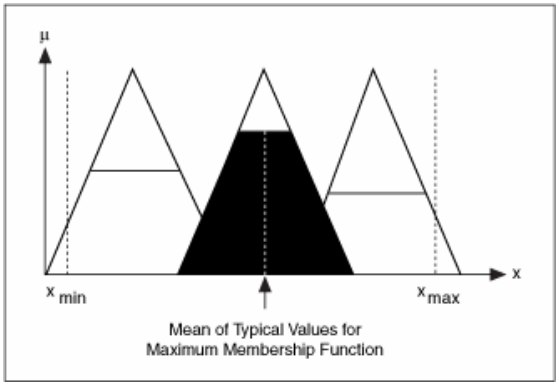


Figure (2.16): Example of MOM Method

2- Center of Area (CoA): - The center-of-area (COA) method is the most popular defuzzification method [15]. We can calculate the real value of the output by the next equation.

$$out = \frac{\int_{x\min}^{x\max} \mu_i(x)\,x\,dx}{\int_{x\min}^{x\max} \mu_i(x)\,dx} \qquad (2.12)$$

An example of CoA method is shown in Figure 2.16. But the main disadvantage of this method is its high computational cost [7].

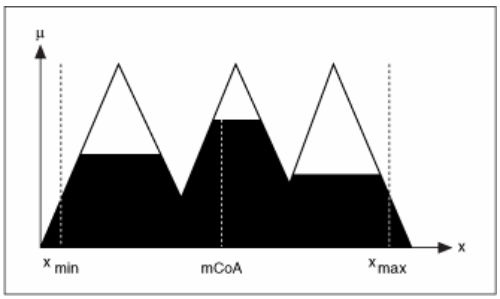


Figure (2.17): Example of COA Method

3- **Center of Maximum (CoM):** In the Center of Maximum (CoM) defuzzification method, the fuzzy logic controller first determines the typical numerical value for each scaled membership function. The typical numerical value is the mean of the numerical values corresponding to the degree of membership at which the membership function was scaled. The fuzzy logic controller then uses the following equation to calculate a weighted average of the typical values.

$$\text{out}= \frac{x_1\mu_1 + x_2\mu_2 + ...x_n\mu_n}{\mu_1 + \mu_2 + ...\mu_n} \qquad (2.13)$$

In this thesis the Mamdani model will be used and will have two inputs error and change of errors and the output will be change of voltage. All of inputs and output will have seven membership functions. And the CoA defuzzification method will be used. The rule base will be as the next form:

IF error is zero AND change of error is zero THEN change of voltage is zero.

These rules will be written by the experience.

CHAPTER 3 GENETIC ALGORITHM

3.1 Introduction

Genetic Algorithms are reliable and robust methods for searching solution spaces [10]. GA is general purpose search algorithm which uses principles inspired by neutral genetic to find solutions to problems [16][17]by using Survival of the fittest principle. The basic idea is to maintain a population of chromosomes, which represent candidate to the concrete problems that will be solve, through a process of computation and controlled variation. Each structure of chromosome in the population represent one of the possible solution of the problem and the fitness test of these chromosomes can determine which chromosomes are used to form a new chromosomes that will be use in computational process. As in natural the new chromosomes are created by some operations such as crossover and mutations. There is another operation which called reproduction. This operation is added to achieve the survival of the fittest principle. In recent years, GA is used in many applications specially in optimization and search problems and had a great measure of success; the main reason of this success that it can start from any solutions, and generate other solutions that converge to the optimal solution in less time versus other classical search tools (enumerative, heuristic). "GA are theoretically and empirically proven to provide a robust search in complex spaces, thereby offering a valid approach to problems requiring efficient and effective searches" [18]. The human designer which want to solve optimization problem using GA must address five issues[18].

1- A genetic representation of candidate solutions,
2- A way to create an initial population of solutions,
3- An evaluation function which describes the quality of each individual,
4- Genetic operators that generate new variants during reproduction, and
5- Values for the parameters of the GA, such as population size, number of generations and probabilities of applying genetic operators.

Figure 3.1 shows a basic model of a genetic algorithm, the algorithm are as follows[19].

1- [Start] Generate random population of n chromosome, the individuals of this population represent the Possible solutions.
2- [Fitness] Evaluate the fitness f(x) of each chromosome x in the population.

3- [New population] Create a new population by repeating the following steps until the New population is complete.

- [Selection] Select two parent chromosomes from a population according to their fitness (the better fitness, the bigger chance to get selected).
- [Crossover] With a crossover probability, cross over the parents to form new offspring (children). If no crossover was performed, offspring is the exact copy of parents.
- [Mutation] With a mutation probability, mutate new offspring at each locus (position in chromosome)
- [Accepting] Place new offspring in the new population.

4. [Replace] Use new generated population for a further sum of the algorithm.
5. [Test] If the end condition is satisfied, stop, and return the best solution in current population.
6. [Loop] Go to step2 for fitness evaluation.

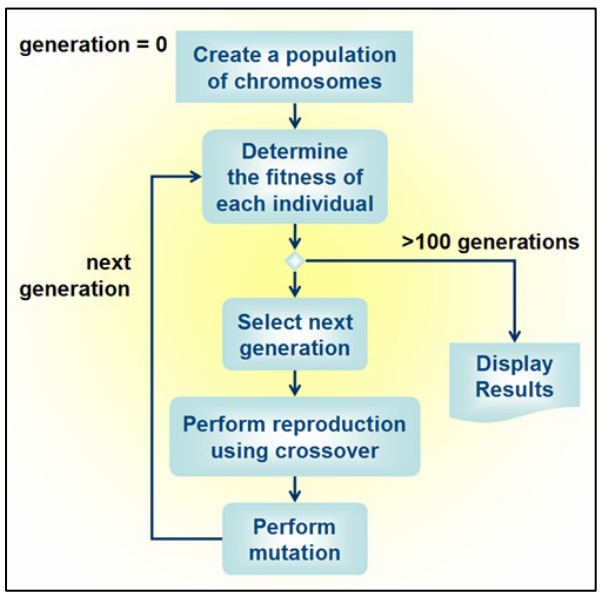


Figure (3.1): The basic Genetic Algorithm

3.2 Genetic Algorithm vs. Other Optimization Techniques

Working with GA is very simple, because its principle depends on emulating genetics and natural selection by software tools. GA does not deals with the real parameters of the problem, but it deals with the code of these parameters. The parameters of the problem are

coded mostly as a binary code. The population of the problem is always chosen randomly, typical population size is from few dozens to thousands [19]. To solve the any problems (searching , optimizations) the cost function or fitness function is needed; this function is responsible for the selection of the best individuals in the population and delete vulnerable individuals. Fitness function in GA can be any functions depending on the applications or the problems and can be nearly anything that can be evaluated by a computer or even something that cannot. For example $f(x) = x^2$ can be evaluated by computer but the case of eyewitness, the human being selects among the alternatives generated by GA [19]. There are three criteria used to classify the optimization algorithm:

1- Discreet or continues.
2- Constrained or unconstrained.
3- Sequential or parallel.

Some optimization algorithms can work with discreet and continues problems, and there are algorithms work with one kind of the problems; and so with the constrained and unconstrained. There are many applications need to work with high speed such control systems, so it needs parallel algorithm; parallel algorithms are used to speed up the processing. GA differs from conventional optimization techniques in following ways [20]:

1- GA does not deal with data directly but works with encoded data.
2- GA uses least information such as fitness function to solve problems an does not need derivation.
3- GA uses probability laws rather than certain laws.
4- GA generate populations of answer not just one answer.
5- Almost all conventional optimization techniques search from a single point but GA always operates on a whole population of points (parallelism).

3.3 GA Operations

The process of genetic algorithm is used to solve the structure of chromosome to represent the possible combinations of answers, according to population size; the population size can be fixed or random according to the optimization problem. The chromosome structure represents all the variables that the designer want to find the optimal values of it.

Every variable represents gene, and the main structure of the genes is 0 and 1 combinations. Real numbers can also be used to indicate the number of floating. The chromosomes of each string represents a separate individual, and each individual represent one solution of the problem, so each population contain series of individual and the decision of each individual will have a so-called fitness value. This value is calculated by fitness function; the individual which have higher fitness value has higher probability to appear in the next generation more than the lower fitness value. The new population is generated after some processes, such selection, replication (reproduction), mating (crossover), Mutation and other evolutionary mechanisms. The process of evolution over generations eventually converges to the optimal solution.

3.4 GA Elements

As shown before there are some terminology associated with GA. This section will gives brief definitions of these terminology.

3.4.1 Individuals

An individual represents one solution of the problem, in two forms.

1- The chromosome or genotype: which consist of genes that GA deals.
2- The phenotype: which expresses the chromosome in the real model.

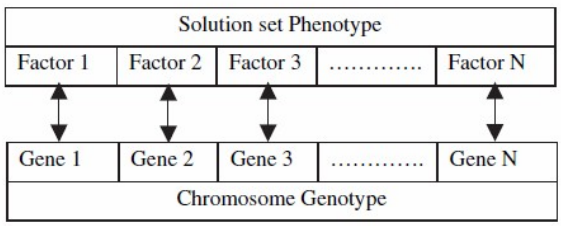


Figure (3.2): Representation of Genotype and Phenotype

Figure 3.2 shows the two forms of the individuals (chromosome). The chromosome consists of a set of genes. Every gene represents a single factor of the problem. A gene is A gene is a bit string of arbitrary lengths. The bit string is a binary representation of number of intervals from a lower bound [19]. Figure 3.3 shows the chromosome architecture

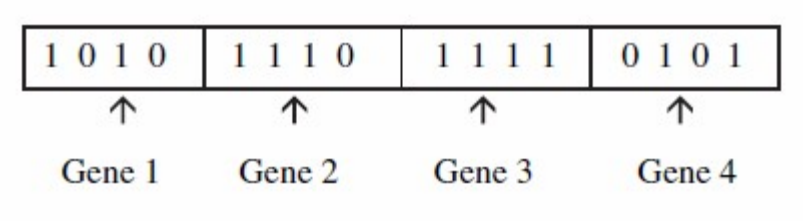


Figure (3.3): Gene Representation

3.4.2 Population

Figure 3.4 shows that the population consists of a group of individuals. Each individual is represented by phenotype parameter. There are two important aspects in the population used in GA.

1- Initial population.
2- Population size.

Population	Chromosome 1	1 1 1 0 0 0 1 0
	Chromosome 2	0 1 1 1 1 0 1 1
	Chromosome 3	1 0 1 0 1 0 1 0
	Chromosome 4	1 1 0 0 1 1 0 0

Figure (3.4): Population Representation

Initial population often consists of random individuals but in some cases the designer can suggest some solutions to be in the population. The size of it depend on the complexly of the problem. In the ideal case the first population must have large number of individuals to cover all the rang of solution space. All possible alleles of each should be present in the population. Sometimes some of the solutions expected can be used to seed the initial population. Thus, the fitness of these individuals will be high which helps the GA to find the solution faster. The population size can cause some problems. The large population is useful to find the best solution but it was established that the time required by a GA to converge is (N x log N) where N is population size [19].

3.5 Chromosome coding

First step of the optimization problem is converting the variables in this problem to code in which the GA can work with it, as shown in Figure 3.2, this code must have relation with the real value of the variable to calculate its fitness value. For example if the variable is integer between 0 and 100 the 0 can be coded to binary such as "0000000" and 100 can be coded to "1100100" and every value between these values can be found by a simple conversion. There are many kinds of coding [19][20]. Figure 3.5 shows the kinds of coding.

1- Binary encoding:

The most common way of encoding [19]. This code consists of binary 0 or 1 indicates chromosome gene, often used in numerical problem as in Figure 3.2 (a).

2- Real-encoding:

This encode is used to represents the numbers or symbols, often used in the problems of arrangement type as in Figure 3.2 (b).

3- Octal encoding:

This encoding uses string made up of octal numbers (0–7) as in Figure 3.2 (c).

4- Hexadecimal Encoding

This encoding uses string made up of hexadecimal numbers (0–9, A–F) Figure 3.2 (d).

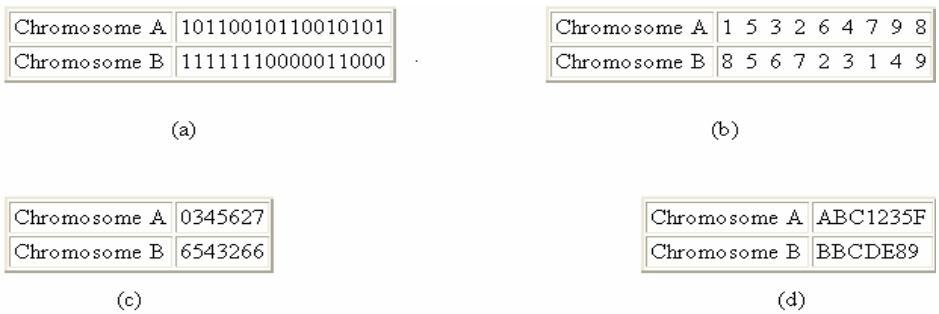


Figure (3.5): a) Binary Coding - b) Real Coding - c) Octal Coding d) Hexadecimal Coding

3.6 Fitness Function.

The fitness of an individual in a genetic algorithm is the value of an objective function for its phenotype. and it is used to evaluate how good the different individuals in the population are. The fitness function depends on the problem that will be solved, For example

in Traveling Sales Man (TSM) problem may be the time that the sales man will take it along traveling. So the fitness value can be defined as function of the objective function g(x).

$$fitness.value = f(g(x)) \quad (3.1)$$

For calculating fitness, the chromosome has to be first decoded and the objective function has to be evaluated. The fitness not only indicates how good the solution is, but also corresponds to how close the chromosome is to the optimal one [19]. When the optimization problem is single criterion, it is simple because there are one goal to achieve, When the problem is multi criterion the optimization problem will be more complex because if the solution is optimal for one criterion it may be worst for another one. The most difficult fitness functions are the ones needed to evaluate non-numerical data [21], as the developer must find other metrics or ways to find a numerical evaluation of non-numerical data. An example of this is provided by Mitchell [22], who describes the problem of finding the optimal sequence of amino acids that can be folded to a desired protein structure. The acids are represented by the alphabet {A, …, Z}, and thus no numerical value can be straightforwardly calculated. The used fitness function calculates the energy needed to bend the given sequence of amino acids to the desired protein. In control applications there are different fitness function that may be used [23].

1- $fitness.value = \int_0^\infty e^2(t)dt$ sum of squared error (3.2)

Where (e) is the error signal, this function can track error quickly, but easily gives rise to oscillation..

2- $fitness.value = \int_0^\infty |e(t)|dt$ sum of absolute error (3.3)

This function can obtain good response, but its selection performance is not good.

3- $fitness.value = \int_0^\infty te^2(t)dt$ sum of time weighted squared error (3.4)

This function can gives fast tracking and good response.

3.7 Selection

Selection or reproduction is the first process after finding the fitness values of the solutions. In this process the developer will choose the pairs of parents that will be crossed. This step is to decide how to perform selection. On other words, who are the individuals of that population will be used to create the next offspring that will be used for next generation. The purpose of this step is to stress the individuals of the population that will be selected with the higher fitness. The problem is how to select the chromosomes that will cross, there are many methods that can be used [19].

3.7.1 Roulette Wheel Selection.

Figure 3.6 shows the Roulette selection method which is one of the traditional GA selection techniques. That is called because this method works in a way that is similar to a roulette wheel. Each individual in a population is allocated a share of a wheel; the size of the share depends on the individual's fitness. The individuals that have higher fitness have big share. The individuals that have lower fitness have small share. That means the lower fitness of the individuals may have no chance to be in the roulette. A pointer is spun (a random number generated) and the individual to which it points is selected. This continues until the requisite number of individuals has been selected. The Roulette wheel will have a problem when the fitness values differ very much. If the best chromosome fitness is 90%, its circumference occupies 90% of Roulette wheel, and then other chromosomes have too few chances to be selected.

3.7.2 Rank Selection

Rank Selection ranks the population and every chromosome receives fitness from the ranking [19]. It results in slow convergence It also keeps up selection pressure when the fitness variance is low [19]. Here, rank selection is programmed as follow.

1. select first pair at random.
2. generate random number R between 0 and 1.
3. if R<r use the first individual as a parent. If the *R>=r* then use the second individual as the parent

4. repeat to select the second parent.

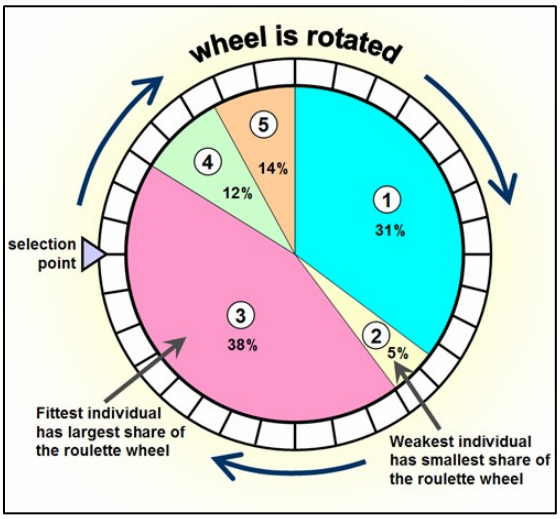


Figure (3.6): Roulette Wheel Selection.

3.7.3 Stochastic Universal Sampling

Figure 3.7 shows Stochastic Universal Sampling method in this method the individuals represent a line that is divided into number of Adjacent segments, such that each individual's segment is equal in size to its fitness exactly as in roulette-wheel selection. Then, create equally space pointers that are placed over the line. The numbers of these pointers (*NPointer*) depends on the number of the individuals that will be selected,; the distance between the pointers is given as 1/NP, and the position of the first pointer is given by a randomly generated number in the range [0, 1/*NPointer*].

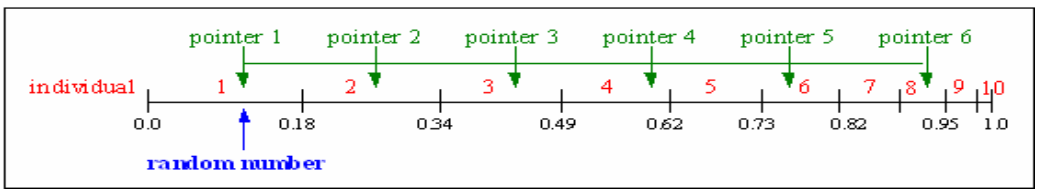


Figure (3.7): Stochastic Universal Sampling

3.8 Crossover

Crossover is the process that takes two parents of solutions and generates a new offspring. The reproduction step filters the individuals but stays without any change so to find better solutions. The crossover process is needed to generate new childes that take the best adjectives from the parents. The crossover needs three steps to complete the process.

1- Choosing the parents randomly (choosing two random individuals).
2- Choosing the parts of chromosomes that will be changed.
3- Finally, swapping the position values between the two chromosomes following the cross site.

Various crossover techniques are discussed as follows:

3.8.1 Single-Point Crossover

Traditional genetic algorithm use this technique. Where the two parents are cut once at corresponding points and the parts that cut exchanged as shown in Figure 3.8. The cross point is selected random, that means the crossover point at two parents may be changed at other two parents in the same population.

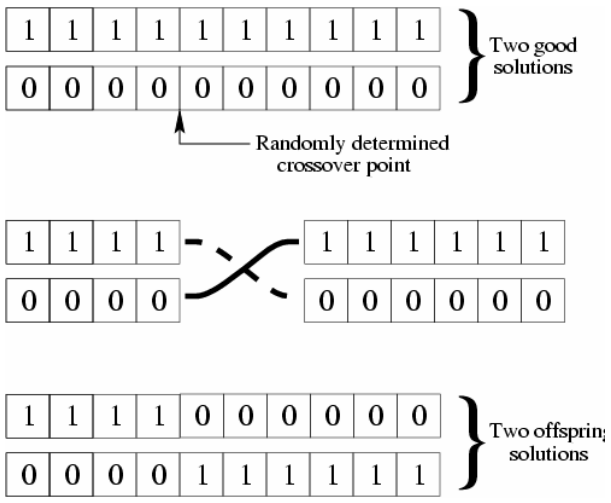


Figure (3.8): Single Point Crossover

3.8.2 Two-Point Crossover

That technique is similar to Single-Point Crossover, but the difference is that there are two cut points these points are selected randomly and the part that are between these points are swapped as shown in Figure 3.9. The disadvantage of this techniques is that building blocks are more likely to be disrupted. But the advantage of it is that the problem space may be searched more thoroughly.

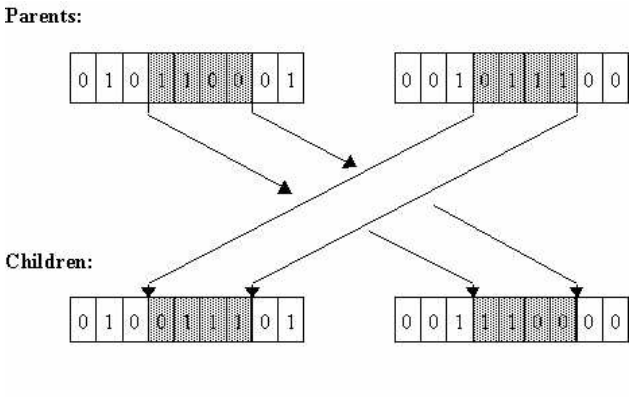


Figure (3.9): Two-Point Crossover

3.8.3 Uniform Crossover

Uniform crossover is another crossover technique. In this technique, the random mask is used, and this mask has same length as the chromosome. This mask consists of 1s and 0s . If a bit in the mask is 1 then the corresponding bit in the first child will come from the first parent and the second parent will contribute that bit to the second offspring. If the mask bit is 0, the first parent contributes to the second child and the second parent to the first child as shown in Figure 3.10.

