Fuzzy Logic Control with the Intel 8XC196 Embedded Microcontroller

Navin Govind
Senior Systems Engineer
Intel Corporation
Chandler, AZ

Abstract: Fuzzy logic control is being increasingly applied to solve control problems in areas where system complexity, development time and cost are the major issues. In the absence of a system mathematical model , a fuzzy system model is described which is analogous to a human operator's behavior, based on approximate reasoning bound by a minimum set of rules. A set of linguistic fuzzy control rules are set up which are conditional linguistic statements which establish the relationships between the inputs and the outputs. The fuzzy system is associated with Binary Input Output Fuzzy Associative Memories which are used for control. This paper discusses the development of a fuzzy inference unit and algorithms for fuzzification, rule evaluation and defuzzification of a fuzzy closed loop control system. Tools and techniques to generate optimized fuzzy based real time code in assembly and C, with short development time, are shown for the Intel 8XC196 microcontroller. Performance and features of the 80C196 for fuzzy-based control are analyzed.

INTRODUCTION TO FUZZY LOGIC

Fuzzy logic is being increasingly used over a wide range of areas such as industrial control, image processing, auto industry as well as commercial products and has become one of the most popular and successful methods used to design and implement complex control systems.

Classical controllers are designed by various techniques for a variety of control systems applications and are modeled on the systems or process being controlled. Fuzzy control is based on the human operator's behavior. Control rules in the case of an anti-lock braking system (ABS) of an automobile may include variables such as the car speed, brake pressure, brake temperature, time of brake application and many other variables which are continuous in nature and the range of these variables are set by the designer. Fuzzy logic easily represents continuous data in digital computers. The linguistic variable, speed may adopt values such as small, positive small, large, negative large and so on. By applying an **IF-THEN** rule fuzzy theory is applied to real time control i.e.; "***if*** *speed is slow* ***then*** *brake pressure is negative small*", hence a single fuzzy rule replaces conventional rules. Fuzzy logic is used to build a controller even when a mathematical model of a system does not exist or is incomplete and creates a control surface by combining rules and fuzzy sets. Fuzzy Associative Memories (FAM) are transformations which map fuzzy sets to fuzzy sets. A FAM system maps antecedents to consequents and is a collection of IF-THEN rules. The universes of discourse for the fuzzy controller are pre-defined and each universe of discourse contains a library of fuzzy sets. Each fuzzy set allows its members to have different grades of membership, each expressed by a number in the interval [0,1], adjacent sets in the universe of discourse overlap. A weighted quantization of antecedent X and the consequent Y are sets in the library which includes the fuzzy set value that the fuzzy variable can have. Defuzzification schemes in the form of centroid defuzzification are used to directly compute the real

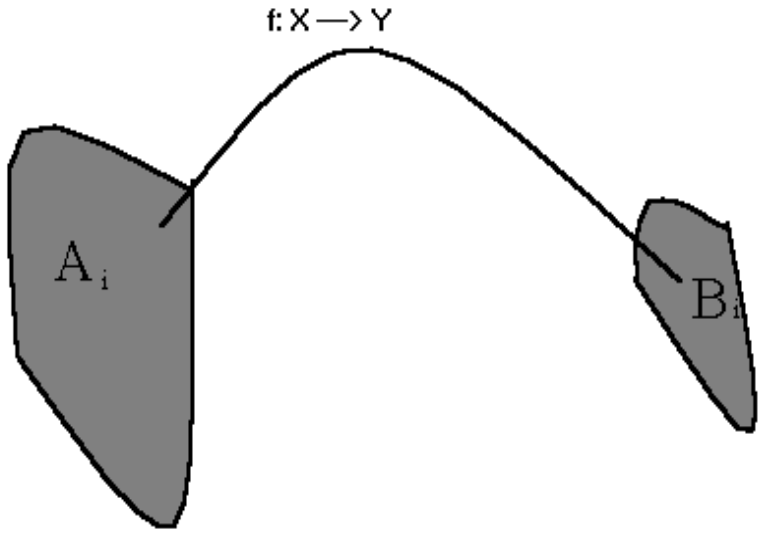

Fig 1. Mapping of Fuzzy Subsets A_i of X to B_i of Y. (A_i, B_i) is a fuzzy association which represents a system structure

valued output as a normalized combination of fit values with respect to the output space. The behavior of fuzzy systems can be associated with associative memories and are model free estimators. Linguistic rules are used to implement a fuzzy system and fuzzy systems require that a linguistic rule matrix be partially filled.

