FPGA-Based Implementation of Genetically Tuned Fuzzy Logic Controller (GA-FLC)

Asst. Lect. Emad A. Hussein
Electrical Engineering Department
Al-Mustansiriya University
Iraq-Baghdad

Asst. Lect. Ammar G. Samir
Electrical Engineering Department
Al-Mustansiriya University
Iraq-Baghdad

Abstract

Fuzzy Logic controller (FLC) contains three operations; the fuzzification of the inputs, the knowledge base (data base and rule base), and the defuzzification of the output. In this paper our fuzzy controller contains two inputs and one output each have five membership functions. This fuzzy controller will pass through two operations; the first is to tune the input/output scaling factor (SF) and the second operation is to tune the membership function parameters. This tuning is done by the use of Genetic Algorithm (GA). The tuned fuzzy controller then will be reduced to a look-up table by taking the whole fuzzy probabilities. The output for the tuned fuzzy controller will be obtained using center of gravity method. To apply this tuned circuit we must translate the resulted table to digital binary values using a special encoder then to a set of boolean functions. Finally FPGA technology will be used to describe the resulted boolean functions by the use of the FPGA programming language (VHDL) hardware description language. The output will then pass through the decoder to get the suitable control action.

The most important advantage of our method is to describe the FLC using digital numbering system applied to FPGA technology to achieve high processing speed for the fuzzified output per second and also the speed of this controller is independent of the number of fuzzy rules.

Key words: ***Fuzzy Logic Controller (FLC), Genetic Algorithm (GA), FPGA and VHDL.***

الخلاصة

تَحتوي مسيطرات المنطقِ الضبابيِ (FLC) على ثلاث عملياتَ؛ fuzzification للمدخلاتِ، قاعدة المعرفةَ (قاعدة بيانات وقاعدة القوانين)، و defuzzification للنتائج. في هذا البحث جهازِ سيطرتنا الضبابيِ يَحتوي على مدخلين ومخرج واحد كُلّ لَهُ خمس دوال عضوية. سيمر جهازِ السيطرة الضبابيِ خلال عمليتان؛ الأولى هي توليف معامل القياس للمداخل والمخارج والعمليةَ الثانيةَ هي توليف معاملات الدوال العضويةَ. سيتم هذا الوليف بإستعمالِ الخوارزميةِ الوراثيةِ (GA). جهاز السيطرة الضبابي المُنَظَّم سيقلص الى جدول منطقيٍ بأَخْذ كافة الإحتمالاتِ الضبابيةِ. المخرج لجهازِ السيطرة الضبابيِ المُنَظَّمِ سَيَستحصل باستخدام نظرية مركز الثقل. لتَطبيق هذه الدائرةِ المُنَظَّمةِ يَجِبُ أَنْ نُترجمَ الجدول النَاتج إلى قِيَمِ ثنائيةِ

رقميةِ (باستعمال محول تناظري/رقمي خاص) ثمّ إلى مجموعة من الدوال المنطقيةِ. أخيراً فأن تقنية (FPGA) سَتَستعملُ لبناء الدوال المنطقية الناتجة بإستعمالِ لغة حاسوبيةِ خاصة بهذه التقنية مثل (VHDL) لغة التوصيف المادي. الناتج بعد ذلك سَيَمر خلال محول رقمي/تناظري للحُصُول على الخرج المناسب.

أنّ الفائدةَ المهمة جداً في طريقتِنا هذه هي عملية وصف المسيطر الضبابي باستخدام النظام الرقمي مطبقاً عل تقنية (FPGA) هي الحصول على سرعة معالجة عاليةِ للخرج بالثّانية والفائدة الثانية هي أن سرعةِ هذا المسيطر ستكون غير معتمدة على عددِ القوانين المستخدمة في المسيطر الضبابي.

1. Introduction

In general fuzzy condition may be classified into two types according to fuzzy controller input. For the first type, the fuzzy controller is based on the traditional control theory; e. g. fuzzy controller. Regarding the second type, the controller is constructed with assistance of some useful approaches, such as fuzzy neural network and genetic algorithm, etc.

In this paper, the fuzzy-genetic controller is used as the working method. This method requires a mathematical model for tuning the scaling factor (SF) for the fuzzy inputs and output with fixing the fuzzy membership functions parameters. The second step is to tune the membership functions parameters with fixing the scaling factors of this controller. The used method for tuning is the genetic algorithm method (GA). This controller has two inputs (error (e) and rate of change of this error (ce)) and single output (control action (ca)). The fuzzy controller will be applied to the plat by a closed loop control system (shown in Fig.(1)) to get a tested parameters for the two inputs scaling factor (SF1, SF2) and the single output scaling factor (SF3). The inputs are then fuzzified, rule is applied and the fuzzy input is defuzzified to get a suitable response with fixing the membership functions parameters. IS1, IS2, and IS3 will be the chromosomes for the genetic algorithm (GA) and the fitness function is the error between the desired and the obtained response (see Fig.(2)). After getting the suitable scaling factors we must now tune the membership functions parameters with fixing the scaling factors. Genetic algorithm now will be used another time to get the suitable base parameters for these functions. These parameters will be the chromosomes for the genetic algorithm. Note that the x-axis parameters that limits the base for each membership function we called it the base parameters.

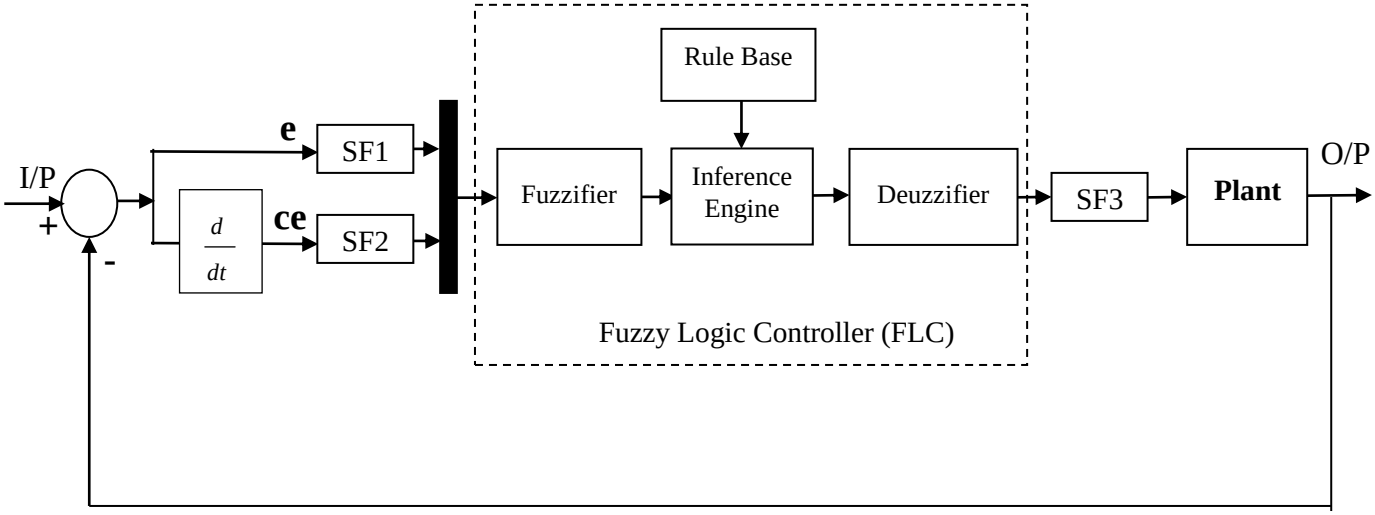

Fig.(1): Structure of fuzzy logic controller with unity feedback control system.