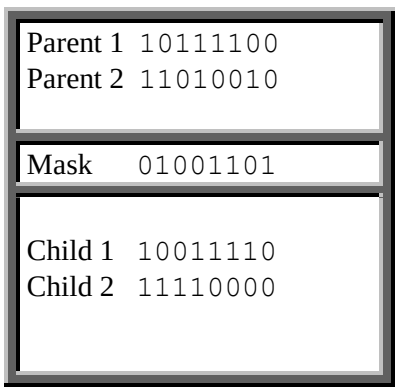


Figure (3.10): Uniform Crossover

3.9 Mutation

Mutation means swap one bit in binary coding or changes one number if the chromosome consists of numbers. For binary coding, for doing this process there are one way; first choosing random number at every individual if the number less than specified number which is chose before by the programmer, then this individual will have mutation process. But how many bits will be changed there are many ways. First choosing random number between 1 and the total numbers of chromosome length and swap the bits which meet that number. Second method is choosing random number between 0 and 1 at every bit of the chromosome if the number less than specified number then this bit will be swapped i.e., if it is a 1 change it to 0 or vice versa. This mutation probability is generally kept quite low and is constant throughout the lifetime of the GA. However, a variation on this basic algorithm changes the mutation probability throughout the lifetime of the algorithm, starting with a relatively high rate and steadily decreasing it as the GA progresses. This allows the GA to search more for potential solutions at the outset and to settle down more as it approaches convergence.

3.10 Elitism

With crossover and mutation taking place, there is probability that the best solution may be lost as there is no guarantee that these operations will preserve fitness. To combat this elitist models are often used [16]. In these method, the best individual from a population is saved before any of the operations take place. After the new population is formed and evaluated, it is examined to see if this best structure has been preserved. If not, the saved copy is reinserted back into the population, usually this individuals is reinserted instead of the weakest individual. The GA then proceeds to perform the operations on this population.

3.11 Genetic Fuzzy Systems

As known before heuristic fuzzy control depends on good knowledge of the system, which is intended to control it. So it must be a control designer with a good experience with this system. The fuzzy controller of any system can be found by try and error. Try and error

method is not simple in fuzzy control and may be take very long time, because there are many parameters have an effect of the fuzzy controller, such as membership shape, rules (number of rules or architecture of rules), inputs gains and outputs gains, and the interval of the membership of the inputs and outputs; so in the recent years Genetic Fuzzy Systems (GFS) are used to solve these problems, and the Hybridization between GA that is one of methods of Evolutionary Algorithms (EA) and FLC so called soft computing. Figure 3.11 shows a family of computing techniques. There are so many ways of how to use GA in fuzzy control. The most extended GFS type is the genetic fuzzy rule-based system (GFRBS), where an EA is employed to learn or tune different components of an FRBS. The objective of a genetic tuning process is to adapt a given fuzzy rule set such that the resulting FRBS demonstrates better performance. The following components of the knowledge base (KB) are potential candidates for optimization [18].

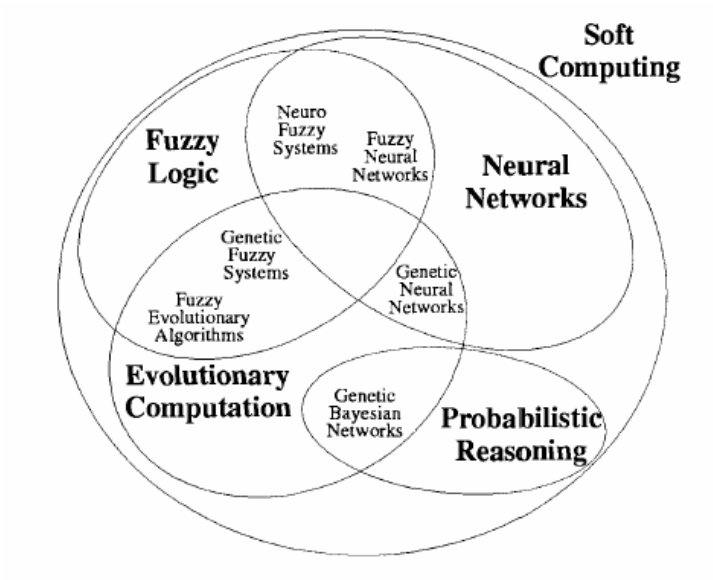


Figure (3.11): Hybridization in Soft Computing

1- Data base (DB) components: scaling functions and membership function parameters.
2- Rule base (RB) components: "IF-THEN" rule consequents.

3.11.1 Genetic Tuning of The Data Base

The tuning of the scaling gains and fuzzy membership functions is an important task in FLC design. Scaling gains applied to the inputs and outputs of an FLC. Because the most

FLC is normalized, the universes of discourse in which the fuzzy membership functions are defined (all inputs and outputs are in the rang [-1 1]. The individual is refer to scaling gain and by using fitness function can calculate the best individual which gives the best scaling function. In the case membership function, the parameters of the membership is tuned; Triangular membership functions are usually encoded by the left, right, and center of the membership

3.11.2 Genetic Learning of The Rule Base

In this way the membership of the fuzzy inputs and outputs and the scaling gain of them do not changed, but the sequence of IF THEN rules will be modified to gives the best result. In this way the individual is represent the one rule or the all rules. The RB is represent by a relational matrix, a decision table or a list of rules.

In this thesis the fuzzy membership inputs and output membership functions will be used as a variables that will be optimized using GA. Every triangular membership has three variable can effect on the shape of it; so the each chromosome will has the number of genes every genes refer to one parameter that effect on the membership shape. For example every triangular membership can represents be three genes because it has three parameters that control of its shape (center edge, left edge and right edge). If the inputs of fuzzy controller have seven triangular membership, then every chromosome of the population will have twenty one genes (7x3) (in this thesis there are ten variable for membership and three variables for gains). And the fitness function will be the integral of the absolute value of error.

CHAPTER 4 FPGA

4.1 Introduction

FPGA is digital integrated circuits (ICs) that have electronics blocks which can be programmed, and these blocks has configurable interconnection between them. These block can be used by the designed engineer to perform a huge ranges of tasks. There are two kinds of FPGA, one kind can be programmed for single time only and it so called one time programmable (OTP). The second kind can be reprogrammed many times. The difference between FPGA and the other devices that have internal hardwire by manufacturer is its flexibility. Since 1960, the ICs are used in the human life, and there are many different types of ICs that have been used such as memory devices, microprocessors (μP). Programmable logic devices (PLDs), application-specific integrated circuits (ASICs), application-specific standard parts (ASSPs), and—of course—FPGAs. Figure 4.1 shows the timeline of the ICs and other devices.

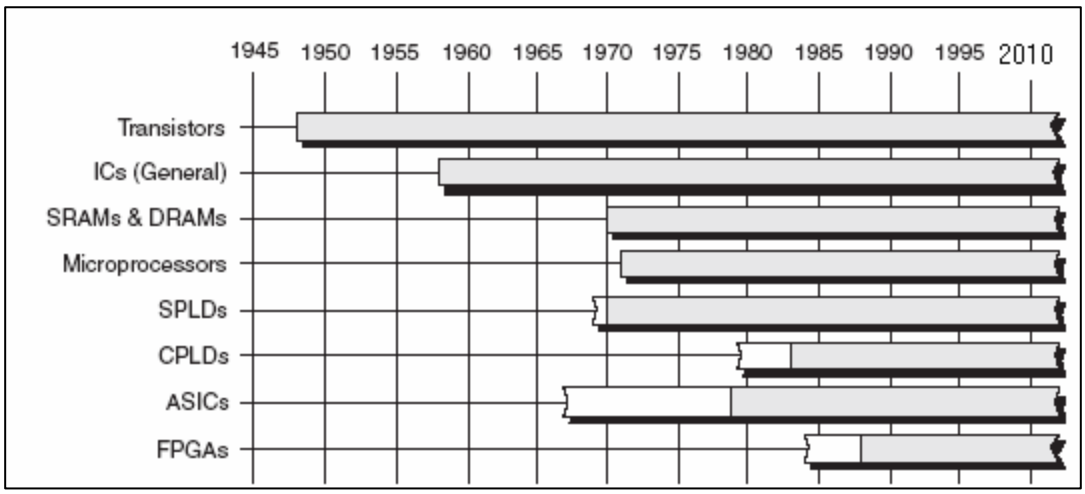


Figure (4.1): Technology Timeline

The white parts in the timeline means the time that the devices are found but were not used by the engineering widely. For example Xilinx has been designed first FPGA in 1984 but started being in use by the engineers in 1990 [24].

4.2 Types of PLD

There are two types of PLDs, simple programmable logic devices (SPLDs) and complex programmable logic devices (CPLDs). There are many SPLDs such as programmable logic arrays (PLAs), programmable read only memory (PROMs), programmable array logic (PALs) and generic array logic (GALs),. as shown in Figure 4.2.

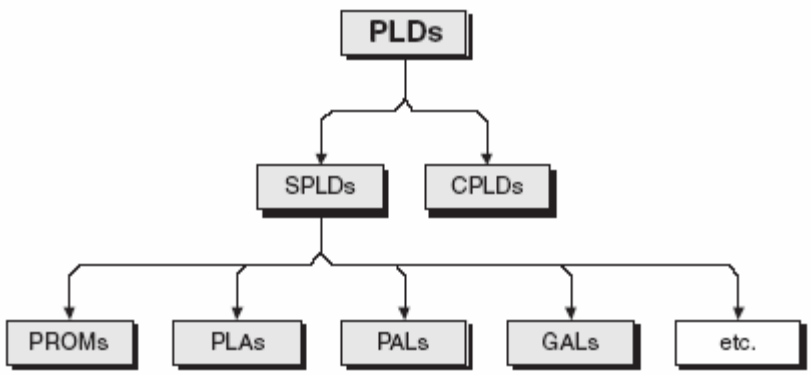


Figure (4.2): PLDs Types

4.3 Types of FPGA

FPGA is silicon chip that has lot of logic gates not connected together, and it is an integrated circuit that have more than 10,000 logic cell; that cells are interconnected by matrix of wires and programmable switches. The function of FPGA is defined by the user program rather than manufacturer. "The program is either 'burned' in permanently or semi-permanently as part of a board assembly process, or is loaded from an external memory each time the device is powered up" [25]. Every FPGA has three major component, configurable logic blocks (CLBs), input/output blocks (IOBs), and interconnects. CLBs is responsible for building the logical circuit for the user. IOBs are responsible for the interface between package pins and internal signal lines. Interconnects are responsible for routing paths to connect the inputs and outputs of the CLB and IOB [25]. Manufacturers use different technologies for the implementation in FPGA. Among these technologies, and the technology

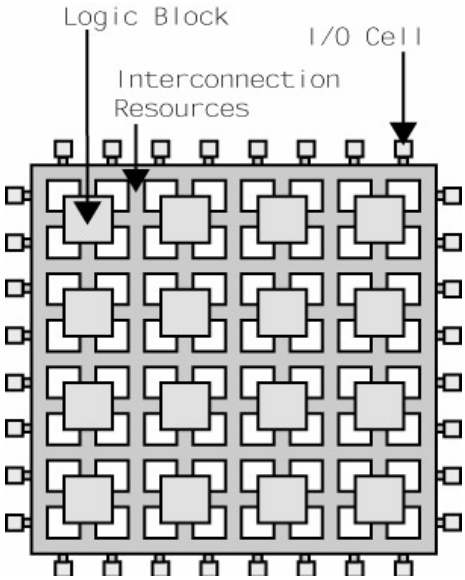


Figure (4.3): FPGA Architecture

of choice that allows programming the device more than once. Examples of these technologies are:

1- **EPROM technology**: This technology uses transistors erasable programmable read-only memory (EPROM). Its main disadvantage is the operation of reconfiguration that requires the use of an ultraviolet source.

2- **EEPROM technology:** This technology uses transistors electrically erasable programmable read-only memory (EEPROM). Compared to the EPROM technology, it has the advantage that it can be reprogrammed electrically.

3- **Technology Static Ram (SRAM):** For this technology, connections are made by making the pass transistors. This technology allows rapid reconfiguration of the FPGA. However, its main disadvantage is the space required for SRAM.

4- **The FLASH technology:** This technology is limited in the number of reconfigurations and has a reconfiguration time compared to longer-SRAM technology. However, the advantage of this technology is that it retains its configuration even if power is removed. Therefore, an FPGA-that depends on Flash technology can turn off its power and the program will not be erased.

Figure 4.4 shows the structure of CLB Xilinx technology. This structure includes a LUT of 4 bits that can implement any function of four variables combinational logic. This LUT can also be configured as a RAM (16 × 1) or register shift of size 16 bits. It also includes a multiplexer and a flip-flop, D flip-flop and all its control inputs (clock, reset, enable).

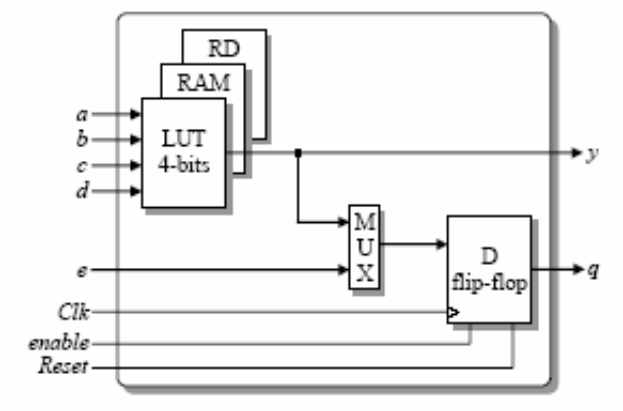


Figure (4.4): Structure of a Logic Cell

These days, FPGAs offer the possibility of using dedicated blocks such as memories RAM, multipliers cabled PCI interfaces and processor cores. The design of control architectures is done using CAD tools. There are two commonly used language, Very high speed integrated VHDL and Verilog . These two languages are standarized and compatible with all FPGA technologies previously introduced. Figure 4.5 shows the FPGA programming steps. This thesis will use the Spartan 3e FPGA from Xilinx company.

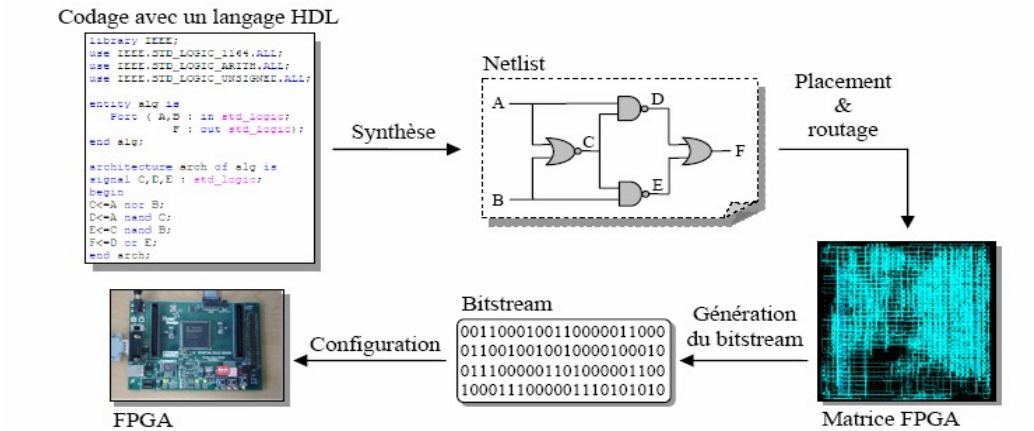


Figure (4.5): Programming a FPGA

*** The Spartan-3E Development System**

The Spartan-3E, a kind of FPGA board, is used to design the very-large-scale integration (VLSI) Test Module, which provides an advanced hardware platform that can be used to create a complex system, (see Figure 4.6). The Spartan-3E family is specifically designed to meet the needs of high volume, cost-sensitive consumer electronic applications [26].

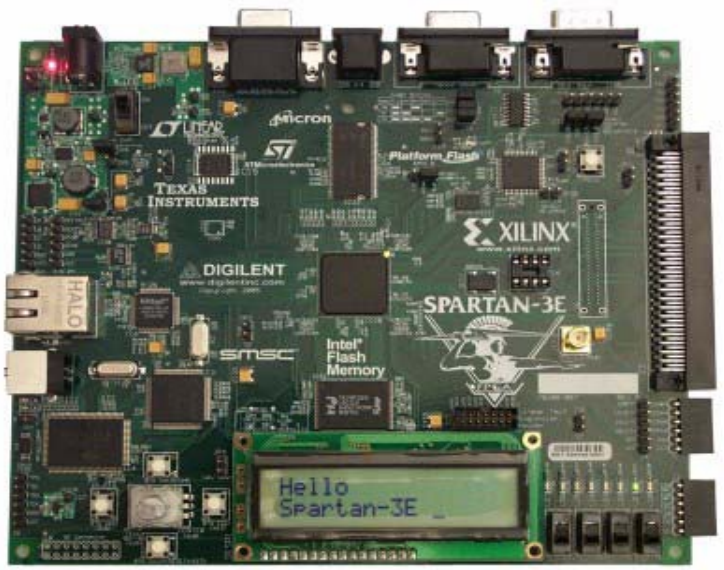


Figure (4.6): The Spartan-3E Development System Board Photo

The Spatan-3E contains these important features:

- Very low cost, high-performance logic solution for high-volume, consumer-oriented applications.
- Proven advanced 90-nanometer process technology.
- Eight discrete LEDs
- Four slide switches
- Four push-button switches
- Xilinx XC3S500E Spartan-3E FPGA: up to 232 user-I/O pins, 320-pin FPGA package, and over 10,000 logic cells.
- Multi-voltage, multi-standard Select IO interface pins.
- Up to eight DCMs.

- Complete Xilinx ISE and Webpack development system support.
- A 2-line by 16-character LCD .
- VGA display port.
- PS/2 mouse or keyboard port.
- On-board USB-based FPGA/CPLD download/debug interface.
- 50 MHz clock oscillator.
- Hirose FX2 expansion connector.
- Three Digilent 6-pin expansion connectors.
- Four-output, SPI-based DAC.

Including the above functions, Spartan-3E has these specific features:

- Parallel NOR flash configuration
- MultiBoot FPGA configuration from parallel NOR flash PROM
- SPI serial flash configuration
- MicroBlaze 32-bit embedded RISC processor
- KCPSM3 8-bit embedded controller
- DDR memory interfaces.

4.4 FPGA and Fuzzy Implementation

As presented in chapter 2 fuzzy systems have became very popular in recent years. Finding many different hardware implementation of FLSs, general-purpose microprocessors and microcontrollers are mostly used for implementing FLS in hardware, but with the complex systems these devices can not perform operations assigned to it as required. There are many researches of FLC implementation on FPGA [2][3][4]. These researches discussed the implementation of FLS onto FPGA, but writing VHDL code is not easy. Because fuzzy systems have many correlated parameters, writing VHDL code needs good knowledge in VHDL language. Reducing design time of FLS causes focusing on optimizing the FLS to give high performance. Software CAD tools are used to deal with the complex tasks of FLS. Some of these tools are [27]:

4.4.1 FuzzyTech

This tool is graphic user interface (GUI). It is used to develop FLSs, and generates different codes that can be useful in many hardware and software tools, these codes are:

1- portable C code that can be used with other software on different target hardware devices.
2- assembly code for microcontroller and DSP such as (PIC and Motorola microcontrollers).
3- code that integrates conventional control techniques.
4- M code that can be used by Matlab™/Simulink™.

4.4.2 Rigel's Fuzzy Logic Applications Software Helper (rFLASH)

It is a code generator that can create a subroutine in assembly code to implement the FLC. rFLASH creates FLC code from the high level control description files. This tools also has a simulator that generates the outputs from given inputs.

4.4.3 Fuzzy Inference Development Environment (FIDE)

It is a CAD tool that run under Windows™. This tool can generate ANCI C code, and assembly code that can work with Motorola micro controllers. As shown these tools can not generate a VHDL code that work with FPGA devices. But since 1992 XFUZZY CAD tool has been developed, and has evolved with the passage of time. This thesis uses Xfuzzy to generate VHDL code for FLC.

4.5 Xfuzzy

Xfuzzy is a CAD tool that was developed using JAVA language. This tool can be used under two operating system (Windows and UNIX). It offers the safety advantages of Java programs. This tool is a combination of several tools, that covering the four stages of FLS design: description, tuning, verification, and synthesis stages. Figure 4.7 shows the Interface Applied of the Xfuzzy [27][28][29].

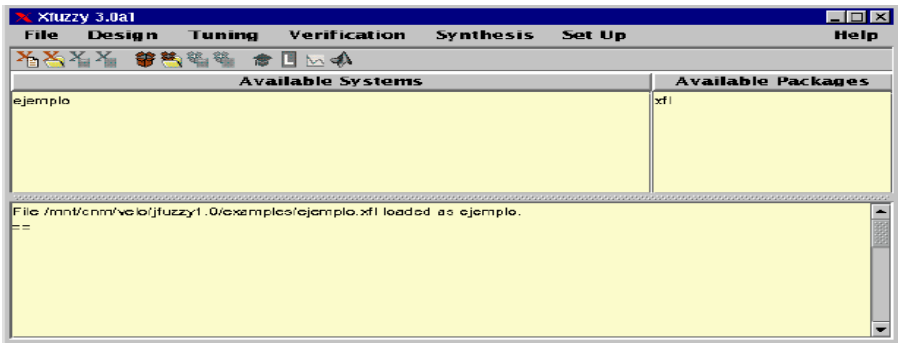


Figure (4.7): Main Window of Xfuzzy 3.0.

4.5.1 Description Stage

Xfuzzy has two tools that are used in this stage (*xfedit and xfpkg*.). The first tool dedicated to the definition of linguistic variables and the logical relations between them. This definition can be designed using GUI or by editing the ".XLF" extension file. Figure 4.8 shows the main windows of *xfedit*. From this window the designer can edit the number of inputs and outputs, types of the membership of the variables, the rule base, and the operators between the variables.

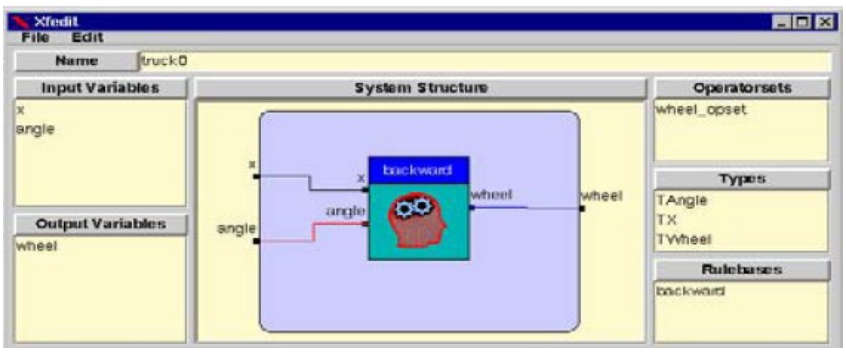


Figure (4.8): Main Window of "xfedit"

The second tool *xfpkg* responsible for all mathematical function of the *xfedit*. For example, the result of OR operator can be programmed to be the minimum of the two variables or can be programmed to another relation. Another example the membership of the variables can be generated by the mathematical function then this function can be modified using *xfpkg* [30].

