Computation scheme for the general purpose VLSI fuzzy inference engine as expert system

Yoshiyasu Takefuji* and Meng-Hiot Lim†

Fuzzy inference engines based on the existing fuzzy theory are inadequate to perform reliable decision making. Besides requiring the fuzzy sets and data to be normalized, the inference engine is also sensitive to noise in observational data. Inaccurate conclusions are produced if noise is present and also when the fuzzy sets are not normalized. In this paper, a new term 'similarity' (σ) and the method to compute σ to enhance the capability of fuzzy set theory for application in expert systems is introduced. Even though the complexity of the hardware engine is slightly increased, it actually reduces the overhead of computation by eliminating the need for normalization of fuzzy data. With reliable fuzzy data manipulation, it is easy to extend to a multi-dimensional membership function which has a wider scope of applications. To implement the Very Large Scale Integration fuzzy inference engine, two general schemes of the hardware architecture that can be easily reconfigured to satisfy given performance requirements are discussed.

Keywords: fuzzy computation, fuzzy inference engines, fuzzy expert systems

Human possesses several distinguished reasoning mechanisms which non-human does not have. Most of these reasoning processes are still not well understood by researchers or scientists. A few of the reasoning mechanisms have been studied and practically applied to real world problems. Inference is a kind of mechanism in reasoning — reasoning can be classified into three categories: exact reasoning; fuzzy reasoning; and a combination of the two. In exact reasoning there is no ambiguity at all in expressions, e.g. 'John has a wife; her name is Nancy; John's wife is pregnant'. An expression in exact reasoning is either an inference rule or a fact.

*Department of Electrical Engineering, Case Western Reserve University, Cleveland, OH 44106, USA
†School of Electrical and Electronic Engineering, Nanyang Technological Institute, Singapore 2263

Paper received 7 April 1988. Accepted 5 January 1989

In this example, there are two facts and one rule;

FACTS: wife(John) and is_pregnant(wife(John))
RULE:EQUAL(wife(John), Nancy)

From these two facts and one rule, we can conclude that Nancy is pregnant: is_pregnant(Nancy). In fuzzy reasoning, an expression contains fuzzy meaning, e.g. 'if x is a very heavy man then x is not suitable to be a jockey'. Note that there is a fuzzy term 'very heavy'. In fuzzy reasoning we have to deal with fuzzy terms to solve problems. To perform an intelligent job, human knowledge can be represented or described by knowledge expressions. However, it is very hard or impossible for us to describe the entire domain of human knowledge even by current high technologies. Thus, each of the existing inference systems must have knowledge information in a very narrow domain so that it becomes feasible to store and manipulate a finite number of inference rules and facts in memory elements such as hard discs, floppy discs, or semiconductor memory chips to manage the knowledge expressions for a specific application. The so called 'inference engine' is a knowledge manipulation system to deal with some specific inference problems.

Inference engines can be classified into the following categories: Synchronous Line Driver (SLD) inference engines such as Prolog machines (so called fifth generation computers in Japan[1-3]); resolution-base machines such as Interactive Theorem Prover (ITP) at Argonne National Laboratory in Illinois[4]; fuzzy inference engines at AT&T[5]; paramodulation inference engines at the University of South Carolina[6]; and abduction engines.

Here, the focus is initially on the theory of fuzzy sets. Then a new fuzzy inference computation scheme and its inference engine as expert system is introduced. We have been developing the prototype of the inference engine based on our improved fuzzy inferencing scheme, thus eliminating the significant drawbacks of the existing fuzzy computation algorithms. The new fuzzy inference computation is suitable for complex inference rule expressions and multi-dimensional membership functions. (A brief explanation of the terminology used in the representation of fuzzy inference rules is provided below.)

0950-7051/89/020109-08 $03.00

FUZZY SET THEORY[7]

The main applications of fuzzy logic lies in economics, management science, Artificial Intelligence (AI), psychology, linguistics, information retrieval, medicine, biology, chemistry, system controls, and other fields. A brief theory of fuzzy logic in expert systems is presented here.