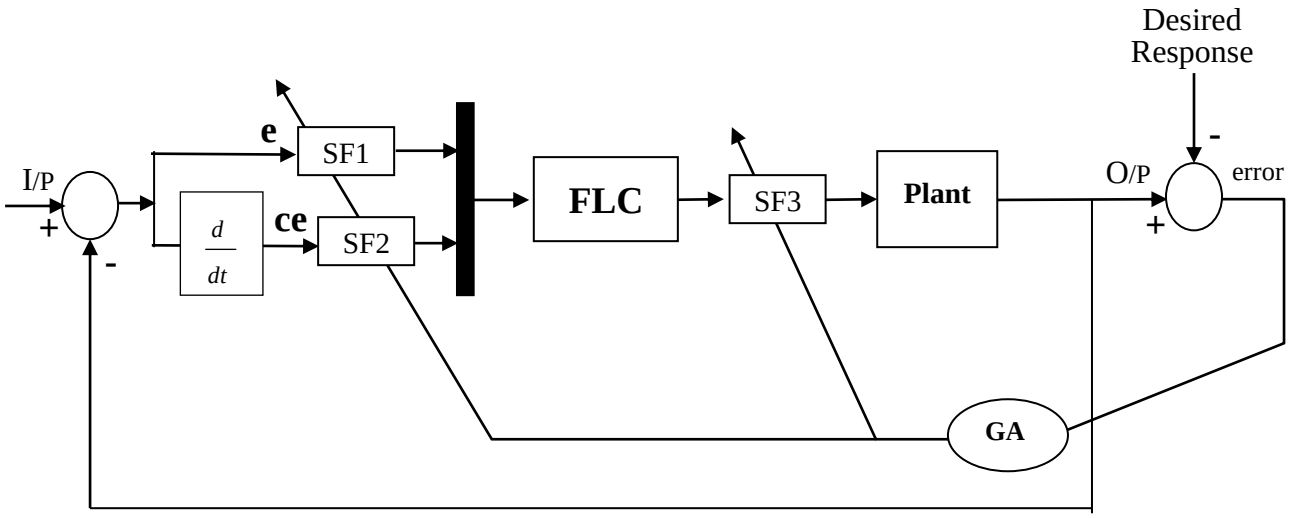

Fig. (2) GA applied to FLC to get optimum scaling factors (SFs)

When the tuning was finished, now we can write a table that contains the base parameters for the error (e) and rate of change of this error (ce). The FLC will be implemented now according to these specifications to get the suitable control action according to Center of Gravity method. We will repeat the same operation for the whole possibilities to cover the whole area and to get a general fuzzy logic controller.

Now we will implement the resulted table by using one of hardware methods. Hardware implementation of the controller can be achieved in a number of ways. The most popular method of implementing fuzzy controller is using a general-purpose microprocessor or microcontroller. General, 8-bit microcontrollers are more economical and flexible, but often face difficulties is dealing with control systems that require high processing and input/output handling speeds. As an option, the controller can be implemented on an FPGA, which is suitable for fast implementation and quick hardware verification. FPGA based systems are

flexible and can be reprogrammed unlimited number of times. Fig.(3) shows the block diagram for our FPGA genetically tuned fuzzy controller system. A special Encoder is used to convert the analog input to digital one depending on the (e) and (ce) base parameters that can the FPGA process it. The Decoder then will give the crisp output depending on the obtained control action by the use of center of gravity method.

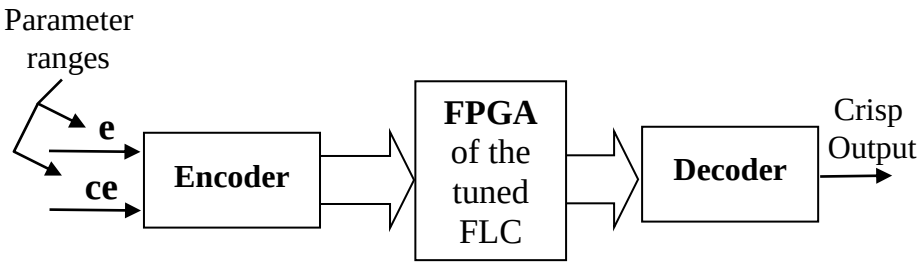

Fig.(3): block diagram for FPGA of genetically tuned fuzzy controller system

For designing the fuzzy rules and simulating the rules, Matlab ® Ver. 7.4.0.287 (R2007a) was used. Rules were written using its fuzzy tools are shown in Table (1). The two inputs (e, ce) and single output (ca) membership functions used in the controller are shown in Fig. (4).

Table-1: Rule base for fuzzy inference system

error (e) \ change of error (ce)	NB	NS	Z	PS	PB
NB	PB	PB	PB	PS	Z
NS	PB	PB	PS	Z	NS
Z	PB	PS	Z	NS	NB
PS	PS	Z	NS	NB	NB
PB	Z	NS	NB	NB	NB