4.5.2 Tuning Stage

Tuning fuzzy system is complex task**,** there are many parameters that affect the performance of fuzzy systems such as the rule base, membership ranges, and the number of memberships for every variable. In Xfuzzy the tuning process depends on changing the membership parameters, but this changing can not be happening randomly. So it needs to use some optimization algorithms. Xfuzzy contains a tool named *xfsl* dedicated to the tuning of fuzzy systems with supervised learning algorithms. This tool has some famous algorithms that can be used in optimization and machine learning, for example Steepest Descent, Backpropagation, Backpropagation with Momentum, Adaptive Learning Rate, Adaptive Step Size, Manhattan, QuickProp and RProp [30]. To work with this tool, users must have two types of data, training data and testing data. This tool works only if the designer will use fuzzy systems as classifier. So in control field this will not work with all the control problems, because not all control problems have training data. Figure 4.9 shows the main window of the *xfsl* tool. In addition, *xfsl* contains also two algorithms to simplify the designed fuzzy system. First algorithm prunes the rules and reduces the membership functions. The second algorithm clusters the output membership functions. This algorithm is used in system identification.

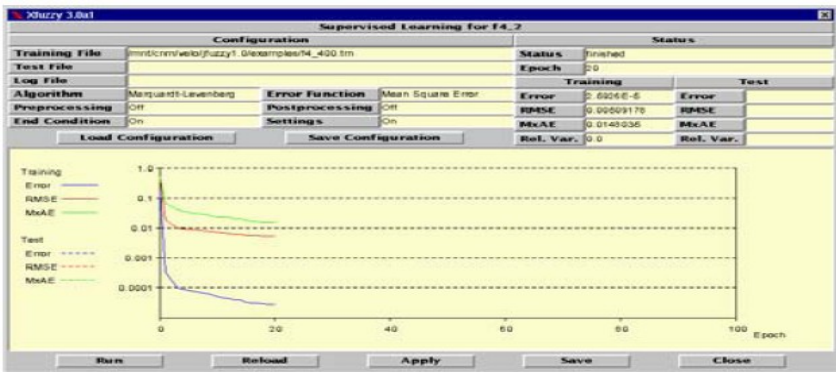
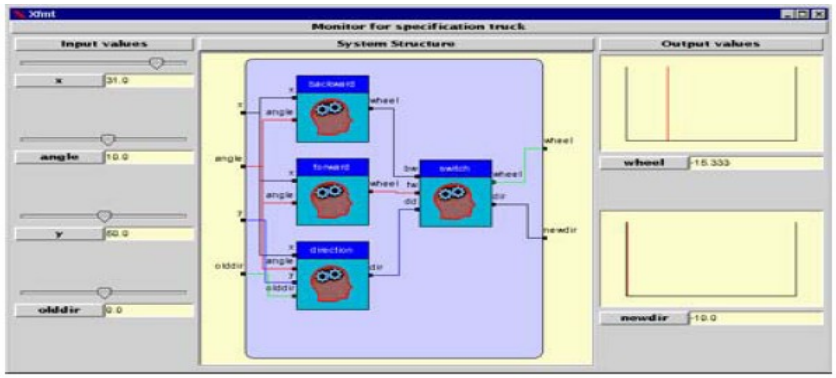


Figure (4.9): Main Window of "xfsl."

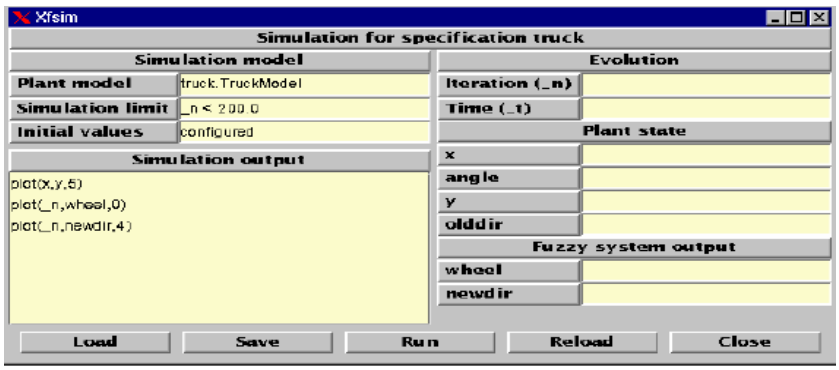
4.5.3 Verification Stage

This stage work as simulation stage which be used to study the behavior of the system, and detect the potential deviations from behavior and the source of this deviations. Xfuzzy has four verification tools for these purposes: *xf2dplot, xf3dplot, xfmt,* and *xfsim*. The first two tools work as surface plotter in 2 dimensions and 3 dimensions. *Xfsm* work as simulator to the FLS, and its behaviors when it connect to the plant in a closed-loop configuration. The tool

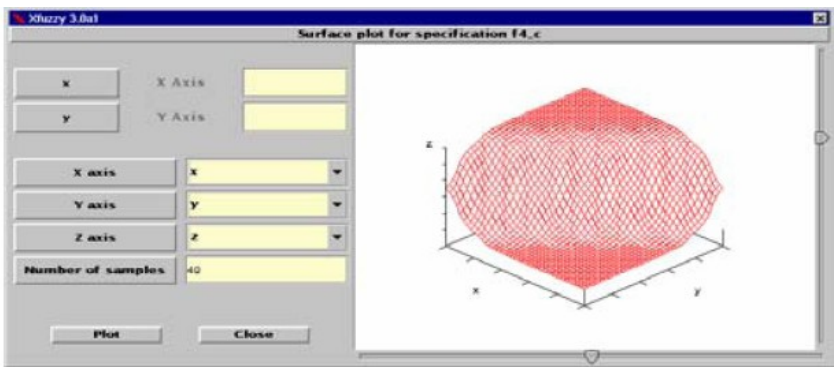
xfmt allows monitoring the system at all the hierarchical levels, and shows the rules that are firing at any value of the input variables. Figure 4.10 shows *xfmt, xfsim, xf3dplot* respectively.



(a)



(b)



(c)

Figure (4.10): a) Main Window of "xfmt" - b) Main Window of "xfsim"- c) Main Window of "xf3dplot".

4.5.4 Synthesis Stage

It is the final stage of the FLS design, in this stage the FLS are converted to different codes depending on the software or hardware target that the designer will use fuzzy system with. This tool can convert the FLS to different codes, these codes is C code, Cpp code, java code, sysgen code that work with Xilinx toolbox in Matlab simulink, and finally VHDL code that used in FPGA devices. This thesis works with VHDL code and sysgen code to simulate the FPGA fuzzy in Matlab simulink, and interfaces the magnetic levitation with this blocks.

CHAPTER 5 MAGNETIC LEVITATION AND FUZZY CONTROLLER

5.1 Magnetic Levitation

Magnetic ball levitation is a difficult task that requires simultaneous manipulation of several controls in order to achieve a desired movement. The question was raised on whether there is a possibility to develop an agent that will act as an intermediate between the user and the magnetic ball levitation. The electro-magnetic levitation system is a mechatronic system accepted both for the specific mechatronic area and for other engineering fields. The magnetic levitation system is a recommended subject for the academic curricula in mechatronic study programs, due to the synergic integration of the sensorial elements, the control subsystem and the actuating subsystem. magnetic levitation systems have many varied uses such as in frictionless bearings, high-speed MAGLEV passenger trains, levitation of wind tunnel models, vibration isolation of sensitive machinery, levitation of molten metal in induction furnaces and levitation of metal slabs during manufacture. These systems have nonlinear dynamics that are usually open-loop unstable and, as a result, a high performance feedback controller is required to control the position of the levitated object. Due to inherent nonlinearities associated with electromechanical dynamics, the control problem is usually quite challenging to the control engineers, since a linear controller is valid only about a small region around a nominal operating point. In this thesis the CE 152 magnetic levitation model will be used.

5.2 Magnetic Levitation Model CE 152:

The CE 152 magnetic levitation model is one of the ranges of educational scale models offered by Humusoft Company for teaching system dynamics and control engineering principles. The model belongs to the range of teaching systems directly controllable by a PC computer in real time. The CE 152 Magnetic Levitation model is one dimensional strongly unstable system designed for studying system dynamics and experimenting with number of different control algorithms based on classical and control theory. Figure 5.1 shows CE 152 model and its components.

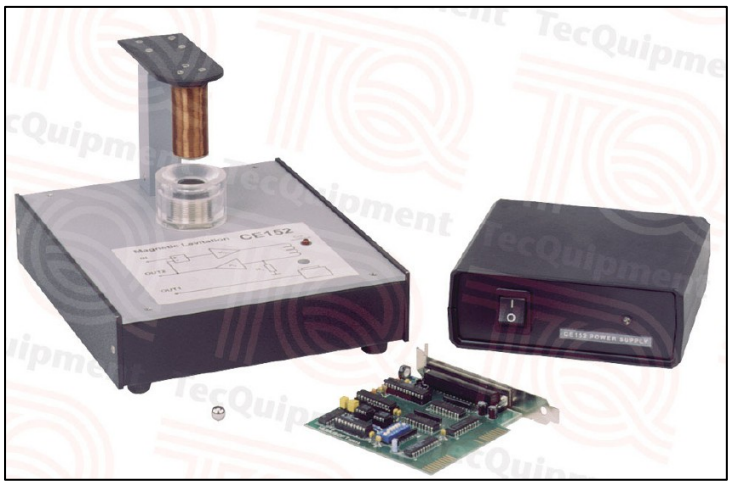


Figure (5.1): CE152 Magnetic Levitation Model

A system configuration for the CE152 follows from the Figure 5.2 where the system is connected to PC compatible computer.

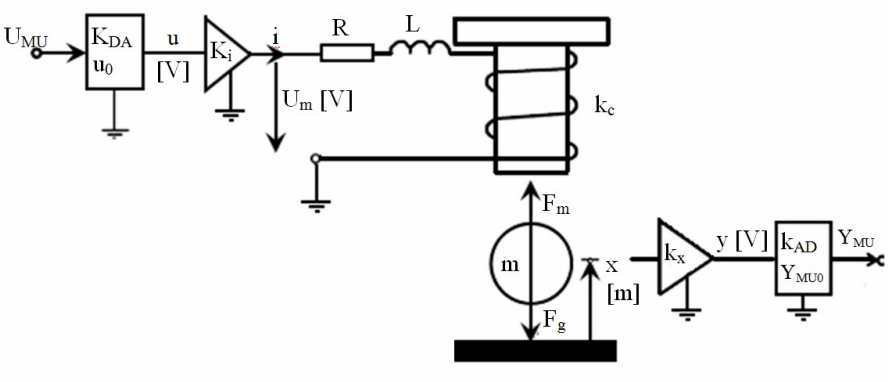


Figure (5.2): Interface to the CE152 Magnetic Levitation Model.

5.3 Model Analysis

The model shown in Figure 5.2 consists of the following blocks [31]:

1. D/A converter.
2. Power amplifier.
3. Ball & coil subsystem.

4. Position sensor.
5. A/D converter.

5.3.1 D/A Converter.

The first part of Magnetic Levitation model is digital to analog (D/A) converter this part is not found in the CE 152 model, but it is refer to the Matlab and data acquisition card (DAQ) that used as the interface between the CE 152 model and the computer. As shown in figure 5.3

$$u = k_{DA} * u_{MU} + u_0 \tag{5.1}$$

Where:

u=Model output voltage [V]

$u_{MU=}$ D/A converter input [MU]

kDA=D/A converter gain [V/MU]

u_0 = D/A Converter Offset [V]

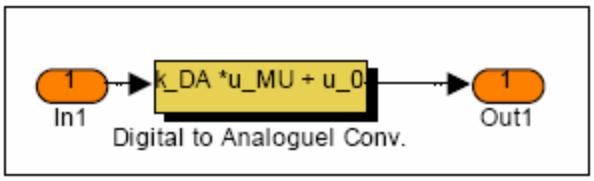


Figure (5.3): Digital to Analog Converter Model.

5.3.2 Power Amplifier

The power amplifier is designed as a source of constant current with the feedback current stabilization. This part gives electric power to the control signal that out from the data acquisition card (DAQ) . The amplifier and coil subsystem can be modeled with the transfer function of 1st order as shown in equation (5.5), and the power amplifier block is shown in Figure 5.4.

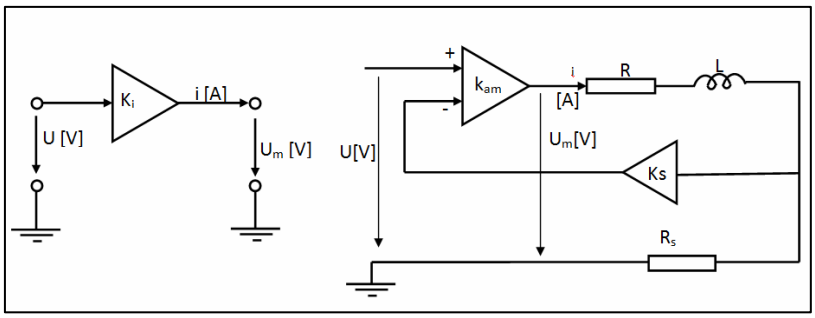


Figure (5.4): The Power Amplifier and its Internal Structure.

$$u_m = iR + L\frac{di}{dt} + R_s i \tag{5.2}$$

$$u_m = K_{am}\left(u - K_s\left(iR\right)\right) \tag{5.3}$$

From equation (5.2) and equation (5.3)

$$\therefore iR + L\frac{di}{dt} + R_s i = K_{am}\left(u - K_s\left(iR_s\right)\right) \Rightarrow IR + LIS + R_s I = K_{am}U - K_{am}K_s R_s I$$

$$\Rightarrow \frac{I}{R} = \frac{K_{am}}{R}\left(\frac{1}{\frac{L}{R}S + 1 + \frac{R_s - K_{am}K_s R_s}{R}}\right)$$

if $R \gg R_s - K_{am}K_s R_s$ such as this system $\Rightarrow \frac{I}{R} = \frac{K_{am}}{R}\left(\frac{1}{\frac{L}{R}S+1}\right)$ (5.4)

simplify the previous relation

$$\frac{I}{U} = K_i \frac{1}{T_a s + 1} \tag{5.5}$$

where K_i is Gain , T_a is time constant

Figure 5.5 shows the matlab simulink block for power amplifier and coil subsystem.

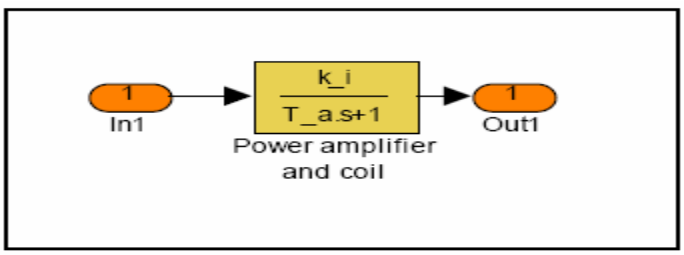


Figure (5.5): Power Amplifier and Coil Model.

5.3.3 Ball & coil subsystem

The motion equation is based on the balance of all forces acting on the ball, i.e. gravity force *Fg*, electromagnetic force *Fm* and the acceleration force as shown in Figure 5.6.

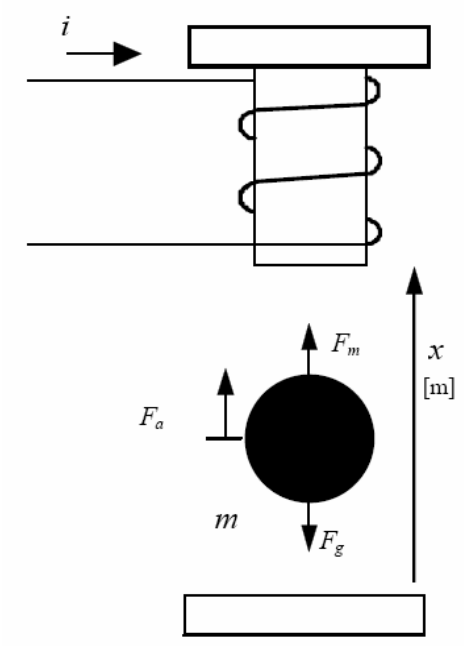


Figure (5.6): The Forces That Effect on the Ball Motion.

The net force

$$F_a = F_m - F_g \tag{5.6}$$

Where;

$$F_m = \frac{i^2 k_c}{(x - x_0)^2} \tag{5.7}$$

$$F_g = m_k g \tag{5.8}$$

$$F_a = m_k \ddot{x} \tag{5.9}$$

Substituting equations (5.7), (5.8) and (5.9) into equation (5.6)

$$m_k \ddot{x} = \frac{i^2 k_c}{(x - x_0)^2} - m_k g \tag{5.10}$$

Limits of the ball movements and ball damping is taken into account. So, to model the damping, the term k_{fv} is introduced into the equation

$$m_k\ddot{x}+k_{fv}\dot{x}=\frac{i^2k_c}{(x-x_0)^2}-m_kg \qquad (5.11)$$

5.3.4 Position Sensor

An inductive position sensor shown in Figure 5.7 is used to measure the ball position. The sensor can be approximated with a linear function:

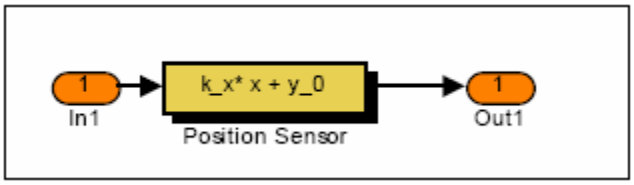


Figure (5.7): Position Sensor Model.

$$Y=k_x x + Y_0 \qquad (5.12)$$

K_x = Position sensor gain [V/m].

Y= Model output voltage [V].

x= Ball position [m].

Y_0= Position sensor offset [m]

5.3.5 A/D Converter

This block includes the influence of the analog to digital (A/D) acquisition card plus the software used for the data acquisition. If the influence of limits is neglected, this relationship is described by a linear function. Figure 5.8 shows the matlab simulink block for A/D subsystem.

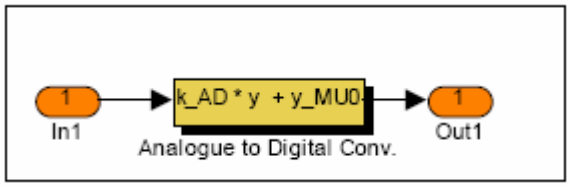


Figure (5.8): A/D Model

$$Y_{MU} = K_{AD} Y + Y_{MU0} \quad (5.13)$$

Y_{MU} = A/D Converter Output [V]

K_{AD} = A/D Converter Gain [MU/V]

Y_{MU0} = Converter Offset [MU]

Y = Model Output Voltage [V]

5.3.6 Complete Model

The final block diagram of the magnetic levitation model CE 152 is shown in the Figure 5.9.

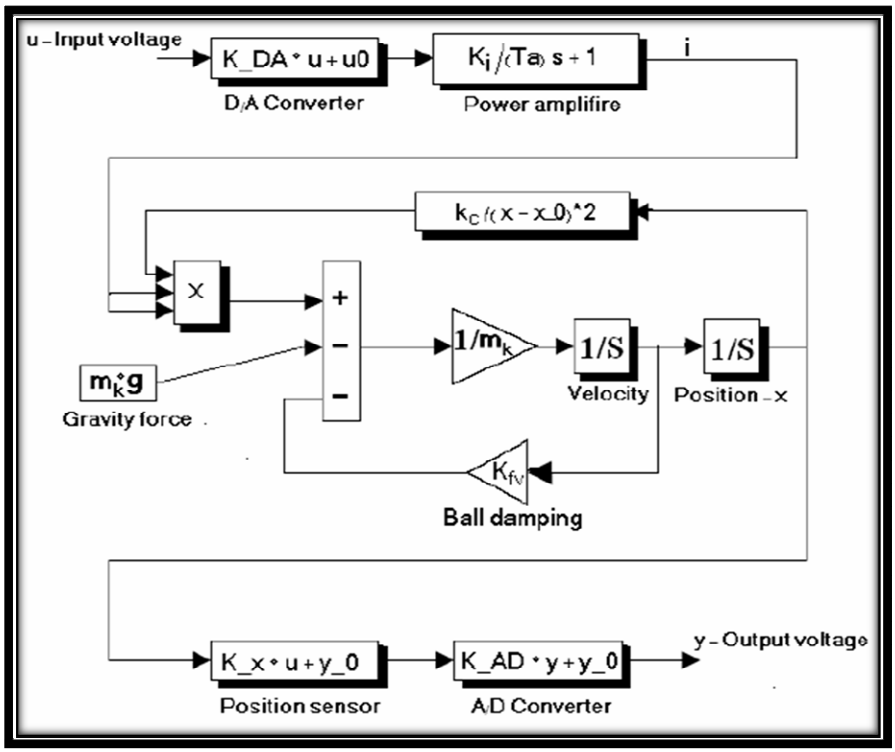


Figure (5.9): The Complete Model of Magnetic Levitation CE152

The magnetic levitation model CE152 can be modeled by second order differential equation 5.11.

let $x_1 = x$ (5.14)

$x_2 = \dot{x}_1 = \dot{x}$ (5.15)

$x_3 = i$ (5.16)

Where x_1 is ball position state, x_2 ball velocity state, and x_3 is the current state.

