Fifth IFSA World Congress (1993), 921 - 924

# An 8-bit Resolution 140 kFLIPS Fuzzy Microprocessor

Mamoru SASAKI Fumio UENO Takahiro INOUE

Computer Science and Electrical Engineering, Kumamoto University

## Abstract

For the purpose of applying to a high-speed control system, such as engine control for automobile application, we propose an architecture of a fuzzy inference processor, which can realize high-speed inference, high-resolution, and can be implemented with small chip area. We have designed a single chip based on the architecture, and confirmed the performance, such as 140 kFLIPS with 8-bit resolution.

## 1 Introduction

Fuzzy inference have been widely applied to automatic control systems and Fuzzy inference processors have been developed to realize high-speed inference [1],[2]. However, when it is applied to high-speed control systems such as engine control of automobile, high-resolution (8-12 bits) is required in addition to high speed. Furthermore, it is important that the processor can be implemented with small hardware resources in order to incorporate it into the control equipment. So, we need an architecture of fuzzy inference processor satisfying above three requirements, i.e. high-speed, high-resolution, and implementation with small hardware resources.

In this paper, in order to realize high-speed inference, we make use of the property of parallelism in fuzzy inference algorithm and we introduce parallel processing with multiple operational elements. We can implement the parallel processing using reasonable hardware resources for two reasons, i.e. (1) the operational element can be implemented very simply because the operations executed in parallel are only MAX and MIN operations. (2) an efficient parallel processing can be executed by preparing only same number of operational elements as the number of labels, which is 5-7. Considering realization of high-resolution, the hardware resources for the table-look-up realizing membership functions, increase exponentially to the resolution. So, we need to implement the membership functions using logic circuits. In this case, however, the configuration for the membership-function circuit is very complicate because a multiplier is required, and the parallel processing with many membership function circuits is impossible because of the restriction of hardware resources. The typical membership functions have a property being that the number of membership functions having non-zero membership value for any input, is less than three. So, we make use of the property and we implement a membership-function circuit outputing a label and a membership value of the membership function having non-zero membership value. The parallel processing with multiple operational elements is possible by preparing only two membership-function circuits. We introduce singleton membership-functions with respect to consequence, in order to implement them with small hardware resources and to realize high-speed defuzzification.

## 2 A fuzzy inference processor

A block diagram of the fuzzy inference processor is described in Fig.1. We explain the membership-function circuit (input stage), the operational element (operation stage), the weighted-average circuit (output stage).

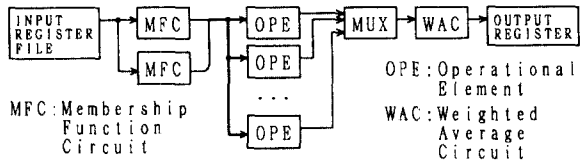


Fig.1 A fuzzy inference processor

### 2.1 A membership-function circuit

#### 2.1.1 Grouping and setting of membership functions

Several membership functions are defined on an input as shown in Fig.2 (a). In many cases, the number of membership functions having non-zero membership value is less than three as shown Fig.2 (a). So, the membership functions can be divided into two groups, in which any membership function does not overlap each another as shown in Fig.2 (b) and (c). Furthermore, we can specify the region where each membership function is defined as shown in Fig.2 (b) and (c). When an input is given, we can obtain all information about all membership functions from two labels and two membership values with respect to two groups. We can reduce memories to 2/( the number of labels) by making a table of Fig.2 (b) and (c) with binary codes representing the labels. However, the method is not useful for high resolution membership functions, because the memories increases exponentially to high resolution. Here, we implement a membership-function circuit by logic circuits, which calculates the label and the membership value of the membership function including the input in own region. Although the circuit configuration is complicate because of a multiplier, increase of hardware resources is very small because only two membership-function circuits realize the parallel processing with multiple operational elements as mentioned above.