Fuzzy systems learn by samples, therefore given samples (A_i,B_i) a fuzzy system can estimate functions given the above fuzzy set samples. The number of FAM rules are 'm'. Given fuzzy set samples (A_i,B_i), the term A_i is defined as an antecedent term or input and B_i is defined as consequent term or output in relation to fuzzy association memory. This is analogous to an IF-THEN rule which represents mapping partial input space to partial output space. Consider the example where motor speed control is being addressed "*if speed is **slow** then voltage is **increased***", the fuzzy association here is (slow, increased), the antecedent term being **slow** for the fuzzy input variable and the consequent term being **increased** for the fuzzy output variable. In a FAM matrix each entry is a defined input output transformation and numerically structured input data is numerically processed. In real world fuzzy systems the inputs activate the FAM rule antecedents, to a varying degree, on which the output variables depend. The encoding technique used in encoding linguistic rules of a fuzzy system into a infinite dimensional dedicated numerical matrix is by **fuzzy Hebb procedure.**

In general a FAM system consists of a FAM bank of different FAM associations where each association corresponds to a different entry in a linguistic FAM bank or a different structured FAM

matrix. The matrices are stored separately and parallel accessed to avoid crosstalk, also, due to binary input-output FAMs, parallel access does not impose a heavy computational burden.

For example, A and B are two finite fuzzy sets represented by structured fit vectors

$A = (a_1, a_2, a_3, ...a_n)$
$B = (b_1, b_2, b_3, ...b_p)$

A and B define membership functions m_A and m_B and are fuzzy subsets of X and Y, the two membership functions m_A and m_B map x_i of X and y_i of Y to different degrees of membership in [0,1]. The state variables of a system are related to output variables and the value of the output variable is determined by the above relationship. Linguistic rules are defined for the variables which are fuzzy in nature and are expressed as fuzzy subsets of the universes. A fuzzy subset F of universe of discourse A has a membership function m: $A \rightarrow [0, 1]$ which determines the degree of membership of m(a) in [0,1]. Three operators that operate on fuzzy sets A, B of A are:

$m_{A \cap B} = \min(m_A, m_B)$: pairwise minimum
$m_{A \cup B} = \max(m_A, m_B)$: pairwise maximum
$m_{A'} = 1 - m_A$: complementation or order reversal of A

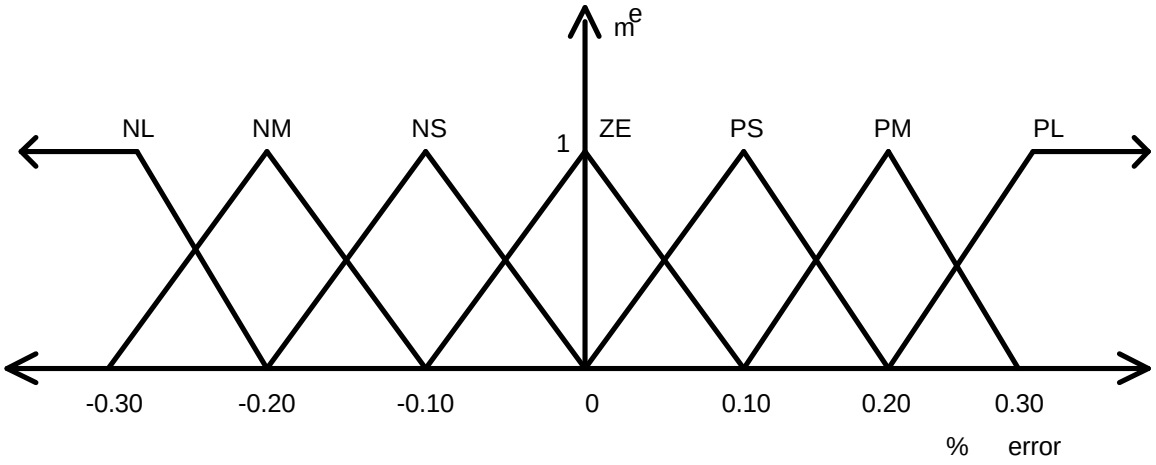

Fig 2. Membership function for percent error

Fuzzy variables used here are quantized as Positve Small, Positive Medium, Positive Large, Zero, Negative Small, Negative Medium and Negative Large. IF-THEN control rules are implemented using fuzzy conditional statements. Using compositional rule of inference the output or control action is derived. Fundamentals of the fuzzy subset theory are used to interpret the linguistic rules and to arrive at a deterministic action point after the decision making process. BIOFAMs are used to control the fuzzy variables. Each universe of discourse is quantized to a small interval centered about zero. Fig 2. represents membership functions for a fuzzy variable "error."