Substituting equations (5.14), (5.15) and (5.16) into equation (5.11)

$$\Rightarrow \dot{x}_2 = \ddot{x} \Rightarrow \left(m_k \dot{x}_2 + k_{fv} x_2 = \frac{i^2 k_c}{(x_1 - x_0)^2} - m_k g \right) \tag{5.17}$$

$$\begin{bmatrix} \dot{x}_1 = x_2 \\ \dot{x}_2 = \dfrac{i^2 k_c}{m_k (x_1 - x_0)^2} - g - \dfrac{k_{fv} x_2}{m_k} \\ \dot{x}_3 = \dot{I} \end{bmatrix} \quad \text{(nonlinear state space model)} \tag{5.18}$$

From equation (5.5) $\Rightarrow (T_a s + 1) I = K_i U$

Where i(t) can be defined such as:

$$T_a \dot{i}(t) + i(t) = K_i u(t) \Rightarrow \dot{i}(t) = \frac{K_i u(t) - i(t)}{T_a} \tag{5.19}$$

Linearization of this nonlinear model can be obtain by Taylor series

from equation (5.17)

Using Taylor to linearize the term $\dfrac{i^2 k_c}{(x - x_0)^2}$ linearization points x_{00} and i_{00}

$$f(x,i) = \frac{i^2 k_c}{(x - x_0)^2} = k_c \left[\frac{i}{(x - x_0)} \right]^2 \tag{5.20}$$

$$f(x,i) \simeq f(x_{00}, i_{00}) + \left(\frac{\partial (f(x_{00}, i_{00}))}{\partial (x_{00})} x(t) + \frac{\partial (f(x_{00}, i_{00}))}{\partial (i_{00})} i(t) \right) \tag{5.21}$$

$$\therefore k_c \left[\frac{i}{(x - x_0)} \right]^2 = \frac{i_{00}^2 k_c}{(x_{00} - x_0)^2} + \left[\frac{-2 k_c i_{00}^2}{(x_{00} - x_0)^3} \right] x(t) + \left[\frac{2 k_c i_{00}}{(x_{00} - x_0)^2} \right] i(t) \tag{5.22}$$

Substituting equation (5.22) into equation (5.17)

$$m_k \ddot{x} = \frac{i_{00}^{\,2} k_c}{(x_{00}-x_0)^2} + \left[\frac{-2*k_c*i_{00}^{\,2}}{(x_{00}-x_0)^3}\right] x(t) + \left[\frac{2*k_c*i_{00}}{(x_{00}-x_0)^2}\right] i(t) - m_k g - k_{fv}\dot{x}$$

$$\because m_k \dot{x}_2 = \left[\frac{-2*k_c*i_{00}^{\,2}}{(x_{00}-x_0)^3}\right] x_1 - k_{fv}\, x_2 + \left[\frac{2*k_c*i_{00}}{(x_{00}-x_0)^2}\right] i(t) + \left(\frac{i_{00}^{\,2} k_c}{(x_{00}-x_0)^2} - m_k g\right) \qquad (5.23)$$

When the ball is fixed at position (x_1) zero; then, the velocity (x_2), acceleration: the derivative of velocity ($\dot{x}_2$) and the coil current (i) are all equal zero.

Substituting $x_1, x_2, \dot{x}_2,$ and $i = 0$ into equation (5.6)

$$\because m_k(0) = \left[\frac{-2*k_c*i_{00}^{\,2}}{(x_{00}-x_0)^3}\right] (0) - k_{fv}\,(0) + \left[\frac{2*k_c*i_{00}}{(x_{00}-x_0)^2}\right] (0) + \left(\frac{i_{00}^{\,2} k_c}{(x_{00}-x_0)^2} - m_k g\right)$$

$$\Rightarrow \left(\frac{i_{00}^{\,2} k_c}{(x_{00}-x_0)^2} - m_k g\right) = 0 \qquad (5.24)$$

subsituting equation (5.24) into equation (5.23)

$$\because m_k \dot{x}_2 = \left[\frac{-2*k_c*i_{00}^{\,2}}{(x_{00}-x_0)^3}\right] x_1 - k_{fv}\, x_2 + \left[\frac{2*k_c*i_{00}}{(x_{00}-x_0)^2}\right] i(t) \qquad (5.25)$$

from equation (5.25) and equation (5.19)

$$\dot{x}_2 = \left[\frac{-2*k_c*i_{00}^{\,2}}{m_k(x_{00}-x_0)^3}\right] x_1 - \frac{k_{fv}\, x_2}{m_k} + \left[\frac{2*k_c*i_{00}}{m_k(x_{00}-x_0)^2}\right] i(t)$$

$$\dot{x}_3 = \dot{i} = \frac{K_i u(t) - i(t)}{T_a}$$

$$\therefore \begin{bmatrix} \dot{x}_1 = x_2 \\ \dot{x}_2 = \dfrac{-2*k_c*i_{00}^{\,2}}{m_k(x_{00}-x_0)^3} x_1 - \dfrac{k_{fv}}{m_k} x_2 + \dfrac{2*k_c*i_{00}}{m_k(x_{00}-x_0)^2}\, i(t) \\ \dot{x}_3 = \dfrac{di}{dt} i(t) \end{bmatrix} \qquad (5.26)$$

$$\begin{bmatrix}\dot{x}_1\\ \dot{x}_2\\ \dot{x}_3\end{bmatrix}=\begin{bmatrix}0 & 1 & 0\\ \dfrac{-2*k_c*i_{00}^{\,2}}{m_k\left(x_{00}-x_0\right)^3} & \dfrac{-k_{fv}}{m_k} & \dfrac{2*k_c*i_{00}}{m_k\left(x_{00}-x_0\right)^2}\\ 0 & 0 & -\dfrac{1}{T_a}\end{bmatrix}\begin{bmatrix}x_1\\ x_2\\ x_3\end{bmatrix}+\begin{bmatrix}0\\ 0\\ \dfrac{K_i}{T_a}\end{bmatrix}u(t) \tag{5.27}$$

$$y=\begin{bmatrix}1 & 0 & 0\end{bmatrix}\begin{bmatrix}x_1\\ x_2\\ x_3\end{bmatrix} \tag{5.28}$$

Adding the gain of DAC, ADC, and sensor gain, the new state space model is .

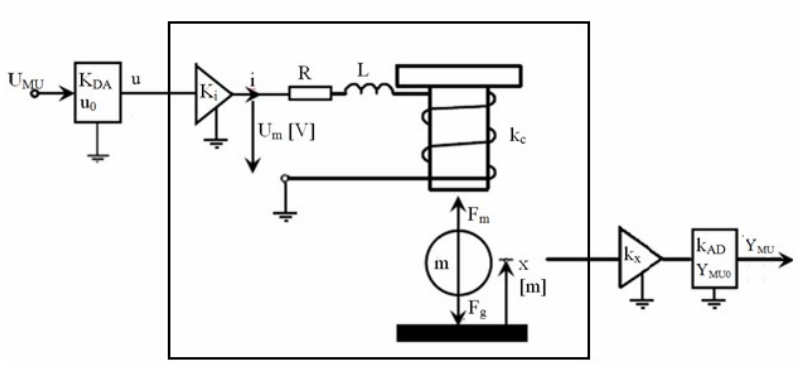


Figure (5.10): Plant of Magnetic Levitation CE152 with ADC, and DAC

$$\begin{bmatrix}\dot{x}_1\\ \dot{x}_2\\ \dot{x}_3\end{bmatrix}=\begin{bmatrix}0 & 1 & 0\\ \dfrac{-2*k_c*i_{00}^{\,2}}{m_k\left(x_{00}-x_0\right)^3} & \dfrac{-k_{fv}}{m_k} & \dfrac{2*k_c*i_{00}}{m_k\left(x_{00}-x_0\right)^2}\\ 0 & 0 & -\dfrac{1}{T_a}\end{bmatrix}\begin{bmatrix}x_1\\ x_2\\ x_3\end{bmatrix}+\begin{bmatrix}0\\ 0\\ \dfrac{K_iK_{DA}}{T_a}\end{bmatrix}u(t) \tag{5.29}$$

$$y=\begin{bmatrix}K_xK_{AD} & 0 & 0\end{bmatrix}\begin{bmatrix}x_1\\ x_2\\ x_3\end{bmatrix} \tag{5.30}$$

by using the practical values which is given in the data sheet of the magnetic levitation

1- Model weights and dimensions:-

Dk = 12.7e-3 m (ball diameter [m])

m_k = 0.0084 (ball mass [kg])

Td = 0.019 m (distance from the ground and the edge of the magnetic coil)

L = 0.019 - Dk (distance of limits [m])

g = 9.81 (gravity acceleration constant [m.s^-2])

2- Coil and amplifier parameters:-

U_DAm = 5 (maximum DA converter output voltage)

Rc = 3.5 (coil resistance [Ohm])

Lc = 30e-3 (coil inductance [H])

Rs =0.25 (current sensor resistance [Ohm])

Ks = 13.33 (current sensor gain)

K_am = 100 (power amplifier gain)

I_am = 1.2 (maximum power amplifier output current)

Ta = L/((Rc+Rs)+Rs*Ks*K_am) = 1.8694e-005 (amplifier time constant [s])

k_i = K_am / ((Rc+Rs)+Rs*Ks*K_am) = 0.2967 (amplifier gain [A/V])

3- Empirical parameters:-

KFv = 0.02 (viscose friction)

D/A converter :-

k_DA =5 (converter gain)

u_0 = 0 (converter offset)

A/D converter:-

k_AD = 0.2 (converter gain)

y_MU0 = 0 (converter offset)

4- Coil and position sensor parameters :-

k_x = 797.4603 (position sensor constant)

x_0 = 8.26e-3 (coil bias [m])

k_f = 0.606e-6 (aggregated coil constant [N/V])

k_c=k_f/(k_i)^2 = 6.8823e-006 (coil constant)

v00 = 0 (ball velocity [m/s] at Equilibrium)

x00= L/2 = 3.2e-3 (ball position at Equilibrium)

u00 = - sqrt(mk * g / k_f) * (x00 - x_0) = 1.8843 (input volt at Equilibrium)

i00 = u00 * k_i = 0.5592 (coil current at Equilibrium)

the final state space matrices is

$$A = 1.0\ e+004 \begin{bmatrix} 0 & 0.0001 & 0 \\ -0.3840 & -0.0002 & 0.0035 \\ 0 & 0 & -5.3492 \end{bmatrix}$$

$$B = 1.0e+005 \begin{bmatrix} 0 \\ 0 \\ 1.5873 \end{bmatrix}$$

C= [159.4921 0 0]

D= [0]

The first step in any control design know if the system is stable or not, by finding the transfer function of it.

$$\textit{The plant} \qquad G(s)= \frac{8.883e8}{s^3 + 5.349e4\ s^2 + 1.312e5\ s - 2.054e8} \tag{5.31}$$

calculating the open loop poles of the system, we get an unstable pole

-53492 (Stable pole), -63 (Stable pole), 61 (Unstable pole)

The system is unstable because it has positive pole (61).

The second step is calculating if the system controllable or not by finding the controllability matrix and test the its rank.

$$\text{Controlability Matrix} = \left[\begin{array}{c|c|c} B & AB & A^2B \end{array}\right]$$

$$= \begin{bmatrix} 0 & 0 & 5.5697e6 \\ 0 & 5.5697e6 & -2.9795e11 \\ 1.5873e5 & -8.4908e9 & 4.519e14 \end{bmatrix}$$

$\Rightarrow \text{Rank} = n = 3$

$\Rightarrow$ Sytem is controllable

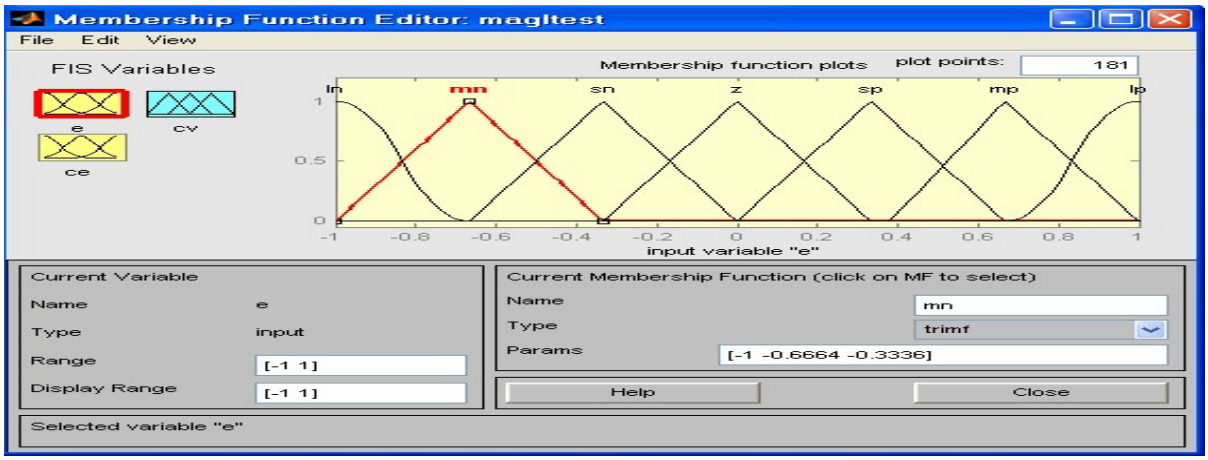
5.4 Fuzzy Controller Design for CE152 Model

To apply the fuzzy logic controller to the magnetic levitation CE 152, certain properties of the system are exploited so that the design of the controller can be made easier. As the system is symmetrical, it is assumed that symmetrical membership functions about the y-axis will provide a valid controller. A symmetrical rule-base is also assumed. Other constraints are also introduced to the design of the FLC:

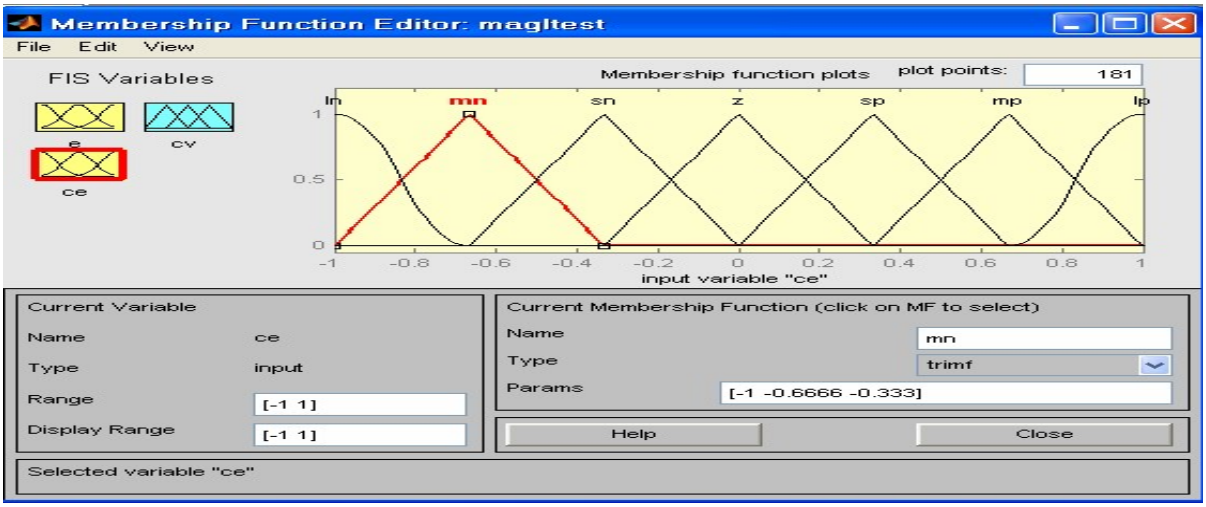
1- All universes of discourses are normalized to lie between –1 and 1 with scaling factors external to the FLC used to give appropriate values to the variables.
2- It is assumed that the first and last membership functions have their apexes at –1 and 1 respectively. This can be justified by the fact that changing the external scaling would have similar effect to changing these positions.
3- Triangular, Z and S membership functions are to be used.
4- The number of fuzzy sets is constrained to be an odd integer greater than unity. In combination with the symmetry requirement, this means that the central membership function for all variables will have its apex at zero.
5- The base vertices of membership functions are coincident with the apex of the adjacent membership functions. This ensures that the value of any input variable is a member of at most two fuzzy sets, which is an intuitively sensible situation. It also ensures

that when a variable's membership of any set is certain, i.e. unity, it is a member of no other sets.

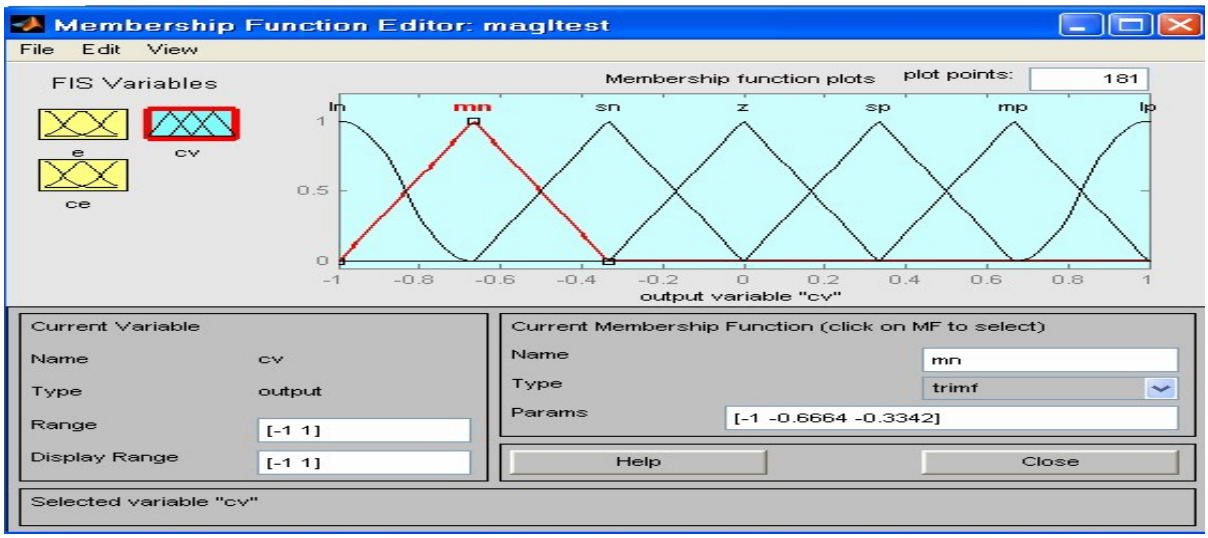
The fuzzy controller of magnetic levitation uses Mamdani model. The FLC has two inputs which are error and change of error and the output is the change of voltage. Figure 5.11 a,b and c shows the membership functions of fuzzy controller using Fuzzy Toolbox of Matlab software. The ranges of the inputs and output is [-1 1]. All have 7 membership function.



(a)



(b)



(c)

Figure (5.11): a) Error - b) Change of Error- c) Change of Voltage

The next step After designing the membership functions is to write the rules of the fuzzy controller. These rules are chosen based on knowledge base and experts. Table 5.1 shows the rules base as a matrix

Table 5.1 Fuzzy Control Rule Base of Magnetic Levitation

e / ce	ln	mn	sn	z	sp	mp	lp
ln	ln	ln	ln	ln	mn	sn	z
mn	ln	ln	ln	mn	sn	z	sp
sn	ln	ln	mn	sn	z	sp	mp
z	ln	mn	sn	z	sp	mp	lp
sp	mn	sn	z	sp	mp	lp	lp
mp	sn	z	sp	mp	lp	lp	lp
lp	z	sp	mp	lp	lp	lp	lp

Table 5.1 shows the matrix form of the fuzzy rules these rules can be written in the next form

IF e is ln AND ce is ln THEN cv is ln

IF e is z AND z is ln THEN cv is z

Figure 5.12 shows the surface of fuzzy controller using the previous rules. Surface shows the relation between the inputs and output at any point in the intervals [-1 1] using the Centeroid defuzzification method.

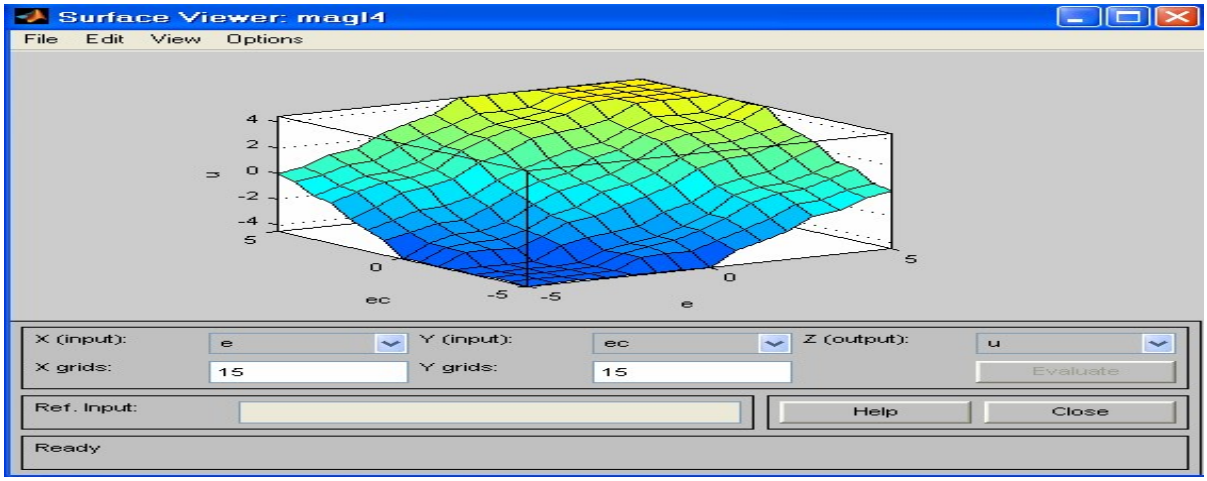
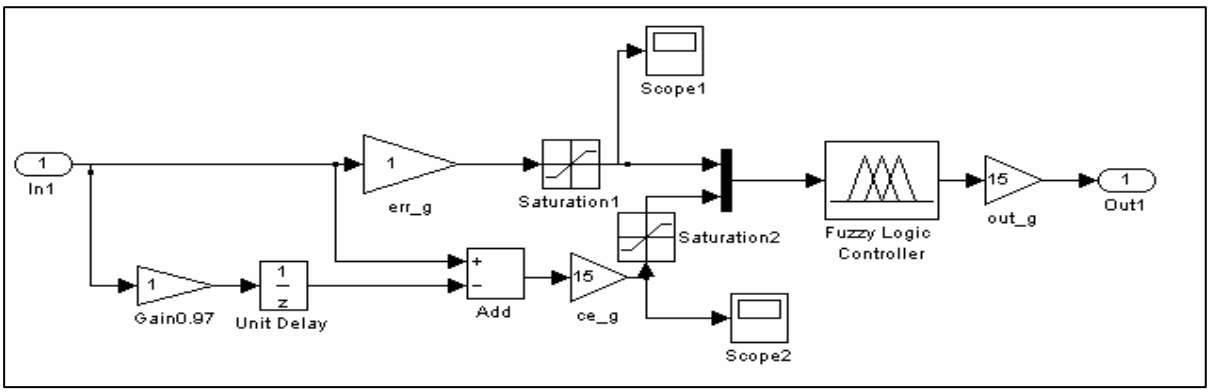
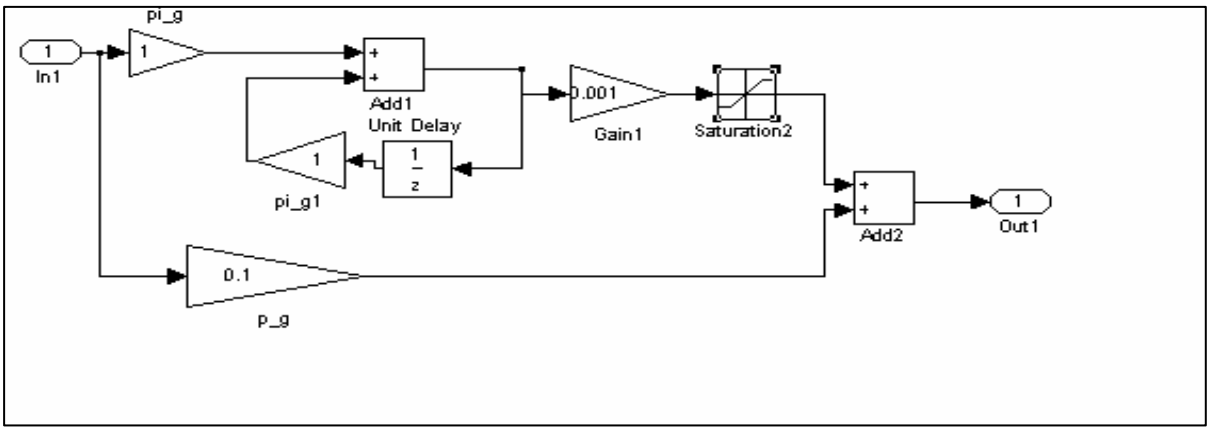