Consider the following sentence: 'he is a very tall man.' We do not know exactly how tall he is. He might be 6'5", 6'8", or more than 7 feet in height. If a given sentence contains ambiguity or fuzzy meaning, then we must know what the attribute is associated with the fuzzy term in order to process fuzzy expressions. In this sentence, 'very tall' is a fuzzy term, and this fuzziness is associated with the attribute 'height'. In general, a linguistic variable is used to express fuzzy attributes, e.g. since the terms very short, short, not tall, tall, and very very tall are values of attribute height, then height must be a linguistic variable. Once we have determined the attribute associated with the fuzzy expression, the attribute can be quantized. There are objections against quantization of fuzzy attributes among some researchers. Their claims are that fuzzy attributes cannot be quantized — doing so would suggest that they are not fuzzy any more. Despite such claims, fuzzy theory strongly supports the quantization of fuzzy attributes[5,7].

The use of fuzzy expressions to represent domain-specific human knowledge can be described by fuzzy conditional statements. Fuzzy conditional statements are expressions of the form '**if** A **then** B' where A and B have fuzzy meaning, e.g. '**if** temperature in the room is too hot **then** turn the thermostat to low' is a fuzzy conditional statement. The A part is called the antecedent while the B part is called the action or conclusion. The antedecent part is compared with the fuzzy observation input (the fuzzy observation input is explained below). If matching between the antecedent part of the fuzzy set and the observation input is perfect then 100% of the action part is generated as fuzzy output. If matching is X%, then X percent of the action part is generated as fuzzy output.

Before fuzzy expressions can be processed based on fuzzy computation, a formal representation of linguistic variables in terms of fuzzy sets is required. Fuzzy set can be characterized by members and a membership function. Members in membership and the membership function play an important role in the quantization of fuzzy attributes. Members lie in a range of attribute domains. The membership function is a probability function or a grade function, e.g. if the attribute is age, a member of membership AGE is an element in a set of values {0,1,2...,150} (note that member 0 means 0 years old and 150 means 150 years old). The membership function AGE(x), where x is a member of membership AGE, expresses the grade of membership of x in AGE. The membership function AGE(x) usually lies in the interval [0,1] (note that AGE(x) = 0 means 0% probability, and AGE(x) = 1 denotes 100% probability).

Some examples of members and membership functions are shown in Figure 1. Consider the attribute 'age'. Fuzzy sets of various linguistic terms (young, old, very young, not young and very old) can be expressed as membership functions (shown in Figures 1a, b, c, d and e respectively).

In fuzzy reasoning, fuzzy conditional statements are

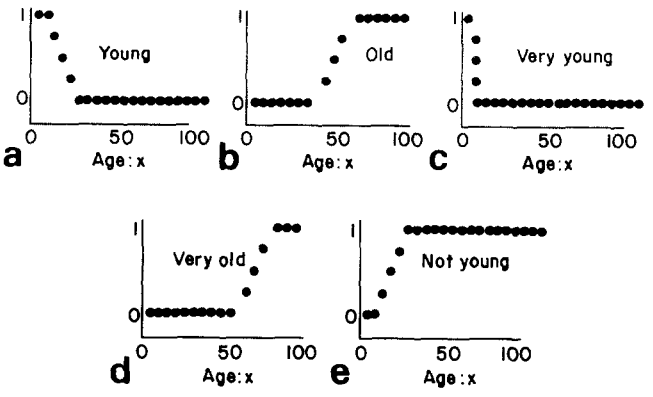

Figure 1. Membership function examples. (a) Young(x), (b) Old(x), (c) Very_young(x), (d) Very_old(x), and (e) Not_young(x)

incorporated into the knowledge base as rules in the following form:

