Hardware realization of Interval Type 2 Fuzzy Logic Controller

A thesis submitted in partial fulfilment of the requirement for

Dual Degree (Bachelor and Master of Technology)

In

Electronics and Communication Engineering

(VLSI & Embedded System)

By

Ashutosh Padhi

710EC2156

Department of Electronics and Communication Engineering,

National Institute of Technology, Rourkela,

Pin- 769008

Hardware realization of Interval Type 2 Fuzzy Logic Controller

A thesis submitted in partial fulfilment of the requirement for

Dual Degree (Bachelor and Master of Technology)

In

Electronics and Communication Engineering

(VLSI & Embedded System)

By

Ashutosh Padhi

710EC2156

Under the supervision of

Prof. Sarat Kumar Patra

Department of Electronics and Communication Engineering,

National Institute of Technology, Rourkela,

Pin- 769008

Dedicated to my father and mother…

Department of Electronics and
Communication Engineering
National Institute of Technology, Rourkela
Pin- 769008

CERTIFICATE

This is to certify that the thesis entitled "***Hardware realisation of Interval Type 2 Fuzzy Logic Controller***" being submitted by **Ashutosh Padhi** bearing *roll no.* ***710EC2156*** to the National Institute of Technology, Rourkela, in the Electronics and Communication Engineering Department is a bona fide work carried out by him under my supervision and guidance. The research reports and the results presented in this thesis have not been submitted in parts or in full to any other University or Institute.

Place: Rourkela
Date: 29 May 2015

Prof. Sarat Kumar Patra
Dept. of E.C.E.
National Institute of Technology,
Rourkela - 769008

ACKNOWLEDGEMENT

I express my sincere gratitude to my supervisor and mentor Prof. S. K. Patra sir for his invaluable guidance throughout the project period. His guidance truly effective for not only the project work but also in my personal development. Whenever I go to him he always gave me a plan, a perfect roadmap. I learn a lots of things in this project period. I am interested in artificial intelligence, and sir allot me this project which is the part of AI. So I am really thankful to him. My interest towards microcontrollers, microprocessors, MCU programming is catalysed and the concepts got totally cleared by the classes he has taken during fourth semester. As during the project I basically deals with the lower level programming or hardware level programming, it is extremely necessary to be very clear in the basic concepts of MCUs.

I also very grateful to Prof. K.K. Mahapatra sir, Prof A.K. Swain sir and Prof D.P. Acharya sir. They are the reason I am able to design the digital circuits, FSMs for this project and other design works. I mostly learn many thing from our Profs during class time which helps me in my project work and also in my day to day life.

I extend my gratitude towards Bhaskar Rao sir and Pallav Majhi sir for their continuous guidance during the project. A special thanks to Bhaskar Rao sir for encouraging and guiding me throughout the work.

How can I forget our dual degree team, Thandava, Harsha, Tuhin, Samresh, Jha, Vimal, Srinivas, Bibeka, Sanjeet. They just made the thing happen.

Without the priceless softwares like Dia, Inkscpae, Gunplot it is impossible to describe or illustrate what I really wanted to say. I am really very grateful to the corresponding developer community.

Ashutosh Padhi

ashutosh.padhi525@gmail.com

ABSTRACT

Form the publication of Prof. Loftis' fuzzy set theory and linguistic approach, fuzzy logic has become one of the major research topic for researchers. The type 1 fuzzy logic controller has already been used in many areas such as consumer electronics, defense, aerospace engineering etc. This type 1 fuzzy system does not model the fuzziness of the membership function or in other terms it does not deal with uncertainty very well. The control problems where more accuracy is needed, type 1 fuzzy system unable to satisfy their requirements. If a type 2 fuzzy controller is designed, it will have great application in control industry. Type 2 fuzzy system can deal with fuzziness in the decisions or in the membership functions. Membership functions are the decisions of an expert. This inclusion of uncertainty in the design will give us much better control behavior.

The design and hardware realization of interval type 2 fuzzy logic controller is aim of this work. The way of design or methods of design much more important as it will give a clear understanding to the reader.

Table of Contents

Index of Figures

Index of Tables

Chapter 1

1 Introduction:

With the development of human race we are going through the different challenges. Some of the problems we have faced in the electronics world are control system and technologies related to decision making. Previously it was considered that we are the best decision makers and computer and/or any artificial system is far behind us. But in the last few decades with the development of the artificial intelligence we came to know about this hidden ability of computer, that computer can assist us in decision making. Control system is one kind of decision making system but in hardware platform. Think about the classic PI, PID controllers. Those controllers used to be made with heavy electronics circuitry and that was under the scope of analog electronics. Size of the circuits make this products or systems unlikable by us. We want small but powerful systems that can be fitted in our palm.

Those designs have another disadvantage of too much design time. As those were hard to implement that cannot be modified after final implementation, so it takes lots of time in testing and verification before implementation. Which is very costly in terms of resource and time. FPGAs come in handy in this context as a FPGA can be re programmed after final deployment also.

Now concentrating on control system, fuzzy logic control system new in the area of control systems but gives us promising results. Let's assume we wanted to control the temperature of a particular system. With the classic PI, PID method it can be designed and manufactured which cannot be further altered, if there is a requirement. Instead in fuzzy logic controller the system behavior can be altered by changing the rule base of the system. Rule base is one of the building blocks of a fuzzy system, it is going to be discussed later. The whole system is mathematical but it can be programmed as mathematical definition normally suggests about logical concepts,

and if something is logically explainable than it can be programmed. Now with the fuzzy system we are getting the benefits of a flexible decision making system. Which can be programmed that implies a FPGA can be programmed to realize a Fuzzy control system. With this approach a smart control system with minimum size is realizable.

In our design process we are going to go through many challenges, many approaches to solve the problems. The main goal of this work is to find suitable methods and procedures to implement interval type 2 fuzzy logic controller in FPGA. The concepts about interval type 2 fuzzy logic controller will be discussed later. Why interval type 2 is chosen and all related concepts will be revealed gradually. In this work all the concepts are explained with illustrations so that this work can be understood by a person without fuzzy background. Mathematical description are given when necessary only to maintain a writing level simplicity of the complex theories. Concepts about the Fuzzy logic controller are discussed in chapter 2 followed by rule-base optimization and rule-base reduction which are necessary to keep the size and complexity of the system to be designed small. Chapter 4 contains all the simulation results and implementation results. Finally concluded in chapter 5.

Chapter 2

2 Concepts behind Fuzzy Logic Controller:

Before jumping to the mind freshening world of fuzzy system let's think of binary logic. It has only 0s and 1s. Any activity must be described with the above two numbers only. A fan can be on or off not fast, faster etc. If one will take the decimal system then also we have 10 discrete values to describe an activity. In real world nothing is discrete and as real world we are living so nothing is perfect so there is always uncertainties. The uncertainties are like the door is half closed, no it little bit more than half closed etc. How to model this type of uncertainties in any control system. Let us say one of our friend wants to control the speed of the motor inside the washing machine according to the requirements. The output of this system depends on the requirements which are the necessary parameter to control the speed of the motor are cloth type, weight of the cloths, dirt level etc. Proceeding, now we have all the parameter necessary how to control? We need input output relationship. Let's assume one of our friend knows one of those experts in washing machine design. We interviewed that washing machine genius and now we have good IO relationship. But even than how to design the final system. Now consider this one of genius scientist gave me a black box and said take this black box and tell him the rules like

> If cloths is cotton and weight is heavy and dirt level is moderate then speed will be normal.

This is an example, and I am not a washing machine expert. If we have such a system we can model our required control system with the given black box. That black-box is technically called the fuzzy logic controller. These are also come in different flavor. Before jumping into that let us go through the history of the fuzzy logic systems.

2.1 History of Fuzzy Logic Controller:

The idea of fuzzy logic was proposed by Prof Lofti A. Zadeh , university of California, Berkeley, in his 1965 paper. This paper was criticized by academician at that time on the basis of its name. But he continued his work on fuzzy logic and extend and elaborate his work in details in his 1973 paper. The concept of linguistic variable was introduced in this paper. Following this worldwide research started on fuzzy system. This idea got its first industrial implementation in a cement kiln made in Denmark on 1975. Adding to this success Takeshi Yamakawa show the power of fuzzy logic by designing the inverted pendulum experiment in 1987. As one must have seen mechanical wall clocks there is pendulum which keep on swinging per second. The upper end of the pendulum is kept fixed, while the lower end swings left to right on a constant time interval. The concept of inverted pendulum is equivalent to this only different is here the upper end virtually fixed. Consider this if we place a stick vertically on floor, due to gravity it will fall down. If we place a small motor vehicle below the stick move the vehicle opposite to the direction of fall, the vertically placed stick is magically remains stable by the counter balance force of motor vehicle. The movement of the motor vehicle is controlled by the fuzzy logic controller. It was a great experiment indeed. It showed the power of the fuzzy logic systems to us. To add some spice to his experiment and to make the experiment more thrilling and interesting Yamakawa placed a glass of wine above the poor inverted pendulum. Still the system able to keep the balance. Following this Yamakawa keep on doing research on Fuzzy Logic systems and he also started his own fuzzy lab and also applied for lots of patents on fuzzy system.

Subsequently our electronics champ Japanese engineers develop affection on Fuzzy Logic Systems. The started doing research on implementation of fuzzy logic system in industrial and computer applications. To express their interest Japanese engineers also established a laboratory named LIFE, Laboratory for International Fuzzy Engineering in 1988. LIFE gave

home to 48 different companies to do their research on fuzzy. Japanese consumer good pioneers started using fuzzy system in their products. Vacuum cleaners, washing machines, cameras, air conditioners were designed with fuzzy logic systems.

Here in western world Computing technology was taking birth. Super computer following microcomputers, personal computer were being designed. Software engineers were fascinated with concept of artificial intelligence. During that time Fuzzy system attract the attention of artificial intelligence enthusiasts and made a place in AI industry.

This Fuzzy System usually comes in two flavours. Type 1 fuzzy system, Type 2 fuzzy system. Mathematically it can have infinite such fuzzy systems. But considering current restriction in technology and resource only the above two types of systems only can be realised.

2.2 Type 1 Fuzzy Logic System:

This is often called simply fuzzy logic controller. Following this building blocks and working of a fuzzy controller was discussed. As it is mentioned in the introduction part, fuzzy suggests uncertainties. To understand about the uncertainties first let us take a simple example. Say it is winter evening, everywhere only freezing cold. Assume our friend wanted to maintain his home temperature. But how much, assume 25 degree Celsius. Now as an engineer it is our duty to design a system which can maintain his room temperature constant. Before coming to us he pay a controller another human friend to the job for him. Our controller friend has a thermo-meter when temperature in that thermometer rises to 25 he switch of the heater and when temperature falls below 25 he switch the heater back on. So he is following some rules as

If temperature is below 25 switch on heater
If temperature is above 25 switch off heater

Now let us add some uncertainties to our system. Assume our controller friend is blind. So he cannot see the thermometer reading he only can feel. So now how he is going to control the heater. He will probably follow some rules as below

If temperature is hotter switch off the heater
If temperature is colder switch on the heater

You must have been marked the uncertainties. The uncertainties are in the hotter and colder part. It is not a particular discrete constant instead it is a linguistic name hotter, colder. But why application of such linguistic variable is good in a control system. The reason is that we human are always better in decision making. In out temperature control case the human controller will definitely control the temperature system better the any non-living system. As we can feel it so we can deal with it. So as evident from my discussion above if we designed a system which can mimic human operator than it will be a great job. This is the work of prof. Lofti, as discussed in the history part. Add a piece of information to the uncertainty part, maximum of the electronics system designed for human or operated by humans. So where we are, there is always uncertainties.

2.2.1 The Fuzzifier:

Now concentrating on the system, but how the system really did its job. The Fuzzy Control System basically consist of four building blocks, fuzzifier, inference engine, rule base, defuzzifier. All of these together are responsible for the overall system performance. Let us consider another simple example, our friend this time gone to buy mango. There are variety of mangoes in the stall. How to decide? His father gave him some instructions regarding the mangoes. Finally he understood that he had to make the decision on the basis of colour and smell of the mango. So in this case he had two parameters colour of the mango and smell of the mango. But he instead made a fuzzy robot to do the job as he had some other jobs. But our fuzzy robot can only understand numbers as ultimately a digital system. So first the fuzzy robot

say Fuzbot, convert the inputs to numbers but how. To discuss further we need to first analyze the situation. He had to buy mango, mango has two parameter, colour and smell. But how to take the decision what is the rule base, say it was as follows

If Colour is A and Smell is B then buy Z

Where A is set of all possible mango colours, B is set of all possible mango smells and Z is set of buy and not buy. Here Colour and Smell are our input variables, and buy is here output variable. Say mango colour can be red, pink and green. Here these are called linguistic variables. As these have no particular crisp value, so it is a great way of dealing with uncertainties. Now we have to convert these linguistic variable to numbers. Here comes the concepts membership functions. We assign a particular membership function to particular linguistic variable.

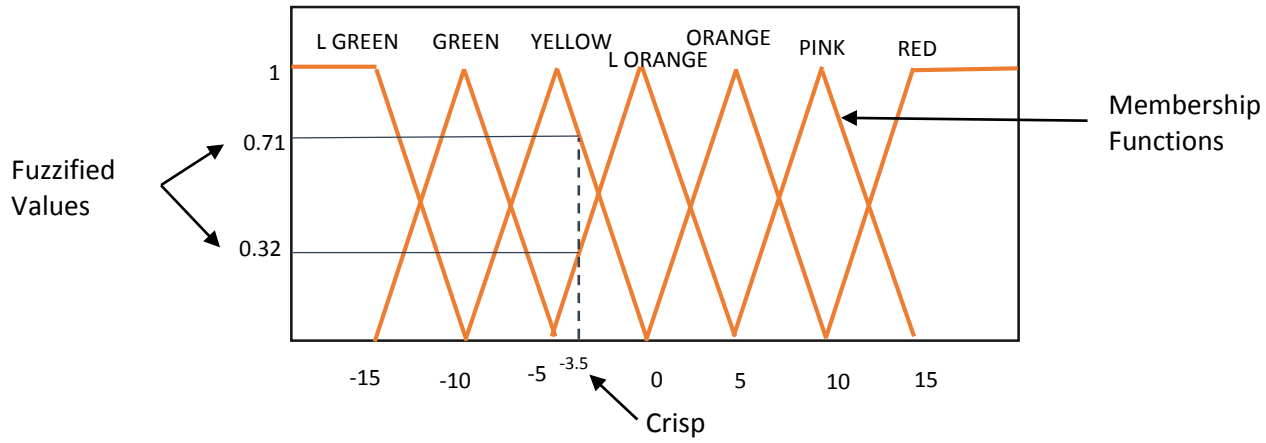

Figure 1: Membership functions for Type 1 Fuzzy system

The above figure illustrate the triangular membership function. Our Fuzbot has computer vision technology, so it sense the colour of the mango form the frequency of the colour it got a mapped value between -15 to +15. Basically this number describes the color of the mango instead of putting the wavelength directly it uses a mapped value. Say the value is -3.5 as per the figure the fuzzified values are 0.71 and 0.32. It tells the Fuzbot that the selected mango is 71% yellow and 32% light orange. This is how our Fuzbot did the fuzzification. It follows the same

procedure to get the fuzzified value for smell. Mathematically fuzzified values is written as $\mu(x)$. x is one of the element of X , X is the set of all possible values that particular variable can have. This X is different than linguistic variables. In our mango example it is the all possible colors the computer vision system on Fuzbot can detect.

A fuzzy set for a particular input and particular membership function can be represented as

$$A_1 = \{(x_1, \mu_{11}(x_1)), (x_2, \mu_{11}(x_2)), (x_3, \mu_{11}(x_3)), (x_4, \mu_{11}(x_4)) \dots\}$$

The Rule-base:

It is also mentioned earlier about the structure of rules. The Fuzbot will follow the rules as per the following format.

> IF COLOUR IS RED AND SMELL IS MAN-1 THEN BUY MANGO; (MAN-1 one of mango smells)

Like this there are many rules. These all rules together called the rule-base.