Figure (5.12): Surface of Fuzzy Controller

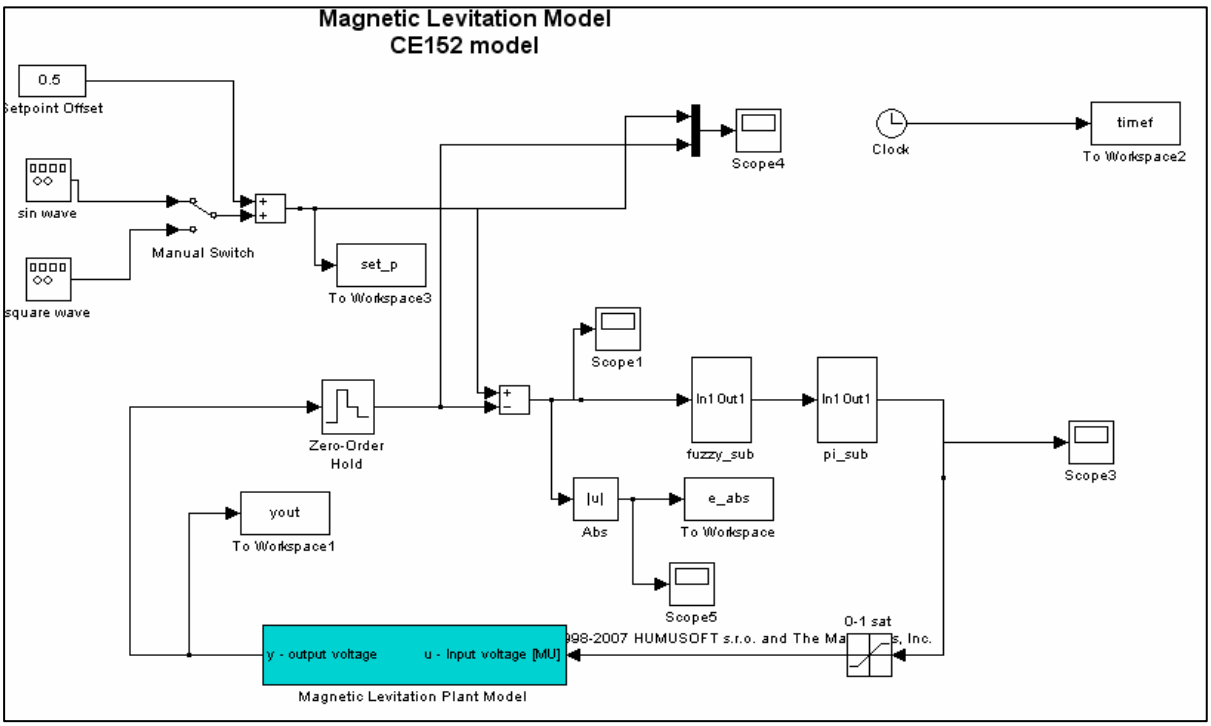
After designing fuzzy controller, the controller will be connected to the CE152 magnetic levitation Matlab simulink model. Figure 5.13 a, b, and c shows the fuzzy controller, proportional integral controller (PI) subsystems and all system respectively. The integral part is used to convert the change of voltage value that come from fuzzy controller to real voltage which will act on the magnetic levitation CE152. Using tuning method the good proportional gain is 0.1 and the integral gain is 1.



(a)



(b)



(c)

Figure (5.13): a) Fuzzy Controller- b) PI Subsystem –c) Magnetic Levitation Model

As shown in Figure 5.13 c there are many signals connected to work space blocks this block will be used with the GA Matlab code to optimize the membership of the inputs and output of the fuzzy controller. Figure 5.14 shows the output of the magnetic levitation after connecting to the fuzzy controller. The set point is unit step has value 0.5 which is in the center of the gap, this point is one of the equilibrium points of magnetic levitation CE152 model

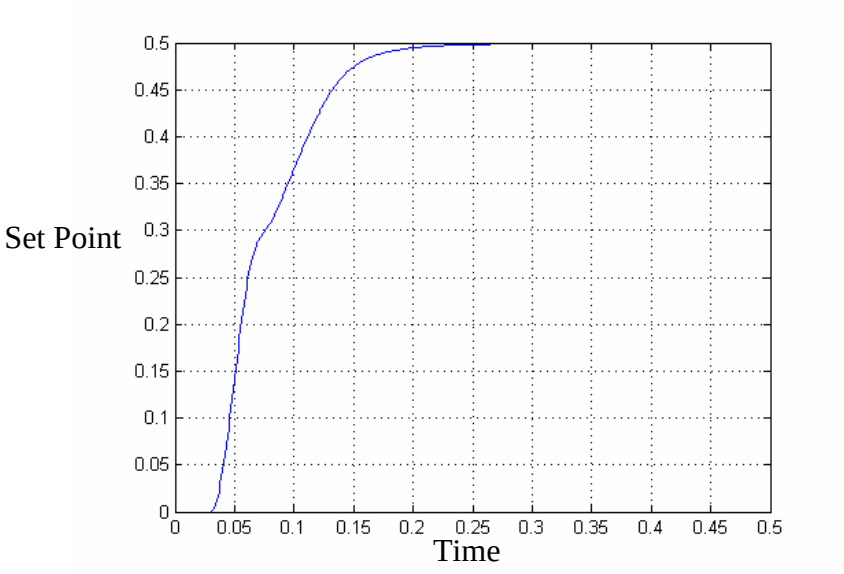


Figure (5.14): Step Response of the System

Figure 5.14 show that there is no overshoot and the settling time is nearly 0.15 sec, rising time is 0.106 sec. The main use of magnetic levitation is high-speed MAGLEV passenger trains. That means there are uncertain parameters will effect on the system, so the controller will be tested with two additional set points signals sin wave and square wave. Figure 5.15 and 5.16 show the output response with that two signals. These figures show that the fuzzy controller can keep the stability of the system with various set points.

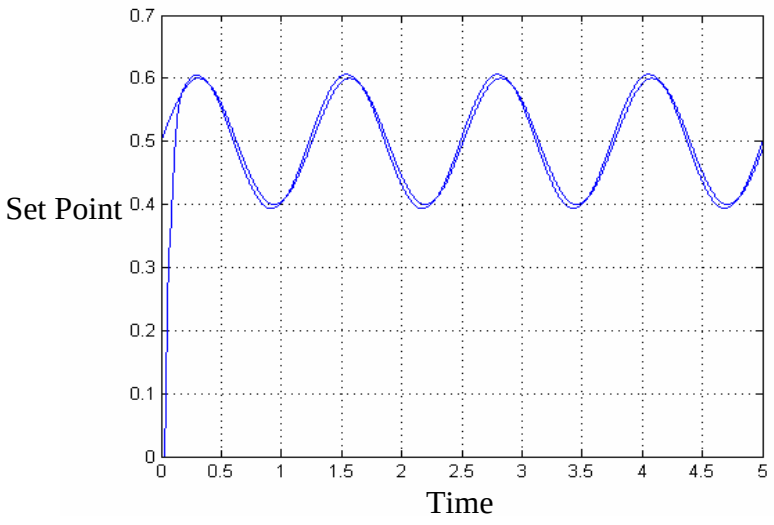


Figure (5.15): Sine Wave Output Response of the System

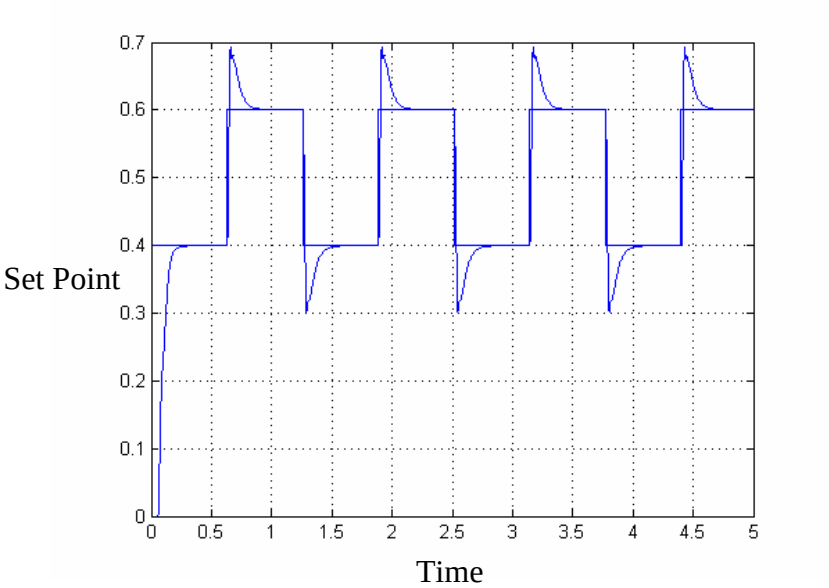


Figure (5.16): Square Wave Output Response of the System

5.5 Fuzzy Controller Design with GA for CE152 Model

There are many parameters effect on the control process of the CE152 model as shown in Figure 5.13 a,b in addition to the shape of the memberships of the inputs and output of fuzzy controller. These parameters is, p_g (proportional gain), pi_g (integral gain), ce_g (change of error gain), err_g (error gain)and out_g (fuzzy output gain).(see Figure 5.13 a,b and c) These gain parameters can be tuned to give near optimal results. GA is used to optimize these parameters and optimize the shape of memberships of fuzzy controller. There are many ways to have good response, first is to optimize gain parameters only without any change in memberships shape. Second is to optimize the memberships shape without any change in gain parameters. The last way is optimize both gain parameters and memberships shape. GA Matlab code (Appendix A) is used to optimize the memberships shape and gain parameters. This program is divided into two codes, main code is responsible for performing the GA steps such as selection, crossover, mutation and make new population. The second code is responsible for testing the new population to calculate the fitness function and out the best fitness value. These codes is very simple GA codes, so that the memberships shape for inputs and output will change equally. In fuzzy controllers, there are two inputs and one output. Every input and output have seven membership functions, five of these membership functions are triangular memberships, one S membership function and one Z membership function. In triangular membership functions there are three parameters that can be modified to change its shape, center, left edge, right edge. In Z and S membership functions there are

two parameters, top edge and left or right edges (depending on if it is S or Z membership). But in this thesis some simplification are used in writing GA code.

1- The interval of the inputs and output will still at [-1 1] range.
2- The symmetric point will still at zero.
3- The inputs and output will have he same shape.

After these simplification the number of variable will be 8 for membership functions and 5 for the gain parameters. The fitness function that will be used is the integral of the absolute values of the error, in the control design the fitness value need to be minimized. The number of individuals per one population will be 50, the number of generation will be 100, the crossover probability will be 0.7. The firs step in matlab program, generate random population consist of 50 individual, every individual consist of 13 variable every variable is coded in 10 bit binary form. Second step, convert the binary code of every variable to real number. Third step, call fuzzy control program and simulink program and find the fitness value of the fitness function (the integral of absolute values of error). Fourth step, implement crossover and mutation, and finally generate the new population. The stop condition can be after exact times or after the exact value of fitness value. in this thesis the stop condition is after 100 generation. Figure 5.17 shows the change of the fitness values after running the Matlab code.

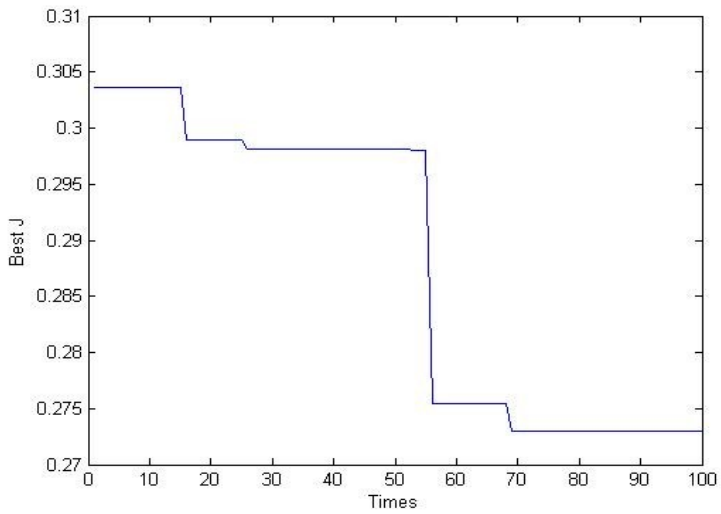


Figure (5.17): Fitness Values

Figure 5.17 shows that the value of fitness function is decreased from 0.305 to 0.275. Figure 5.18 shows the memberships shape of the inputs and output.

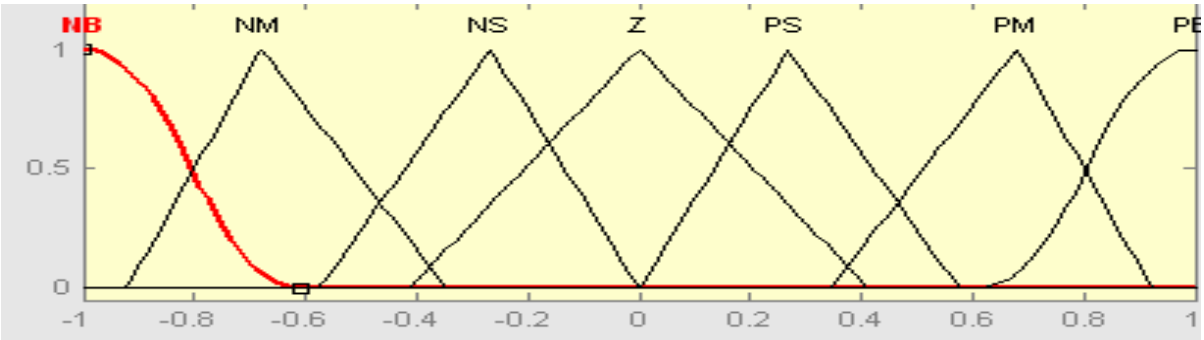


Figure (5.18): Membership Functions of the Fuzzy Controller with GA

Table 5.2 shows the difference between gain parameters with and without GA optimization

Table 5.2 Gain Values with and without GA optimization

Gains	with GA optimization	without GA optimization
err_g	1.0955	1
ce_g	15.4941	15
out_g	15.4932	15
p_g	0.15797	0.1
pi_g	2.7363	1

Figure 5.19, 5.20 and 5.21 show the system response of step, sin wave and square wave inputs with these new values.

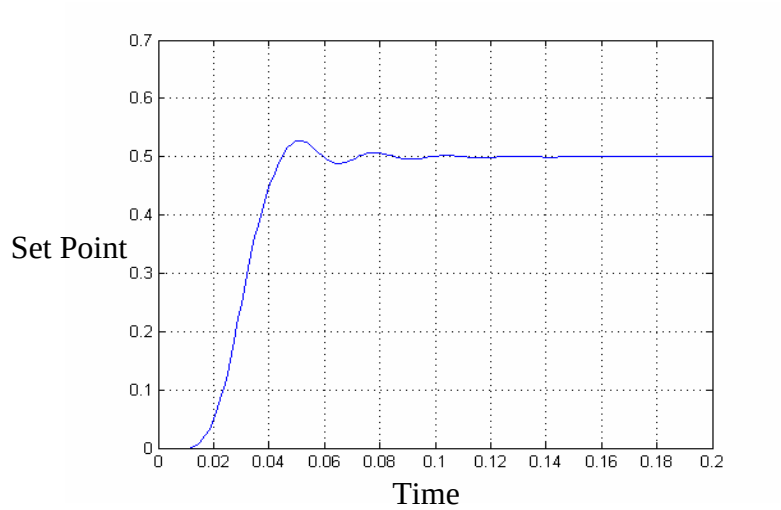


Figure (5.19): Step Response of the System with GA

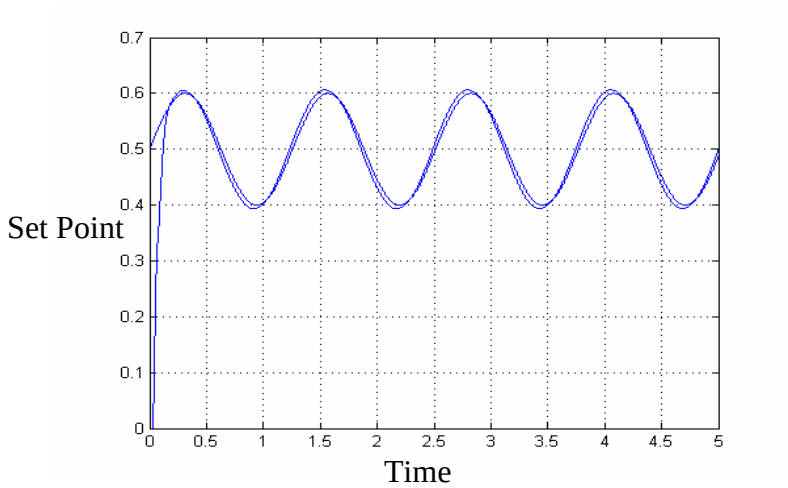


Figure (5.20): Sine Wave Output Response with GA

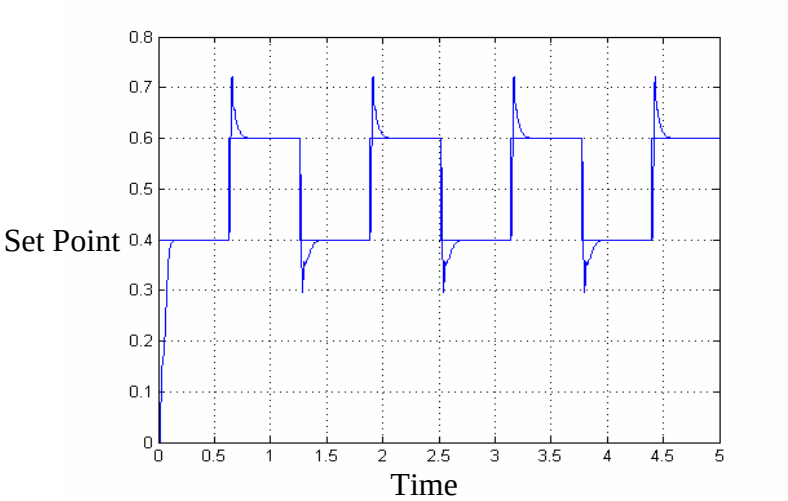


Figure (5.21): Square Wave Output Response with GA

From Figure 5.19 the overshoot is nearly 6%, the settling time is 0.046 sec and rising time is 0.023 sec. table 5.3 shows the comparison between the system response with and without GA optimization

Table 5.3 System Response with and without GA optimization

System response for step input	With GA optimization	Without GA optimization
Overshoot	6%	No overshoot
Settling time	0.046 sec	0.15 sec
Rising time	0.023 sec	0.106 sec

by comparison between these results with without GA optimization can find that GA give better results and improved the response of the system.

5.6 Comparison between Fuzzy GA controller and H_2 Controller

H_2 controller one of the famous optimal control techniques, This type of controllers is used to reach the optimal state of the system. But the design of this technique is difficult task, because it works with linear systems, needs the mathematical model of the system and it depends on weighted functions. Figure 5.22 shows the block diagram of H_2 controller [32].

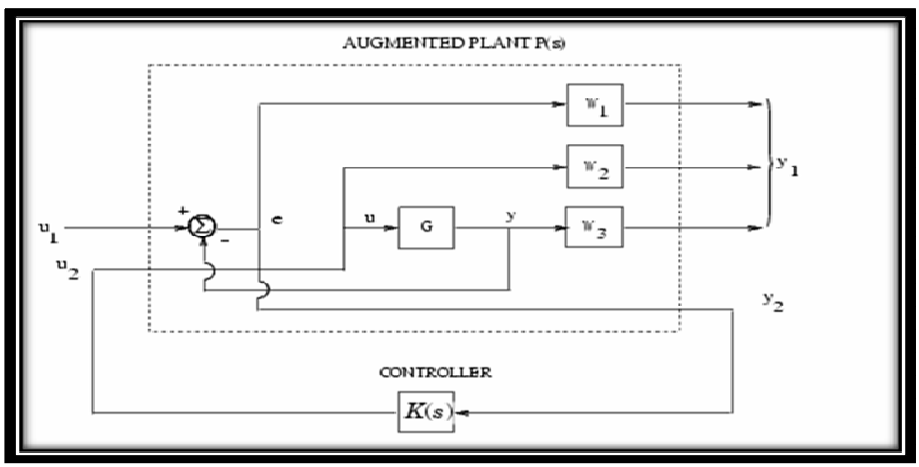


Figure (5.22): H_2 Controller Block Diagram

In this section the H_2 controllers will be designed and will be compared with fuzzy GA controller. First step, find the linear approximation model of the magnetic levitation equations (5.29), (5.30). Second step, test the stability and controllability of the system as shown in 5.3.6. Third step, convert single input SISO system to MIMO system because H_2 controller works as MIMO controller [33]. Fourth step, finding Figure 5.23 shows the matlab simulink blocks of H_2 controller for CE152 magnetic levitation. The H_2 controller block in this figure is stat space block has 4 parameters A, B,C and D matrices. These matrices is generated after running the matlab code in appendix (c).