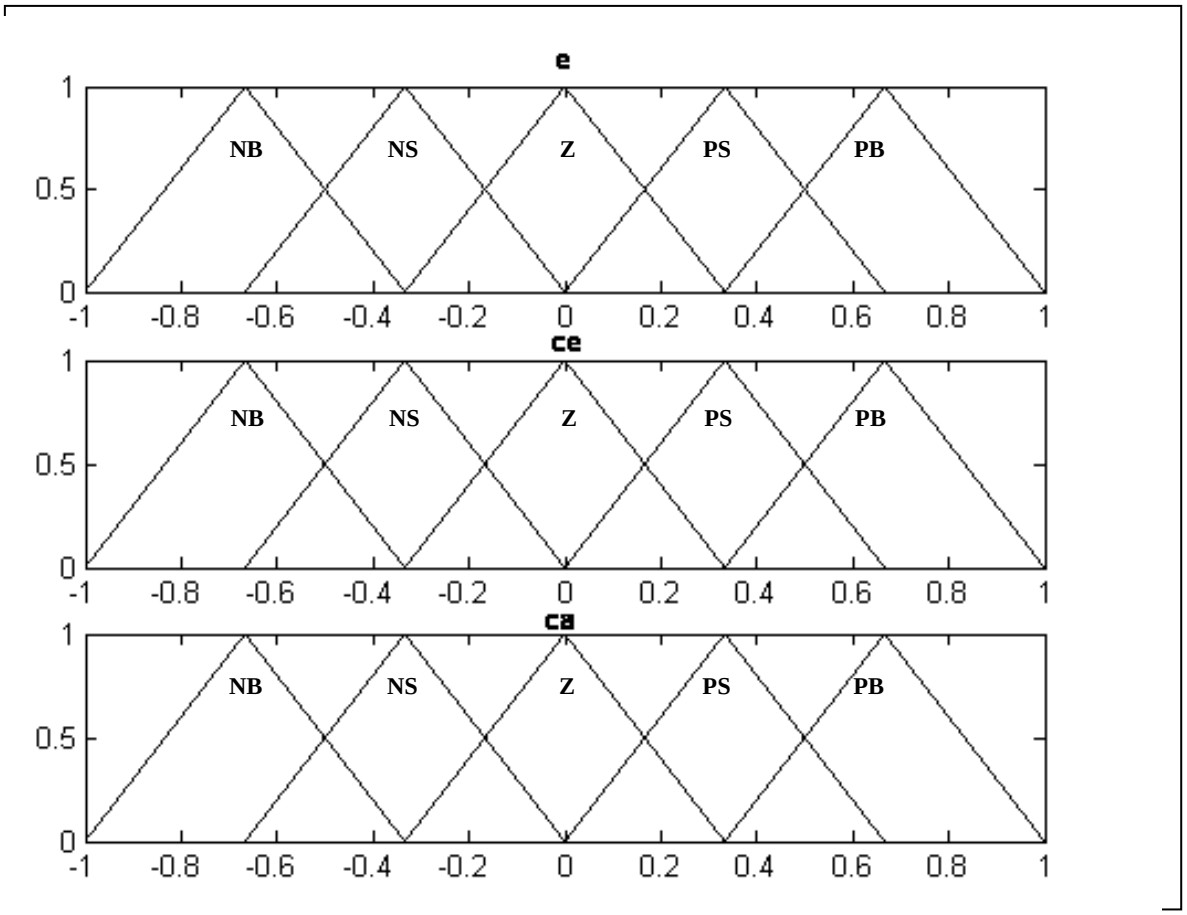

Fig. (4): Initial membership functions of controller inputs (e, ce) and output (ca)

2. Fuzzy_Genetic controllers

The last decade has seen a large interest in technologies that have as their motivation some aspect of human function. Some of these, like artificial intelligence, can be seen to be rooted in the psychological domain. Others, like neural networks, genetic algorithms, and evolutionary programming, are inspired by reconsiderations of biological processes. Common to all these so-called "intelligent technologies" is a need to represent knowledge in a manner that is both faithful to the human style of processing information as well as a form amenable to computer manipulation.

Genetic Algorithms (GAs) are stochastic search techniques that can perform optimization without relying on gradient information or becoming trapped in local minima [1]. The trade-off in using GAs for optimization is that their robust global search, which can also be applied to discrete landscapes, cannot guarantee an optimal solution, but only regions of good solutions. Soft Computing (SC) is an approach to computing which parallels the ability of the human mind to reason and learn in an environment of uncertainty and imprecision. SC combines

knowledge, techniques, and methodologies from Fuzzy Logic, Neural Networks, Probabilistic Reasoning, and Evolutionary Algorithms to create intelligent systems [2].

The use of GAs in Fuzzy Logic systems goes back to the early 90's, when researchers began to use attributes with fuzzy values and a fuzzy pattern matcher for case retrieval [3][4]. In fact, FL techniques have proven to be very useful in addressing many other problems. For example, FL can be used in case representation to provide a characterization of imprecise and uncertain information. It can also be used for case adaptation through the concept of gradual rules [4]. The task of global minimization of numerical functions has paramount importance in several areas of Knowledge. It appears in fields like Engineering, Finance, Management, Medicine, etc.
Our goal was to combine Fuzzy system and GA techniques into a generic, Self-Optimizing Fuzzy capable of handling a wide variety of problems in which an existing case base would be used to build solutions to new cases.

Problems start to rise when the given function presents several local minima, each one having its own attraction basin, making, typically, the final result to depend on the starting point. Unfortunately, most real problems originate very complex objective functions that are nonlinear, discontinuous, multi-modal, high dimensional, etc. To solve such a class of problems, stochastic methods seem to be a good (sometimes, the only) way to go. Genetic algorithms and simulated annealing are among the most popular approaches to stochastic global optimization. The problem in that case is related to speed of convergence and, in the genetic approach, warranty of the ability to reach a global optimum, under general conditions. Pure annealing methods, by the other side, have results assuring its convergence to a global minimum with probability 1, but the performance presented by most implementations is not very encouraging [5].

4. Field Programmable Gate Array (FPGA) technique and its computer programming languages

When they first arrived on the scene in the mid-1980s, FPGAs were largely used to implement glue logic, medium-complexity state machines, and relatively limited data processing tasks. During the early 1990s, as the size and sophistication of FPGAs started to increase, their big markets at that time were in the telecommunications and networking arenas, both of which involved processing large blocks of data and pushing that data around. Later, toward the end of the 1990s, the use of FPGAs in consumer, automotive, and industrial applications underwent a humongous growth spurt.
FPGAs are often used to prototype Application-Specific Integrated Circuits (ASICs) designs or to provide a hardware platform on which to verify the physical implementation of new algorithms. However, their low development cost and short time-to-market mean that they are increasingly finding their way into final products (some of the major FPGA vendors actually have devices they specifically market as competing directly against ASICs).

High-performance FPGAs containing millions of gates are currently available. Some of these devices feature embedded microprocessor cores, highspeed input/output (I/O) devices, and the like. The result is that today's FPGAs can be used to implement just about anything, including communications devices and software-defined radio; radar, image, and other digital signal processing (DSP) applications; and all the way up to System-on-Chip (SoC) components that contain both hardware and software elements [6][7].

With the maturity and availability of hardware description language (HDL) and synthesis software, using them to design custom digital hardware has become a mainstream practice. Because of the resemblance of an HDL code to a traditional program (such as a C program), some users believe incorrectly that designing hardware in HDL involves simply writing syntactically correct software code, and assume that the synthesis software can automatically derive the physical hardware. Unfortunately, synthesis software can only perform transformation and local optimization, and cannot convert a poor description into an efficient implementation. Without an understanding of the hardware architecture, the HDL code frequently leads to unnecessarily complex hardware, or may not even be synthesizable.