2.2.2 Inference Engine:

It first gets an overall input on the basis of the rule base, and this input will be then mapped to output. But a creative mind always questions how. So there are two method to get the input value to be mapped to output. These are

1. Min-method (min t-norm): $\min(\mu(x_1), \mu(x_2))$
2. Product –method (product t-norm): $\mu(x_1).\mu(x_2)$

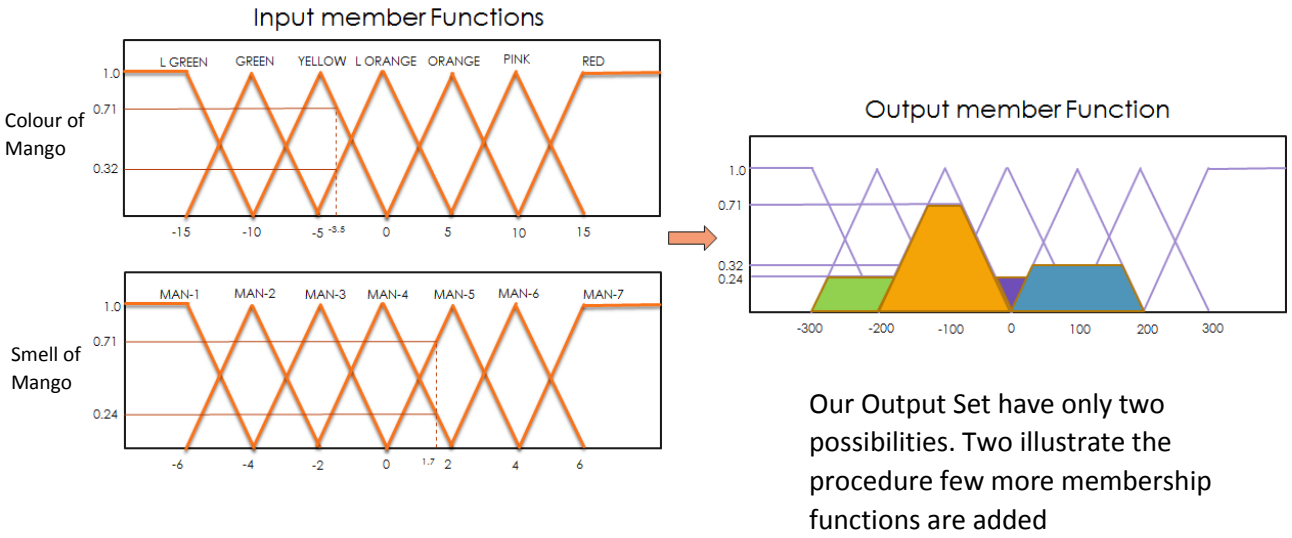

Figure 2: Illustration of Inference Procedure

To unfold the mechanism behind the inference just take one rule at a time. Say in that rule mango is yellow and smell is MAN-4. Yellow has membership value 0.71 and MAN-4 has membership value 0.24 as per the current input to the system or current selected mango for the Fuzbot. At the first stage of inference it choose the minimum of these two, i.e. in our case 0.24 and map this value to output linguistic variable as per the rule say it is buy. So the 'Buy' triangle is clipped at 0.24. This procedure is carried out for all the rules. After that different clipped version of 'Buy' must have remained. Finally the inference engine will choose the maximum one. In the figure 2 this is illustrated but to give a good visual description the number of output membership function is increased to 7.

2.2.3 Defuzzifier:

This is the final step in Fuzzy Logic system. As illustrated in figure 2 after inference engine we have some clipped triangles. Defuzzifier only finds the centroid of the shaded part or selected parts. To find the centroid also there are many methods like center of mass, root mean square method, height method etc.

Mathematically, $y = \frac{\Sigma_{n=1}^{N} y_n f_n}{\Sigma_{n=1}^{N} y_n}$

Here y is the defuzzified output, y_n output per rule f_n is the membership grade of the output variable for nth rule. N is total number of rules.

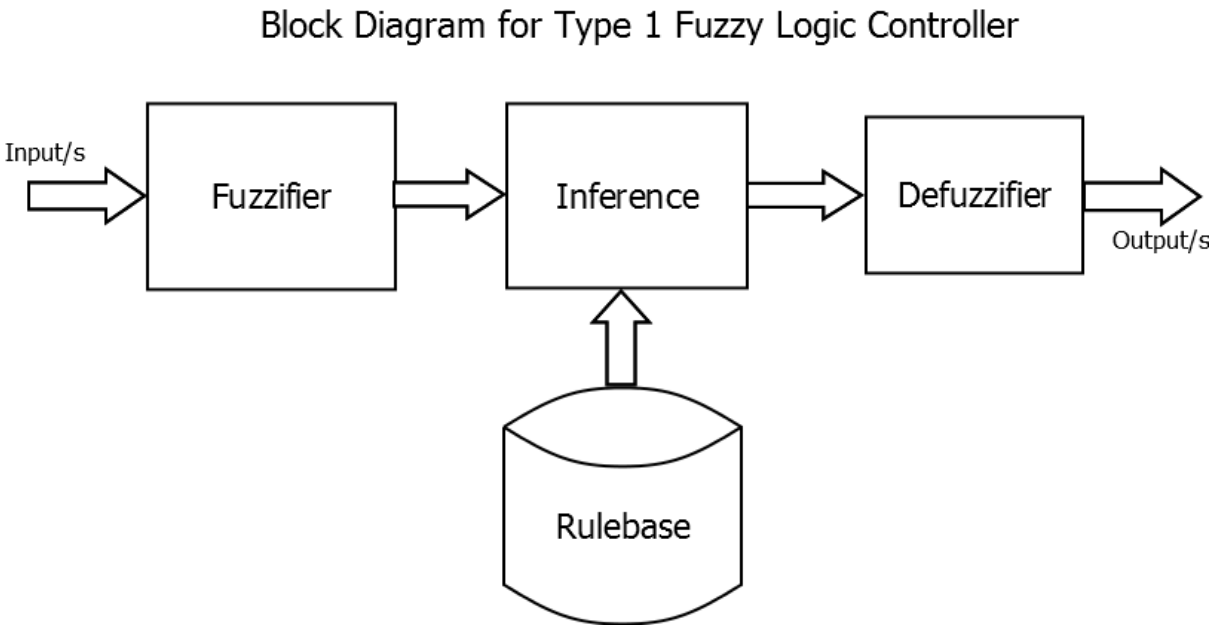

Figure 3: Block diagram for Type 1 Fuzzy Logic Controller

The block diagram of the overall system is shown in figure 3.

2.3 Type 2 Fuzzy Logic System:

As it is mentioned earlier, to model the uncertainties in the design fuzzy logic system is there. But further some researcher consider the fact that the membership function should not be that perfect which we assumed earlier like triangular. Here come the concept of type 2 fuzzy logic system. In the mango example we assumed that for each input it colour of mango the fuzzifier give us value which is the membership grade. But some may argue how the fuzzifier is so perfect in deciding the membership grade. In reality the camera used in the Fuzbot is not perfect so there may be some error, it may also happen that same linguistic variable has different meaning for different people. As the rule base is usually designed from the human experts. So this error enters into the system. To model all these error into the system type 2 fuzzy logic system is the best choice.

Considering the above Fuzbot mango selection example. Suppose it happened to hold a mango of colour little bit greener. The above type 1 fuzzifier will give single membership grade per membership function. But here in type 2 fuzzy logic fuzzifier at that green colour point unlike type 1 it itself an another fuzzy set. So at that particular input we have another fuzzy set which describes some other uncertainties as discussed earlier. From here the name type 2 fuzzy logic controller came. Fuzzy inside fuzzy, i.e. fuzzy to the power 2 or type 2. But realizing the type 2 fuzzy logic controller is difficult due to the complicated mathematics behind that. This type 2 membership function also increase the complexity of the inference and defuzzification part. Now came the gem interval type 2 fuzzy logic controller.

2.4 Interval type 2 fuzzy logic controller:

2.4.1 Fuzzifier:

Instead of taking s fuzzy set in the second level why not we just take the interval. Linking to our past example, for green colour point we had a fuzzy set which is difficult to handle with. So instead we took the interval i.e. membership grade for that particular input lies within a range say lower to upper. In Fuzbot case it may like 0.27 to 0.83. So here the lower value is called the lower membership grade $\underline{\mu_1}(x_1)$ and the other is the upper membership grade $\overline{\mu_1}(x_1)$. Here μ_1 implies the membership function 1. Below is the graph of interval type 2 fuzzy membership functions. If interval type is not considered than the graph will be a 3D graph. If we assume triangular membership function for second degree fuzzy system than imagine a triangular like shaped is raised above the plane of paper. It will look like a v shaved triangular hill.

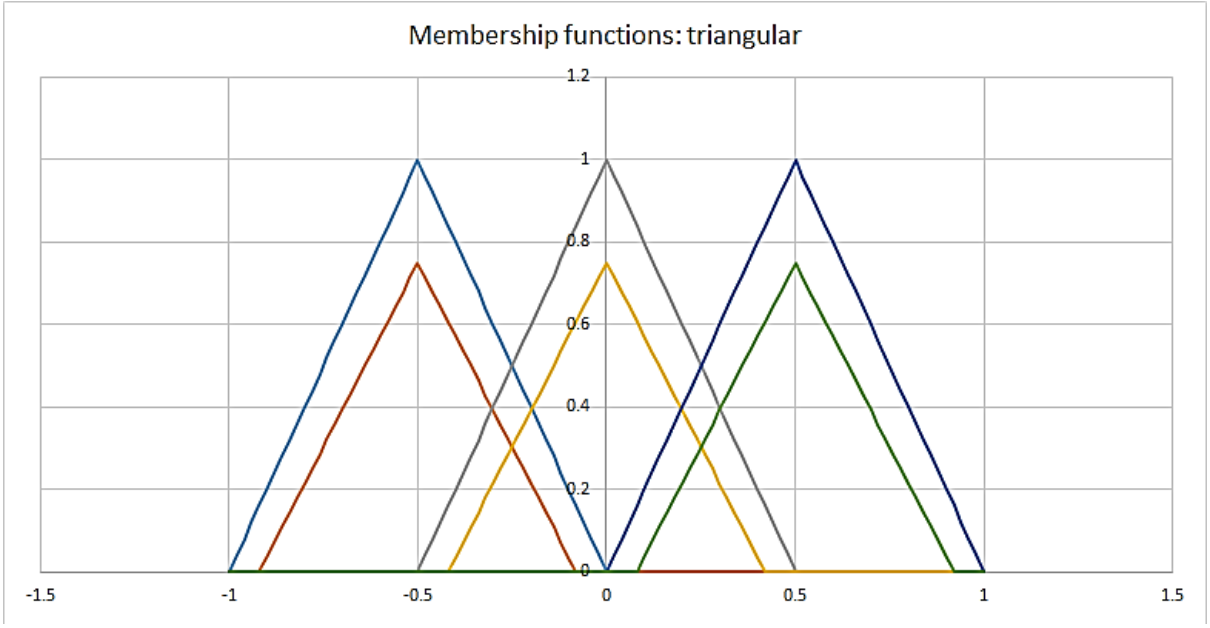

Figure 4: Interval type 2 triangular membership function

2.4.2 Inference:

The inference part is equivalent to the type 1 counterpart. It has too two option to get the inference value. In type to case the mapping part of the inference engine is not there. Here the inference block only finds the min of all triggered inputs per rule. In set theory it is t-norm of the sets. These are explained in detail in [1], [2], [3], [4].

Considering our Fuzbot case if the fuzzifier of the bot fuzzified the colour input as $\underline{\mu_{11}}(x_1)$ and $\overline{\mu_{11}}(x_1)$ and for smell it give us $\underline{\mu_{23}}(x_1)$ and $\overline{\mu_{23}}(x_1)$. Here subscript 11 suggests input 1 membership function 1 and 23 suggests, input 2 membership function 3. So finally we got results as

$$\underline{f_2} = \min(\underline{\mu_{11}}(x_1), \underline{\mu_{23}}(x_1)) \text{ and}$$

$$\overline{f_2} = \min(\overline{\mu_{11}}(x_1), \overline{\mu_{23}}(x_1))$$

As shown here with the equations it find the minimum of lower membership grades and minimum of upper membership grades per particular input.

The rule base is similar for both type 1 and type 2 fuzzy logic system. So now going with the next concept i.e. how to get the defuzzified value. But at this particular point we don't have simple area like type 1 so that we can find centroid of the area to get defuzzified the value. Type reducer comes into picture at this point.

2.4.3 Type-Reducer:

Continuing the discussion from the above para, we need something or some kind of tool which took these inferred antecedent input upper and lower membership grades and the consequence upper and lower membership grades to type 1 type area. From which we can easily find the centroid. This is the job of type reducer. The consequence membership grade are foreign here. In interval type 2 fuzzy systems, the output membership function which are also called consequence membership function, are not the type 2 fuzzy sets as antecedent counterpart. In a way these are fuzzy set but their left and right centroid is pre-calculated and available to us. Now we have consequence parts as $(\overline{f_n}, \underline{f_n})$ and consequence parts as $(\overline{y_n}, \underline{y_n})$. There are few methods to do the type reduction part. In this work only KM algorithm and WU – uncertainty bound set method is only considered.

As we are discussing about type reducer now we have to find what we have and what we are trying to find. So what we have is set of antecedent parts F and set of consequence parts Y. These sets F and Y consists of interval pairs as described earlier as

$$\left[\overline{y_n}, \underline{y_n}\right] \in Y \quad \text{and}$$

$$\left[\overline{f_n}, \underline{f_n}\right] \in F.$$

If we want to visualize this data then we need to graph (1) between $\underline{y_n}$ and F, and another (2) between $\overline{y_n}$ and F. Limiting out discussing only to $\underline{y_n}$ vs F.

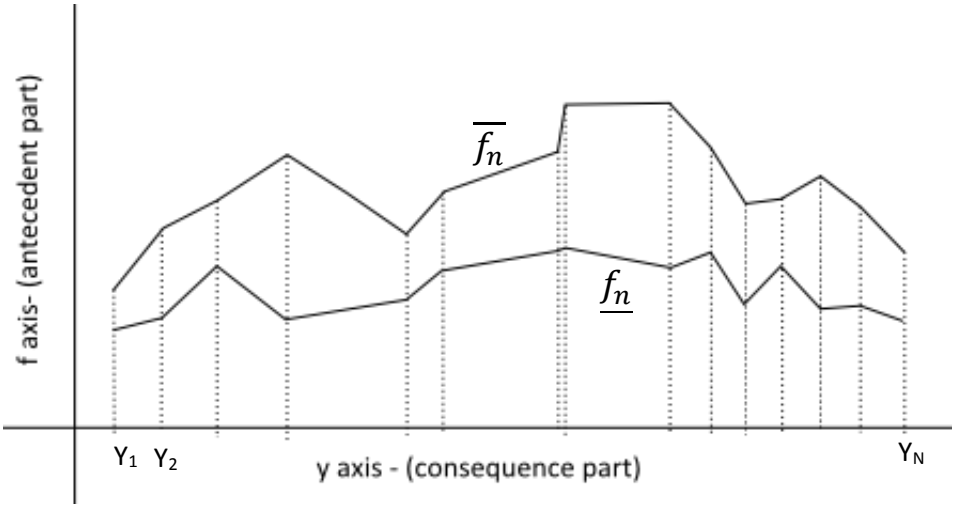

Figure 5: Illustratiing Interval Type 2 inference graph

This how a typical graph will look like. But we need to find out the defuzzified value. One thing we can do is, we convert this to type 1 inference graph. From which the centroid can be found by finding the center of mass of the graph. Still it is not clear how this type 1 fuzzy inference graph will be generated from this type 2 graph. We are only concerned about interval type system so here let us only find a y interval i.e. y_l and y_r. Here y_l is the minimum possible centroid and y_r is the maximum possible centroid. Before finding the maximum and minimum possible centroid what is the function which is going to be maximized or minimized.

$$y = \frac{\sum_{n=1}^{N} y_n f_n}{\sum_{n=1}^{N} y_n},$$

Here our concern is to find the maximum and minimum y value by choosing different combinations of $\overline{f_n}$ and $\underline{f_n}$. Calculus is handy in this regards we can differentiate this function with respect to f to find the maximum and minimum value. All this proof can be found in [5].