8XC196 BASED FUZZY CONTROLLER

With microcontrollers being available in different configurations, low cost, small die sizes, power management features and high clock rates, it is reasonable to apply fuzzy logic in an inexpensive microcontroller to implement the fuzzy control loop. The Intel 8XC196 microcontroller can be used to implement a closed loop fuzzy control system. Typical applications of the 8XC196

microcontrollers include closed-loop control and signal processing applications such as motor control, hard disk drives, ABS and medical instrumentation. The powerful instruction set makes use of the register-register based architecture with various addressing modes and a rich set of peripherals. Data acquisition and processing is done easily and efficiently.

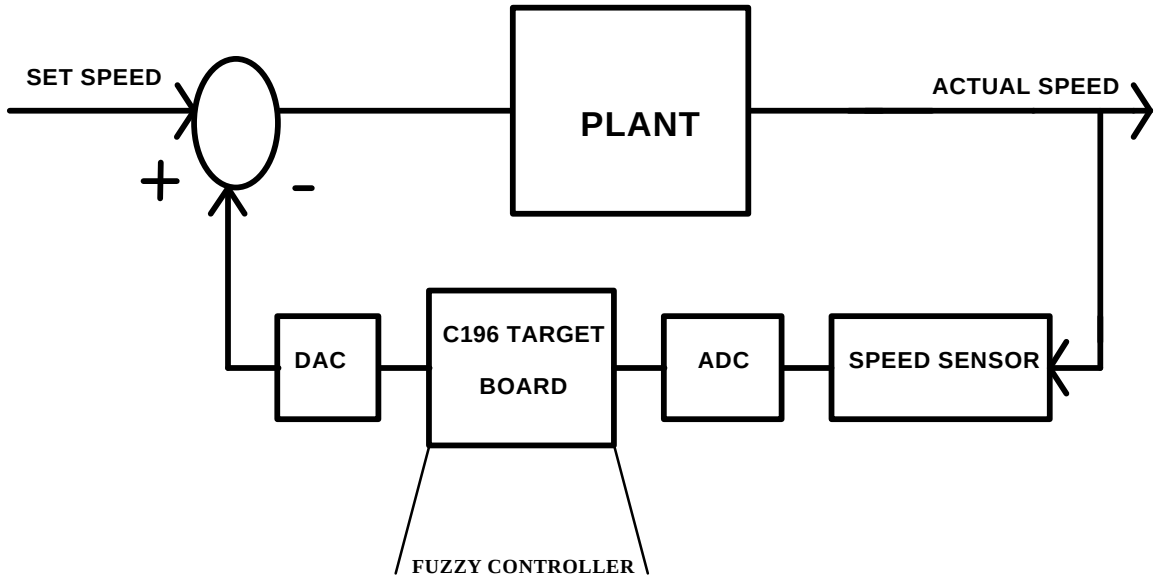

Fig 3. 8XC196 Closed Loop Fuzzy Controller

Software can be implemented with minimum memory and fast execution (upto 50MHz) by locating the code/data in internal chip memory. The convenience of using tools to expedite system design and reduce or eliminate time consuming coding makes the 8XC196 a viable choice.

A converter circuit designed for a fuzzy based dc motor control in the form of a single phase two pulse bridge converter with two SCRs and two diodes connected in bridge type is an example. In the rectifying mode the converter controls the mean output dc voltage from a maximum positive corresponding to 90 degrees to zero. The maximum voltage applied to each thyristor is the applied maximum voltage. The speed of the dc motor is set to a value which is denoted by set_speed. The actual speed is detected by the speed sensor and the algorithm computes the real speed which is denoted by actual_speed. The ***error*** between the set_speed and the actual_speed is the first fuzzy state variable. The error value is determined by the difference between the actual_speed and the set_speed. Fig 3. shows a setup of the fuzzy controller system with the target board implementing the fuzzification, rule evaluation and defuzzification algorithms.

