Performance of MC68HC812A4's specialized Fuzzy instructions in Fuzzy logic programming.

Ratcho M. Ivanov, Member IEEE, G.E.M.D.C.Bandara, Member IEEE,
Department of Electronics Engineering, Technical University, Sofia, Bulgaria.
E.mail : dcb@aero.vmei.acad.bg

Abstract.

Microcontrollers with specialized fuzzy instructions being introduced in recent years by various microcontroller manufacturers. Performance of fuzzy programs increase considerably due to simplicity , less execution time and compactness based on such instructions

.

Motorola's latest 8bit microcontroller MC68HC812A4 comes with four fuzzy logic instructions and seven other instructions for custom fuzzy logic programming.

Instruction MEM evaluates trapezoidal membership functions, REV and REVW perform unweighted and weighted MIN-MAX rule evaluation and WAV performs weighted average difuzzification.

During this work fuzzy inference engine is developed based on these instructions for this processor and is tested, compared for performance with fuzzy inference engine for MC68HC11A1. Advantages of specialized fuzzy instructions being evaluated and modified inference engine with very much less execution time and improved performance is verified.

Introduction

Fuzzy logic in fuzzy control systems is implemented in three phases. (Fig.01)

1. Fuzzification (crisp input to fuzzy set mapping)
2. Inference (Fuzzy rule generation)
3. Defuzzification (Fuzzy output to crisp output transformation)

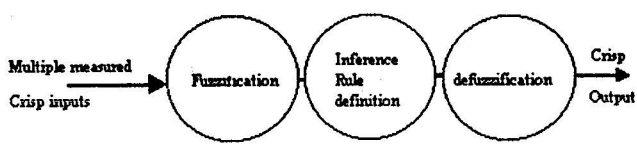


Fig.01 - Fuzzy logic phases.

Analysis of Fuzzy inference kernal for MC68HC812A4.

A block diagram (Fig.02) of a fuzzy logic system can be derived while considering three basic steps in Fuzzy Logic programming.

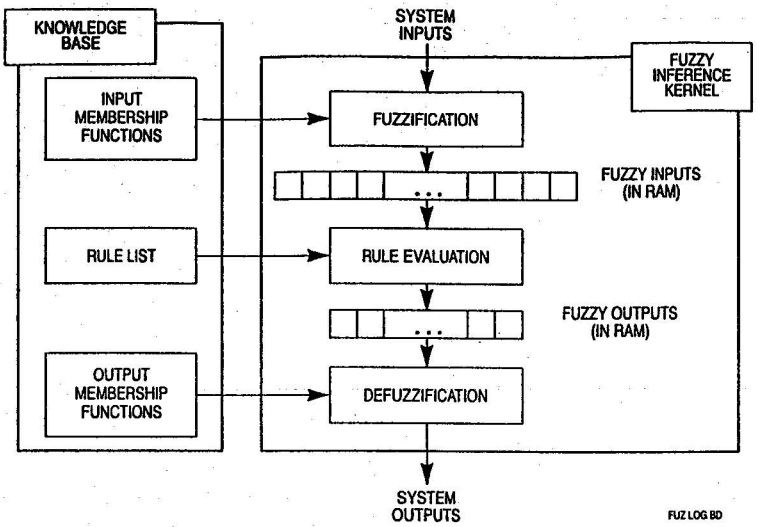


Fig. 02 - Fuzzy inference kernal of a Fuzzy Control System.

MC68HC812A4 has specialized unique instructions for all three steps in fuzzy programming. Given below is a brief description of these instructions and the algorithm of the inference kernal.

Fuzzification (MEM)

At this stage actual measured input values are mapped into fuzzy membership functions and each input's grade of membership in each membership function is evaluated. Final result is then stored in RAM as a table.

Given below existing three definitions for membership functions. [Lotfi Zadeh, Fuzzy Sets Theory, 1965, p. 310].

Definition 1.

The membership function $\mu_c(x)$ of the intersection $C = A \cap B$ is pointwise defined by;

$$\mu_c(x) = \min\{\mu_A(x), \mu_B(x)\}, x \in X \quad ; \text{where}$$

A and B are two fuzzy sets.

Definition 2.

The membership function $\mu_D(x)$ of the union $C = A \cup B$ is pointwise defined by;

$$\mu_D(x) = \max\{\mu_A(x), \mu_B(x)\}, x \in X \quad ; \text{where}$$

A and B are two fuzzy sets.