This work uses the VHDL hardware description language. VHDL code is simply one of the methods to describe a hardware design. The same design can also be described by a schematic or code in other HDLs. VHDL and synthesis software will not lead automatically to a better or worse design. However, they can shield designers from low-level details and allow them to explore and research better architectures.

The emphasis of the work is on hardware rather than language. Instead of treating synthesis software as a mysterious black box and listing "recipe-like" codes, we explain the relationship between the VHDL constructs and the underlying hardware structure and illustrate how to explore the design space and develop codes that can be synthesized into efficient cell-level implementation. The discussion is independent of technology and can be applied to both ASIC and FPGA devices. The VHDL codes listed in this work largely follow the IEEE 1076.6 RTL synthesis standard and can be accepted by most synthesis software. Most codes can be synthesized without modification by the free "demo-version'' synthesis software provided by FPGA vendors. For more information about this language see [8].

5. Fuzzy controller design using FPGA technique

There are several types of control systems that use FLC as an essential system component. The majority of applications during the past two decades belong to the class of fuzzy controllers. These fuzzy controllers can be further classified into three types: the direct action (DA) type, the gain scheduling (GS) type and a combination of DA and GS types. The majority of fuzzy applications belong to the DA type; here the fuzzy controller is placed within the feedback control loop, and computes the actions through fuzzy inference. In GS type controllers, fuzzy inference is used to compute the individual gains [9][10][11]. The simplest and most usual way to implement a fuzzy controller is to realize it as a computer program on

a general purpose computer. However, a large number of fuzzy control applications require a real-time operation to interface high-speed Constraints.

Software implementation of fuzzy logic on general purpose computers can not be considered as a suitable design solution for this type of application, in such cases, design specifications can be matched by specialized fuzzy processors. Higher density programmable logic devices such as FPGA can be used to integrate large amounts of logic in a single IC [12][13]. Semi-custom and full-custom application specific integrated circuit (ASIC) devices are also used for this purpose but FPGA provide additional flexibility: they can be used with tighter time-to-market schedules. The Field-Programmable Gate Array (FPGA) places fixed logic cell son the wafer, and the FPGA designer constructs more complex functions from these cells. The term field programmable highlights the customizing of the IC by the user, rather than by the foundry manufacturing the FPGA [14][15][16].

A large numbers of fuzzy control applications with the physical systems require a real-time operation to interface high speed constraints; higher density programmable logic devices such as field programmable gate array (FPGA) can be used to integrate large amounts of logic in a single IC. This paper reviews the state of the art of FPGA with the focus on FPGA-based fuzzy logic controller. The paper starts with an overview of FPGA in order to get an idea about FPGA architecture, and followed by an explanation on the hardware implementation with both type analogue and digital implementation, a comparison between fuzzy and conventional controller also provided in this paper. A survey on fuzzy logic controller structure is highlighted in this article with the focus on FPGA-based design of fuzzy logic controller with different applications. Finally, we provided the simulation and experimental results form the literature and concluded the main differences between software-based systems with respect to FPGA-based systems, and the main features for FPGA technology and its real-time applications [17].

6. Our new genetically tuned Fuzzy-FPGA controller

We start our approach using the genetic search by finding the scaling factors (SF1, SF2 and SF3) then we use it another time to find the fuzzy membership base parameters for inputs and output that gives minimum performance index. A suitable performance index is the Integral of the Square of the Error, ***ISE***, which is defined as :

$$ISE = \int_0^T e^2(t)\,dt \qquad \ldots\ldots(1)$$

Where the upper limit T is a finite time chosen somewhat arbitrarily so that the integral approaches a steady– state value. It is usually convenient to choose T as the setting time, T_s.
The fitness value for the j-th chromosome is inversely proportional to the ISE,:

$$fitness = \frac{1}{ISE} \quad \ldots\ldots (2)$$

After applying the genetic algorithms operations (encoding, mutation, selection, and crossover) the survived chromosome will have the optimized fitness that gives a minimum ISE (or optimal solution). The obtained regions from the survived chromosomes then will be encoded to five bits (A, B, C, D, E) according to all possibilities.

The FPGA then will receive a digital signal of five bits and outputs another four bits (f_o, f_1, f_2, f_3) represents the equivalent output membership function after processing using center of gravity method. Before we use the output bits (f_o to f_3) we can reduce them using karnof-map to get a suitable Boolean functions (F, G, H, L) as shown:

$f_o = F(A,B,C,D,E)$
$f_1 = G(A,B,C,D,E)$
$f_2 = H(A,B,C,D,E)$
$f_3 = L(A,B,C,D,E)$

The Boolean functions (F, G, H, and L) will be implemented using the VHDL computer language. The Integrated Software Environment (ISE™) is the Xilinx® design software suite that allows us to take our design from design entry through Xilinx device programming. To use this FPGA device we must have the following hardware: Spartan-3 Startup Kit, containing the Spartan-3 Startup Kit Demo Board. This board will interfaced with our PC to translate the VHDL program to the Xilinx FPGA device which then will represent the genetically tuned fuzzy logic controller.

The special Encoder then will convert the four bit output bits to single analog signal represent the control action that leads the controlled plant as shown in Fig.(5). Note that the black triangle represents a suitable scale multiplied by the signal entering it.

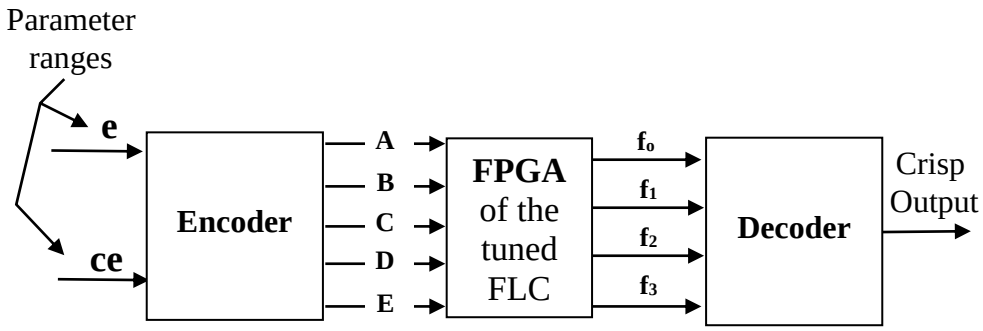

Fig.(5): Signal representation in the FPGA genetically tuned fuzzy controller system

7. Simulated Example

In this section we will implement an FPGA fuzzy controller tuned by the genetic algorithm that controls a plant having the transfer function:

$$G(s) = \frac{1}{s(s+1)} \qquad \ldots\ldots (3)$$

We desire to obtain an optimum controller that have a step response shown in Fig,(6) with settling time (Ts) of (0.6) and a percentage overshoot (PO.) of (32%).