FUZZIFICATION

The value of error is defined as positive value if actual speed is greater than set speed and negative if actual speed is less than set speed. To determine the actual rate of change of error, the value obtained by the difference of the second difference of error and the first difference of error is computed. This value is the second fuzzy state variable and is called ***rt_spd_chng*** which refers to the rate of speed change. The manipulated variable or the control variable called ***gate_output*** is manipulated so that if the actual speed is greater than the set speed gate_output has a value where the firing angle is greater than 60 degrees. When the actual speed is less than

the set speed, the firing angle is less than 60 degrees but greater than 15 degrees to maintain the speed at a constant level. These conditions are arrived at after observing the nature of the system by an human operator's point of view and using basic human reasoning to adjust the fuzzy controller behavior.

Fig 4. describes the output waveform for a firing angle set at 90 degrees. The entire cycle has a period of 16.66 msecs and the effective range of control for the dc motor is for 4.16 msecs. Therefore controlling the firing angle of the SCR bridge between 90 -15 degrees will control the output voltage to a desired level.

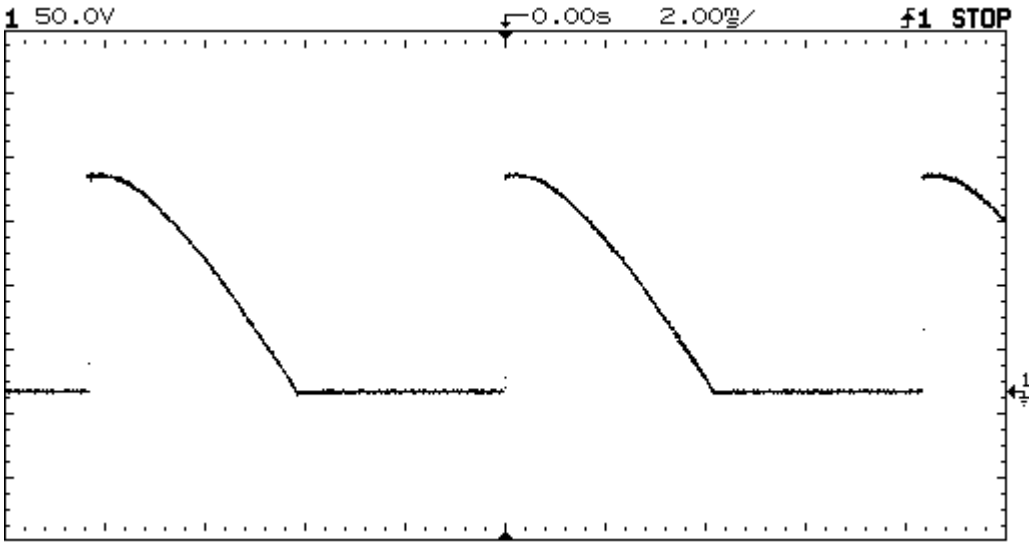

Fig 4. The output waveform for two thysistors firing at a delay angle of 90 degrees

Each compositional rule of inference involves the three fuzzy variables, error, rt_spd_chng and gate_output. For effective speed control the FAM bank is a 7x7 matrix with 343 possible two input FAM rules. The fuzzy variables are quantized to a magnitude of small, medium and large. The quantization of these variables leads to a 7x7 FAM matrix.

7X7 FAM Matrix

	NEG_L	NEG_M	NEG_S	ZERO	POS_S	POS_M	POS_L
NEG_L	**POS_L**	**POS_L**	**POS_L**	**POS_L**	**POS_M**	**POS_S**	**ZERO**
NEG_M	**POS_L**	**POS_L**	**POS_M**	**POS_M**	**POS_S**	**ZERO**	**NEG_S**
NEG_S	**POS_L**	**POS_M**	**POS_S**	**POS_S**	**NEG_S**	**NEG_M**	**NEG_L**
ZERO	**POS_L**	**POS_M**	**POS_S**	**ZERO**	**NEG_S**	**NEG_M**	**NEG_L**
POS_S	**POS_L**	**POS_M**	**POS_S**	**NEG_S**	**NEG_S**	**NEG_M**	**NEG_L**
POS_M	**POS_L**	**ZERO**	**NEG_S**	**NEG_M**	**NEG_M**	**NEG_L**	**NEG_L**
POS_L	**ZERO**	**NEG_S**	**NEG_M**	**NEG_L**	**NEG_M**	**NEG_L**	**NEG_L**