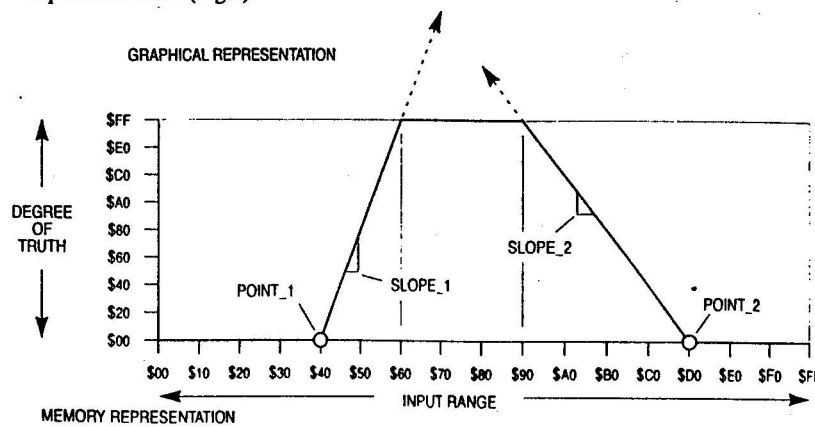
Definition 3.

The membership function of the complement of a fuzzy set A, $\mu_{\complement A}(x)$ is defined by,

$$\mu_{\complement A}(x) = 1 - \mu_A(x), x \in X, \text{ where}$$

A is a fuzzy set. .

shape of the membership function for MC68HC812A4 calculations is assumed as a trapezoidal form. (Fig.3).



MEMORY REPRESENTATION

ADDR	$40	X-POSITION OF POINT_1
ADDR+1	$D0	X-POSITION OF POINT_2
ADDR+2	$08	SLOPE_1 ($FF/(X-POS OF SATURATION – POINT_1))
ADDR+3	$04	SLOPE_2 ($FF/(POINT_2 – X-POS OF SATURATION))

NORM MEM FNCTN

Fig.3 - Defining a normal membership function

When fuzzification begins, the current value of the system input is in an accumulator of the CPU12, one index register points to the first membership function in the knowledge base, and a second index register points to the first fuzzy input in RAM. As each fuzzy input is calculated by executing a MEM instruction , the result is stored to the fuzzy input and both pointers are updated automatically to point to the locations associated with the next fuzzy input. Figure 4 shows how the instruction MEM calculates the grade of membership.

$\mu(grade) \Rightarrow M_{(Y)}$

if $(A) < P_1$ *or* $(A) > P_2$ *then* $\mu = 0$

else

$\mu = MIN\left[((A) - P_1) \times S_1, (P_2 - (A) \times S_2, \$FF\right]$, where

A = current crisp input value.

$(X) + 4 \Rightarrow (X)$; X points at the membership function

$(Y) + 1 \Rightarrow (Y)$; Y points at fuzzy input (RAM location)

Fig. 04 - Membership function evaluation using MEM instruction.

At the end the result of the fuzzification step is a table of fuzzy inputs representing current input conditions.

Rule evaluation (REV and REVW instructions).

In fuzzy logic control, the dynamic behaviour of a fuzzy systems is characterized by a set of linguistic description rules based on expert knowledge. The expert knowledge is usually of the form;

IF (a set of conditions are satisfied) THEN (a set of consequences can be inferred).

Since the antecedents and consequents of these if-then rules are associated with linguistic terms, they are called fuzzy conditional statements.

A fuzzy control rule such as "if (x is Ai and y is Bi) then (z is Ci)" is implemented by fuzzy implication (fuzzy relation) Ri and is defined as follows;

$$\mu_{R_i} = \mu_{(A_i \text{and } B_i \rightarrow C_i)}(u, v, w)$$
$$= \left[\mu_{A_i}(u)\ \mu_{B_i}(v)\right] \rightarrow \mu_{C_i}(w) \text{ , where}$$

Ai and Bi is a Fuzzy set $A_i \times B_i$ in $u \times v$ and $R_i = (A_i \text{ and } B_i) \rightarrow C_i$ is a fuzzy implication in $u \times v \times w$ and $\rightarrow$ denotes a fuzzy implication function.

Instructions REV and REVW use MIN-MAX rule evaluation method, hence it finds smallest rule input and stores to rule outputs unless fuzzy output is already larger (MAX). Each rule input is an 8 bit offset from the base address in Y. Each rule output is an 8 bit offset from the base address in X. Reserved value $FE separates the rule inputs from the rule outputs. Termination of the rule list is marked by the value $FF.