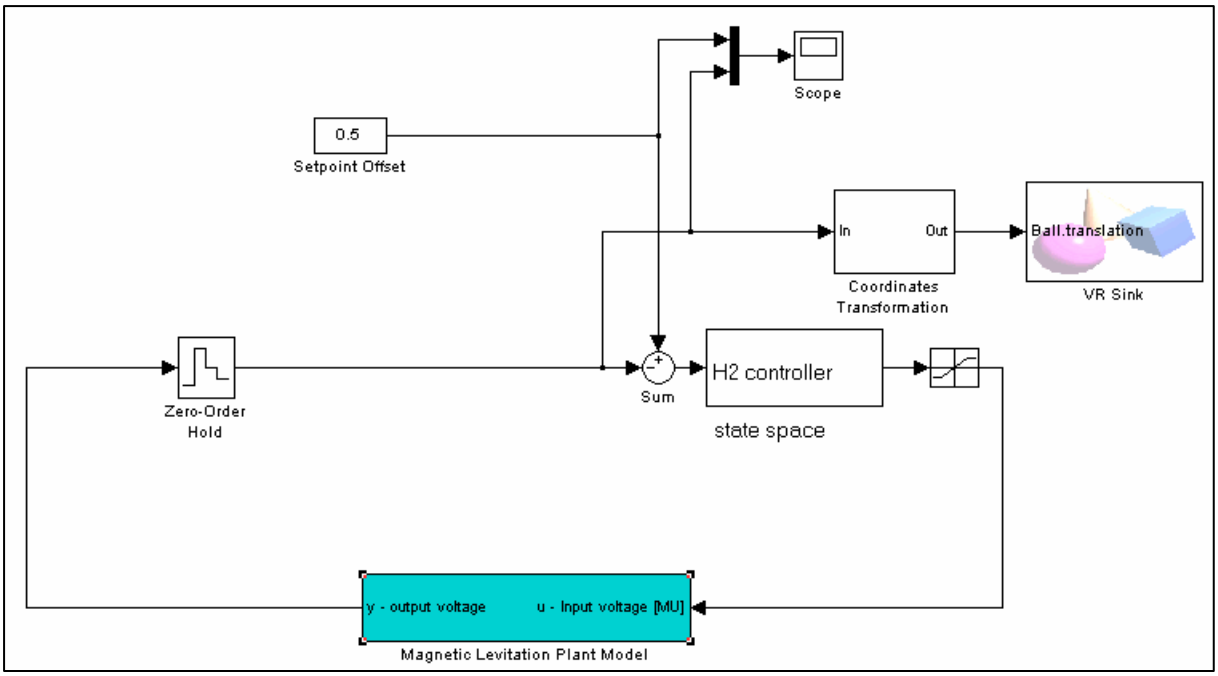


Figure (5.23): H_2 Controller for Magnetic Levitation CE 152 model

Figure 5.23 shows that the set point is 0.5 and the H_2 controller can be implemented as stat space model. Figure 5.24 shows the output response of the magnetic levitation after using H_2 controller, the rising time is nearly 0.0235, settling time is nearly 0.156, overshoot is nearly 20%. Table 5.4 shows comparison between H_2 controller fuzzy controller with GA optimization. This table shows that the rising time of two controller is nearly equal, but the overshoot and settling time of fuzzy controller using GA optimization is less than the settling time and overshoot of H_2 controller. These results prove that the GA optimization with fuzzy control can work properly with nonlinear system and give better results.

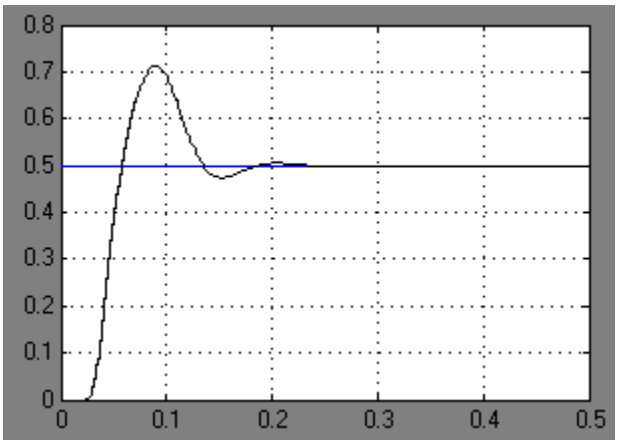


Figure (5.24): Step Response of the System with H_2 Controller

Table 5.4 Comparison between H_2 Controller and Fuzzy-GA controller

System response for step input	Fuzzy controller With GA optimization	H_2 controller
Overshoot	6%	No overshoot
Settling time	0.046 sec	0.15 sec
Rising time	0.023 sec	0.106 sec

CHAPTER 6 FPGA IMPLEMINTATION FOR FUZZY CONTROLLER

6.1 VHDL Fuzzy Controller Implementation

This chapter will discuses the implementation of the fuzzy controller without GA optimization using VHDL code that will use to program the Xilinx Spartan 3e FPGA. Figure 6.1 shows the block diagram of our controllers using FPGA. It consist of three main parts, ADC, FPGA, DAC. ADC and DAC that will be used in this thesis have 8 bits resolution. The integrated circuits (ICs) that will be used is PIC 16f877 as ADC, and DAC 0800. These two ICs have eight bits parallel interface with FPGA.

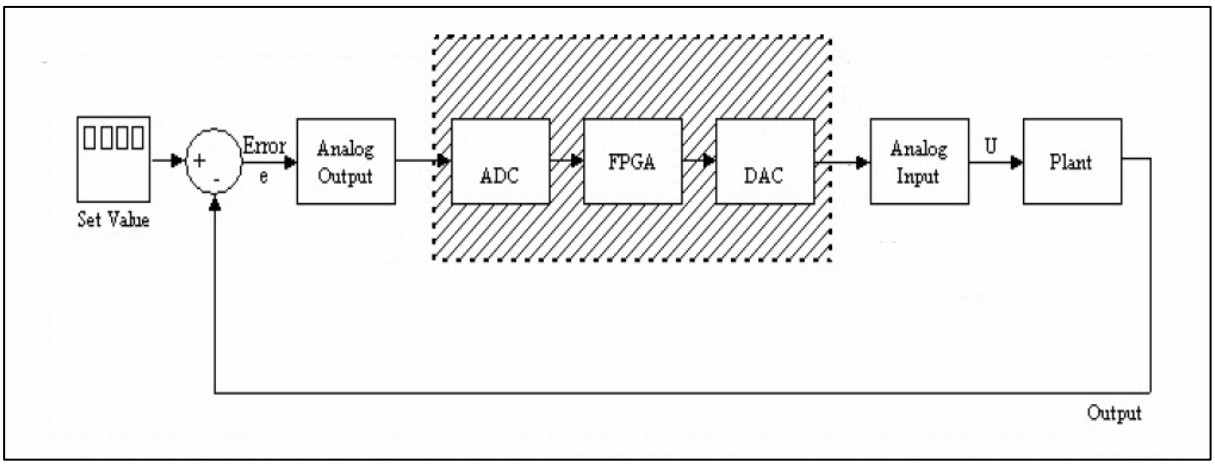


Figure (6.1): The Block Diagram of the FPGA and its Interface with the System

The controller consists of three parts. Summation part has two inputs set point and feedback, and generates the error and change of error that are inputs to fuzzy controller part. Fuzzy controller is generated using Xfuzzy CAD tool and has two inputs (error and change of error) and one output (change of voltage). PI part has one inputs (change of voltage) and one output (effective voltage). Xilinx company had developed Matlab simulink tools (Xilinx System Generator) that work with different kinds of Xilinx FPGA, this thesis uses this tool to test the controller with the magnetic levitation CE152 model. Figure 6.2, 6.3, 6.4, 6.5 show the fuzzy controller design using Xfuzzy tool

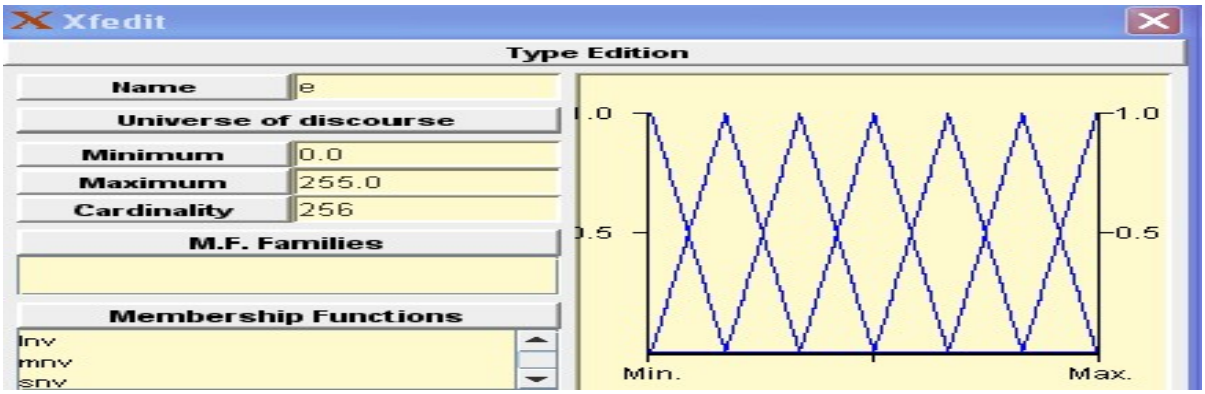


Figure (6.2): Error Input Using Xfuzzy Tool

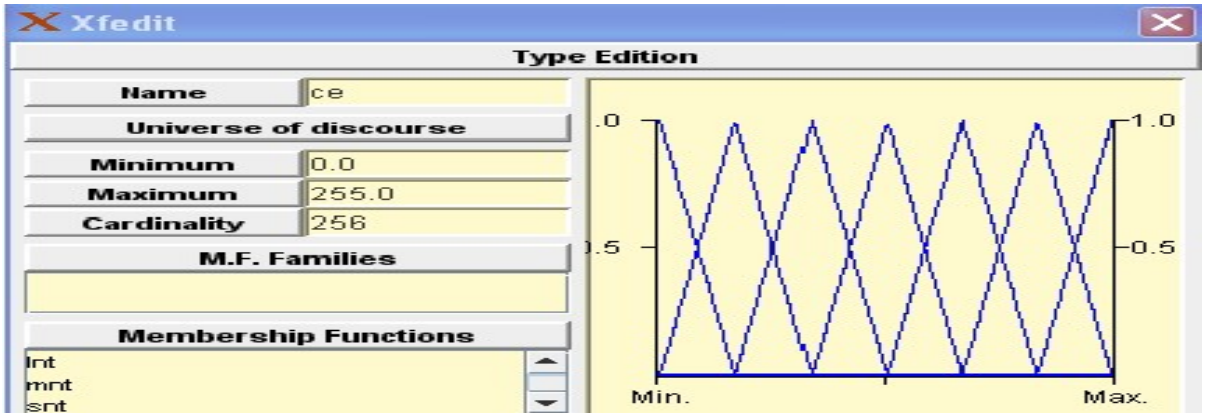


Figure (6.3): Change of Error Input Using Xfuzzy Tool

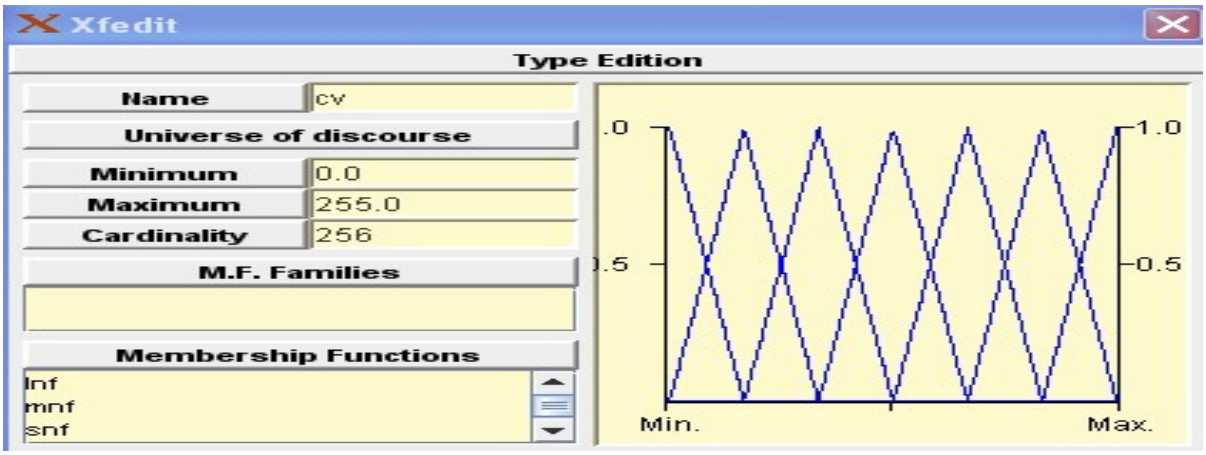


Figure (6.4): Change of Voltage Output Using Xfuzzy Tool

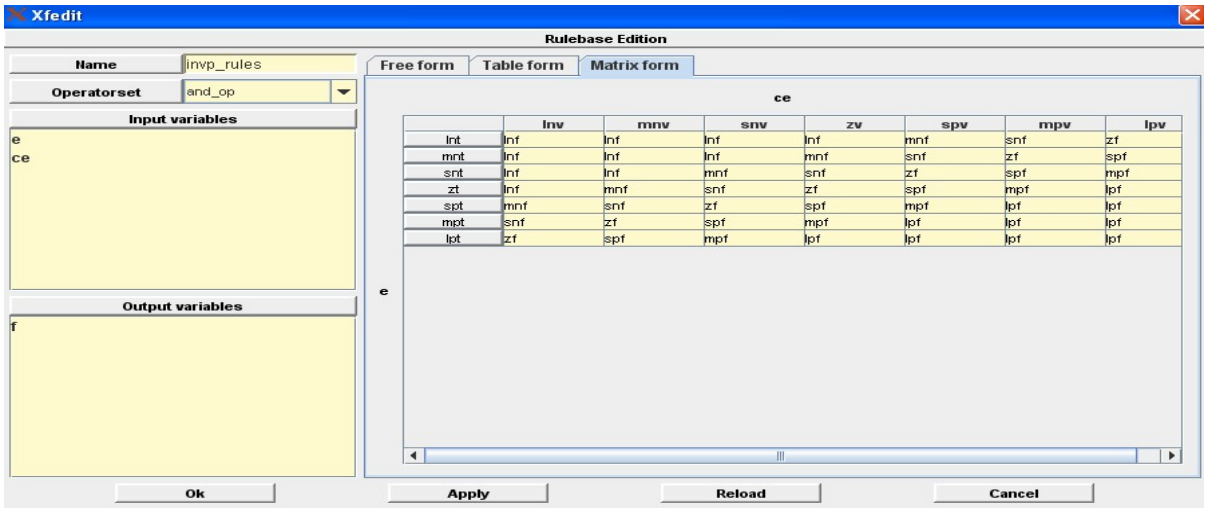


Figure (6.5): Fuzzy Rules Xfuzzy Tool

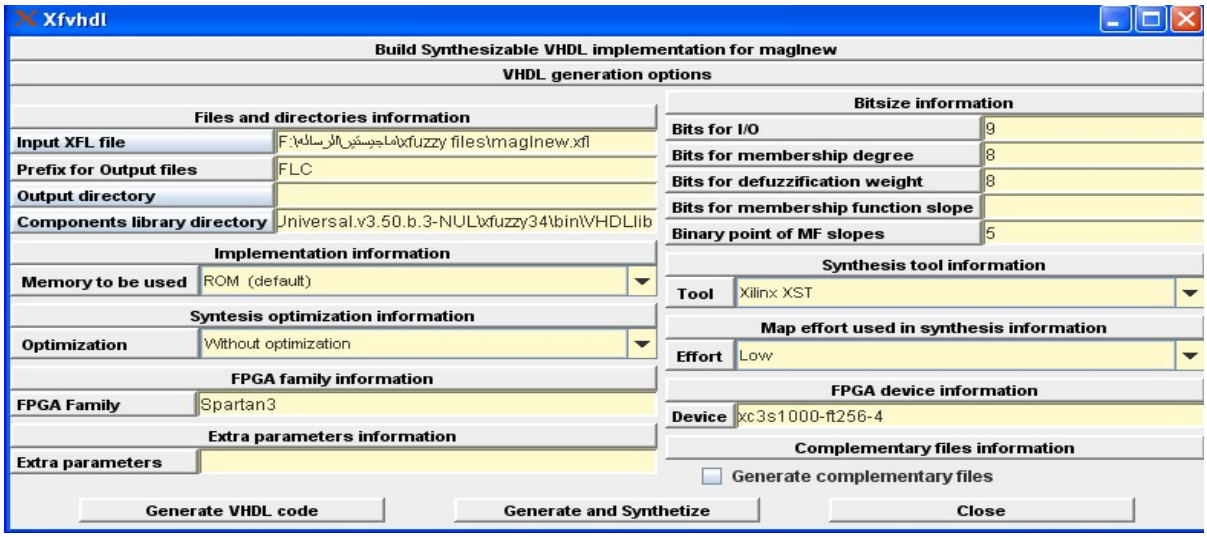


Figure (6.6): VHDL Code Generation Using Xfuzzy Tool

Figure 6.6 shows that the number of bits of input and output is 9 bits, because the set point will be from 0 to 255 and the error is in interval [255 -255]. After generating the VHDL code of fuzzy controller this code is used with "sum.vhd" and "pi.vhd" (appendix B) files to complete the controller. These files will be used with ISE 10.1 software to generate bit file that is uploaded to FPGA. Xfuzzy tool will generate System Generator blocks of fuzzy controller that is used with Matlab simulink. "Black Box" block that is in the System Generator Matlab toolbox will be used to insert the "sum.vhd" and "pi.vhd" into Matlab simulink as shown in Figure 6.7 and 6.8 respectively.

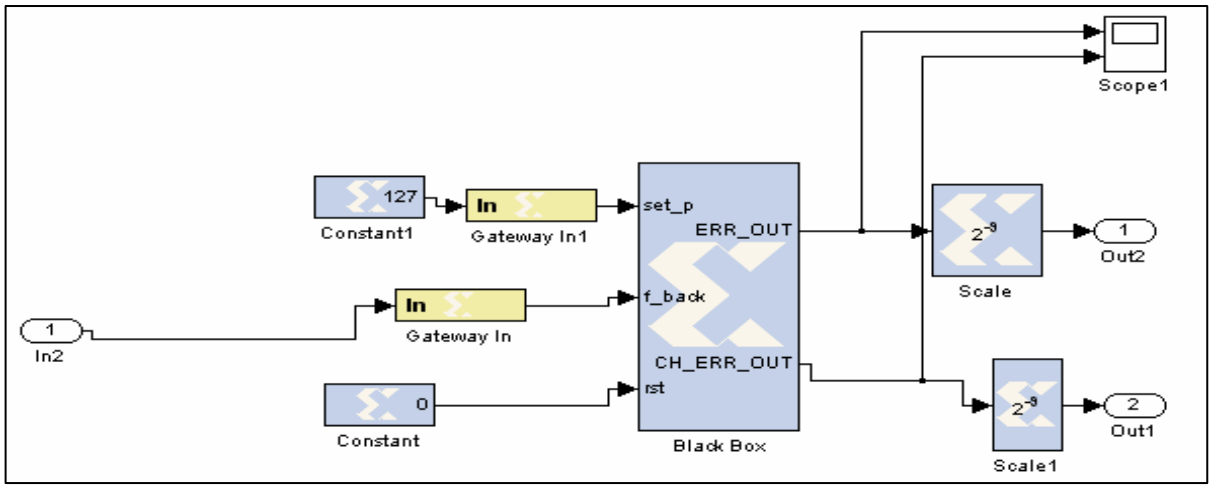


Figure (6.7): Block Diagram of Summation Subsystem

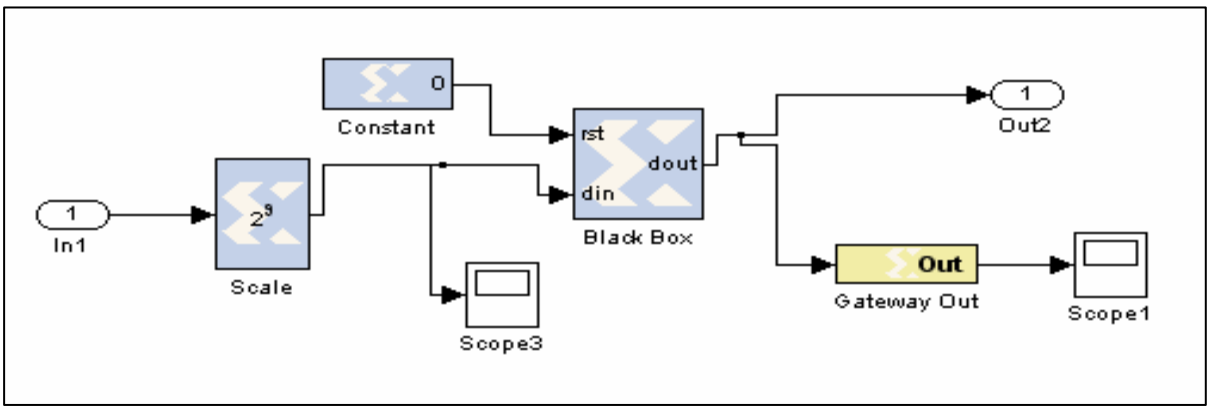


Figure (6.8): Block Diagram of PI Subsystem

Figure 6.9 shows the blocks of fuzzy controller that is generated using Xfuzzy tool. figure 6.10 shows the final system after connecting all sub systems. After building the simulink model of fuzzy controller using System Generator toolbox, the system runs at three different set points, step, sine wave and square wave to test the VHDL code if it work properly or not. Figure 6.11 shows the output response of the system at set point equal 127. (0.5 at normalize mode). Figure 6.12 shows the error and change of error of the ball position, the two values are shifted by 255, because the VHDL code of fuzzy controller works only in the interval [0 510], Where 0 means that the -255 and 510 means 255, so 0 error in matlab equal 255 in vhdl code. We need to set the simulation time in System Generator block to be 0.00000002 sec to work as real FPGA because the FPGA clock that is used in vhdl code is 50 mhz