The universe of discourse for the three fuzzy variables are centered about zero and restricted to a small interval. For the three fuzzy variables in this study the real line R is the universe of discourse. Quantization of the three fuzzy variables corresponding to the three universes of discourse accomplished by five overlapping fuzzy sets with the magnitude of the fuzzy variables defined coarsely. The elements of the base matrix base_FAM contain clock ticks. These clock

ticks determine the firing angle of the SCR's. For a small overlap value of sets, there will be excessive undershoot and overshoot due to control assuming the form of bivalent control. For excessive overlap, the distinction between each fuzzy set values get blurred, therefore the default value of 25 percent overlap is chosen.

The matrix columns are indexed by the seven fuzzy sets quantizing the fuzzy variable error. The rows are indexed by the seven fuzzy sets that quantize the rate of change of speed (rt_chng_spd). For every pair of error indexed by columns and rt_chng_spd there is precisely one value of output voltage, which explains FAM rule as a mapping. The first FAM rule that is chosen is the steady state FAM rule: (ZE, ZE, ZE), which states that no action is taken when the state of the system is in equilibrium. The 49 entries explain the 343 possible two input FAM rules. In the equilibrium state when error is zero, the rate of change of speed is zero, speed of the motor remains steady at the set value, this would include the steady-state FAM rule (ZE,ZE; ZE). If error is zero but there is a difference between set speed and sensed speed, the value of the rate of change of speed is positive then the manipulated variable has to be negative to compensate. If the rate of change of speed is negative then the manipulated variable or control variable is positive, therefore the fourth column of the base_FAM matrix which corresponds to error=ZE equals the ordinal inverse of the rate of change of speed column values and also, includes the steady state FAM rule. When the value for error is negative and rate of change of speed is positive, the control variable has positive values with FAM rule (NS, PS; PS), symmetrically when the error is positive and the rate of change of speed is negative, the control variable has negative values represented by the FAM rule (PS, NS; NS). The rest of the FAM bank matrix is filled using this approach.

RULE EVALUATION

The algorithm shows rule evaluation control flow. Initialization is required to clear registers that store the consequent value before rule evaluation. Loops compare the antecedent value depending on the rule being evaluated in a repeated fashion until all rules are evaluated. Pointers point to values where labels are stored in appropriate registers.

Begin
    Initialize pointers and loop counters
For error less than ZERO **do**:
    check for error value in region NEGATIVE SMALL
For error in NEGATIVE SMALL **do**:
    Assign error value to ZERO and NEGATIVE SMALL
For error value not in NEGATIVE SMALL **do**:
    Assign error value to region ZERO
End
Begin
For rate of speed change less than ZERO **do**:
    check rate of speed change value in region NEGATIVE SMALL
For error in NEGATIVE SMALL **do**:
    Assign rate of speed change value to ZERO and NEGATIVE SMALL
For rate of speed change value not in NEGATIVE SMALL **do**:
    Assign rate of speed change value to region ZERO
End

Similarly this Rule evaluation and error region assignment would follow towards the left end of the x-axis to NEGATIVE LARGE. The else statements would be for the false conditions and would assign error and rate of change of speed values if greater than ZERO to the POSITIVE SMALL region and work its way towards the right end of the x-axis to POSITIVE LARGE.

Fuzzy Rule Area Evaluation

```
//function determines the region of the input fuzzy variable - rate of speed change

 void Rate_spd_chng(int *chng_regn, float *diff_one, float *diff_two, float ang_velocity)
{ float ZE_spd_chng_mbshp, PS_spd_chng_mbshp, PM_spd_chng_mbshp;
float PL_spd_chng_mbshp, NS_spd_chng_mbshp, NM_spd_chng_mbshp;
 float NL_spd_chng_mbshp;
 if(ang_velocity >= 0) {if(ang_velocity <= Spdchng_PSS)
{ ZE_spd_chng_mbshp = acquire_mbsp(0,Spdchng_PSS, -1, ang_velocity);
            *diff_one=ZE_spd_chng_mbshp;
            *diff_two=0;
            *chng_regn=0; }
             else if(ang_velocity > Spdchng_PSS && ang_velocity <= Spdchng_ZEE)
                {ZE_spd_chng_mbshp = acquire_mbsp(0,Spdchng_ZEE, -1, ang_velocity);
             *diff_one=ZE_spd_chng_mbshp;
            PS_spd_chng_mbshp=acquire_mbsp(Spdchng_PSS,Spdchng_PSM, 1, ang_velocity);
             *diff_two=PS_spd_chng_mbshp;
             *chng_regn=1; }
```