If two or more rules affect the same fuzzy output the rule that is most true governs the value in the fuzzy output, because the rules are connected by an implied fuzzy or operation.

Defuzzification (WAV).

The final step in the fuzzy logic program combines the raw fuzzy outputs into a composite system output. Unlike the trapezoidal shapes used for system inputs the CPU12 typically uses singletons for output membership functions. As with the inputs the X-axis represents the range of possible values for a system output. Singleton membership function consist of the X-axis position for a label of the system input.

Fuzzy outputs corresponds to the Y-axis height of the corresponding output membership function.
The WAV instruction calculates the numerator and the denominator sums for weighted average of the fuzzy outputs according to the formula;

$$SystemOutput = \frac{\sum_{i=1}^{n} S_i F_i}{\sum_{i=1}^{n} F_i} \quad \text{;where}$$

n is the number of labels of a system output, Si are the singleton positions from the knowledge base and Fi are fuzzy output from RAM.
The final divide is performed with a separate EDIV instruction placed immediately after the WAV instruction.
Following is the source code of complete fuzzy inference kernal for 8 inputs with 8 labels for each input and 4 outputs with 8 labels. This example use the same number of inputs and outputs as used in HC11 inference kernal for comparison.

```
* Fuzzy Logic inference Kernal for HC12
* Uses 8 inputs with 8 labels and 4 output with 8 labels.
                ORG $1021           ;uses on chip RAM
CURRENT_INS     RMB 8               ;storage for 8 8bit inputs
FUZZ_INS        RMB 64              ;storage for fuzzy inputs
FUZZ_OUTS       RMB 32              ;storage for fuzzy outputs
COG_OUT         RMB 4               ;defuzzified outputs
MEM_IDX         RMB 1               ;index for processing input

                ORG $2000
INPUT_MFS       EQU *               ;input membership functions
                ORG $2200
RULE_START      EQU *               ;fuzzy rule base
                ORG $2600
SGLTN_POS       EQU *               ;output singleton positions


* Begining of source code of the inference kernal
                ORG  $3000
                LDS  #$1119         ;initialize the stack
                LDAA #8             ;number of inputs to process
                STAA MEM_IDX
NXT_INPUT       LDX  #CURRENT_INS   ;x point to the fist input
                LDAA 1,X+           ;loads current inputs and x = x+1
                PSHX                ;we need x later
                LDX  #INPUT_MFS     ;input membership function address
                LDY  #FUZZ_INS      ;address of fuzzy inputs
                LDAB #8             ;number of labels for an input
GRAD_LOOP       MEM                 ;execute MEM for one label
                DBNE B,GRAD_LOOP    ;execute MEM for all the labels
                PULX
                DEC  MEM_IDX        ;decrement pointer to load next input
                BNE  NXT_INPUT      ;process the loop for nex input
```

```
* Rule evaluation

                LDAB #8
RULE_EVAL       CLR  1,Y+          ;clr fuzzy output and increment pointer
                DBNE B,RULE_EVAL   ;loop to clear all fuzzy outputs
                LDX  #RULE_START   ;points at 1st rule element
                LDY  #FUZZ_INS     ;points at fuzzy inputs and outputs
                LDAA #$FF          ;initializes A and clears the V bit
                REV                ;process rule list
* Defuzzification

DEFUZZ          LDY  #FUZZ_OUTS    ;points at fuzzy outputs
                LDX  #SGLTN_POS    ;point at sigleton positions
                LDAB #8            ;8 fuzzy outputs per COG output
                WAV                ;calculate sums for weighted average
                EDIV               ;final divide for weighted average
                TFR  Y D           ;move result to A:B
                STAB COG_OUT       ;store system output
```

Conclusion.

1. New fuzzy inference kernal fits in 63 bytes of code while HC11 inference kernal takes 268 bytes of code. Therefore using new instructions reduce the code space and increase the readability, simplicity of the program almost four times.

2. For a system containing 8 inputs with 8 labels and 4 outputs with 8 labels it takes only 867 cycles to process one rule byte while HC11 kernal takes 6 times more time than the estimated time above.

3. Finally specialized fuzzy instructions in CPU12 make fuzzy logic programs smarter and work faster, saving important processor time for other tasks.

Reference:

1. MC68HC812A4 reference manual, Motorola Inc., 1995

2. Neural Networks and Fuzzy Systems, Bart Kosko, Prentice Hall 1987

3. Texas Instruments Application report on Fuzzy Logic Control

4. Fuzzy sets and systems, L. Zimmerrman, 1990