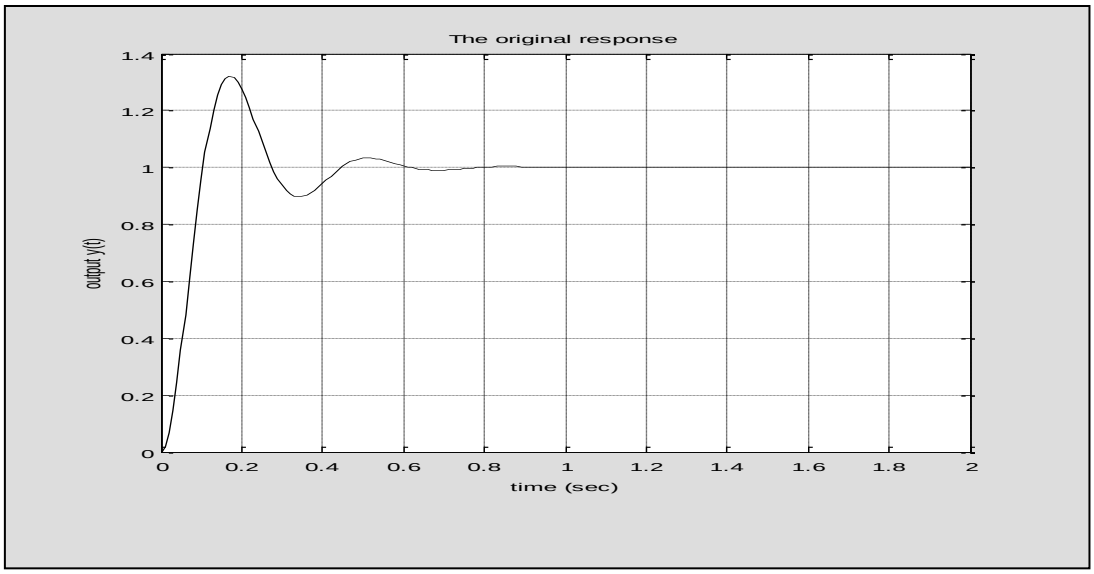

Fig. (6): The original step response

The Integral of the Square Error (***ISE***) between the original step response and the obtained one, dedicated previously in equ.(1), is selected to be minimized by the use of GA in order to obtain optimum scaling factors (SF1, SF2, SF3) with fixing the membership base parameters. The specifications that were used in GA tuning are:

Number of generation =500.

Population size =50.

Chromosome length =3 (for SF1, SF2, SF3)

Crossover probability =0.95 (Simple crossover).

Mutation probability =0.01 (Uniform mutation).

After completing the following scaling factors are obtained that gives minimum ISE =1.9362.

SF1 = 1.0043

SF2 = 1.019

SF3 = -9.7989

Now we will use the GA again to obtain the membership base parameters with fixed scaling factors. The specifications that were used in GA tuning are:

Number of generation =500.

Population size =200.

Chromosome length =35 (for e, ce, ca membership base parameters)

Crossover probability =0.95 (Simple crossover).

Mutation probability =0.01 (Uniform mutation).

The obtained membership functions according to GA optimization method for the two inputs (error (e) and change of error (ce)) and the single output (control action (ca)) are shown in Fig.(7) that gives a minimum ISE of 1.9362. Fig.(8) shows the training steps curve between the fitness and generations during the GA operations and Fig. (9) shows the ISE curve against the GA generations. Fig.(10) shows the step response that we obtained it according to GA operations.

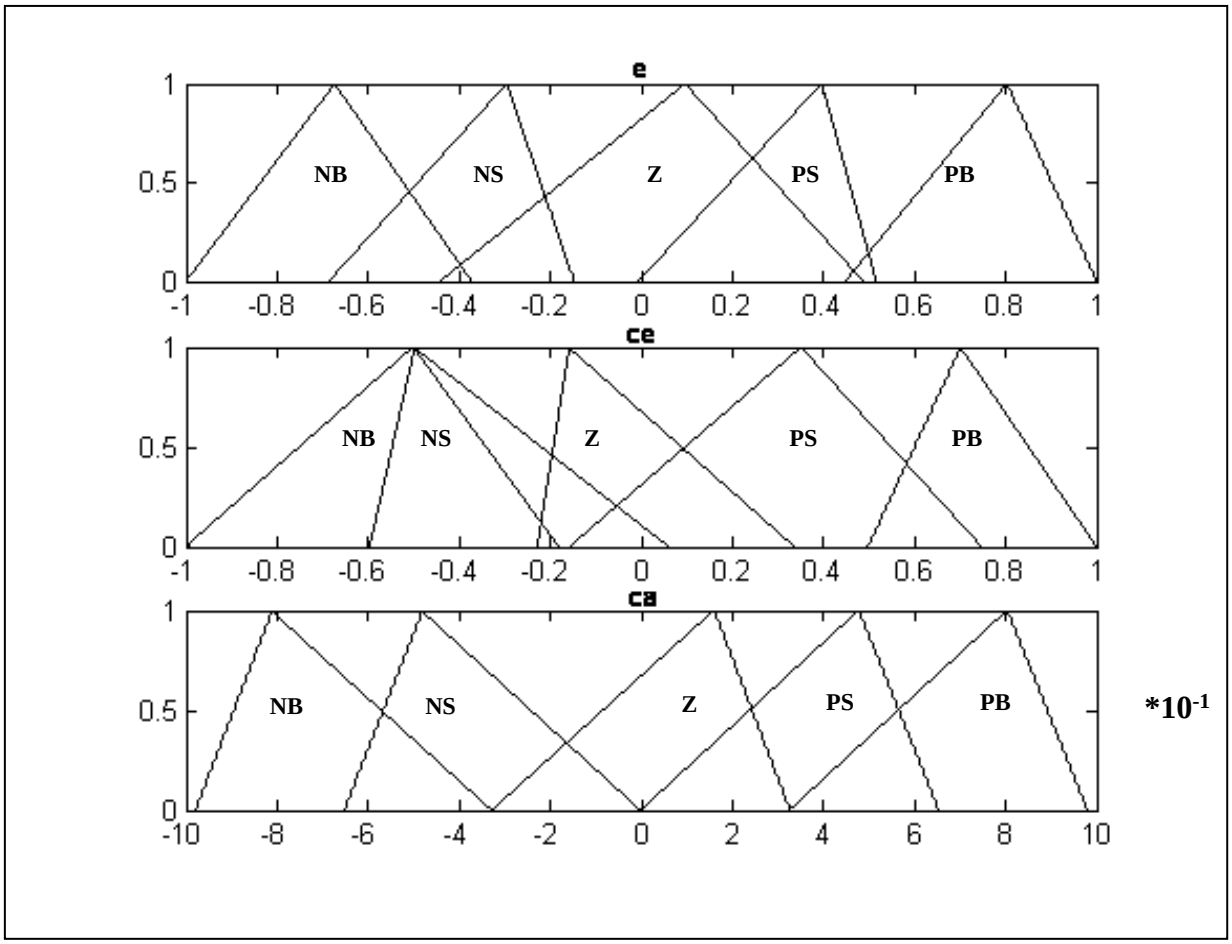

Fig. (7): Optimized membership functions using scales: e=1.0043, ce=1.019, ca=-9.7989