A membership function is represented by a piece-wise linear function specified with two turning points $a_i$, $a_{i+1}$ and two

slopes $\alpha_i$, $\alpha_{i+1}$ as shown in .Fig.2 (c). In order to guarantee the operational accuracy, the slope $\alpha_i$ is a floating point decimal as follows :

$$\alpha_i = A_i \times 2^{-B_i} \quad (1)$$

Where the number of bits $N$ of $A_i$ represents the resolution and $B_i = \log_2 N$. In this case, the membership value for the input $x$ can be calculated as follows :

$$g = \begin{cases} min\{(x - a_i) \times \alpha_i, F\} & (x \leq a_{i+1}) \quad (2) \\ max\{F - (x - a_{i+1}) \times \alpha_{i+1}, O\} & (x > a_{i+1}) \quad (3) \end{cases}$$

Where $F$, $O$ represent full-bit and zero-bit, respectively.

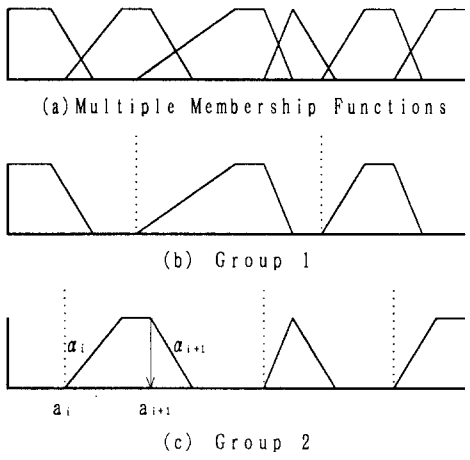


Fig.2 Grouping of membership functions

### 2.1.2 Circuit configuration

A membership-function circuit is described in Fig.3. We explain the operation of the circuit.

(1) An input is sent from the input-register-file to the register 1.

(2) $a_i$, $A_i$ and $B_i$ are read from the membership-function memory using the content $i$ of the counter as an address, and they are sent to the register 2, 3 and 4, respectively.

(3) $a_i$ is stored to the register 2 and (register 1 - register 2) is executed by the subtractor.

(a) In case that the result of subtraction is positive, $A_i$, $B_i$, the content $i$ and the result of subtraction are stored to the register 3,4,5 and 6, respectively.

(b) In case of negative, no data is stored.

After that, the content $i$ is incremented ($i = i + 1$),

(a) In case that $i \leq M$, return to (2).

(b) In case that $i > M$, go to (4).

Where M represents the number of the labels. The decision for positive or negative, and the timing signals for storing can be generated from the sign-bit of the subtractor.

(4) The content of the register 6 are multiplied with the content $A_i$ of the register 3 by the shift-adding type multiplier, and the result is sent to the shifter.

(5) The result of multiplication is shifted $B_i$ times toward LSB according to the content of the register 4. In the shifting, zero is substituted into MSB. After shifting, the overflow is detected by logical summation among upper N bits. Note that N represents the resolution and the result of the multiplication becomes $2 \times N$ bits.

(a) In case of no overflow, lower N bits are output .

(b) In case of overflow, full-bit are output .

(6) (a) In case that LSB of the content of the register 5 is 1 ( $i$ is odd ), all bits of the output are inverted.

(b) In case of 0 ( $i$ is even ), no operation.

The label ( the binary code ) of the firing membership-function can be obtained as the upper bits without LSB of the content $i$ of the register 5.

In the circuit shown in Fig.3, the above operations can be executed on three stage pipeline composed of the subtractor, the multiplier and the shifter. In this case, the number of clock cycles every stage becomes max(M,N), where M and N are the number of the labels and bits representing the resolution. In case of no pipeline process, the number of clocks for all operations is (M + 2×N). So, the pipeline process speeds up by approximately three times.