Finally we have,

$$y_l = \min_{\forall f_i \in [\underline{f_i}, \overline{f_i}]} \frac{\sum_{i=1}^{N} \underline{y_i} f_i}{\sum_{i=1}^{N} f_i},$$

$$= \frac{\sum_{i=1}^{L} \underline{y_i} \overline{f_i} + \sum_{i=L+1}^{N} \underline{y_i} f_i}{\sum_{i=1}^{L} \overline{f_i} + \sum_{i=L+1}^{N} \underline{f_i}},$$

$$y_r = \max_{\forall f_i \in [\underline{f_i}, \overline{f_i}]} \frac{\sum_{i=1}^{N} \overline{y}_i f_i}{\sum_{i=1}^{N} f_i},$$

$$= \frac{\sum_{i=1}^{R} \overline{y}_i \underline{f_i} + \sum_{i=R+1}^{N} \overline{y}_i \overline{f_i}}{\sum_{i=1}^{R} \underline{f_i} + \sum_{i=R+1}^{N} \overline{f_i}},$$

Here L and R are the switching points for y_l and y_r respectively. Now again in terms of graph it can be shows as follows

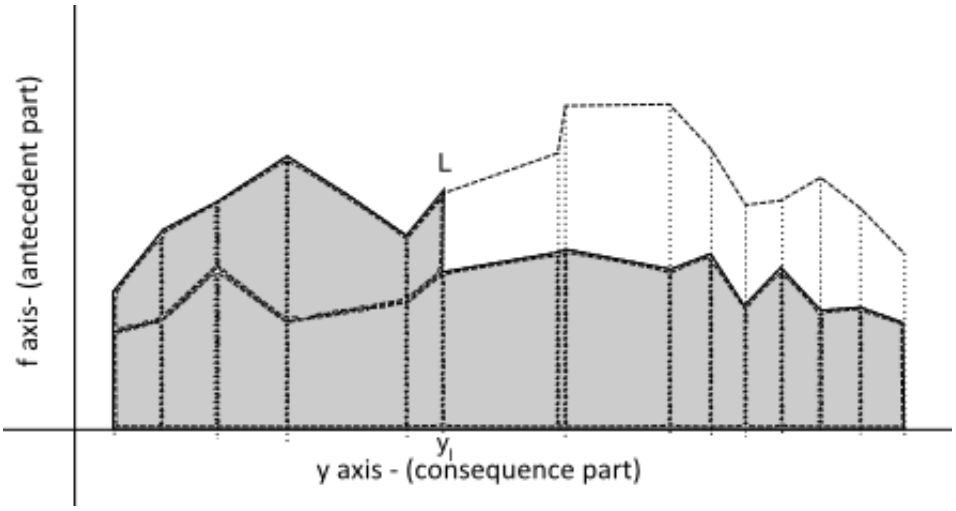

Figure 6: illustraing the left centroid of the output.

From the figure it is self-understandable that y_l is the centroid of the shaded part. So first we need to find the switch point L and R, from where y_l and y_r can be calculated by the expressions shown above.

2.4.4 KM algorithm:

KM algorithm stands for Karnik–Mendel algorithm. Kernik under the guidance of Prof Mendel proposed this iterative algorithm to find y_l and y_r. The algorithm is as follows.

Algorithm: KM (flowchart) (following page)

KM Algorithm to find y_l

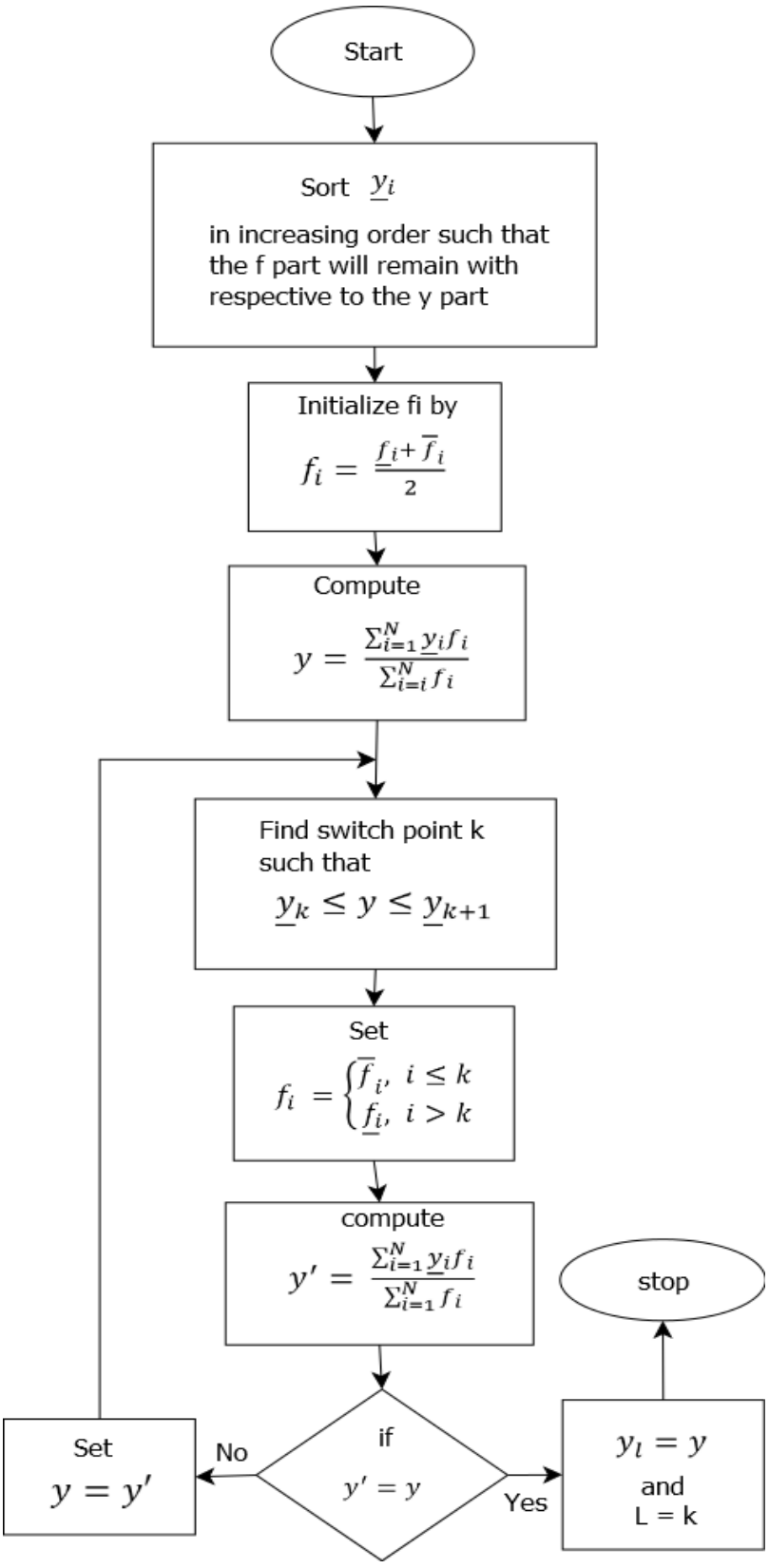

Figure 7: Flow chart for KM algorithm To find out yl

KM algorithm to find y_r

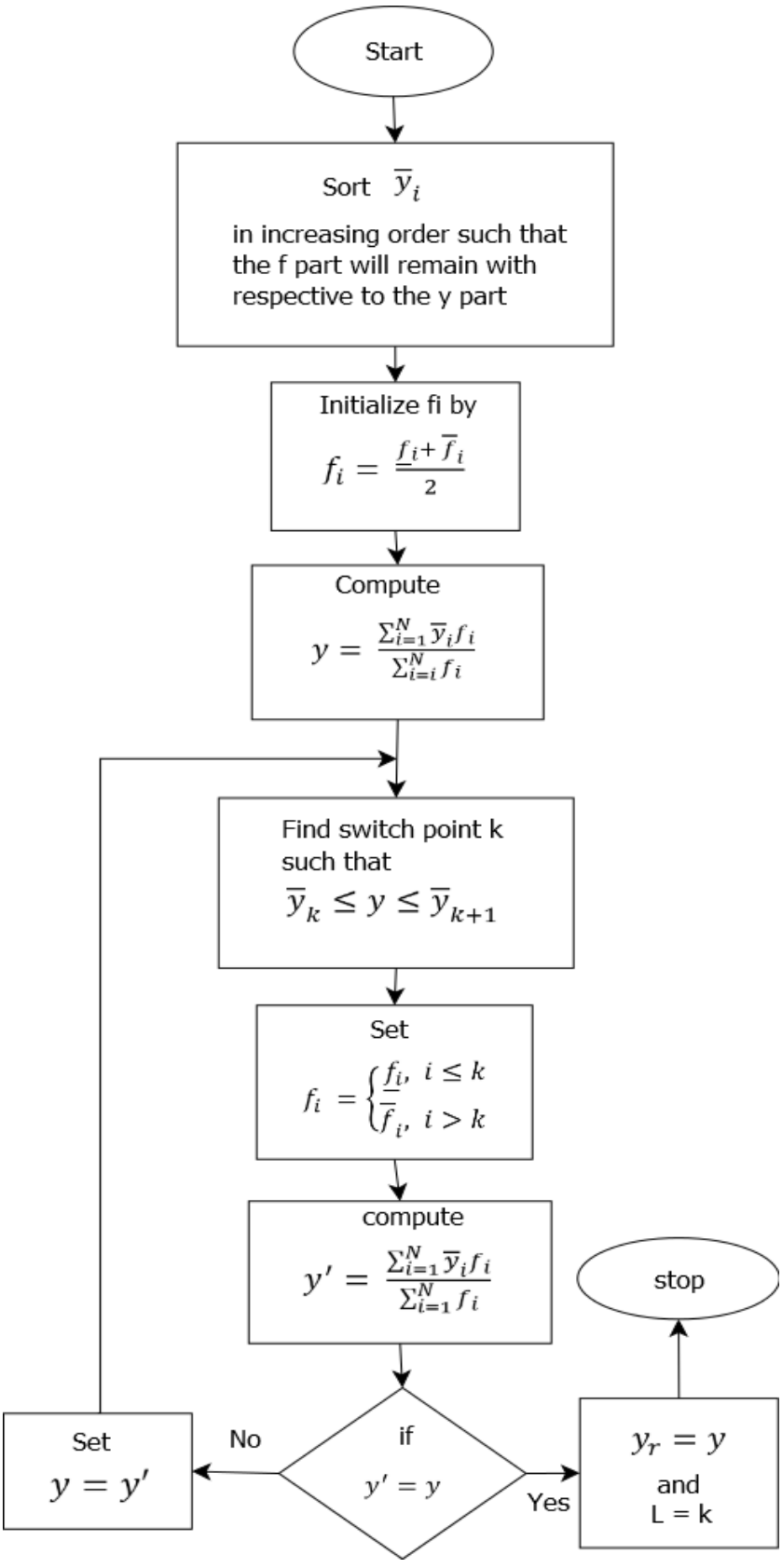

Figure 8: Flow chart for KM algorithm to find out yr

The above shown the flow chart for KM algorithm to find out left and right grades of the final output. Now we have a type one fuzzy inference graph whose centroid can easily be found.

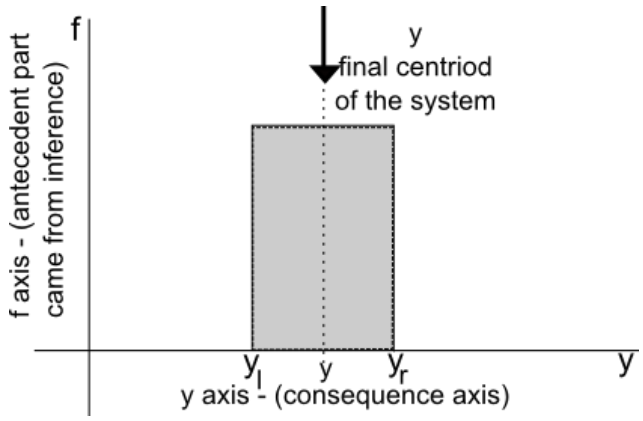

Figure 9: Final type reduced system

From the graph it is sure that by taking the average of y_l and y_r final defuzzified value can be obtained as below,

$$y = \frac{y_l + y_r}{2}.$$

This KM iterative method is used extensively by researchers in fuzzy domain but it got a disadvantage too. Due to its iterative nature its hardware realization is complex. Its design consumes lots of space in FPGAs. Space is serious parameter in FPGA based design as space is very costly in this context. But this problem too has a solution. Another algorithm for type reduction. Basically the main objective of defuzzification is to find a unique output value in accordance with the inference graph. So it only demands that, there must be a relation between the shape of the inference graph and the defuzzified value. If we have a method other than the classic centroid methods, and which also gives us satisfactory results than this method can be used as type reducer. One such method is Wu-Mendel closed form method.

2.4.5 Wu-Mendel closed form method

The Wu-mendel closed form method [6] finds the bound sets for the interval type 2 fuzzy logic controller. The calculation were performed with some closed form expressions. Here closed form suggest there is a limit or one can say a finite number of steps for the calculation unlike KM algorithm which keep on iterating until it gets the switch points. Here the outputs are bound set these are not the type reduced set as KM's method. The bound set gives us the foot print of uncertainty for that particular output. Making it more simple in KM's method we got y_l and y_r and average of these two gave us the defuzzified value y. But later method give us a range for y_l as $[\underline{y}_l, \overline{y}_l]$ and a range for y_r as $[\underline{y}_r, \overline{y}_r]$. The physical significance of these interval are they gave us the region in which probability of finding y_l and y_r is maximum. Here we don't know where y_l or y_r lies, but we have knowledge about a narrow regions in which they are there. If we take the average of these the left and right foot point of uncertainty or $\underline{y}_l$ and $\overline{y}_l$ for y_l , and $\underline{y}_r$ and $\overline{y}_r$ for y_r we will get approximate y_l and approximate y_r . These approximate values may not be perfect but they are giving us satisfactory results while saving our resource in terms of time and silicon space. Few comparison of these two algorithm is shown in the simulation section.

The Wu-mendel Algorithm can be expressed as follows

Step1:

$$y_l^{(0)}(x) = \frac{\sum_{i=1}^{N} \underline{f}^i y_l^i}{\sum_{i=1}^{N} \underline{f}^i}$$

$$y_l^{(m)}(x) = \frac{\sum_{i=1}^{N} \overline{f}^i y_l^i}{\sum_{i=1}^{N} \overline{f}^i}$$

$$y_r^{(m)}(x) = \frac{\sum_{i=1}^{N} \underline{f}^i y_r^i}{\sum_{i=1}^{N} \underline{f}^i}$$

$$y_r^{(0)}(x) = \frac{\sum_{i=1}^{N} \overline{f}^i y_r^i}{\sum_{i=1}^{N} \overline{f}^i}$$

Step 2:

$$\overline{y}_l(x) = min\,\{y_l^{(0)}(x), y_l^{(m)}(x)\}$$

$$\underline{y}_r(x) = \min\{y_r^{(0)}(x), y_r^{(m)}(x)\}$$

Step 3:

$$\underline{y}_l(x) = \overline{y}_l(x) - \left[\frac{\sum_{i=1}^{N}\left(\overline{f}^i - \underline{f}^i\right)}{\sum_{i=1}^{N} \overline{f}^i \sum_{i=1}^{N} \underline{f}^i} \times \frac{\sum_{i=1}^{N} \underline{f}^i \left(y_l^i - y_l^1\right) \sum_{i=1}^{N} \overline{f}^i \left(y_l^N - y_l^i\right)}{\sum_{i=1}^{N} \underline{f}^i \left(y_l^i - y_l^1\right) + \sum_{i=1}^{N} \overline{f}^i \left(y_l^N - y_l^i\right)}\right]$$

$$\overline{y}_r(x) = \underline{y}_r(x) + \left[\frac{\sum_{i=1}^{N}\left(\overline{f}^i - \underline{f}^i\right)}{\sum_{i=1}^{N} \overline{f}^i \sum_{i=1}^{N} \underline{f}^i} \times \frac{\sum_{i=1}^{N} \overline{f}^i \left(y_r^i - y_r^1\right) \sum_{i=1}^{N} \underline{f}^i \left(y_r^N - y_r^i\right)}{\sum_{i=1}^{N} \overline{f}^i \left(y_r^i - y_r^1\right) + \sum_{i=1}^{N} \underline{f}^i \left(y_r^N - y_r^i\right)}\right]$$

Step 4:

$$\left[\overline{y}_l(x), \underline{y}_r(x)\right], Inner\ bound\ set$$

$$\left[\underline{y}_l(x), \overline{y}_r(x)\right], Outer\ bound\ set$$

$$y_l(x) \cong \frac{\overline{y}_l(x) + \underline{y}_l(x)}{2}$$

$$y_r(x) \cong \frac{\overline{y}_r(x) + \underline{y}_r(x)}{2}$$

$$y = \frac{y_l(x) + y_r(x)}{2}$$

Finally the block diagram for type 2 fuzzy logic controller.