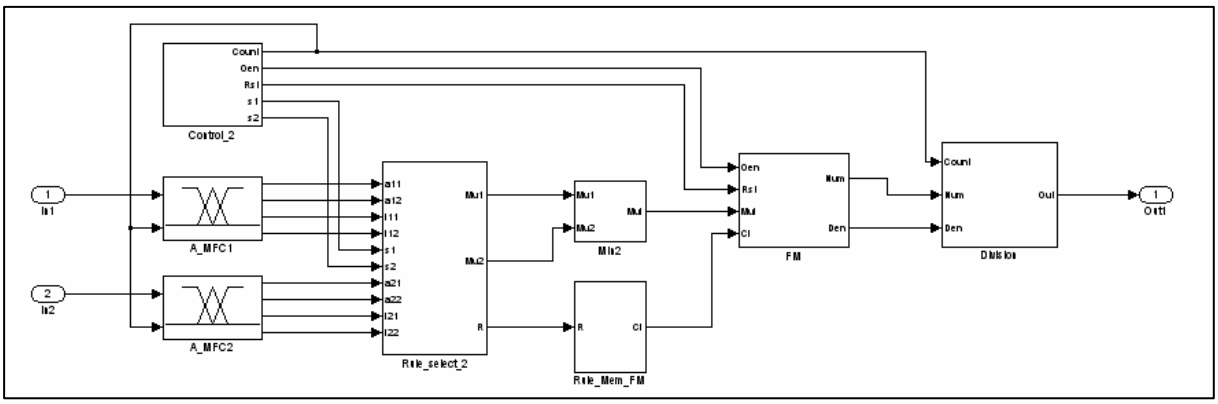


Figure (6.9): Block Diagram of Fuzzy Controller Subsystem

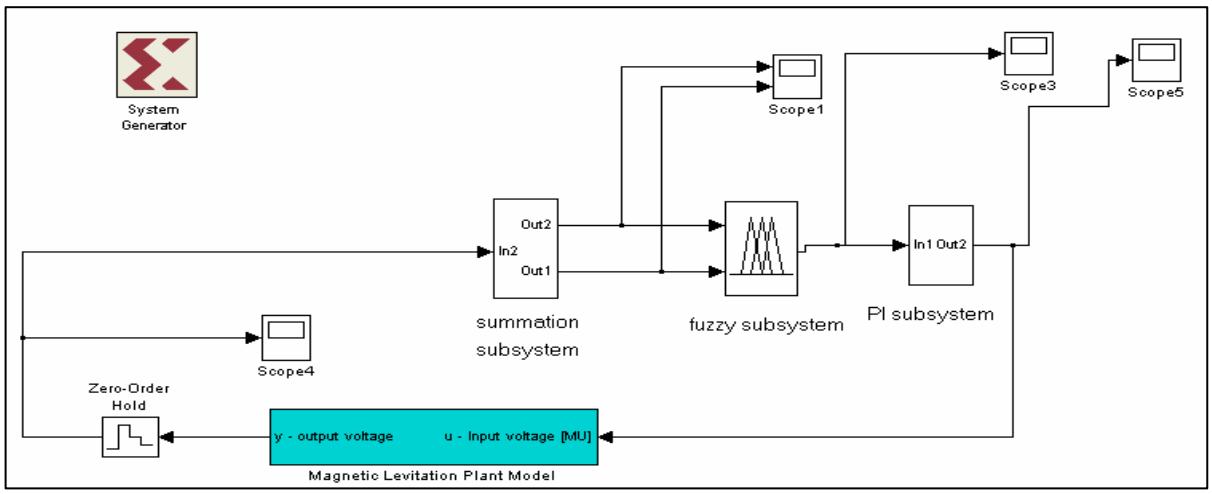


Figure (6.10): Block Diagram of Fuzzy Controller Using VHDL Code

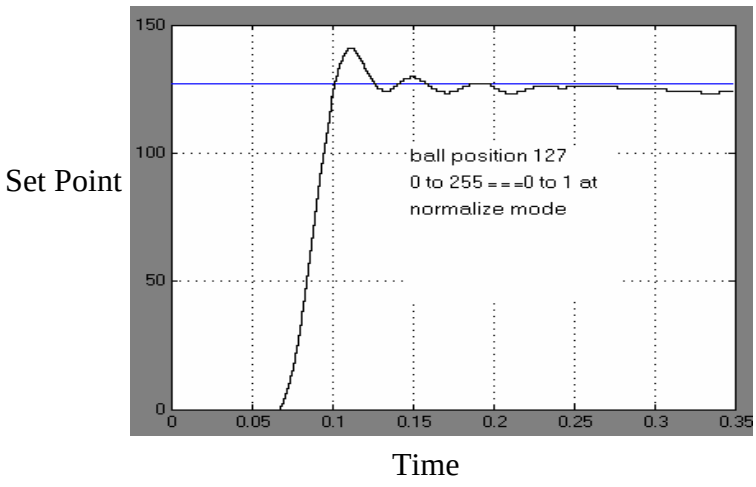


Figure (6.11): Step Response of the System Using VHDL Code

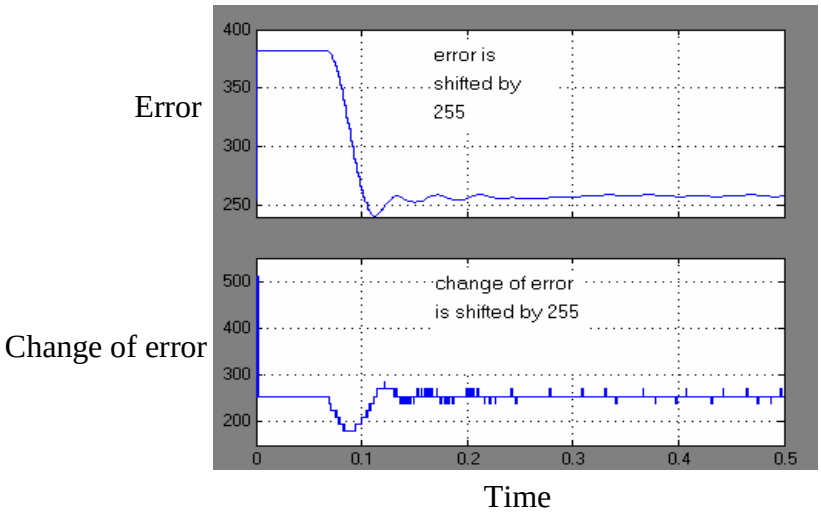


Figure (6.12): Error and change of error using VHDL code

Figure 6.13 and 6.14 show the output response when the set point is sine wave and square wave respectively. But there are steady state error in all set points, there are to reasons for these error. First the approximation in vhdl code for example, 0 error and 0 change of error and 0 change of voltage in fuzzy memberships equal 127.5 but FPGA works only with integer number, so that we set 0 value of all memberships equal 128. Second CE 152 model work with 14 bits DAQ and the difference between all steps equal 0.0000061.3 $=1/2^{14}$, but in this thesis the DAC and ADC is 8 bits and the difference between steps $0.0039=1/2^{8}$. The large difference between steps case low accuracy.

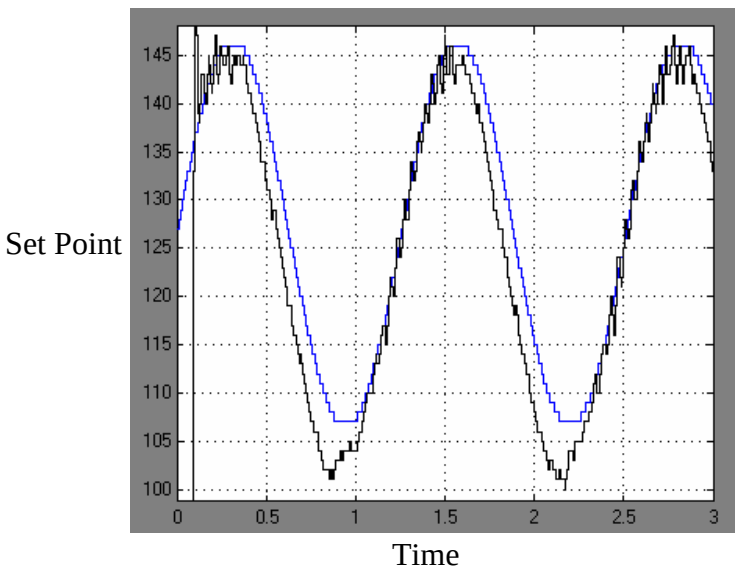


Figure (6.13): Sine Wave Output Response Using VHDL Code

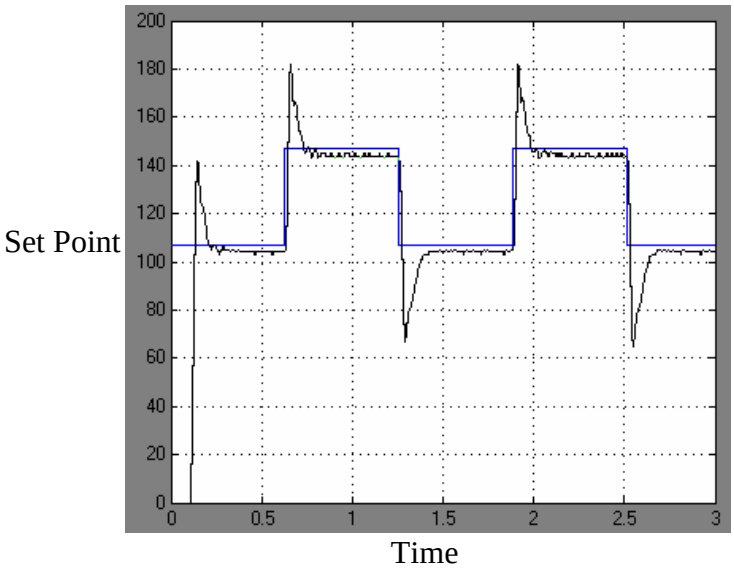


Figure (6.14): Square Wave Output Response Using VHDL Code

Previous Figures show that the VHDL code work properly with the CE152 model and can work with different set points shape. After testing the VHDL code of fuzzy controller using Matlab. There is another software that is used to program the FPGA. ISE 10.1 software is developed by Xilinx Company to work with most of FPGAs that are designed by this company. Figure 6.15 and 6.16 show the main windows of the ISE 10.1 and the schematic diagram of the all fuzzy controller blocks respectively.

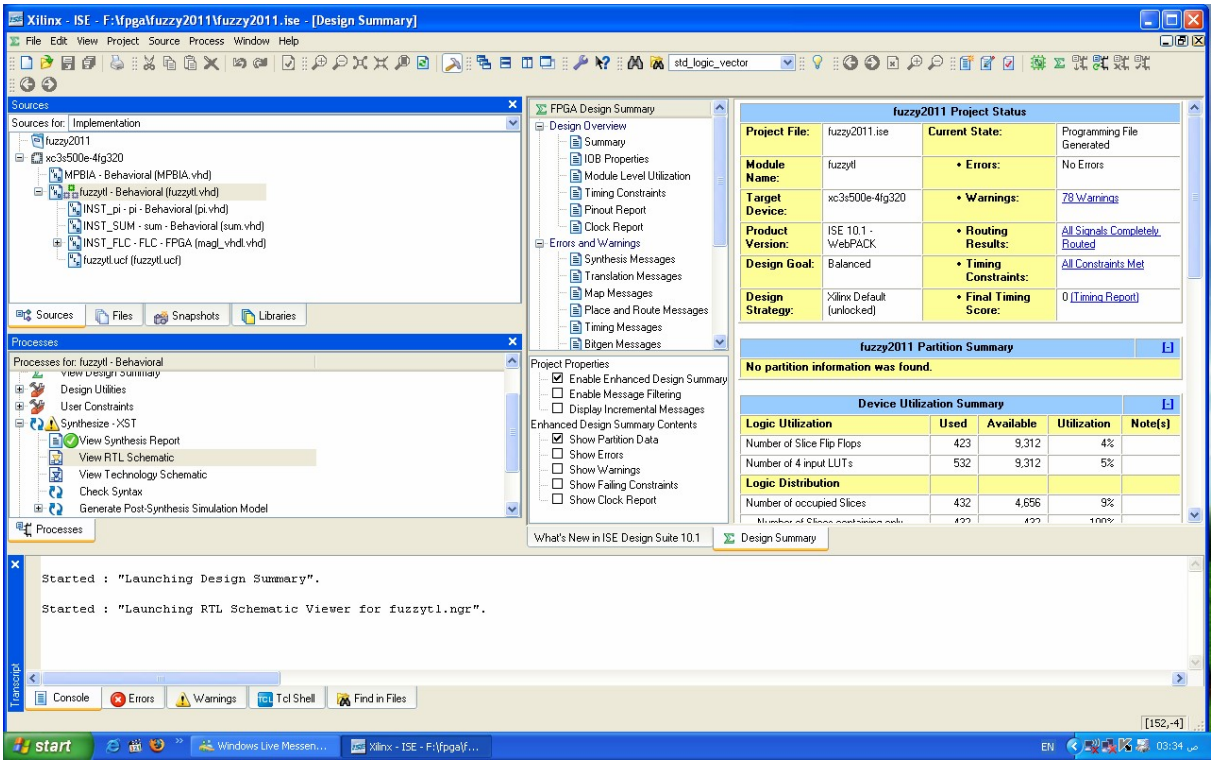


Figure (6.15): ISE 10.1 Software Main Window

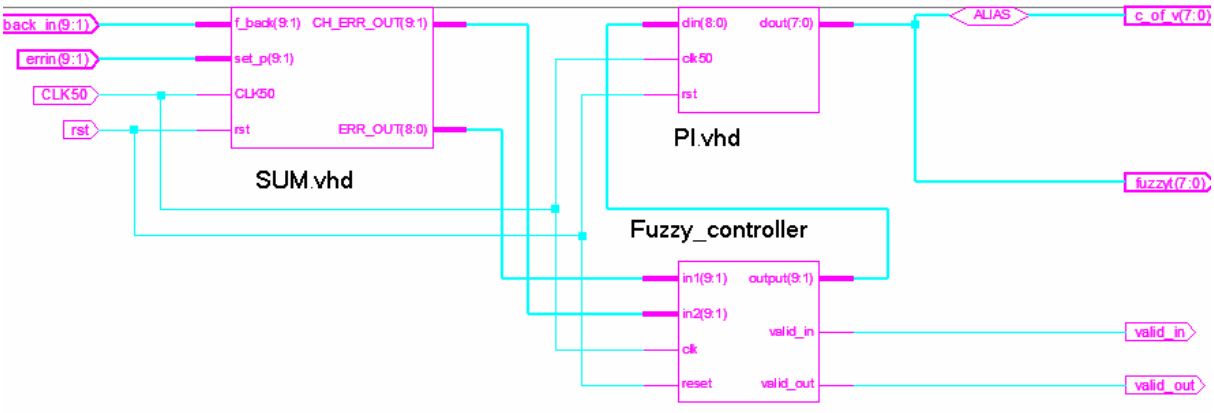


Figure (6.16): Schematic Diagram of Fuzzy Controller

CHAPTER 7 CONCLUSION

7.1 Conclusion

In recent years, tremendous progress has been made in the techniques of control field. The control field entered almost all areas such as medical applications, military and civilian. Fuzzy control is one of those techniques. This technique in recent years has become involved in many applications and integrated with other technologies to improve their performance. GA is one of this technologies, that can be used to optimized the fuzzy controller. With development of fuzzy control there became a need to develop hardware to deal with it. FPGA is one of these hardware, that is programmed using VHDL language, because it has high speed time process, and its program work in parallel mode. But the representation of fuzzy logic in FPGA is not easy, Therefore, there is a need to develop CAD tools to help the controller designer and safe the time of design the fuzzy controller code. Xfuzzy one of these tools, The most important features of Xfuzzy that can generate VHDL code of fuzzy logic.

- In this thesis the magnetic levitation CE152 Model is used as practical example of nonlinear systems, because it is one of the famous systems in control engineer. The mathematical equation of the model showed that the Magnetic Levitation is unstable system and controllable. There are many controllers used to control the magnetic levitation in this thesis fuzzy logic control was used. Because fuzzy controller work with the system as black box so the designer does not need to find the mathematical model of the system. The genetic algorithm is used to improve the performance of the fuzzy controller. The biggest problem face fuzzy controller is how to implement it in hardware controllers so this thesis used FPGA as embedded system to implement fuzzy controller. The fuzzy controller was designed with Matlab software and this controller was tested with the CE152 Model that is found in Matlab the fuzzy controller stabilized the Magnetic Levitation CE152 Model under different set points. The GA optimization method was used to optimize the membership function of the inputs and output of the fuzzy controller and also to optimize the gains p_g, pi_g, ce_g, err_g and out_g, The CE152 was tested with the new membership functions and new gains and the results was better than the results of old fuzzy controller under different set points. The results of fuzzy controller using GA was compared with H_2 controller that is one of the famous techniques in optimal control. GA

optimization gave better results than H_2 controller. The Xfuzzy tool was used to generate the VHDL code of fuzzy controller, this code was used with "sum.vhd" and "pi.vhd" VHDL codes to give the over all controller of Magnetic Levitation CE152 Model. These codes were tested using Matlab/Simulink under different set points and the results were good. After design the VHDL codes of fuzzy controller this code was programmed and tested using Spartan 3E FPGA and the CE152 Model was stable.

7.2 Future Works

1- Sugeno method can be used instead of Mamdani model and make comparison between two methods.
2- Using fuzzy supervised PID techniques.
3- In the future research the GA can be used in learning of rule base with Tuning of the data base to give better results.
4- Change the 8 bits of The ADC and DAC with 12 bit or higher to give more accuracy.
5- The researcher can try to change the CE 152 model with practical model such as small Magnetic Levitation train.

REFERENCES

[1] Phillips C.L, & R.D. Harbor, *Feeedback Control System*, 2nd, Prentice hall, Englewood Cliffs, 1988.

[2] Michael McKenna and Bogdan M. Wilamowski, "Implementing a Fuzzy System on a Field Programmable Gate Array," IEEE International Joint Conference on Neural Networks, Volume: 1, p.189-194, 2001.

[3] MJ. Patyra & J.L. Grantner, "Hardware Implementation of Digital Fuzzy Logic Controller,". Information Science – An International Journal (113): pp.19-54, 1999.

[4] T. Philip, M. Vuong, asad, Madni and b. jim Vuong, ” VHDL implementation for a fuzzy logic controller”, BEI technologies, inc. 13100 telfair avenue, sylmar, 2006.

[5] J.E. Bonilla, V.H. Grisales and M.A. Melgarejo, "Genetic tuned FPGA based PD fuzzy LUT controller," IEEE International Conference. Fuzzy Systems, Vol 3,pp :1084 – 1087, 2001.

[6] Chia-Feng Juang; Chun-Ming Lu, "Ant Colony Optimization Incorporated With Fuzzy Q- Learning for Reinforcement Fuzzy Control," IEEE Trans. Systems, Vol 39,pp.597-475, May 2009.

[7] John Yen & Langari Reza, *Fuzzy Logic Intelligence Control and Information*, Prentic-Hall, Englwood Cliffs,1999.

[8] Z. Kovacic & S. Bogdan, *Fuzzy Controller Design: Theory and Application*, Taylor & Francis Group: CRC Press, 2006.

[9] P. Basehore, "Fuzzy Logic Outperforms PID Control," PCIM, Vol 17, pp. 40-46, March, 1993.

[10] Robert Fulle´r, "Fuzzy Reasoning and Fuzzy Optimization," TUCS General Publications, No. 9, Turku Centre for Computer Science, Abo, 1998.

[11] M. Ibrahim, *Fuzzy Logic for Embedded Systems Applications*, Elsevier Science, MA, USA, 2004.

[12] C. W. Silva, *Intelligent Control-Fuzzy Logic Application*, CRC, Boca Raton, FL, 1995.

[13] Lotfi A.Zadah, "Fuzzy Logic," University of California, Berkeley,1989**.**

[14] Prasad, Ram., "Fuzzy Logic Control," class handout, New Mexico State University, Electrical & computer Engineering, 1996.

[15] Peri, Vamsi Mohan, "Fuzzy Logic Controller for an Autonomous Mobile Robot," Jawaharlal Nehru Technological University, India May, 2002.

[16] D.E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley, MA,1989.

[17] J.H. Holland, *Adaptation in Natural and Artificial systems*, University of Michigan Press, Ann Arbor, 1975.

[18] O. Cordon, F. Herrera, F. Hoffman and L. Magdalena, “Genetic Fuzzy System Evolutionary Tuning and Learning of Fuzzy Knowledge Bases," Worlds Scientific, Singapore, 2001.

[19] S.N. Sivanandam and S.N. Deepa, *Introduction to Genetic Algorithms*, Springer, New York, , 2008.

[20] A. Alibeiki and S.S. Fallahi, "Genetic Algorithm and Comparison with Usual Optimization Methods," World Applied Sciences journal 11 (6):752-754, 2010.

[21] Outi Raiha, Applying Genetic Algorithms in Software Architecture Design, University of Tampere, Department of Computer Sciences Computer Science ,M.Sc thesis, February, 2008.

[22] M. Mitchell, *An Introduction to Genetic Algorithms*, MIT Press, 3rd edition. Springer. New York, 1996.

[23] Hung-Cheng Chen† and Sheng-Hsiung Chang, "Genetic Algorithms Based Optimization Design of a PID Controller for an Active Magnetic Bearing," IJCSNS International Journal of Computer Science and Network Security, VOL.6 No.12, December 2006.

[24] Maxfield, Clive, "The Design Warrior's Guide to FPGAs," Newnes, 2004.

[25] Balasaheb S. Darade, Abhishek Singh Chauhan and Tarun A.Parmar, "Paper Presentation On Programming FPGA's Using Handel-C," Jawaharlal Nehru Engineering College Aurangabad.

[26] Xilinx Inc. "Using Digital Clock Managers (DCMs) in Spartan-3 FPGAs," XAPP462 (V1.1), Jan 2006.

[27] K. Mabasa, Mohamed Akil, Thierry Grandpierre, B.J. Van Wyk and M.A. Van Wyk "Automatic VHDL Code Generation for Fuzzy Logic Systems,", African Journal of Science and Technology, 2008.

[28] Gersnoviez and M. Brox, "Using Xfuzzy Environment for the Whole Design of Fuzzy Systems," Proc. IEEE International Conference on Fuzzy Systems, London, July 23-26, 2007.