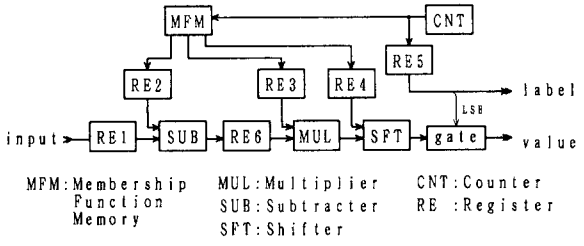


Fig.3 A membership function circuit

## 2.2 Operation stage

In the operation stage, the antecedent process of the following fuzzy rules, that is the matching score is calculated.

$$\begin{aligned} IF \quad & \{A_{11}(x_1), \ A_{12}(x_2), \ \ldots, A_{1j}(x_j), \ \ldots, \ A_{1J}(x_J)\} \ or \\ & \{A_{21}(x_1), \ A_{22}(x_2), \ \ldots, A_{2j}(x_j), \ \ldots, \ A_{2J}(x_J)\} \ or \\ & \quad \vdots \qquad\qquad\qquad\qquad\qquad\qquad \vdots \\ & \{A_{i1}(x_1), \ A_{i2}(x_2), \ \ldots, A_{ij}(x_j), \ \ldots, \ A_{iJ}(x_J)\} \ or \\ & \quad \vdots \qquad\qquad\qquad\qquad\qquad\qquad \vdots \\ & \{A_{I1}(x_1), \ A_{I2}(x_2), \ \ldots, A_{Ij}(x_j), \ \ldots, \ A_{IJ}(x_J)\} \\ & \qquad THEN \quad S_k \end{aligned} \quad (4)$$

Where $A_{ij}(x_j)$, $S_k$ represent a label and a singleton, respectively. And, $i$, $j$ and $k$ represent a kind of a sub-rule, an input and a singleton, and I and J represent the numbers of the sub-rules and inputs, respectively.

The equation (4) can be obtained by combining typical rules having same consequence membership-function. A configuration of the operational element processing the rule is described in Fig.4. Every label in the equation (4) are stored into the label-memory in the order of $A_{11}, A_{21}, \ldots, A_{I1}$,

$A_{12}, A_{22}, \ldots, A_{I2}, \ldots, A_{1J}, A_{2J}, \ldots, A_{IJ}$. Next, we explain the operation of the element.

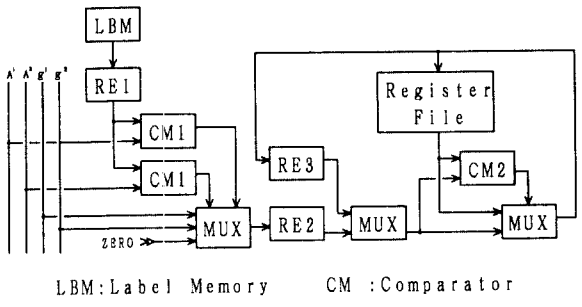


Fig.4 An operational element

(1) Two labels $A_j^1$, $A_j^2$ and two membership values $g_j^1$, $g_j^2$ of the firing membership-functions with respect to the input $j$ are output on local bus from two membership-function circuits. Where the initial value of $j$ is 0.

(2) The $A_{ij}$ is read from the label-memory and it is sent to the register 1. Where the initial value of $i$ is 0.

(3) The $A_{ij}$ is compared with two labels $A_j^1$ and $A_j^2$ of the firing membership-functions by two equal-comparators.

  (a) In case that $A_{ij}$ is equal to $A_j^1$ ($A_j^2$), the membership value $g_j^1$ ($g_j^2$) is stored into the register 2.

  (b) Otherwise, the register 2 is reset to zero.

(4) The content of the register 2 is compared with the content of ith register of the register-file by grater-less-comparator and the less one is stored into ith register of the register-file again. Where all contents of the register-file are set to full-bit as initial values.

(5) (a) In case that $i < I$, $i = i + 1$ and return to (2).

  (b) In case that $i = I$, $j = j + 1$ and go to (6).

(6) (a) In case that $j \leq J$, return to (1).

  (b) Otherwise, go to (7).

(7) The greatest value among the register file is calculated by the greater-less-comparator and it is store into the register 3. The greatest value is the matching score .

As mentioned in section 2.1, the membership-function circuit requires max(M, N) clock cycles with respect to one input process. So, in the above process (1), the labels and the membership values are output on the local bus between max(M, N) clock cycles. So, when the number I of the iteration of the above processes (2),(3),(4) and (5) is set to max(M, N), the most efficient pipeline process on the membership-function circuit and the operation stage is realized. In the fuzzy inference processor shown in Fig.1, same number of the operational elements as one of the labels are prepared and the fuzzy rules described in the equation (4) are processed in parallel.