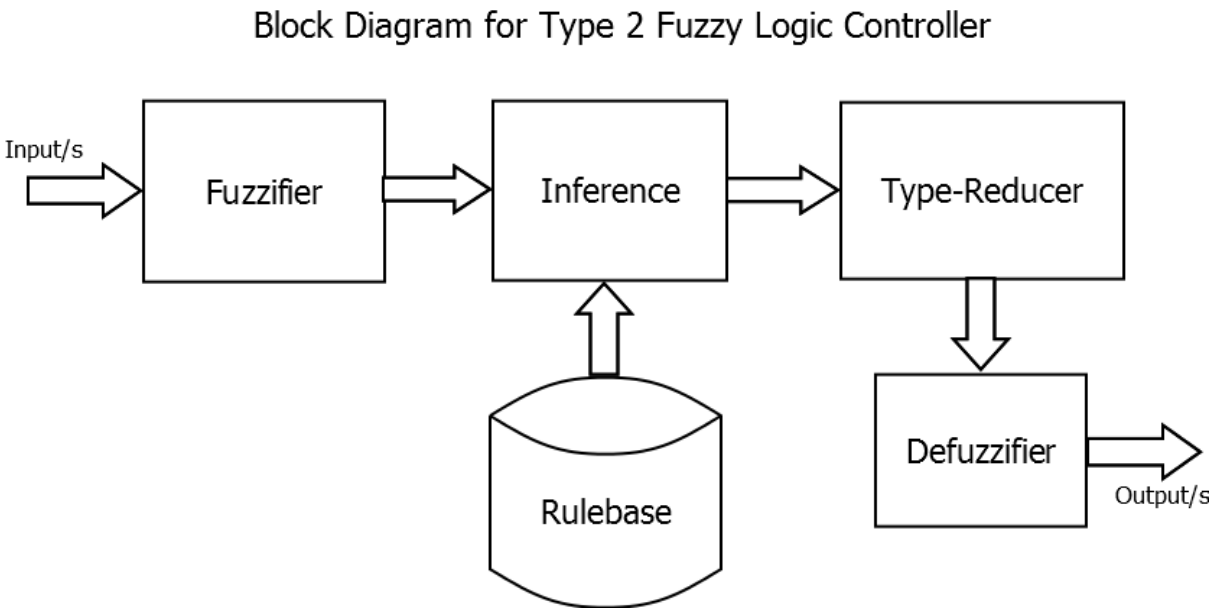

Figure 10: Block Diagram for type 2 Fuzzy Logic Controller

Compering with the block diagram for the Type 1 fuzzy logic controller it has only one extra part which is type reducer. All these concepts were already discussed earlier. Now this block diagram is giving us a complete over all diagram of the working of the system. In this work however the type reducer is replaced by the bound set approximation, the second method for defuzzification. For simplicity in this document the word type reducer will be used in place of bound set approximation system.

Chapter 3

3 Design:

Earlier lots of theoretical stuffs are discussed. Now here we are going to see how the system can be implemented physically i.e. real world realization of the system. Interval type 2 fuzzy logic controller design is given more priority as it was the objective of this work. Now step by step the design procedure of all he building blocks of the fuzzy system will be described and illustrated when there is necessity. All the simulations were done in C programming language and the implementation in Verilog.

3.1 Fuzzifier:

Input data entered to the fuzzy system first goes through the fuzzifier unit. Working of the fuzzifier unit is already discussed. In converts the crisp domain in to fuzzy domain i.e. the output of the fuzzifier will be between 0 and 1. But how it really calculates these membership grades. Before knowing let us think of a type 2 fuzzifier. The visualization is as follows

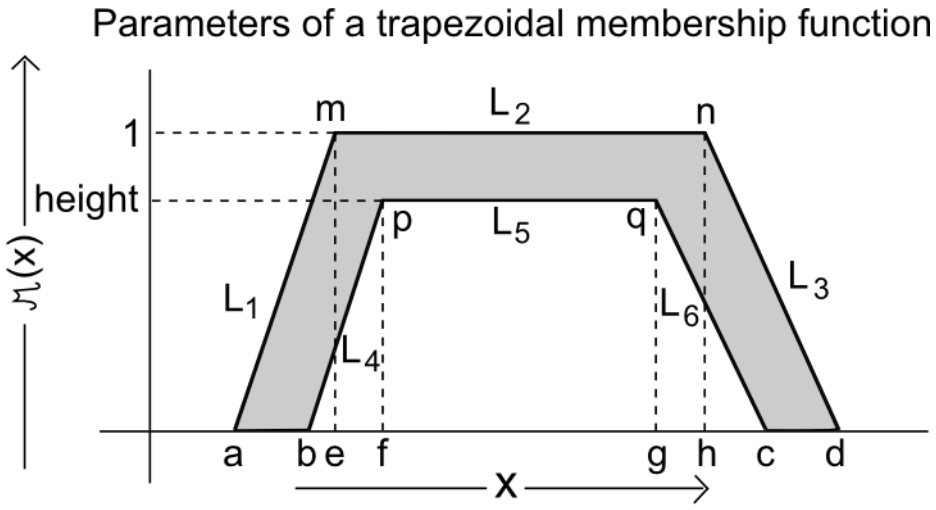

Figure 11: Parameter of a type 2 fuzzifier

Here in this example it is a trapezoidal membership function. In simple mathematical point of view it is basically a function as knowns as $\mu(x)$. There are many other membership functions

as triangular, Gaussian, bell shaped, sink function etc. But how to model these? Gaussian, bell shaped, sink function are complex to realize as they contain exponential terms. It is always difficult for computer to calculate the exponents. Computer may be super-fast but it still takes time. So in real world problem it is better to consider only the trapezoidal and triangular membership functions. These are linear equations so easy to implement.

Generally for trapezoidal interval type 2 fuzzy membership functions are consists of only line segments. As shown in the figure 11. In this example this particular membership function consists of 6 line segments L_1, L_2, L_3, L_4, L_5 and L_6. Consider the line L_3. . If we know the slope of this line segment we can model this line as $y = mx + c$ or

$$(y - y_1) = \frac{(y_2 - y_1)}{(x_2 - x_1)} \times (x - x_1)$$

Here all the points like a, b, c, d, m, n, p, q are set by us so we knew their co-ordinates. So from the above equation the y can be calculated for each x, but this is for one line there are six such line in the membership function. All these lines can be combined using if-then rules in c and Verilog. This method is also called decision tree as programmatically it has a tree structure. In simple English it can be expressed as follows – if x lies in between a and b upper membership grade will be found from the line L_1 and lower membership grade will be calculated from the line equation L_2. A small snippet of the c code,

```
else if (x>=a && x<b)
    {
        *UB = Line1(x,a,0,e,1);
        *LB = 0;
    } else if …
```

Following a small snippet of Verilog code is as follows,

```
else if (x>=h && x<c)
begin
     upper <= line(x,d,slope_hd);
     lower <= line(x,c,slope_cg);
     busy <= 0;
end
else if …
```

In both the code snippet the line is a function it takes the co-ordinates of two points, internally calculates the slope and from the line equation it find the y value corresponds to x value. The c doe will run in Intel core system so the division in the equation is not a big problem for it but for a FPGA it is a big issue. Using a divider demand more clock cycles, hence larger reaction time. The divider are pipelined but still it consumes more space so we will reserve the divider for some other use where it is impossible to proceed without divider. Then what is the solution how to do fuzzification without divider. Simple solution, with the parameter we will send the slope values too. Parameters? Before use of the fuzzifier they must be configured. They need all the 9 parameters which are a, b, c, d, e, f, g, h, height from the figure. To avoid division we will send the slopes of lines $L_1, L_3 L_4$, and L_6 also. Now after coding our fuzzifier is ready. In C it is just a function but in Verilog it is a module as in figure 12.

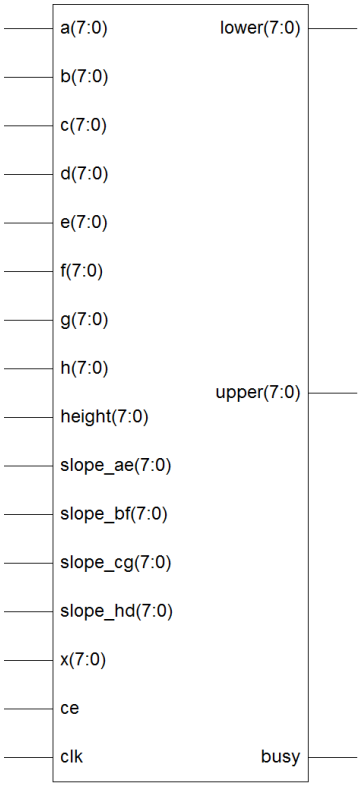

Figure 12: The fuzzifier block

Still there are few thing seem fuzzy here. How one will get those membership function parameters? It is not like that he has to draw it in pen and a paper and got those parameter and send them to the system. It will be time taking, and if one want to test the system with different sets of membership function, then it will too time consuming. So GUI is designed to do the membership function design part easier. That part discussed in the GUI

section. To assist the researcher to play with the system easily a simulator is also designed. The simulator is written in C. The simulator is discussed in the simulation section. Below here is one of the membership function designed as described above, it is designed with the MATLAB GUI and simulated with the simulator.

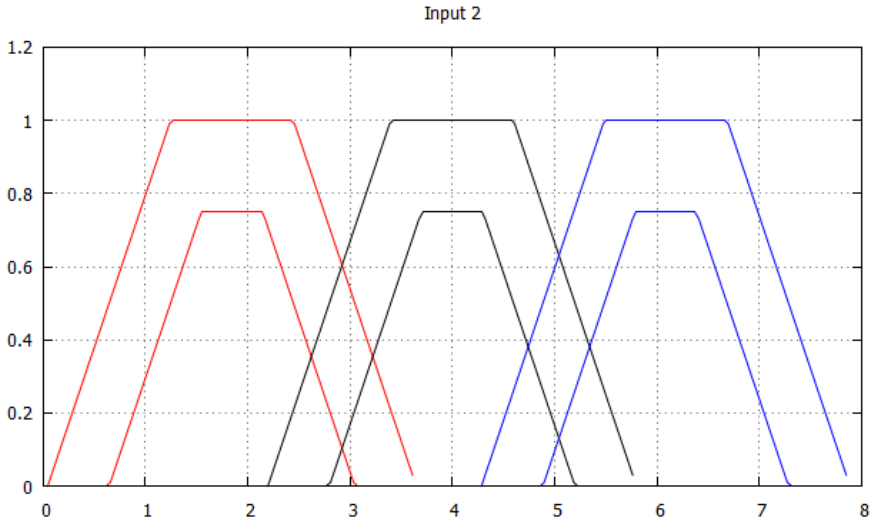

Figure 13: Trapezoidal membership function plotted

It is the membership function for input to for the testing of a benchmark problem discussed in the simulation section.

3.2 Inference engine:

As discussed earlier in the design section mathematically inference is the min or prod t-norm of the concerned sets. In a programmer's point of view we need to find the minimum of the chosen triggered values, chosen according to the rule-base. Snippet of C and Verilog code is given below.

In C:

```
for (rul=0;rul<RSIZE;rul++)
    {
                inf[rul][2] =
find_min4(member1[rulebase[rul][RANT1]][XU],
member2[rulebase[rul][RANT2]][XU],
```

```
member3[rulebase[rul][RANT3]][XU],
member4[rulebase[rul][RANT4]][XU]);
        inf[rul][3] =
find_min4(member1[rulebase[rul][RANT1]][XL],
member2[rulebase[rul][RANT2]][XL],
member3[rulebase[rul][RANT3]][XL],
member4[rulebase[rul][RANT4]][XL]);
        inf[rul][0] = conse[rulebase[rul][RCON]][XU];
        inf[rul][1] = conse[rulebase[rul][RCON]][XL];
    }
```

In Verilog:

```
else if (rule_state == 1)
begin
consed <= din;
    rulebase[counter3][2] <=
min(membf1[in1_index[i]][0],membf2[in2_index[j]][0],memb
f3[in3_index[k]][0],membf4[in4_index[l]][0]);
    rulebase[counter3][3] <=
min(membf1[in1_index[i]][1],membf2[in2_index[j]][1],memb
f3[in3_index[k]][1],membf4[in4_index[l]][1]);
    rulebase[counter3][0] <= conse[consed];
    rulebase[counter3][1] <= conse[consed];
    l <= l+1;
    counter3 <= counter3+1;
    if (counter3 > 15)
    begin
         state1cm <= 1;
    end
    rule_state <= 0;
end
```

The Verilog code given above is the part of a state machine. Inference, rule-reduction and type reduction all the three are done in a single module. So definitely we need a Finite State Machine. The module discussed above is the InfTD module. This module and about the finite state machine will be discussed later in this section.

3.3 Rule-Base:

It may be for C or Verilog the rule base is a bunch of memories to hold the rule data. In C it achieved with the help of an array and in Verilog RAM is used to store rule data. Xilinx core

generator is used to generate the RAM. This RAM has 32 bit wide data bus and 8 bit wide address bus. It can be visualized with the help of following diagram.

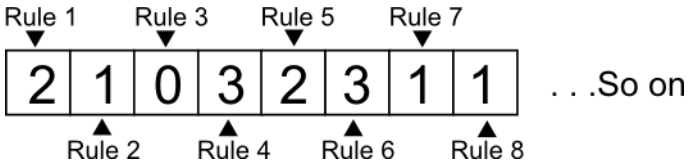

Figure 14: Visualization of Rulebase

Here above each box is a memory location physically, which can be accessed by its physical address. The content of the memory location is the number indicating the output membership function index. We may have rule like

> Rule 2: IF input 1 IS 2 AND input 2 IS 0 THAN OUTPUT WILL BE 1

Here the indices are the numbers implying the membership function. In our FuzBot example, if there are 3 mango colors say green, yellow and red, than index of green will be 0, 1 for yellow and 2 for red. Start is of index is self-choice here in this work it is zero and another reason for starting zero is that both Verilog and C following this starting zero rule for arrays.

3.4 Type Reducer:

The algorithm for type reductions are already discussed. For C simulation both the KM and Wu-Mendel algorithms are implemented but as I have already mentioned the KM algorithm is complex enough for FPGA realization. It is not impossible, it is just the out of scope for this work. Code for KM algorithm is big enough for showing here, so only code snippets of Wu-Mendel algorithms are given below.