[29] D. R. Lopez, S. Sanchez-Solano, and A. Barriga, "Xfuzzy: A Design Environment for Fuzzy Systems," Proc.7th IEEE International Conference on Fuzzy Systems, pp.1060-1065, Anchorage, May 1998.

[30] F.J.M. Velo, L. Baturone, S.S. Solano and A. Barriga, "Rapid design of fuzzy systems with Xfuzzy," Proceedings of the IEEE 12th International Conference on Fuzzy Systems, May 25-28, Sevilla, Spain, pp: 342-347.2003.

[31] CE 152 magnetic levitation model-education manual. *Humusoft* s.r.o 2002.

[32] Math work, *Help Manual, Matlab R2008a* .

[33] Tongwen Chen and Bruce Francis, *Optimal Sampled-Data Control System*, Berlin: Springer-Verlag, 1995.

APPENDIX A GA MATLAB PROGRAMS

1- GA FUZZY MATLAB CODE MAIN PROGRAM

```
%-------------------------------------------------------------------------------------------------------
clc
clear
global rin yout timef
Ts=0.001;
magl1=readfis('magl');
open_system('fuzzy_mag');
MAXGEN = 100; % maximum Number of generations
NVAR = 13; % Generation gap, how many new individuals are created
GGAP = .5; % Generation gap, how many new individuals are created
PRECI = 10; % Binary representation precision
NIND = 50; % No. of individuals per subpopulations
% First, a field descriptor is set up
FieldD = [rep([PRECI],[1, NVAR]); rep([-0.1;0.1],[1, NVAR]);...
rep([1; 0; 1 ;1], [1, NVAR])];
%-------------------------------------------------------------
% The population is then initialized
%-------------------------------------------------------------
FieldD(2,1)=0.9;FieldD(2,2)=14.5;FieldD(2,3)=14.5;FieldD(2,12)=2.4;FieldD(2
,13)=0;
FieldD(3,1)=1.1;FieldD(3,2)=15.5;FieldD(3,3)=15.5;FieldD(3,12)=2.8;FieldD(3
,13)=0.2;
FieldD;
BsJ=0;
E = crtbp(NIND, NVAR*PRECI);

E(1,:)=[1,0,0,0,0,0,0,0,1,1,0,1,1,0,0,1,0,1,0,0,1,1,0,0,0,1,1,1,1,1,1,1,1,0
,0,1,1,0,1,0,0,0,0,1,1,0,0,1,0,0,1,0,1,1,0,0,1,0,0,0,1,1,1,1,1,1,0,0,0,1,1,
1,0,1,0,0,0,1,0,1,0,1,0,1,1,1,0,0,1,1,1,0,0,1,0,0,0,1,0,0,1,0,0,0,0,0,0,0,0
,1,1,1,0,0,1,0,1,0,1,0,1,1,0,1,1,1,0,1,0,1;];

E(2,:)=[1,0,0,0,0,0,0,0,1,1,0,1,1,0,1,0,1,1,0,1,1,0,0,0,0,0,0,0,0,0,1,1,1,0
,0,1,1,0,1,0,0,0,0,1,0,1,0,1,1,1,0,1,1,1,0,0,0,0,1,0,0,1,0,1,0,0,1,0,0,1,1,
0,1,1,0,0,0,0,1,1,1,0,0,1,1,1,0,0,0,1,1,1,0,1,1,0,0,1,1,0,1,1,1,1,0,1,0,0,1
,0,1,1,0,1,1,0,1,0,1,0,0,1,1,1,0,0,1,0,1,1;];
```

```
E(3,:)=[1,0,0,0,1,1,1,1,0,0,0,0,0,0,0,0,0,1,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,1,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,0,1,
0,1,1,0,0,1,1,1,0,0,0,0,0,0,1,0,0,0,0,1,1,1,1,0,1,0,1,0,1,0,1,0,0,1,0,0,1,1
,0,1,0,1,1,1,1,0,0,1,0,1,0,1,0,1,1,1,1,0,0;];
E(4,:)=[1,0,0,0,0,0,1,1,0,1,0,0,1,1,0,0,0,0,0,1,1,0,0,0,1,0,1,1,0,1,1,1,0,0,0
,0,0,0,0,1,0,0,1,0,1,0,0,1,1,1,0,0,0,0,1,1,0,0,1,0,1,1,0,0,0,0,1,1,0,1,1,0,
1,1,1,1,0,1,0,0,0,0,0,1,0,1,0,0,1,0,0,0,0,1,1,0,0,0,1,0,1,1,1,0,0,1,0,1,0,0
,1,0,0,1,1,0,0,0,0,1,1,0,1,1,0,1,1,0,0,0,0;];

for  kg=1:1:MAXGEN
     time(kg)=kg;
       Kfuzzy=bs2rv(E,FieldD);

    for s=1:1:NIND

%*************** Step 1 : Evaluate BestJ ****************************

        Kfuzzyi=Kfuzzy(s,:);

        [Kfuzzyi,BsJ]=mychap5_2f3(Kfuzzyi,BsJ);

        BsJi(s)=BsJ;
    end
    [OderJi,IndexJi]=sort(BsJi);
    BestJ(kg)=OderJi(1);
    BJ=BestJ(kg);
    Ji=BsJi+1e-10;
    fi=1./Ji;
    [Oderfi,Indexfi]=sort(fi);
    Bestfi=Oderfi(NIND);
    BestS=E(Indexfi(NIND),:);
    kg
    BJ
    BestS;

%********** Step 2 : Select and Reproduct Operation******************

    fi_sum=sum(fi);
    fi_Size=(Oderfi/fi_sum)*NIND;
```

```
    fi_S=floor(fi_Size);        %Selecting Bigger fi value
     kk=1;
    for i=1:1:NIND
        for j=1:1:fi_S(i)        %Select and Reproduce
            TempE(kk,:)=E(Indexfi(i),:);
            kk=kk+1;              %kk is used to reproduce
        end
    end

    %************ Step 3 : Crossover Operation ************
    pc=0.99;
    n=ceil(100*rand);
    for i=1:2:(NIND-1)
        temp=rand;
        %if pc>temp                  %Crossover Condition
            for j=n:1:100
                TempE(i,j)=E(i+1,j);
                TempE(i+1,j)=E(i,j);
            end
        %end
    end
    TempE(NIND,:)=BestS;
    E=TempE;

%*********** Step 4: Mutation Operation ****************************

    pm=0.001-[1:1:NIND]*(0.001)/NIND; %Bigger fi, smaller pm
        for i=1:1:NIND
        for j=1:1:3*PRECI
            temp=rand;
            if pm(i)>temp               %Mutation Condition
                if TempE(i,j)==0
                    TempE(i,j)=1;
                else
                    TempE(i,j)=0;
                end
            end
        end
    end
    TempE(NIND,:)=BestS;
    E=TempE;
```

```
%*************************************************************************
end
Bestfi
BestS
Kfuzzyi
Best_J=BestJ(MAXGEN)
figure(1);
plot(time,BestJ);
xlabel('Times');ylabel('Best J');
```

2- GA FUZZY MATLAB CODE CALL SIMULINK PROGRAM

```
%-----------------------------------------------------------------------
function [Kfuzzyi,BsJ]=fuzzy_gaf(Kfuzzyi,BsJ)
global rin yout timef
%-----------------------------------------------------------------------
a=newfis('magl');

a=addvar(a,'input','e',[-1,1]);              %Parameter e
a=addmf(a,'input',1,'NB','zmf',[-1,-0.6668-Kfuzzyi(10)]);
a=addmf(a,'input',1,'NM','trimf',[-1-Kfuzzyi(9),-0.6668-Kfuzzyi(8), -
0.3332-Kfuzzyi(7)]);
a=addmf(a,'input',1,'NS','trimf',[-0.6668-Kfuzzyi(6),-0.3332-
Kfuzzyi(5),0]);
a=addmf(a,'input',1,'Z','trimf',[-0.3332-Kfuzzyi(4),0,0.3332+Kfuzzyi(4)]);
a=addmf(a,'input',1,'PS','trimf',[0,0.3332+Kfuzzyi(5),0.6666+Kfuzzyi(6)]);
a=addmf(a,'input',1,'PM','trimf',[0.3332+Kfuzzyi(7),0.6666+Kfuzzyi(8),1+Kfu
zzyi(9)]);
a=addmf(a,'input',1,'PB','smf',[0.6666+Kfuzzyi(10),1]);

a=addvar(a,'input','ec',[-1,1]);            %Parameter ec
a=addmf(a,'input',2,'NB','zmf',[-1,-0.6668-Kfuzzyi(10)]);
a=addmf(a,'input',2,'NM','trimf',[-1-Kfuzzyi(9),-0.6668-Kfuzzyi(8), -
0.3332-Kfuzzyi(7)]);
a=addmf(a,'input',2,'NS','trimf',[-0.6668-Kfuzzyi(6),-0.3332-
Kfuzzyi(5),0]);
a=addmf(a,'input',2,'Z','trimf',[-0.3332-Kfuzzyi(4),0,0.3332+Kfuzzyi(4)]);
a=addmf(a,'input',2,'PS','trimf',[0,0.3332+Kfuzzyi(5),0.6666+Kfuzzyi(6)]);
a=addmf(a,'input',2,'PM','trimf',[0.3332+Kfuzzyi(7),0.6666+Kfuzzyi(8),1+Kfu
zzyi(9)]);
a=addmf(a,'input',2,'PB','smf',[0.6666+Kfuzzyi(10),1]);

a=addvar(a,'output','u',[-1,1]);            %Parameter u
a=addmf(a,'output',1,'NB','zmf',[-1,-0.6668-Kfuzzyi(10)]);
a=addmf(a,'output',1,'NM','trimf',[-1-Kfuzzyi(9),-0.6668-Kfuzzyi(8), -
0.3332-Kfuzzyi(7)]);
a=addmf(a,'output',1,'NS','trimf',[-0.6668-Kfuzzyi(6),-0.3332-
Kfuzzyi(5),0]);
a=addmf(a,'output',1,'Z','trimf',[-0.3332-Kfuzzyi(4),0,0.3332+Kfuzzyi(4)]);
a=addmf(a,'output',1,'PS','trimf',[0,0.3332+Kfuzzyi(5),0.6666+Kfuzzyi(6)]);
```

```
a=addmf(a,'output',1,'PM','trimf',[0.3332+Kfuzzyi(7),0.6666+Kfuzzyi(8),1+Kf
uzzyi(9)]);
a=addmf(a,'output',1,'PB','smf',[0.6666+Kfuzzyi(10),1]);

%--------------------------------------------------------------------------
 rulelist=[1 1 1 1 1;          %Edit rule base
    1 2 1 1 1;
    1 3 1 1 1;
    1 4 1 1 1;
    1 5 2 1 1;
    1 6 3 1 1;
    1 7 4 1 1;

    2 1 1 1 1;
    2 2 1 1 1;
    2 3 1 1 1;
    2 4 2 1 1;
    2 5 3 1 1;
    2 6 4 1 1;
    2 7 5 1 1;

    3 1 1 1 1;
    3 2 1 1 1;
    3 3 2 1 1;
    3 4 3 1 1;
    3 5 4 1 1;
    3 6 5 1 1;
    3 7 6 1 1;

    4 1 1 1 1;
    4 2 2 1 1;
    4 3 3 1 1;
    4 4 4 1 1;
    4 5 5 1 1;
    4 6 6 1 1;
    4 7 7 1 1;

    5 1 2 1 1;
    5 2 3 1 1;
    5 3 4 1 1;
```

```
    5 4 5 1 1;
    5 5 6 1 1;
    5 6 7 1 1;
    5 7 7 1 1;

    6 1 3 1 1;
    6 2 4 1 1;
    6 3 5 1 1;
    6 4 6 1 1;
    6 5 7 1 1;
    6 6 7 1 1;
    6 7 7 1 1;

    7 1 4 1 1;
    7 2 5 1 1;
    7 3 6 1 1;
    7 4 7 1 1;
    7 5 7 1 1;
    7 6 7 1 1;
    7 7 7 1 1];
a=addrule(a,rulelist);
a=setfis(a,'DefuzzMethod','centroid');
writefis(a,'magl');
magl1=readfis('magl');
ke=num2str(Kfuzzyi(1));
kce=num2str(Kfuzzyi(2));
ku=num2str(Kfuzzyi(3));
ki=num2str(Kfuzzyi(12));
kp=num2str(Kfuzzyi(13));
set_param('fuzzy_mag/fuzzy_sub/err_g','Gain',ke);
set_param('fuzzy_mag/fuzzy_sub/ce_g','Gain',kce);
set_param('fuzzy_mag/fuzzy_sub/out_g','Gain',ku);
set_param('fuzzy_mag/pi_sub/pi_g','Gain',ki);
set_param('fuzzy_mag/pi_sub/p_g','Gain',kp);


[t,x,y]=sim('fuzzy_mag');
clear t;
clear x;
clear y;
```

```
BsJ=0;
ts=0.001;
  for k=1:1:201
      timef(k)=k*ts;
      Ji(k)=(e_abs.time(k)*e_abs.signals.values(k));

     BsJ=BsJ+Ji(k);
  end


end
```

APPENDIX **B VHDL CODES**

---------- "sum.vhd"----------

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_SIGNED.ALL;


entity sum is
   Port ( set_p : in  STD_LOGIC_VECTOR (9 downto 1;(
          f_back: in  STD_LOGIC_VECTOR (9 downto 1;(
          CLK50 : in  STD_LOGIC;
                      rst: in std_logic;
          ERR_OUT : out  STD_LOGIC_VECTOR (8 downto 0):="011111111;"
                      CH_ERR_OUT    :    out        STD_LOGIC_VECTOR    (9    downto
1):="011111111;("
end sum;

architecture Behavioral of sum is

signal set_p_b: std_logic_vector(8 downto 0):="000100100;"
signal f_back_b: std_logic_vector(8 downto 0):="010000000;"
signal e_b: std_logic_vector(9 downto 0):="0000000000;"
signal e_b1: std_logic_vector(9 downto 0):="0000000000;"
signal ch_err_b: std_logic_vector(14 downto 0):="000000000000000;"
signal sh_err: std_logic_vector(9 downto 0):="0000000000;"
signal CH_ERR_OUTb: std_logic_vector(14 downto 0):="000000000000000;"



begin
process (clk50,set_p,f_back,set_p_b,f_back_b(
variable count: integer :=0;
begin
  if clk50' event and clk50='1' then
    count:=count+1;
          if count=50000 then
          count:=0;
--   e_b<=('0'&set_p_b)-('0'&f_back;(
          ERR_out<=sh_err(8 downto 0;(
          ch_err_out<=ch_err_outb(8 downto 0;(
    e_b1<=e_b;
          end if;
          end if;
```

```
            end process;

process (clk50,set_p,f_back,set_p_b,f_back_b(
begin
  if clk50' event and clk50='1' then
 e_b<=('0'&set_p_b)-('0'&f_back;(
 sh_err<=e_b+"0011111111;"

-----------------------------------------------
  ch_err_b<=((e_b-e_b1)*"001111")+255;
    if ch_err_b>=510 then
      CH_ERR_OUTb<="000000111111110;"
    elsif ch_err_b<0 then
      CH_ERR_OUTb<="000000000000000;"
    else
      CH_ERR_OUTb<=ch_err_b;--(8 downto 0;(
    end if;
--     e_b1<=e_b;
end if;
end process;
end Behavioral;
```

----------"PI.vhd"----------

```
----------------------------------------------------------------------------------------------------------

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_SIGNED.ALL;

---- Uncomment the following library declaration if instantiating
---- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity pi is
    Port ( clk50 : in  STD_LOGIC;
                rst: in  STD_LOGIC;
           dout : out  STD_LOGIC_VECTOR (7 downto 0):=(others=>'0');
           din : in  STD_LOGIC_VECTOR (8 downto 0):=(others=>'0'));
end pi;

architecture Behavioral of pi is
signal eb: std_logic_vector(9 downto 0):=(others=>'0');
signal u1: std_logic_vector(31 downto 0):=(others=>'0');
signal u: std_logic_vector(31 downto 0):=(others=>'0');

signal acu_in: std_logic_vector(14 downto 0):=(others=>'0');
signal ebgain: std_logic_vector(14 downto 0):=(others=>'0');
signal p_out: std_logic_vector(22 downto 0):=(others=>'0');
signal c_out: std_logic_vector(31 downto 0):=(others=>'0');
signal c_outb: std_logic_vector(31 downto 0):=(others=>'0');
signal doutb: std_logic_vector(7 downto 0):=(others=>'0');

begin
process (clk50,din,acu_in,ebgain,p_out,c_out,c_outb,u)
begin
if clk50' event and clk50='1' then
eb<=('0'&din)-"0011111111";
acu_in<=eb*"000111";
ebgain<=eb*"000111";
u<=(eb*"01111")+u1;--acu_in+u1;
p_out<=ebgain*"01100100";
c_outb<=u+p_out;
  if c_outb(31)='1' then
     c_out<=(others=>'0');
  else
     c_out<="000000000"&c_outb(31 downto 9);
  end if;
     if c_out>=255 then
```

```
      doutb<="11111111";
    else
     doutb<=c_out(7 downto 0);
    end if;

end if;
end process;


process (clk50,din,acu_in,ebgain,p_out,c_out,c_outb,u)
variable count: integer :=0;
begin
if clk50' event and clk50='1' then
count:=count+1;
if count>50000 then
count:=0;
dout<=not(doutb);
u1<=u;
if u1(31)='1' and u1(30)='0' then
--u1<="00000000000000000000000011111111";
u1<="10111111111111111111111111111111";
elsif u1(31)='0' and u1(30)='1' then
u1<="01000000000000000000000000000000";
--u1<="11111111111111111111111100000000";
end if;
end if;
end if;
end process;
end Behavioral;
```

APPENDIX C H_2 CONTROLLER MATLAB CODE

```
% All of the parameters are read in from MATLAB workspace variables.
% The plant model is [a,b,c,d].
%   -1      num1(s)          num2(s)      -1      num3(s)
% w1  (s) = -------;  w2(s) = -------;   w3  (s) = -------
%           den1(s)          den2(s)              den3(s)
% are the performance weighting function, input weighting function, and
% robustness weighting function respectively. The curves inside the
% weighting function blocks are the magnitudes of the bode plots
% of each weighting function.
%
% The designed H_2 controller is given by [ae,be,ce,de]. During the
% simulation system measurement noise is added.
%
% By changing the plant and the weighting function parameters, you
% can convert the example to solve a problem of your own.
%
% Re-Load Data
% Re-load data from file. This refreshes the data in the workspace.
%
% Re-Design
% After changing the workspace parameters, you should redesign the
% controller to fit your data.
%
% In this design, the following commands in the Robust Control
% Toolbox are used:
% augss     --- state space plant augmentation with weighting function
% obalreal  --- balance realization
% h2lqg     --- H_2 optimal controller design
%
% A MIMO control system can be designed using a similar structure.
%
format short
clear
clc
Ts=0.001
 %--------- Magnetic Levitation Parameters -----------

Dk = 12.7e-3;           % ball diameter [m]
mk = 0.0084;            % ball mass [kg]
g = 9.81;               % gravity accelleration constant [m.s^-2]
L = 19e-3 - Dk;         % distance of limits [m]
U_DAm = 5;              % maximum DA converter output voltage
Rc = 3.5;               % coil resistance [Ohm]
Lc = 30e-3;             % coil inductance [H]
Rs=0.25;                % current sensor resistance [Ohm]
Ks = 13.33;             % current sensor gain  [-]
K_am = 100;             % power amplifier gain [-]
I_am = 1.2;             % maximum power amplifier output current
Ta =0.003;% L/((Rc+Rs)+Rs*Ks*K_am);     % amplifier time constant [s]
k_i = K_am / ((Rc+Rs)+Rs*Ks*K_am);  % amplifier gain [A/V]
KFv=0.02;           % viscose friction
k_DA= 10;           % D/A converter gain
k_AD=0.2;           % A/D converter gain
k_x=797.4603;       % position sensor constant
k_f =0.606e-6;      %aggregated coil constant [N/V])
k_c=k_f/(k_i)^2;    %coil constant
x_0= 8.26e-3;       % ball position at equilibirum point (x00,v00,i00)
```

```
v00 = 0;            % ball velocity [m/s] at equilibirum point
(x00,v00,i00)
x00= L/2;           % coil limit bias [m]
u00 = - sqrt( mk * g / k_f ) * ( x00 - x_0 );   % model input voltage
[m/s^2]  at equilibirum point (x00,v00,i00)
i00 = u00 * k_i;    %coil current at equilibirum point (x00,v00,i00)

%-------------------------------------------------

fprintf('             Feedback Linearization of Magnetic Levitation
System\n\n')
fprintf('The desired over-shoot = 10%%    and settling Time = 0.5 sec\n')

fprintf('The state space equation of the system is:\n ')
% State Space Equation of Magnetic Plant
a = [0 1 0 ; -2*i00^2*k_c/(x00-x_0)^3/mk  -KFv/mk  2*i00*k_c/(x00-x_0)^2/mk
;  0 0 -1/Ta ]
b = k_DA*[ 0  0  k_i/Ta  ]'
c = (k_x*k_AD)*[ 1  0  0  ]
d = [ 0 ]


% performance weighting function W1

num1=[1 300];
den1=[0.5 0.01];

% input weighting function W2

num2=[0.01];
den2=[1];

% robust weighting function W3

num3=[0.1];
den3=[1] ;

% Construct the wieghting functions

sys = ss(a,b,c,d)
[aw1,bw1,cw1,dw1] = tf2ss(num1,den1); sysw1 = ss(aw1,bw1,cw1,dw1);
[aw2,bw2,cw2,dw2] = tf2ss(num2,den2); sysw2 = ss(aw2,bw2,cw2,dw2);
[aw3,bw3,cw3,dw3] = tf2ss(num3,den3); sysw3 = ss(aw3,bw3,cw3,dw3);

s=zpk('s');

sys_=augtf(sys,sysw1,sysw2,sysw3)   % augss     --- state space plant
augmentation with weighting function

[A,B1,B2,C1,C2,D11,D12,D21,D22]=branch(sys_);
[K,CL,GAM]=h2syn(sys_);
[ss_cp,ss_cl]=h2lqg(sys_);   % h2lqg     --- H_2 optimal controller design
[A,B,C,D]=ssdata(K)
[ae,be,ce,de]=ssdata(ss_cp)
p=[A B1 B2;C1 D11 D12;C2 D21 D22] %
disp('H2 Controller is now ready')
```