Fuzzy Region Evaluation Loop

```
//function determines total area of the numerartor of  two i/p fuzzy variables for centriod
//evaluation

 void Area_of_num(float *upper, float *boundary, int Respond_spd,  int FAM_col, float
speed_membshp, float diff1_spd_mbsp, float diff2_spd_mbsp)
{ float least, boundary1, boundary2, upper1, upper2;
 int FAM_row;
if (Respond_spd == ZERO)
{
   FAM_row = 3;
   if (speed_membshp <= diff1_spd_mbsp)
   least = speed_membshp;
   else
   least = diff1_spd_mbsp;
   boundary1 = (least*(base_FAM[FAM_row][FAM_col]
   +((1- least)*base_FAM[FAM_row][FAM_col])))/2;
   *boundary = boundary1;
   upper1 = scr_delay[FAM_row][FAM_col]*boundary1;
   *upper = upper1;
}
    else if (Respond_spd == ZERO_POSMALL)
    {
      FAM_row = 3;
      if(speed_membshp <= diff1_spd_mbsp)
      least = speed_membshp;
      else
      least = diff1_spd_mbsp;
    }
```

DEFUZZIFICATION

The input fuzzy sets that are defined for error and rate of change of speed are to be converted to an output fuzzy set and then into a crisp output for controlling the speed of the dc motor. The determination of a crisp output is illustrated by considering an example. Suppose the value for error was determined to be 0.03 and the rate of change of speed was 80 rps, the steady state FAM rule (ZE, ZE; ZE) is considered to be activated. The antecedent fuzzy sets are combined conjunctively with AND and the data to satisfy the compound antecedent to degree is:

$\min (m^{e}_{ZE}(0.03), m^{r\_c\_s}_{ZE}(8)) = \min (0.03, 8) = 0.03$

$m^{e}_{ZE}(0.03)$ is the membership of error in ZERO

$m^{r\_c\_s}_{ZE}(8)$ is the membership of rate of change of speed in ZERO.

The consequent fuzzy set is now activated to a degree 0.03, which in this case is the gate_output fuzzy set. Correlation-minimum inference procedure is used to activate the consequent fuzzy set to degree 0.3 by taking the pairwise minimum of 0.3 and the ZE fuzzy set for gate_output $m^{g\_o}_{ZE}$. The FAM rule (PS, ZE; NS) is similarly activated by (0.03, 8). So for all output values the output, gate_output fuzzy set values which is small but negative control variable values is scaled by the lesser of the two antecedent fit values, and is given by

$\min(m^{e}_{PS}(0.03), m^{r\_c\_s}_{ZE}(80)) \wedge m^{g\_o}_{NS}(g\_o) = \min(0.20, 0.70) = 0.20$

$\min(m^{e}_{PS}(0.03), m^{r\_c\_s}_{ZE}(80)) \wedge m^{g\_o}_{NS}(g\_o) = \min(0.30,0.20) = 0.30$

$m^{g\_o}_{NS}(g\_o)$ is the membership of gate_output in NEGATIVE SMALL. The other two remaining FAM rules are similarly activated and the resulting minimum-scaled consequent fuzzy sets are combined and summed pointwise using the relation:

$$R = \sum_{k=1}^{m} w_k B_k \qquad (1)$$

The corresponding crisp value for the input pair (0.03, 80) is determined by the combined summation. This crisp value which is either positive or negative is added to the fuzzy controller's previous output to control the speed of the dc motor. The fuzzy centroid is computed with simple discrete terms even if the fuzzy sets are continuos in nature. The only step that requires division in the entire FAM inference procedure is while the centroid is being computed. The centroid is computed by replacing the discrete sums with integrals if necessary. The fuzzy centroid defuzzification scheme is used to directly compute the real valued output as a normalized combination of fit values with respect to the output space

$$R = \sum_{j=1}^{p} y_j \, m_R(y_j) / \sum_{j=1}^{p} m_R(y_j) \qquad (2)$$

The centriod is a crisp output value (Fig 5.) that is added to the previously computed value of the fuzzy controller. If, for example, the speed of the dc motor is sensed to be below the set speed the crisp value is added to the previous value of the controller to provide constant balance of the output or the speed that is set at a rated level.