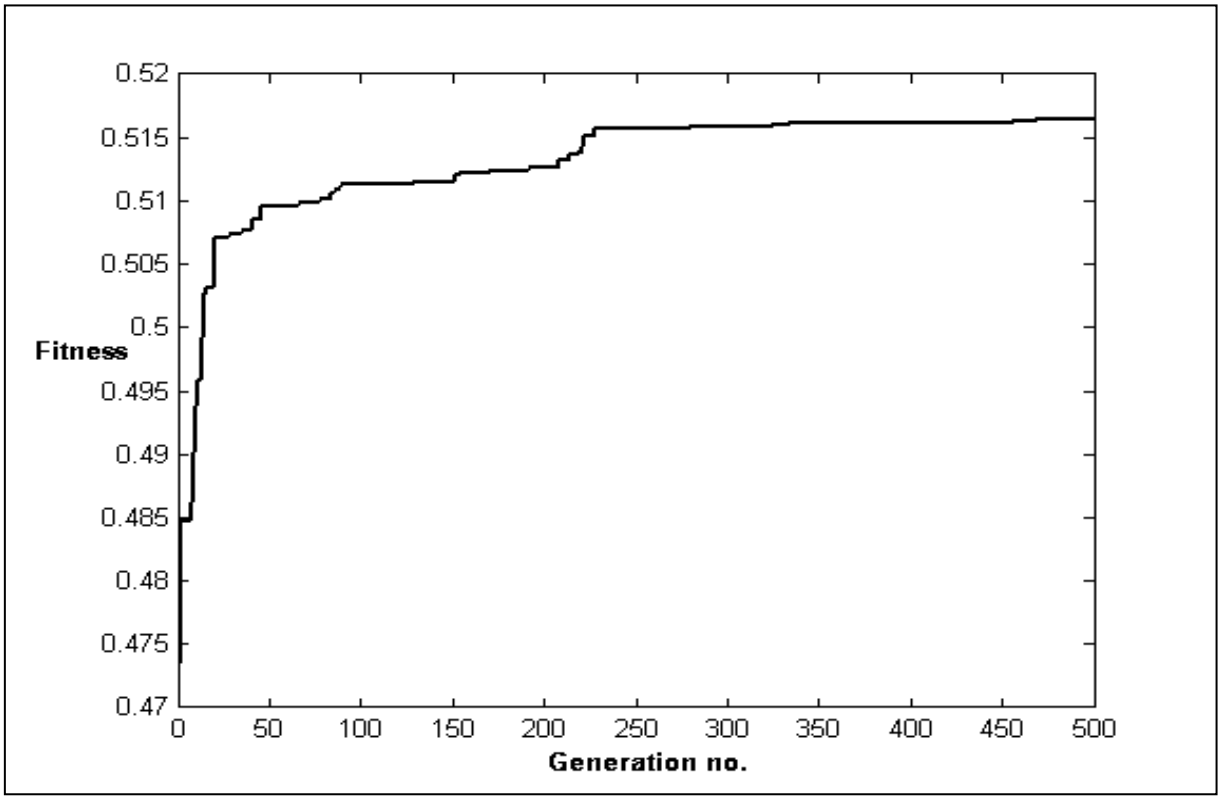

Fig. (8): A curve between fitness and generations during the GA operation

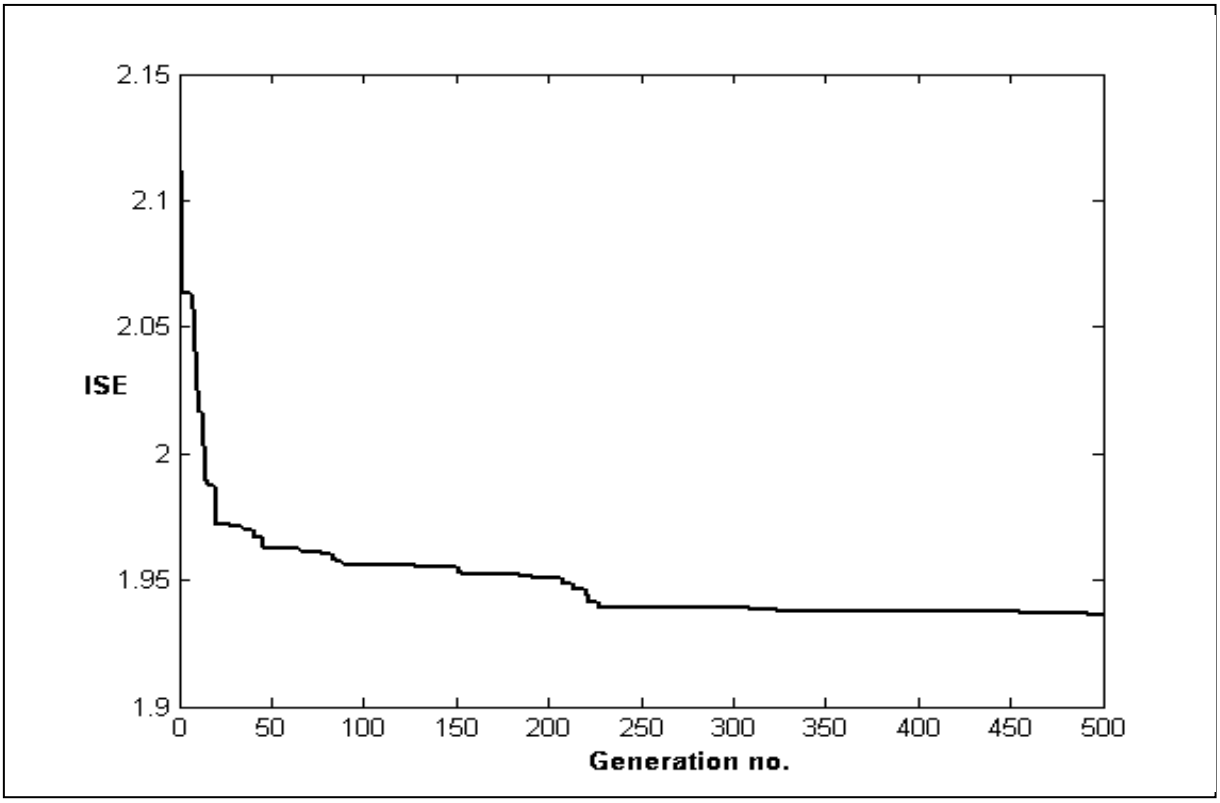

Fig. (9): A curve between ISE and generations during the GA operation

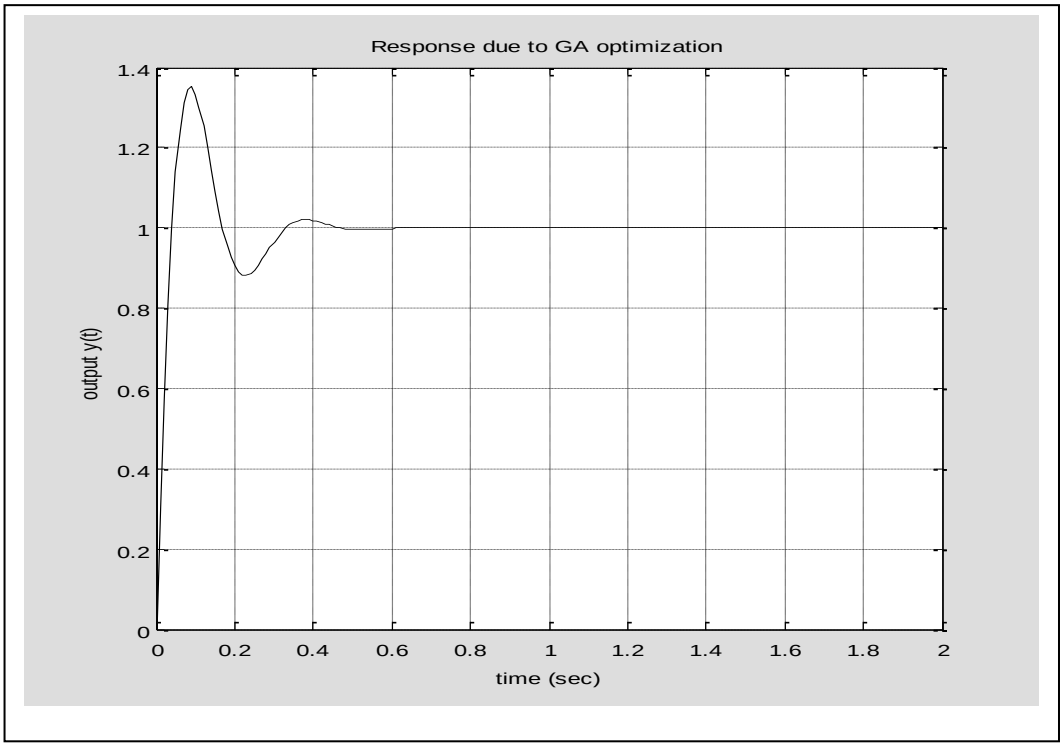

Fig. (10): The GA response

Now we must divide the resultant membership functions to a region then take all the possible choices between (e) and (ce). Table-2 shows the way that we use it to select each region in (e) and applied it in each region in (ce) (divided by SF1 and SF2 respectively) to cover the whole probabilities. Then we will apply these values to our fuzzy logic controller to get the suitable crisp output shown in the third column of the same table after multiply it with SF3.

The encode operation now is begin; as shown in Table-2 there are five regions divided to 25 probability (or rows). Each row will be encoded to binary representation contain five digits (A, B, C, D, E) shown in Table-3. The output also will be encoded to four digits (f_o, f_1, f_2, f_3) after approximation.

Table-2: A lookup table for the (e), (ce) and the crisp value of (ca)

error (e) / SF1	change of error (ce) / SF2	crisp value (ca) / -1	Approx.	Digital Conversion
-1 **to** -0.6	-1 **to** -0.6	0.694 * 9.7989 = 6.8	7	0111
-1 **to** -0.6	-0.6 **to** -0.2	0.707 * 9.7989 = 6.9	7	0111