C code:

```
for (i=0;i<RSIZE;i++){
        fl_sum += inf[i][WLOWER];
        fu_sum += inf[i][WUPPER];
        fl_yl += inf[i][WLOWER]*inf[i][XLOWER];
```

```
        fl_yr += inf[i][WLOWER]*inf[i][XUPPER];
        fu_yl += inf[i][WUPPER]*inf[i][XLOWER];
        fu_yr += inf[i][WUPPER]*inf[i][XUPPER];
        fdiff_sum += (inf[i][WUPPER]-inf[i][WLOWER]);
        fl_yl1_diff += inf[i][WLOWER]*(inf[i][XLOWER]-
inf[0][XLOWER]);
        fu_ylm_diff += inf[i][WUPPER]*(inf[RSIZE-
1][XLOWER]-inf[i][XLOWER]);
        fu_yr1_diff += inf[i][WUPPER]*(inf[i][XUPPER]-
inf[0][XUPPER]);
        fl_yrm_diff += inf[i][WLOWER]*(inf[RSIZE-
1][XUPPER]-inf[i][XUPPER]);
    }
    yl0 = fl_yl/fl_sum;
    ylm = fu_yl/fu_sum;
    yrm = fl_yr/fl_sum;
    yr0 = fu_yr/fu_sum;
    ylu = find_min(yl0,ylm);
    yrl = find_max(yr0,yrm);

    yll = ylu -
(fdiff_sum*fl_yl1_diff*fu_ylm_diff)/((fl_yl1_diff+fu_ylm
_diff+0.00001)*fl_sum*fu_sum);
    yru = yrl +
(fdiff_sum*fu_yr1_diff*fl_yrm_diff)/((fu_yr1_diff+fl_yrm
_diff+0.00001)*fl_sum*fu_sum);
    yl = (yll+ylu)/2;
    yr = (yrl+yru)/2;
    return (yl+yr)/2;
```

Verilog Code:

```
if (part_one_complete==0)
 begin
    for (z=0;z<=15;z=z+1)
     begin
         fl_sum <= fl_sum + rulebase[z][2];
         fu_sum <= fu_sum + rulebase[z][3];
         fl_yl <= fl_yl + rulebase[z][2]*rulebase[z][0];
         fl_yr <= fl_yr + rulebase[z][2]*rulebase[z][1];
         fu_yl <= fu_yl + rulebase[z][3]*rulebase[z][0];
         fu_yr <= fu_yr + rulebase[z][3]*rulebase[z][1];
         fdiff_sum <= fdiff_sum + (rulebase[z][3]-
rulebase[z][2]);
         fl_yl1_diff <= fl_yl1_diff +
rulebase[z][2]*(rulebase[z][0]-rulebase[0][0]);
         fu_ylm_diff <= fu_ylm_diff +
rulebase[z][3]*(rulebase[15][0]-rulebase[z][0]);
         fu_yr1_diff <= fu_yr1_diff +
rulebase[z][3]*(rulebase[z][1]-rulebase[0][1]);
```

```
          fl_yrm_diff <= fl_yrm_diff +
rulebase[z][2]*(rulebase[15][1]-rulebase[z][1]);
    end
    yll_part_dvnd <=
(fdiff_sum*fl_yl1_diff*fu_ylm_diff);
    yll_part_dvsr <=
((fl_yl1_diff+fu_ylm_diff+1)*fl_sum*fu_sum);
    yru_part_dvnd <=
(fdiff_sum*fu_yr1_diff*fl_yrm_diff);
    yru_part_dvsr <=
((fu_yr1_diff+fl_yrm_diff+1)*fl_sum*fu_sum);

    part_one_complete <= 1;
 end
if (all_div_complete==1)
begin
ylu <= min2(yl0,ylm);
    yrl <= max2(yr0,yrm);

yll <= ylu - yll_part_qnt;
    yru <= yrl + yru_part_qnt;
    yl <= (yll + ylu) >>> 1;
    yr <= (yrl + yru) >>> 1;
    fuzzout <= (yl + yr) >>> 1;
    state2cm <= 1;
 end
```

In both the code c and Verilog, the for-loop part is same. The algorithm is analyzed the reoccurring part, for example the summation parts are calculated at once using one for loop. If they were coded separately we might have many loop and each loop will take rule number of cycles. In the code given above it is 15, i.e. number of rules is 15 here it is the number of reduced rules. Concept of reduced rule is going to be discussed later in this section. For Verilog it is not a problem as the complete for-loop will be executed in one clock cycle, but for C it will be a problem. Common code is written which can be applicable in both the languages.

3.5 InfTD module:

Rule reducer, type reducer, inference and defuzzifier all the 4 unit are combined in one Verilog module, the InfTD module. As it is mentioned earlier divider consumes lots of clock cycle, but to Implements the Wu-Mendel algorithm it is necessary. So a divider unit is generated from

Xilinx core generator and is used here. These divider have latency period which depends upon the dividend length and number of divisions per cycle. From the divider database it is confirmed that latency for the instance of divider will be 19. It implies after giving the divider all the input it will take 19 clock cycles to give me first division result. Division per cycle is 1 here, so after latency i.e. 19 clock cycles it will give the division results per clock cycle. This is achieved internally by pipelining. Before explaining how the complete system works, first the unknown part i.e. the rule reducer is discussed.

3.6 Rule Reducer:

It is evident from the Wu-Mendel algorithm is that the complexity of the hard ware increases with the number of number of rules. If say one of our friend have a system having 4 input fuzzy system each having 4 membership function. So in this case we have 256 possible rules. The for-loop in Verilog will be synthesizable but it will consume lots of resource. If the number of rule will be reduced then it will be huge decrease in silicon consumption. To coming the state where we can reduce rules. Let's think how this inference data, which is going to be processed by the type reducer has come.

The inference engine scans all the rules and find corresponding t-normed values referring the rule-base. Here for inference min t-norm is used. So if among the input one input has zero membership grade then the inference engine will produce a zero for that rule as zero is the lowest one. The rules with zero antecedent grade has no effect in the final output of the type reducer. So our concern here is to scan out the zero valued rules. This is illustrated as follows,

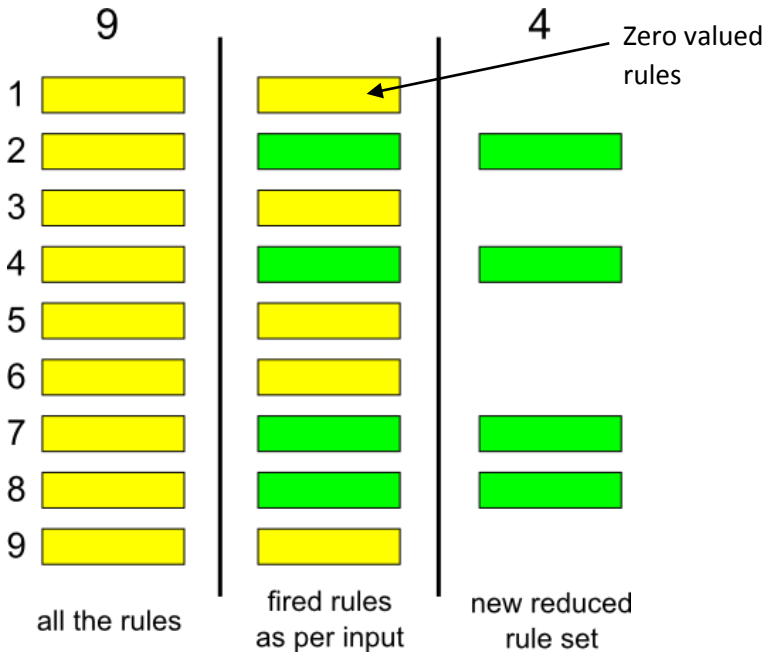

Figure 15: Illustrating rule reduction

In rule-base section it is shown how rule base is implemented in the memory. Memory contains the output membership function index for each rule, and rules are 1, 2, 3, … from the starting of the rule-base memory. Below show a rule arrangement table for a 4 input system with each having 3 membership function.

Table 1:Rule arrangement table

Input 1	Input 2	Input 3	Input 4	Output/ Rulebase	Rule number
0	0	0	0	2	1
0	0	0	1	1	2
0	0	0	2	2	3
0	0	1	0	0	4
…	…	…	…	…	…
2	2	2	2	1	81

Output or the rule-base value given here are random. We have only set of output for the rule-base. Now consider the fuzzifier output. If we scans out the zeroed value fuzzifier output and now we are having a set of valid input membership indices. Than we only choose the rules with the valid membership input indices. If input 1 has 2 valid membership indices say 1 and 2, then

only those rules which include one of these will only be considered. So now we store the valid indices for each inputs, in our case 4 sets were there each for each input. Now taking the combinations of these valid input memberships will be considered for inference. We need to find a relationship between the input indices and the rule number as rule can be accessed by its rule number. Here is the relation,

$$RuleNo = input1 \times 3^3 + input2 \times 3^2 + input3 \times 3^1 + input4 \times 3^0$$

Now it is easy we will take the valid input indices and take their different combinations and for the particular combination selected rule index can be found from the above expression. From the content of the rule-base inference will be carried-out. If the overlapping of membership function is at most two than for a particular input only two of three membership functions will give non zero value. 2 for each input so finally we have only 2^4 i.e. 16 rules. A huge reduce in number of rules.

Now for every change in input or for every input trigger we will need small memory for storing the inference data which is going to be used by the rule reducer.

3.7 The complete InfTD module:

The complete InfTd module is controlled by a Finite state machine. There are only three stpes in that finite step machine as follows

1. Find the valid membership indices
2. For each combination of indices do the inference and find the inference data and execute all the summation parts
3. Execute all the required divisions and find the quotient for each division
4. Finish the process by finding out the y (output)

From the synthesis of the inference block the RTL as shown in figure.

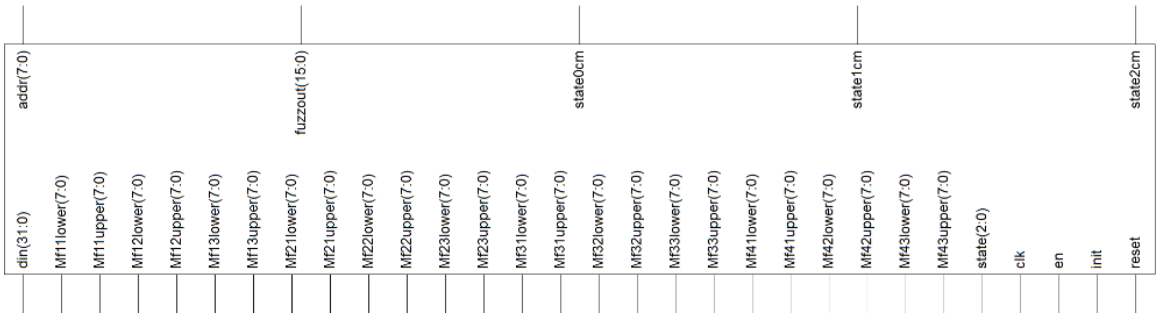

Figure 16: The InfTD Module

3.8 Rule-base extraction:

The rule base extraction is included in this work as without the rule base the fuzzy system is useless. A good rule base is a necessity for the fuzzy system. Let us discuss about the benchmark problem which is simulated. The details of the benchmark problem is discussed in the simulation section. Only to show the necessity of the rule base few information about the problem is discussed here. That is the 4 input 1 output problem. Three membership functions are taken for each input, so basically our system have total 81 rules. But how to find the rule-base which will give us the best possible solution. We will test all the rules one by one. It is good idea but our search space contains 3^{81} elements. If all the computer of the world will search then also we need millions of years to completely search the search space. So it is basically an np-hard problem. There some method to find solutions for these type np-hard problems. Genetic algorithm is one such method to get best possible solution. It may not give us the best solution which is impossible to get instead it will give us a good solution which is almost nearer to the best solution. It is bio-inspired method, inspired by the evolutionary theory. Here first the algorithm is given then the DNA structure will be disclosed. Next all the operations of the genetic algorithms will be discussed.

3.8.1 The GA Algorithm:

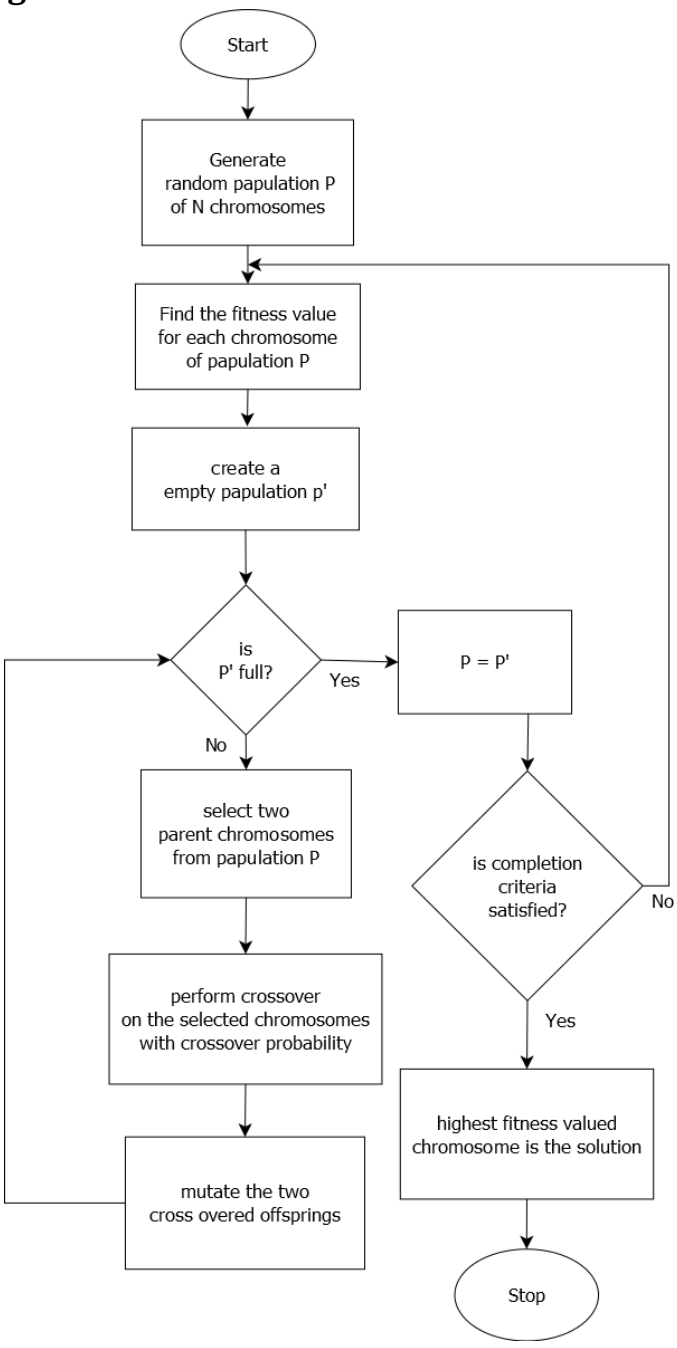

Figure 17: Flow chart for Genetic Algorithm

As it is shown in the flow chart the algorithm demands some chromosomes. We need some chromosome structures for our fuzzy engine. As we are finding the rule base here it is good idea to take the rule-base itself as the chromosome. So our chromosome will look like as follows,

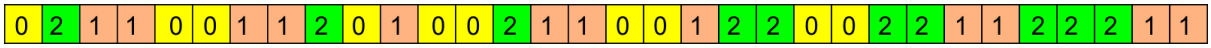

Figure 18: Fuzzy rule base chromosome

As shown in the figure yellow is for 2, brown for 1 and green for 2. Here these numbers indicate the consequence membership function. Green or 2 suggests the 2nd membership function will be considered during inference data generation. Following are the few operation as evident from the algorithm.

3.8.1.1 Selection:

From the pull of population 2 chromosomes are selected on the basis of their fitness value. The selection should be such that with the fitness value probability of selecting that particular chromosome increases. Roulette selection suits our need. This method is the inspiration from the roulette table of casinos.

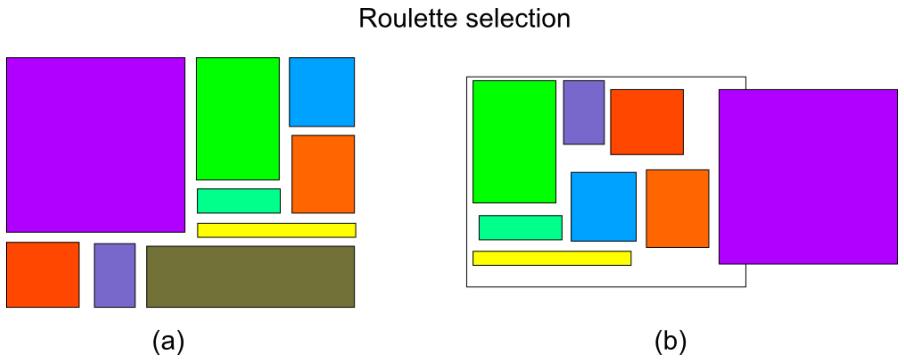

Figure 19: Illustration of roulette selection

Assume that there are coloured boxes as shown in figure 19 (a). If we a throw a ball from outside most probably which box it will be felt. Of course in the purple one or on the big box. How to implement it in the coding. Now assume the coloured boxes as shown in the figure 19 (a) are toys. Purple is the largest toy. Now if we add all the volume of the toys and find a box of that much volume then all the toys can be filled into the box. Let's randomly choose a box lesser in volume and start filling it with the toys. It is obvious that the toys of bigger in size may be remain unfilled, as shown in figure 19 (b). These are out selection. In programming

point of view a random is generated say R with lesser in value from the total of all fitness values. Than randomly chromosomes were selected and subtracted from the randomly generated R. when R value drop below zero selects the current chromosome.