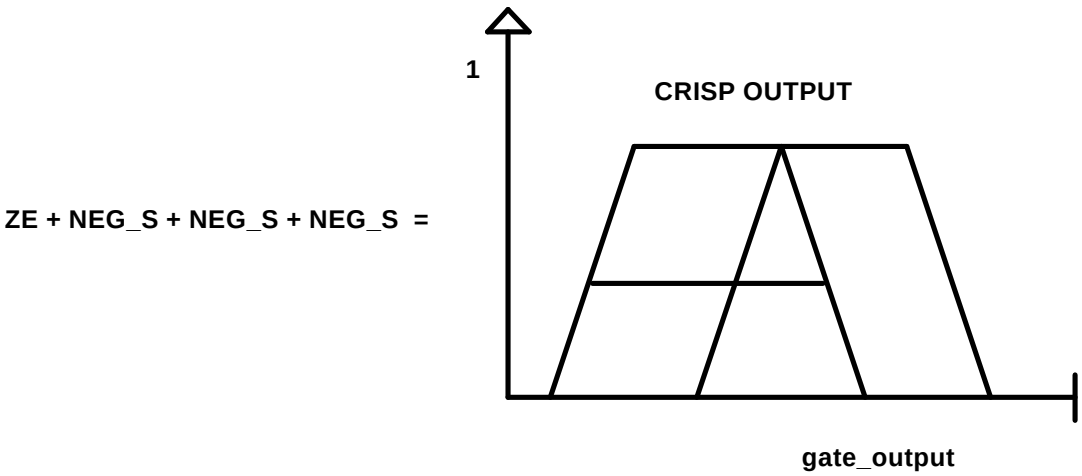

Fig 5. The combination of activated FAM triples with the output as the defuzzified crisp output using centroid defuzzification

The two antecedent values error and rate of change of speed and the consequent gate output are stored as 8-bit values. Fuzzy centroid defuzzification computes the real valued output as shown in (2). The numerator which contains the sum of products of areas for each antecedent rule and degree for each rule is a 16-bit value as well as the denominator which is the sum of areas for each antecedant rule. This is convenient since the Intel 8XC196 supports a 16x16 divide instruction hence eliminating bit manipulations.

OPTIMIZATION OF FUZZY CODE

Inform 8XC196 fuzzy logic design tools emulate fuzzy algorithms on Intel 8XC196 family of 16-bit microcontrollers. This allows defining linguistic variables describing input/output values and IF/THEN rules representing control strategy. Offline optimization allows system performance analysis using model simulation while on-line optimization allows process hardware to be connected to the host system and optimization of the fuzzy controller during runtime. Simulation of a control loop is initialized and the simulation fills in input values to the fuzzy system, invokes the computation of the output values and outputs the result of the fuzzy inference simulation. Single control cycles are executed and the changes in antecedents and consequents can be observed, this allows a defining of a real world control strategy after observing different control strategies to obtain a humanistic reasoning on how to control the system being designed. Overlap and underlap of fuzzy sets can be changed and the right percent overlap determined through observation.

Since simulations tend to be approximations of actual systems behavior as they are model based, appropriate optimization of the fuzzy system is done on-line taking feedback into consideration. On-line optimization can be realised in real-time mode and enables a system to be visualized and modified in real-time on the running process. The 80C196 embedded controller can be connected to the host system enabling “on the fly” optimization by changing system parameters and visualizing inference flow in real-time. The generated code is recompiled without the on-line option and integrated into the 8XC196 hardware system in either ANSI C or assembly. The recompiled code compacts the code since code optimization features are enabled during recompilation.

Assembly code example is shown in the table below for implementing the defuzzification algorithm. Hand coding can be used to optimize specific functions for code execution performance, improved code density, or a combination of code performance and density.