-1 **to** -0.6	-0.2 **to** 0.2	0.583 * 9.7989 = 5.7	6	0110
-1 **to** -0.6	0.2 **to** 0.6	0.374 * 9.7989 = 3.6	4	0100
-1 **to** -0.6	0.6 **to** 1	0.0427 * 9.7989 = 0.4	0	0000
-0.6 **to** -0.2	-1 **to** -0.6	0.694 * 9.7989 = 6.8	7	0111
-0.6 **to** -0.2	-0.6 **to** -0.2	0.672 * 9.7989 = 6.58	7	0111
-0.6 **to** -0.2	-0.2 **to** 0.2	0.261 * 9.7989 = 2.5	3	0011
-0.6 **to** -0.2	0.2 **to** 0.6	0.0467 * 9.7989 = 0.45	1	0001
-0.6 **to** -0.2	0.6 **to** 1	-0.361 * 9.7989 = -3.53	-4	1100
-0.2 **to** 0.2	-1 **to** -0.6	0.683 * 9.7989 = 6.7	7	0111
-0.2 **to** 0.2	-0.6 **to** -0.2	0.536 * 9.7989 = 5.25	5	0101
-0.2 **to** 0.2	-0.2 **to** 0.2	-0.0528 * 9.7989 = -0.5	-1	1111
-0.2 **to** 0.2	0.2 **to** 0.6	-0.389 * 9.7989 = -3.8	-4	1100
-0.2 **to** 0.2	0.6 **to** 1	-0.711 * 9.7989 = -6.9	-7	1001
0.2 **to** 0.6	-1 **to** -0.6	0.48 * 9.7989 = 4.7	5	0101
0.2 **to** 0.6	-0.6 **to** -0.2	0.285 * 9.7989 = 2.8	3	0011
0.2 **to** 0.6	-0.2 **to** 0.2	-0.318 * 9.7989 = -3.11	-3	1101
0.2 **to** 0.6	0.2 **to** 0.6	-0.625 * 9.7989 = -6.1	-6	1010
0.2 **to** 0.6	0.6 **to** 1	-0.711 * 9.7989 = -6.9	-7	1001
0.6 **to** 1	-1 **to** -0.6	0.0301 * 9.7989 = -0.3	0	0000
0.6 **to** 1	-0.6 **to** -0.2	-0.177 * 9.7989 = -1.73	-2	1110
0.6 **to** 1	-0.2 **to** 0.2	-0.655 * 9.7989 = -6.4	-6	1010
0.6 **to** 1	0.2 **to** 0.6	-0.719 * 9.7989 = -7.04	-7	1001
0.6 **to** 1	0.6 **to** 1	-0.711 * 9.7989 = -6.9	-7	1001