3.8.1.2 Crossover:

Crossover is genetic phenomena. The two selected chromosome were taken and some portions of the chromosomes are exchanged. How much portion are going to be exchanged is known from the cross over probability, in our case we generate a random number which indicate the crossover point. Crossover illustrated,

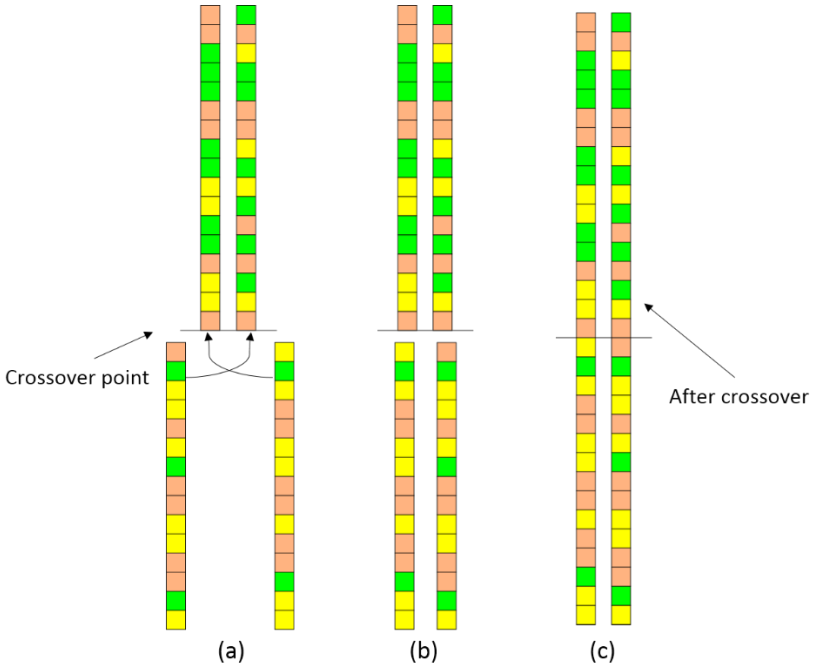

Figure 20: Crossover of fuzzy chromosomes

3.8.1.3 Mutation:

Mutation is the phase where few unrepairable changes were done to the particular genes of the selected chromosomes. How many genes have to be changed is known from the mutation probability in this work it is too randomly generated. The information about the index of the genes to be is changed is also randomly generated. All the random numbers were generated using C rand() function. Mutation illustrated as follows,

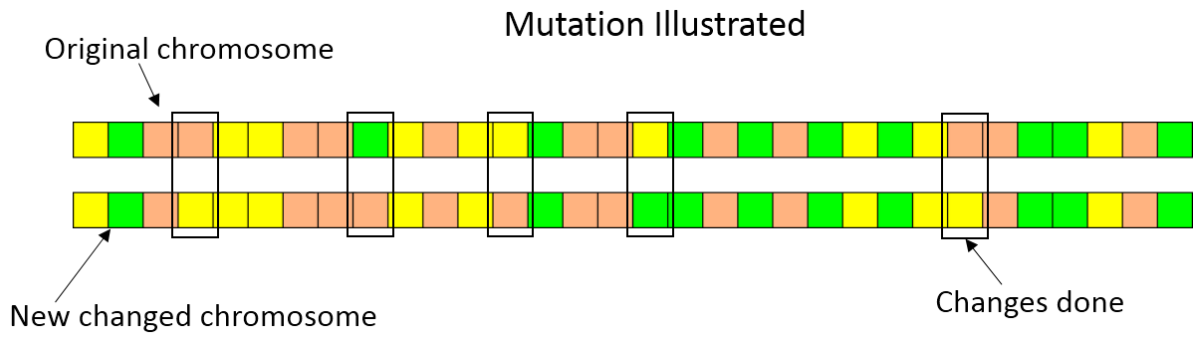

Figure 21: Mutation over fuzzy chromosomes

3.8.1.4 Elitism:

As the chromosome are changed generation by generation. It may happen that best chromosome got by now will be lost in next generation. To save the best chromosomes the best two chromosomes according to their fitness value will transferred to the population without any modification. This concept is called elitism.

The complete view of the genetic processed used here is as illustrated below,

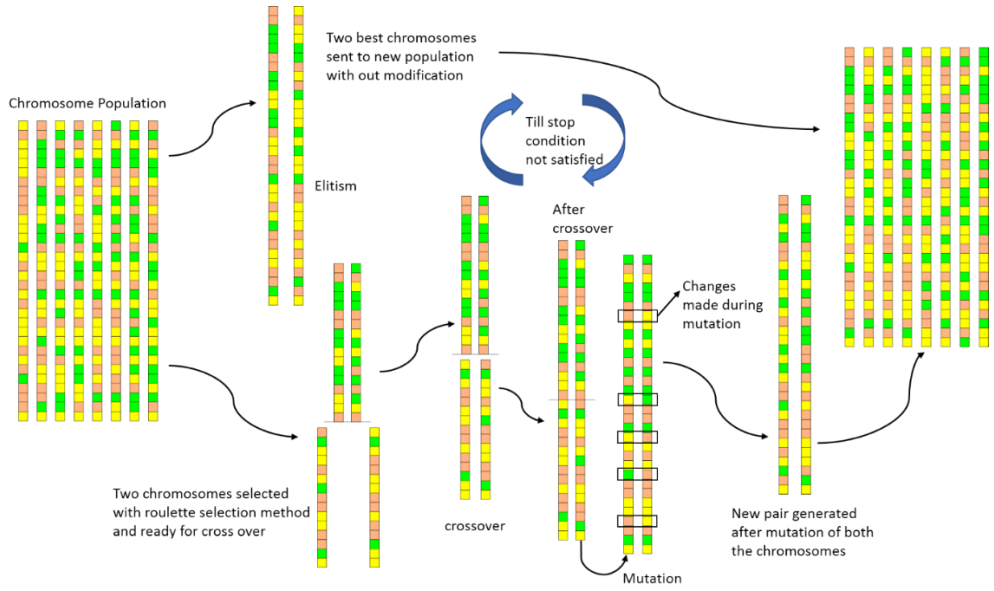

Figure 22: Complete View of Genetic Algorithm

3.9 Control Unit:

Every processor need a control unit. It controls the movement of the data within the system. This is basically a finite state machine (FSM). As name suggests all the states must be finite i.e. as a designer we do not allow unexpected behavior with in a processor. A finite state machine ensure all the state that the processor will go through are finite and our processor will work fine. It is designed in paper and pencil then coded. The FSM is as shown in the figure below.

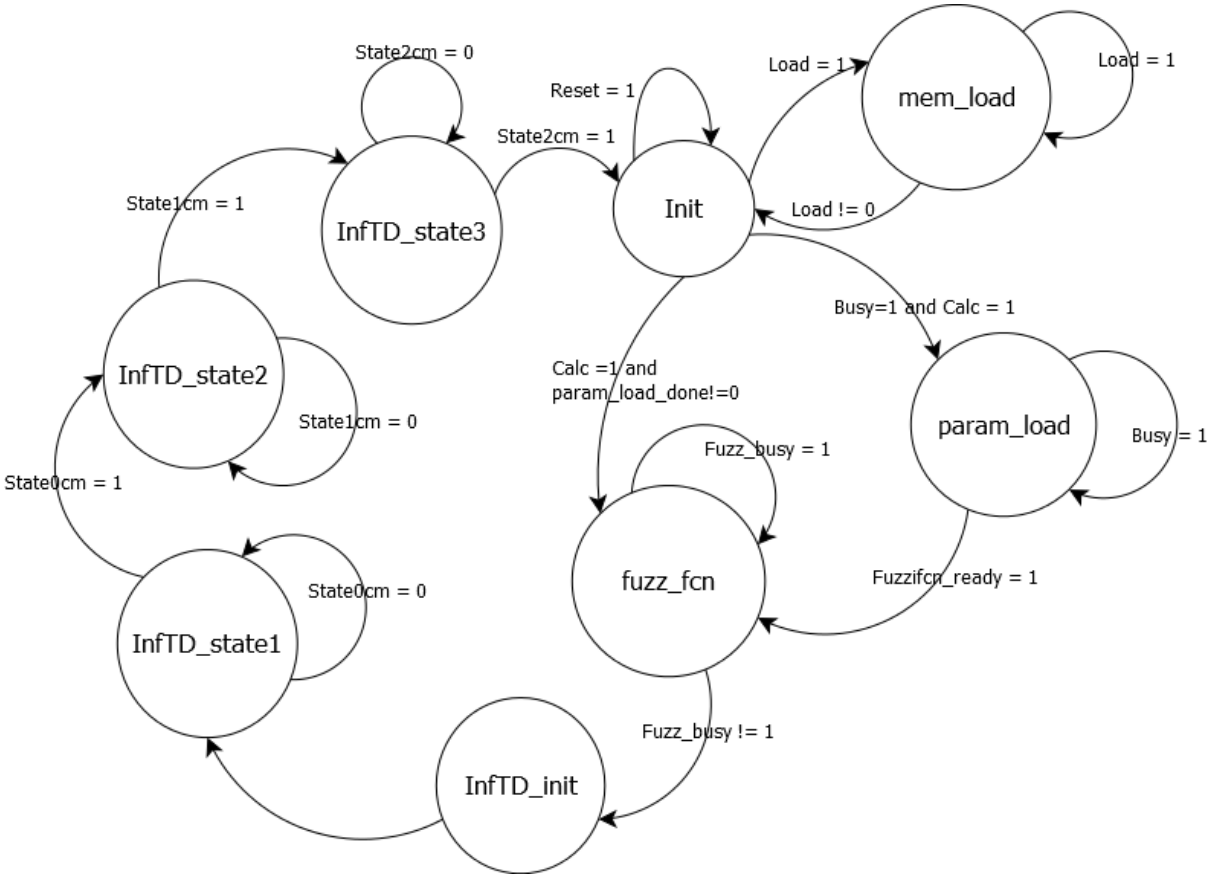

Figure 23: Finite State Machine for Fuzzy Controller

The above shown state machine starts from the initial state ‘Init’. Then proceed as shown in the state diagram. Here reset, calc and load are the input to the fuzzy controller rest are the internal control signal. ‘Init’ state initiate all the internal control signal to initial value. ‘mem_load’ state the get data from outside and save them to internal RAM. ‘param_load’ state load the parameter values for fuzzifier from the RAM. ‘fuzz_fcn’ do the fuzzification and return the fuzzified value. ‘InfTD_init’ initiate the InfTD module. ‘InfTD_state1’, ‘InfTD_state2’, ‘InfTD_state2’ is for controlling the InfTD module. On completion of each task InfTd module produces acknowledgement signals as state0cm, state1cm, and state2cm. Finally the RTL view of the complete fuzzifier is shown in figure 24.

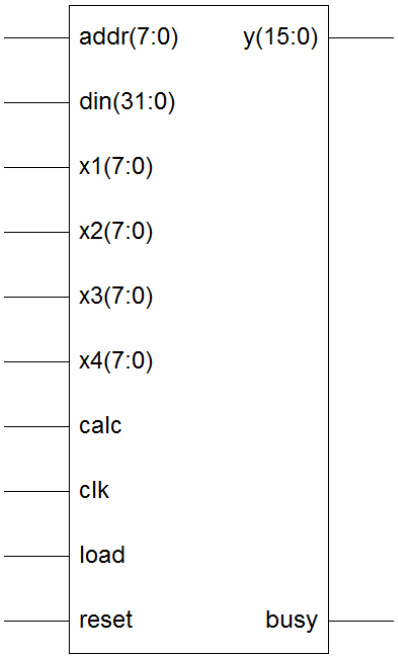

Figure 24: RTL of complete fuzzifier

3.10 The RAM:

As we are using a RAM as it was stated before, we have to decide how RAM will be arranged i.e. on which location what data will be placed. The controller will follow this arrangement to fetch required data for the Fuzzy system. So the memory map for the system is as below.

Table 2: Memory Map for the Fuzzy Controller

Memory Address	**Bit Location**	**Description**
00h	0:7	Input1 membf1 parameter a
	8:15	Input1 membf1 parameter b
	16:23	Input1 membf1 parameter c
	24:31	Input1 membf1 parameter d
01h	0:7	Input1 membf1 parameter e
	8:15	Input1 membf1 parameter f
	16:23	Input1 membf1 parameter g
	24:31	Input1 membf1 parameter h
02h	0:7	Input1 membf1 parameter height
	8:15	Input1 membf1 parameter slope_ae
	16:23	Input1 membf1 parameter slope_bf
	24:31	Input1 membf1 parameter slope_cg
03h	0:7	Input1 membf1 parameter slope_hd
	8:31	reserved
04h	0:7	Input1 membf2 parameter a
	8:15	Input1 membf2 parameter b
…	…	…
30h	0:31	conse 1
31h	0:31	conse 2
32h	0:31	conse 3
33h	0:31	Rule 1
34h	0:31	Rule 2
…	…	…
83h	0:31	Rule 81

Chapter 4

4 Simulation & Implementation:

For the ease of simulation a Simulator is designed in C. It can do fuzzification, inference, type reduction with KM algorithm, Type reduction with WU-Mendel algorithm, GA based learning and it can also generate graphs to see the data. For plotting the data GNU Plot is used here. The GNU Plot program can be invoked with in C with the help of operating system pipes. A snap shot of the simulator,

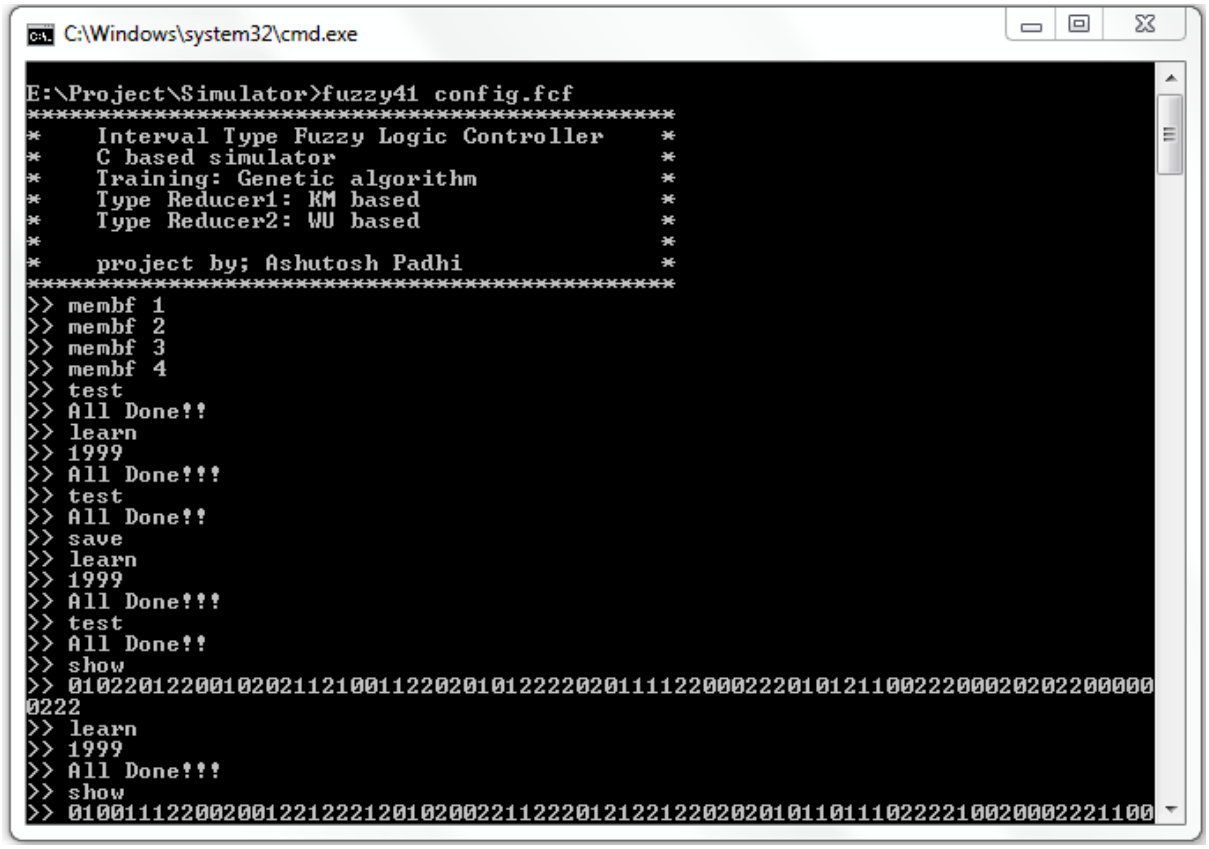

Figure 25: C type 2 Fuzzy Simulator

4.1 Benchmark problem:

The system is tested with a benchmark problem and a simple heat control problem to verify its functionality. IRIS bench mark problem is selected for testing purpose. IRIS is a multivariate dataset. It is introduced by Sir Ronald Fisher in 1963. The data set contains a 50 samples each

of 3 IRIS flowers named Iris setosa, Iris verginica and Iris versicolor. Four parameter of these flowers as sepal length, sepal width, petal length and petal width, with the corresponding flower name is in the data set. This dataset is a good starting point for simulation. A four input one output fuzzy system can be designed which will take the inputs as flower parameters and in output it will give us the flower name. From the data set it is sure that the output can only be one of the three different kinds of flower. So our consequence matrix or the $\underline{y}_i, \overline{y}_i$ values as discussed earlier in the type reducer part will be as given below.