### 2.3 A weighted-average circuit

A diffezzification is done according to the following equation:

$$output = \frac{\sum_k G_k \times P_k}{\sum_k G_k} \tag{5}$$

Where $G_k$, $P_k$ are matching score of the rule with the singleton $S_k$, and co-ordinate of the singleton $S_k$, respectively. This operation is executed by a shift-adding multiplier (shift-subtracting divider) and an accumulator.

### 2.4 Discussion on the inference speed

In this section, we discuss the inference speed of the proposed processor. In the following discussion, $J$, $M$ and $N$ represent the number of inputs, labels and bits representing the resolution, respectively. The pipeline composed of the membership-function circuits and the operational elements have four stages and each stage requires max(M, N) clock cycles. Therefore, the number of clock cycles for the process of all J-input is :

$$max(M, N) \times (4 + J - 1) clockcycles \tag{6}$$

And, the number of clock cycles for the max operation in the operational elements is :

$$I = max(M, N) clockcycles \tag{7}$$

Furthermore , the number of clock cycles for the weighted-average operation is :

$$M \times N + N clockcycles \tag{8}$$

Hence, the total number of clock cycles is :

$$max(M, N) \times (4 + J) + M \times (N + 1) \quad clockcycles \tag{9}$$

In the above clock cycles,

$$I \times M = max(M, N) \times M ruleswithJ - input \tag{10}$$

are processed.

In case that we process multiple groups of rules, the pipeline is more efficient and other pipeline process on the matching-score operation and the weighted-average operation can be introduced. Therefore, the throughput for the multiple groups of rules is :

$$max\{max(M, N) \times (J + 1), M \times (N + 1)\} clockcycles/group \tag{11}$$

## 3 LSI implementation

The fuzzy inference processor have been developed as LSI using 1.2um CMOS standard cells. In the design, the number of inputs, labels and bits representing the resolution $J$, $M$ and $N$ are set to 8. The performance is :

(1) The rule format and the number of rules
(8-input, 1-output, 32-rule) × 4 rule-groups

(2) The number of clock cycles for the process
First consequence (group) : 174 clock cycles
Second consequence (group) : 246 clock cycles
Third consequence (group) : 318 clock cycles
Forth consequence (group) : 390 clock cycles

**(3) The clock frequency : 10MHz**

Although the number of rules is $8 \times 8 = 64$ rules according to equation (10), the number has been 32 rules in the above implementation. The reason is that the number of sub-rules has been max(M,N)/2 = 4 rules because read and write for RAM, by which the register file in the operational element is realized, require more one clock cycle. Although the total number of clock cycles is $8 \times 12 + 8 \times 9 = 168$ clock cycles according to equation (9), the number has been 174 clock cycles because synchronization requires more six clock cycles. The throughput has been $8 \times 9 = 72$ clock cycles according to equation (11). Hence, the inference speed is :

$$10000k\ (Hz)/72\ (clockcycles) = 140k\ FLIPS \qquad (12)$$

## 4 Conclusion

With the aim of applying to high-speed control system, we have proposed an architecture of fuzzy inference processor realizing high-speed, high-resolution and being able to be implemented with small hardware resources. On basis of the architecture, a single-chip fuzzy inference processor has been fabricated and the efficiency has been confirmed.

The authors wish to thank the electronic devices group of the Oki electric Industry corporation, especially T.Katashiro, for fabricating the circuits.

## REFERENCE

(1) H.Watanabe, W.Dettloff and K.Yount : "A VLSI fuzzy logic controller with reconfigurable, cascadable architecture", IEEE J. SC vol.25, no.2, pp.376-382, (1990).

(2) M.Sasaki, F.Ueno and T.Inoue : "7.5MFLIPS fuzzy microprocessor using SIMD and logic-in-memory structure",Second IEEE Int. Conference on Fuzzy Systems, pp.527-534, (1993).