Table-3 shows the lookup table for the encoded input/output memberships after converting it to five and four bits respectively. By the use of karnaugh-map we can reduce the output to form the following logic functions:

$$
\begin{aligned}
f_0 &= \overline{D}BC + B\overline{D}E + BC\overline{E} + AB + AD + AE \\
f_1 &= \overline{ABC} + \overline{D}E + BE + \overline{AC}D + \overline{ABD}\,\overline{E} + BC\overline{D} \\
f_2 &= \overline{BCE} + \overline{ABD}E + B\overline{D}\,\overline{E} + \overline{AB}CD + \overline{B}\,\overline{C}DE + BC\overline{D}\,\overline{E} + \overline{C}D\overline{E} \\
f_3 &= B\overline{E} + BD + \overline{A}CD + ADE + \overline{BCD} + \overline{ABD}E
\end{aligned}
\qquad \ldots\ldots (4)
$$

Table-3: A lookup table for the encoded input/output memberships

A	B	C	D	E		f_o	f_1	f_2	f_3
0	0	0	0	0		0	1	1	1
0	0	0	0	1		0	1	1	1
0	0	0	1	0		0	1	1	0
0	0	0	1	1		0	1	0	0
0	0	1	0	0		0	0	0	0
0	0	1	0	1		0	1	1	1
0	0	1	1	0		0	1	1	1
0	0	1	1	1		0	0	1	1
0	1	0	0	0		0	0	0	1
0	1	0	0	1		1	1	0	0
0	1	0	1	0		0	1	1	1
0	1	0	1	1		0	1	0	1
0	1	1	0	0		1	1	1	1
0	1	1	0	1		1	1	0	0
0	1	1	1	0		1	0	0	1
0	1	1	1	1		0	1	0	1
1	0	0	0	0		0	0	1	1
1	0	0	0	1		1	1	0	1
1	0	0	1	0		1	0	1	0
1	0	0	1	1		1	0	0	1
1	0	1	0	0		0	0	0	0
1	0	1	0	1		1	1	1	0
1	0	1	1	0		1	0	1	0
1	0	1	1	1		1	0	0	1
1	1	0	0	0		1	0	0	1

The final step is to program the Xilinx FPGA instrument depending on the combinational logic functions obtained earlier in Equ.(4) using the VHDL programming language that concerning the board (Spartan-3 Startup Kit Demo Board) interfaced with Pentium-4 PC of speed 3.02 GHz and RAM 2.0 GB

8. Advantages of this approach

1. Fewer values, rules, and decisions are required.
2. More observed variables can be evaluated.
3. Linguistic, not numerical, variables are used, making it similar to the way humans think.

4. It relates output to input, without having to understand all the variables, permitting the design of a system that may be more accurate and stable than one with a conventional control system.
5. Rapid prototyping is possible because a system designer doesn't have to know everything about the system before starting work.
6. They're cheaper to make than conventional systems because they're easier to design.
7. They have increased robustness.
8. They simplify knowledge acquisition and representation.
9. A few rules encompass great complexity.

References

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

[2] Bonissone, P., Chen, Y.-T., Goebel, K. and Khedkar, "**Hybrid Soft Computing Systems: Industrial and Commercial Applications**", Proceedings of the IEEE, vol. 87, no. 9, pp 1641-1667, 1999

[3] Plaza, E. and Lopez de Mantaras, "**A Case-Based Apprentice that Learns from Fuzzy Examples, Methodologies for Intelligent Systems**", 5th edition, Ras, Zemankova and Emrich, Elsevier, pp 420-427, 1990

[4] Dubois, D. and Prade, "G**radual Inference Rules in Approximate Reasoning**", Information Science, vol. 61, pp 103-122, 1992

[5] "**Fuzzy Control of Stochastic Global Optimization Algorithms and Very Fast Simulated Reannealing**", Hime Aguiare Oliveira Junior Electronic and Control Engineer , Independent Researcher Rio de Janeiro , RJ , BRAZIL hime @ engineer.com

[6] Clive " Max " Maxfield, "**FPGAs Word Class Design**", www.newnespress.com

[7] Steve Kilts, "**Advanced FPGA Design Architecture, Implementation, and Optimization**", Copyright # 2007 by John Wiley & Sons, Inc. All rights reserved. Published by John Wiley & Sons, Inc., Hoboken, New Jersey. Published simultaneously in Canada.

[8] PONG P. CHU, "**RTL HARDWARE DESIGN USING VHDL Coding for Efficiency, Portability, and Scalability**", Copyright 0 2006 by John Wiley & Sons, Inc. All rights reserved. Published by John Wiley & Sons, Inc., Hoboken, New Jersey. Published simultaneously in Canada.

[9] I. del Campo, R. Callao, and J. Tarela, "**Automatic Implementation of Different Inference Architectures for Fuzzy Control on PLDs**", Computer and Electrical Engineering Vol. 24, No.1/2, January/March 1998.

[10] G. K. Mann, B. G. Hu, and R. G. Gosine, "**Analysis of Direct Action Fuzzy PID Controller Structures**", IEEE Transactions on Systems, Man, and Cybernetics-Part B: Cybernetics, Vol. 29, No. 3, pp. 371-388, June, 1999.

[11] J. Li, and B. S.Hu, "**The Architecture of Fuzzy PID Gain Conditioner and its FPGA Prototype Implementation**", Second International Conference on ASIC, pp. 61-65, 21-24 October, 1996.

[12] Michael Mc Kenna and Bogdan M. Wilamowski, "**Implementing a Fuzzy System on a Field Programmable Gate Array**", IEEE International Joint Conference on Neural Networks, 2001. Proceedings. IJCNN '01, ISBN: 0-7803-7044-9, Volume: 1, p.189-194, 2001.

[13] Alessandro Gabrielli , Enzo Gandolfi and Massimo Masetti," **Design of a Very High Speed Fuzzy Processor by VHDL Languag**e", Physics Department University of Bologna Viale Berti Pichat 6/240127 Bologna Italy www.bo.infn.it/dacel/papers/96_03_parigi_EDTC.pdf

[14] Jose l. Gonzalez-vazquez, oscar Castillo and luis t. Aguilar-bustos," **A Generic Approach To Fuzzy Logic Controller Synthesis On FPGA**", IEEE international conference on fuzzy systems, p- 2317 - 2322, 2006.

[15] Valentina Salapura and Volker Hamann," **Implementing Fuzzy Control Systems Using VHDL and Statecharts**", Design Automation Conference, 1996, with EURO-VHDL '96 and Exhibition, Proceedings EURO-DAC '96, European, ISBN: 0-8186-7573-X, p. 53-58, Geneva, Switzerland, 1996.

[16] Masmoudi n., hachicha m. and kamoun l. "**Hardware Design of Programmable Fuzzy Controller on FPGA**", IEEE International Fuzzy Systems Conference Proceedings, Seoul, Korea, p. -1675- -1679, August 22-25, 1999.

[17] "**Nasri Sulaiman, Zeyad Assi Obaid, Member, IACSIT, M. H. Marhaban and M. N. Hamidon**", FPGA-Based Fuzzy Logic: Design and Applications – a Review, IACSIT International Journal of Engineering and Technology Vol.1, No.5, ISSN: 1793-8236. December, 2009