Table 3: Consequence matrix

Consequence	$\underline{y}_i$	$\overline{y}_i$
0	0	0
1	1	1
2	2	2

The $\underline{y}_i, \overline{y}_i$ values are reconfigurable i.e. they can be changed to modify the system behavior. Membership function parameter for the particular Problem is as given below.

Table 4: Membership Function Parameters

MembF	***a***	***b***	***c***	***d***	***e***	***f***	***g***	***h***	***height***
11	3.45	3.85	5.45	5.85	4.25	4.45	4.85	5.05	0.75
12	4.80	5.20	6.80	7.20	5.60	5.80	6.20	6.40	0.75
13	6.17	6.57	8.17	8.57	6.97	7.17	7.57	7.77	0.75
21	0.04	0.64	3.04	3.64	1.24	1.54	2.14	2.44	0.75
22	2.20	2.80	5.20	5.80	3.40	3.70	4.30	4.60	0.75
23	4.28	4.88	7.28	7.88	5.48	5.78	6.38	6.68	0.75
31	0.00	0.59	2.99	3.59	1.19	1.49	2.09	2.39	0.75
32	2.20	2.80	5.20	5.80	3.40	3.70	4.30	4.60	0.75
33	4.40	5.00	7.40	8.00	5.60	5.90	6.50	6.80	0.75
41	-0.48	-0.18	1.10	1.31	0.11	0.26	0.56	0.71	0.75
42	0.60	0.90	2.10	2.40	1.20	1.35	1.65	1.80	0.75
43	1.71	2.01	3.21	3.51	2.31	2.46	2.76	2.91	0.75

These parameter values draw the virtual membership function inside our fuzzy system. Initially these parameter were decided by examining the test data. But it is hard and time consuming so

a GUI is developed for this purpose, which is going to be discussed later in this section. These parameters are discussed in the design section.

4.2 Simulator:

A snapshot of the simulator is shown in the figure 25. It has six commands as '*membf*', '*algo'*, '*show'*, '*learn'*, '*test*' and '*update'*. Command reference are listed the following table. The simulator demands one argument the '*config.fcf'* file. This *'config.fcf'* file will be generated from matlab GUI. Working with simulator can be understood from the snap shot as per figure 25.

Table 5: command reference for Fuzzy41 simulator

Command	**Attribute**	**Function**
update	No attributes	Update the internal variables from the new config.fcf file
membf	Input number as: 1, 2, 3, 4	Plot the membership functions for corresponding input
algo	1: Wu-Mendel algorithm 2: KM algorithm	Selects the particular algorithm for simulation
learn	No attributes	Learn using GA
show	No attributes	Show the current chromosome
test	1: plot only the test data 2: plot only the train data 3: plot all the IRIS data	For testing the correctness of the Fuzzy System.

4.3 Matlab GUI:

A matlab GUI is developed one for generating the *'config.fcf'*. The GUI gives the user to model the membership function and change them easily. Matlab GUI as shown below.

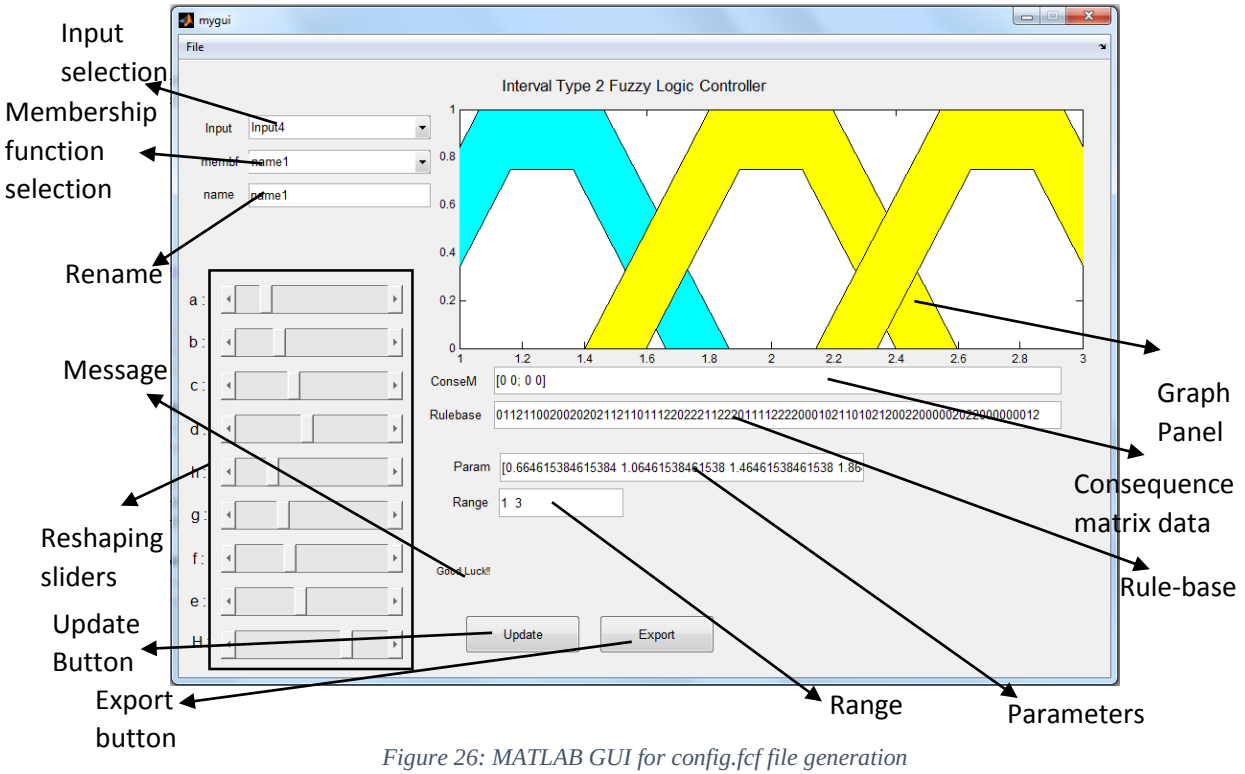

Figure 26: MATLAB GUI for config.fcf file generation

Input and membf which stands for membership function will be selected first. By doing so their membership functions will be displayed in the graph panel. The sky blue one as shown in the graph panel is the selected one. Before changing anything first the range has to be set first. By default the membership functions have been drawn with range 0 to 10. By changing the range the membership functions will be redrawn. Now the position of the selected membership function, the sky blue one, can be set by dragging it. Only selected membf is draggable. After seting its new position update button have to be pressed. The update button will update the current states or one can say it saves the changes. The selected membership function further can be reshaped with the help of reshaping sliders a, b, c, d, e, f, g, h and height. Name of current membership function i.e. its linguistic name can be modified with the rename field. Consequence matrix data, rule base data must be specified before exporting. The consequence matrix data is given in the table 2 for IRIS problem. Export button will generate the *'config.fcf'* file which will be further used to simulate the behavior of the fuzzy system. If the require fields are empty on exporting an error message will be displayed at the message area.

Before directly jumping to the benchmark problem let us simulate a much simple controller. It is temperature controller.

4.4 Simulation of Heat Controller:

Here the target is to control the temperature of a heat element with the help of fuzzy logic controller. We have a reference temperature say 15 degree Celsius. The temperature of the heat element should not be above the reference. By being realistic, little bit error is allowed. Here is the block diagram of the complete system.

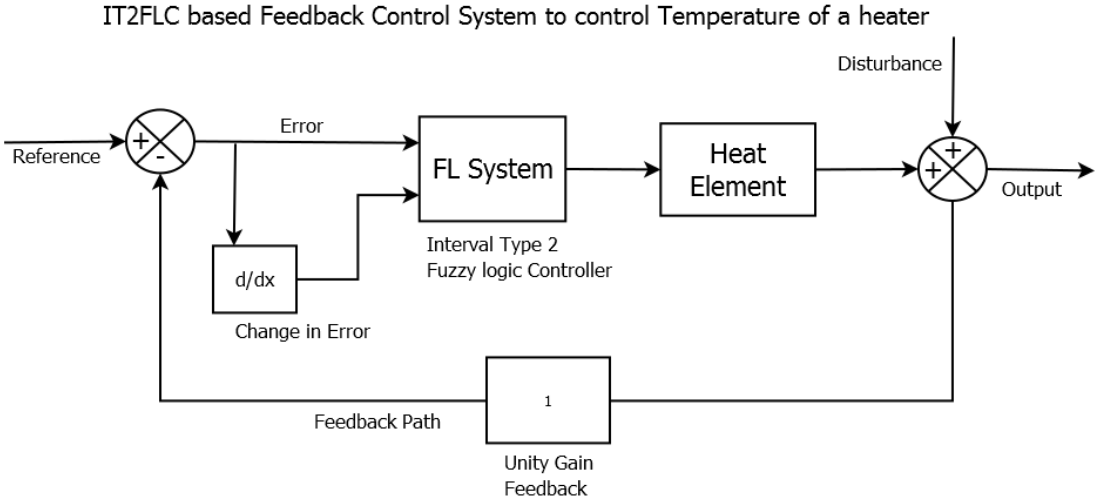

Figure 27: Temperature controller using the Interval type 2 Fuzzy System

It is a simple feedback control system with unity gain feedback. Reference point is set at 15 degree Celsius. The fuzzy logic system is a 4 input 1 output fuzzy system. The fuzzy system takes two input as error and change in error and produces control signal to control the temperature of heat element. Disturbance is added at the output terminal, now as the disturbance try to change the temperature of the system the fuzzy controller must keep it by generating suitable control signal. The unit is tested by giving step input as disturbance. The result for both the KM algorithm and Wu-Mendel algorithm is as shown below.

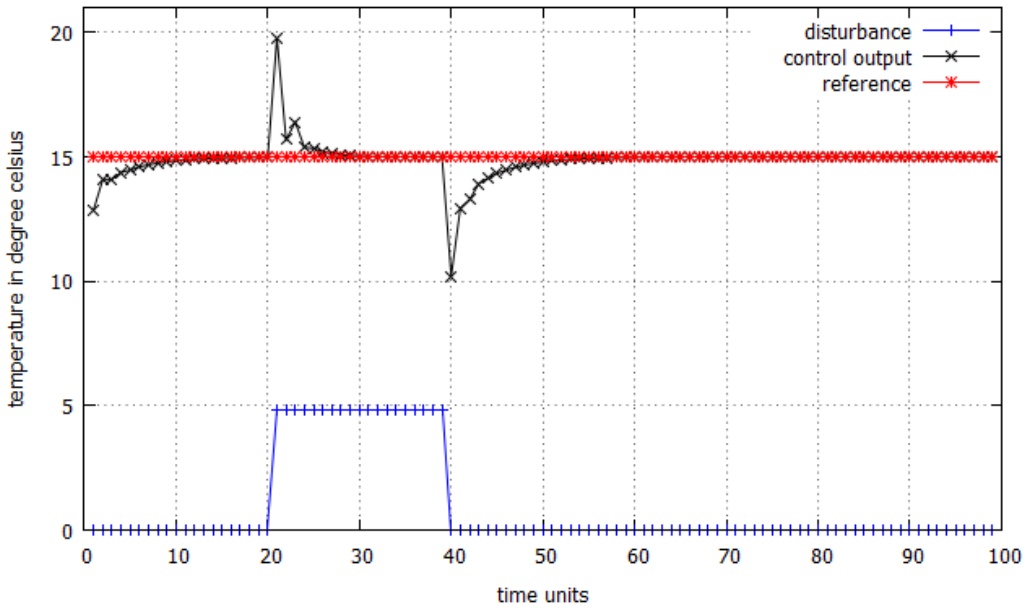

Figure 28: Simulation of temperature controller with step disturbance (KM algorithm)

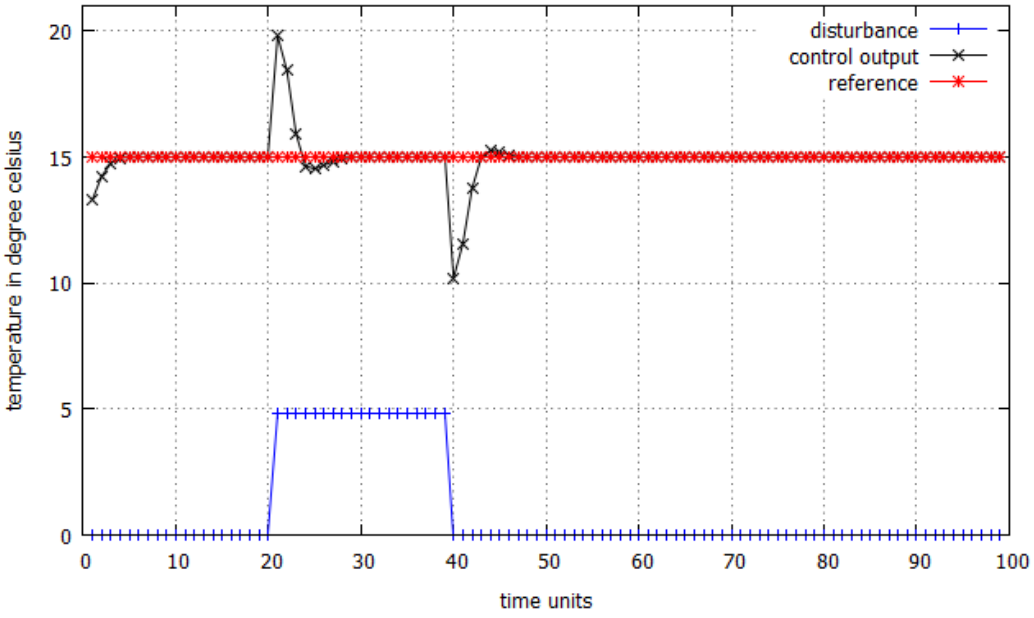

Figure 29: Simulation of temperature controller with step disturbance (Wu-Mendel algorithm)

From the graph both the algorithm give us approximately same result. So Wu-Mendel algorithm may not give us the accurate result, but it saves our costly silicon space and result is in front of us. From the point of view of the controller's correctness, from the graph it is self-explanatory. The algorithm is working fine.

4.5 IRIS Problem Simulation:

The problem is already discussed above. We know that it is 4 input 1output system. Previously we are dealing with a 2 input 1 output system. Number of input and number output never changes the overall algorithm. If number input is more at inference stage we need to find the minimum of more number of inputs and if number of output is more, then our system have output number of type reducer, one per each output. The thing that is going to be changed is the control unit of the processor i.e. the finite state machine. The block diagram for the complete system is as below.