Begin
For regions of error and rate of change of speed computed **do**:
    Set register R0 to activate regions using correlation minimum inference
Begin
    compute centroid value for the combination regions
End
Begin
    Add centroid value to the previous computed actual value of speed to maintain constant speed level
End
End

Fuzzy centroid evaluation

```
LOC     SOURCE STATEMENT
  ; fuzzy.a96 --- Fuzzy centroid defuzzification
0BE0   E70000   ljmp DEFUZZ
0BE3   DEFUZZ:
0BE3   di    ;disable interrupts
0BE4   ld    SP, #500h   ;initialize stack
0BE8   clr   INDEX_1   ;initialize index
0BEA   ld    temp, PREV_OUTPUT   ;load previous o/p
0BED   ld   current, output_1   ;get new fuzzy output
0BF0   add   temp, current   ;add previous and new
0BF3   st   new_value, temp   ;store new value
0BF6   clr   current   ;clear current register
0BF8   ld   current, output_ 1   ;get another fuzzy o/p
0BFB   jnc   RULE_EVAL   ;jump to rule eval loop
0BFD   add  area, current   ;add computed area
0C00   ld   weight, temp   ;load weight of rule
0C03   mul   SOP, weight, area ;multiply upper terms
0C08   st   active, SOP   ;store numerator
0C0B   inc   GATE_OUTPUT   ;point to next output
0C0D   dec   LABEL   ;decrement loop
0C0F   clr   INDEX_2    ;initialize index
0C11   ld   denom, SIGMA   ;sum of activated areas
0C14   div   SOP, denom    ;divide upper/lower terms
0C18   st    RESULT,SOP   ;store result
0C1B   je    get_output       ;jump to area eval loop
0C1D   st    output, RESULT    ;store final result
0C20   ret                          ;return
0C21   END
```

CONCLUSION

Fuzzy controllers take a very short time to design and develop. Fuzzy controllers are model free estimators and are not based on transfer functions like classical controllers, also a mathematical model of the system to be controlled need not be available.

Fuzzy control is not an alternative to classical design and stability analysis cannot be ignored. When rigid design methods are being followed it is necessary that stability analysis be included.

Stability analysis can be done qualitatively to prevent runaway instability. To prevent oscillatory form of instability, stability analysis in the frequency domain should be carried out but in most fuzzy controllers frequency analysis is difficult to perform since mathematical models are rarely available.

Conclusions to be drawn here are: the fuzzy speed controller can be designed and implemented in a much easier and quicker way than a classical controller. The 8XC196 16-bit microcontroller is designed to handle high-speed calculations and fast I/O operations. The large register file with single cycle multiply/accumulate instructions was used to efficiently implement a low cost, simple, model free fuzzy controller. Transfer functions were not used and a mathematical model was not considered. While fuzzy control is an extremely successful means of controlling complex processes, heuristics to control processes that cannot be controlled by a fuzzy system should not be considered. In humanistic systems fuzzy control is the best approach and extremely reliable.

REFERENCES

E. Cox, *Fuzzy Fundamentals.* IEEE Spectrum, pp. 58-61, Oct 1992.

B. Kosko, *Neural Networks and Fuzzy Systems*. Englewood Cliffs: Prentice Hall, 1992.

N. Govind, "A Fuzzy Rule-Based 5 HP DC Motor Speed Controller", in Fuzzy Logic Applications Handbook*,* 1994, Intel Corporation, No 272589-001, pp 95-112

N. Govind and A. R. Hasan, "Fuzzy Logic Speed Controller for a Industrial size DC Motor", in Proceedings of the Fifth Workshop on Neural Networks: Academic/Industrial/NASA/ Defence*,* Nov. 1993, SPIE, vol. 2204, pp 3-8

N. Govind and A. R. Hasan, "Real Time Fuzzy Logic Speed Control Using Conventional, Assembly and Simulation Methods for Industrial DC Motors", in Proceedings of the 1995 IEEE/IAS International Conference on Industrial Automation And Control, IA&C '95, Jan. 1995, Library of Congress No. 94-77908, pp 203-208

E. H. Mamdani and S. Assilian, *An Experiment in Linguistic synthesis with a Fuzzy Logic Controller, Fuzzy Reasoning and its Applications.* Academic Press, 1981 .

S. Murakawi and M. Maeda, *Automobile Speed Control System using a Fuzzy logic Controller,*Industrial Applications of Fuzzy Control, North-Holland, 1985.

L. A. Zadeh, "Outline of a New Approach to the Analysis of Complex Systems and Decision Process*",* in IEEE Transactions on Systems, Man, and Cybernetics, Vol. SMC-3, No. 1, Jan. 1973.