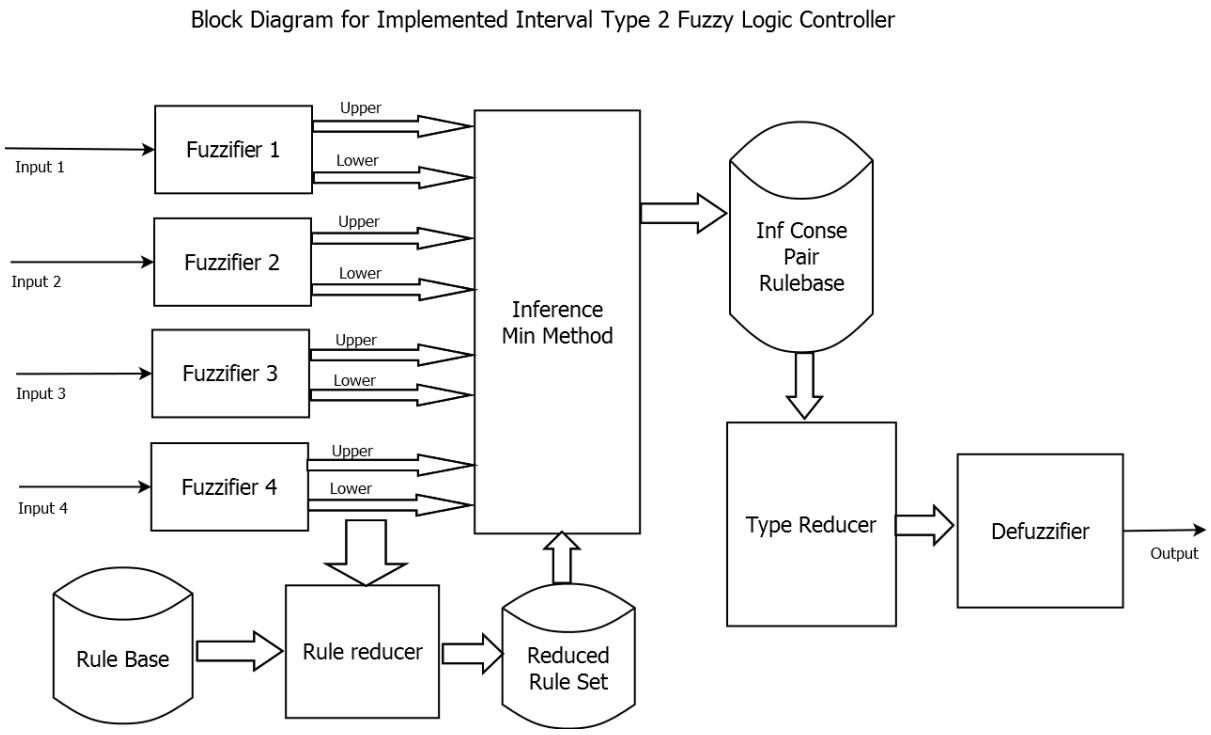

Figure 30: Block Diagram for IRIS simulation of Interval Type 2 Fuzzy Logic Controller

All the resultin graph generated from the C simulator is as below. All the membership function is ploted first.

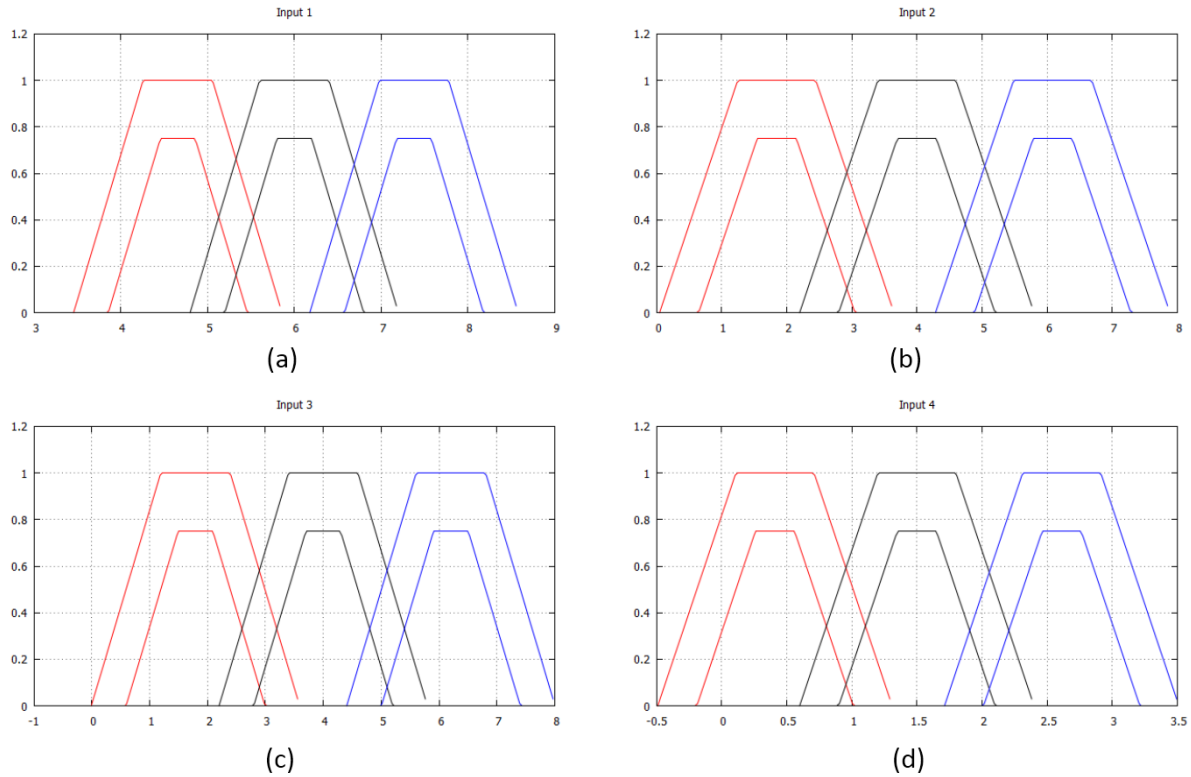

Figure 31: Membership function for IRIS problem

The membership functions are shown for particular testing result, they further can be changed and simulated again with the designed C simulator. After training the training data plotting of the test data is as below.

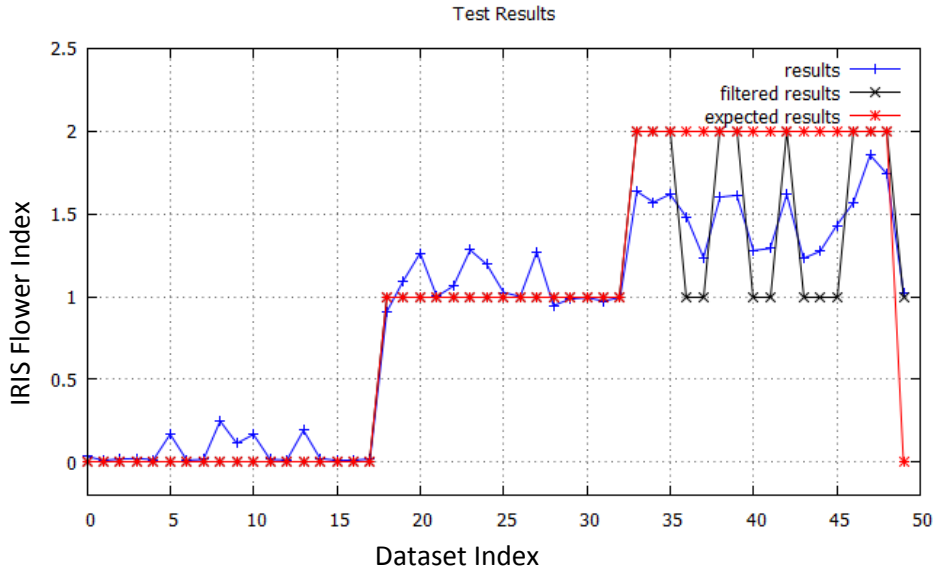

Figure 32: Testing results for IRIS problem (only test data)

From the graph the red line is expected result, blue line is the result and black line is filtered results. One can mark for 70% of data or until x index 32 we are getting all must expected result. But on x index 32 to 50 range we have little bit error. The error is indicating that the learning method should be improved to get perfect result. But as it is sure that the fuzzy system is working fine and we have success in designing it. The plot of total IRIS data as below.

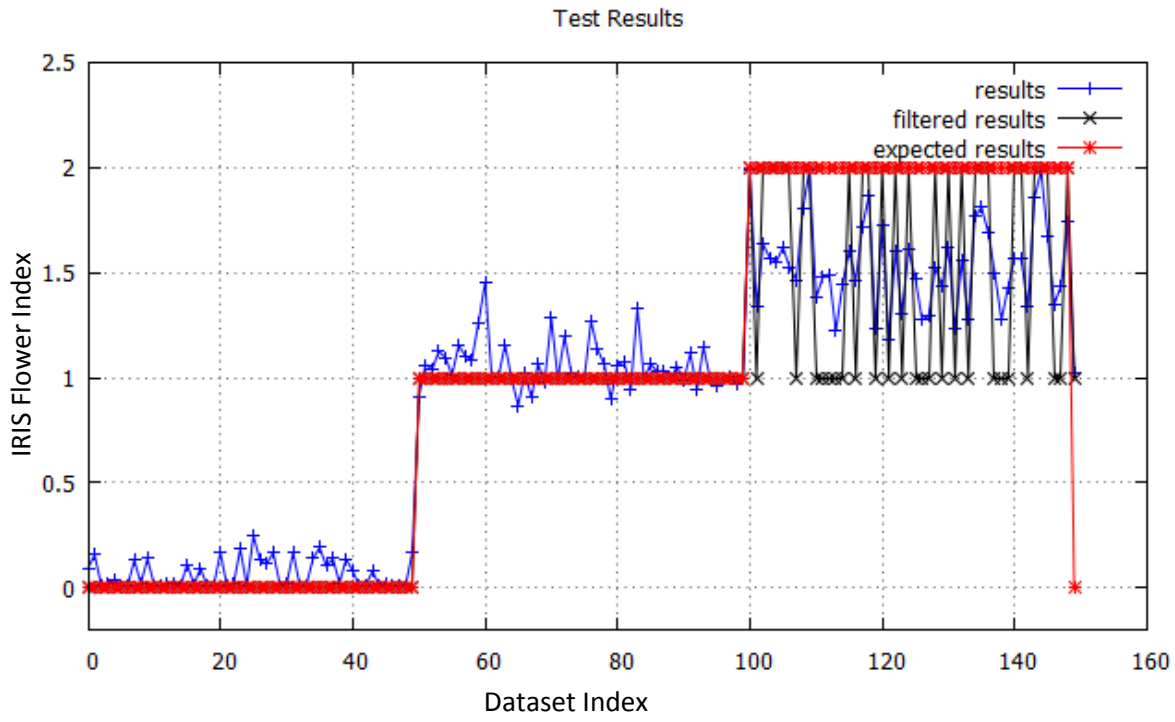

Figure 33: Plot of fuzzy simulation data for complete IRIS data set

And finally behavioral simulation of the fuzzifier as given below.

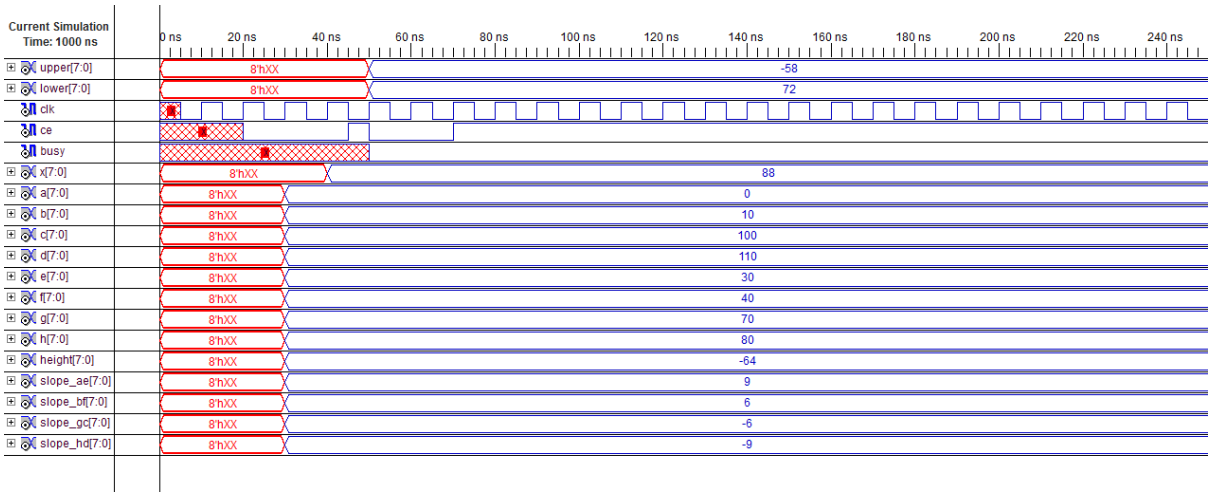

Figure 34: behavioral simulation of the fuzzifier module

Chapter 5

5 Conclusion:

Finally the very expected Interval Type 2 Fuzzy Controller is ready. Clear-cut idea about the interval type 2 fuzzy controller has come from chapter 2. The designed procedure is described step by step in chapter 3. We finally simulate our design and got out expected results by doing simulation and implementation of the work. The complete design procedure is given very priority in this work. The procedure is itself very important from the result, as it gave us much understanding of the underlying concepts.

The study of the two algorithm implies the simplicity of Wu-Mendel closed form method than the KM iterative procedure for the type reduction purpose. The Graphs of the heat controller shows the control behavior of the two algorithms and they are almost identical. So Wu-Mendel closed form method is used here for hardware realization. To reduce the complexity trapezoidal membership function is chosen for fuzzification.

Finding a proper rule-base is very important as important as the fuzzy system itself. Here the approach was Genetic Algorithm. The GA gave a suitable candidate for rule-base searching from the huge search space (NP Hard Problem). Further the Waveform of a fuzzifier is given at final chapter.

So the complete system starting from the simulator, fuzzifier, defuzzifier, type-reducer, rule reducer, GUI are designed with the goal of ease of use, keeping in mind that it can be improved further in future.

6 Bibliography

[1] N. N. Karnik and J. M. Mendel, "Introduction to Type-2 Fuzzy Logic Systems," *IEEE,* 1998.

[2] N. N. Karnik and J. M. Mendel, "Type-2 Fuzzy Logic Systems," *IEEE,* 1999.

[3] J. M. Mendel, "Fuzzy Logic Systems for Engineering: A Tutorial," *IEEE,* 1995.

[4] J. M. Mendel and R. I. B. John, "Type-2 Fuzzy Sets Made Simple," *IEEE,* 2012.

[5] N. N. Kranik and J. M. Mendel, "Type 2 Fuzzy Logic Systems: Type-Reduction," *IEEE,* 1998.

[6] D. Wu and J. M. Mendel, "Enhanced Karnik-Mendel Algorithms," *IEEE,* 2009.

[7] M. A. Melgarejo R. and C. A. Pena-Reyes, "Hardware Architecture and FPGA Implementation of a Type-2 Fuzzy System," *ACM,* 2004.

[8] H. Wu and J. M. Mendel, "Uncertainty Bounds and Their Use in the Design of Interval Type-2 Fuzzy Logic Systems," *IEEE,* 2002.

[9] O. Castillo and P. Melin, Recent Advances in Interval Type-2 Fuzzy Systems, Springer, 2012.

[10] Q. Liang and Jerry M. Mendel, "Interval Type-2 Fuzzy Logic Systems: Theory and Design," *IEEE,* 2000.

[11] D. Wu and J. M. Mendel, "Designing Practical Interval Type-2 Fuzzy Logic Systems Made Simple," 2014.

[12] D. Wu, "A Brief Tutorial on Interval Type-2 Fuzzy Sets and Systems," 2014. [Online]. Available: https://sites.google.com/site/drwu09/home.

[13] Y. Maldonado and O. Castillo, "Genetic Design of an Interval Type-2 Fuzzy Controller for Velocity Regulation in a DC Motor," *International Journal of Advanced Robotic Systems,* 2012.

[14] S. D. Kaehler, "Fuzzy Logic - An Introduction," March 1998. [Online]. Available: http://www.seattlerobotics.org/encoder/mar98/fuz/fl_part1.html.

[15] V. K. Dhar, A. K. Tickoo, R. Koul and B. P. Dubey, "Comparative performance of some popular artificial neural network algorithms on benchmark and function approximation problems," *Pramana- journal of physics,* 2010.

[16] M. Obitko, "Genetic Algorithms," 1998. [Online]. Available: http://www.obitko.com/tutorials/genetic-algorithms/index.php.

[17] Community, "Iris flower data set," Wikipedia, [Online]. Available: http://en.wikipedia.org/wiki/Iris_flower_data_set.

[18] "Block Memory Generator v2.7," Xilinx Logicore.

[19] "Divider v1.0," Xilinx Logicore.

[20] J. Bhaskar, A Verilog HDL Primer, BS Publications, 2013.

[21] S. Srivastava and D. Srivastava, C in Depth, BPB Publication, 2011.