Rule1: **if** A_1 **then** B_1,
Rule2: else **if** A_2 **then** B_2
.
.
RuleN: else **if** A_N **then** B_N

e.g. suppose that you are a teacher giving advice to a student who sat for an examination. Let x be a score of the student. An example of the compositional rules of inference is:

Rule1: **if** x is very low **then** tell him to study very very hard;
Rule2: else **if** x is low **then** tell him to study very hard;
Rule3: else **if** x is not high **then** tell him to study hard;
Rule4: else **if** x is not low **then** tell him to study;
Rule5: else **if** x is high **then** tell him to keep on studying;
Rule6: else **if** x is very high **then** ask him to teach in his class.

Note that each of the antecedent parts such as 'x is low' is compared with fuzzy observation input. Remember that fuzzy observation input is not an exact number, but a fuzzy set. Depending on a given fuzzy observation input, we can generate a fuzzy output by incorporating each of the action parts in rules such as 'tell him to study very hard'. Each of the action parts is a component of the final fuzzy output to be produced by an inference engine. Comparing the magnitude between a fuzzy observation input and the fuzzy set representing the antecedent determines what percentage of the action part in a particular rule is used for the final fuzzy output, e.g. if fuzzy observation input is an expression containing 'x is not low but not high', then a high percentage of Rule3 and Rule4 are activated to generate the final fuzzy output. This is because the observation input hits the highest matching magnitude of Rule3 and Rule4. Of course, Rules 1, 2, 5, and 6 contribute a little to the output too.

In fact, the computation of fuzzy conditional statements described above is based on very simple minimum

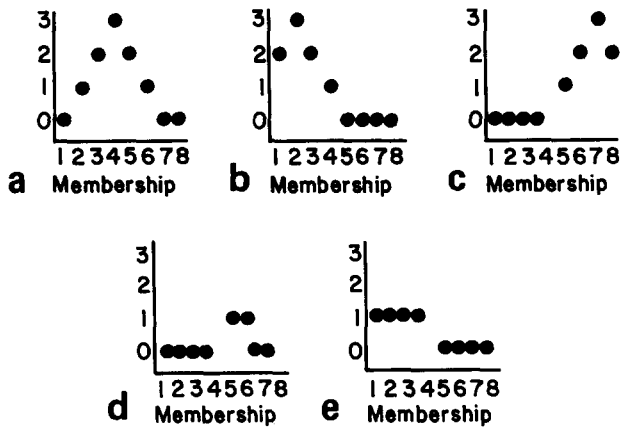

Figure 2. Resolution of membership grades. (a) Antecedent A, (b) Action B, (c) Observation A', (d) Computed α, (e) Final output

and maximum functions. The min function 'min(x,y)' chooses the minimum value in x and y, e.g. min(5,9) returns 5. The max(5,9,7) returns the maximum value 9 among 5, 9, and 7. (Details of the fuzzy computation using min and max functions are explained below.)

Before describing the details of the mechanism of fuzzy computation, the resolution of the grades of membership and the resolution of membership must first be determined. The so called resolution of probability and the number of members in membership are decided based on applications, e.g. if we use four bits for expressing probability, we will have 16 levels in grade (2^4 = 16) (level starts from 0th to 15th grade in this case). If we use 18 members for membership, a total of 18×4 bits of memory cells are required to represent one fuzzy set.

Consider an example with a single rule: **if** x = A **then** y = B where A and B have 8×2 resolution (note that A and B have four levels of probability and eight members for membership). Inferring from a single rule is based on computation of the value of α:

$$\alpha = \max(\min(\text{p(i) of A, p(i) of A'}))$$

where p(i) is the probability of the ith member of membership. The value of α corresponds to the degree of matching between the fuzzy observation input and the antecedent part mentioned previously (note that i is in the range from 1 to 8). Level-0, level-1, level-2 and level-3 in grade correspond to 0%, 33%, 66% and 100% in probability respectively, and is represented graphically by the vertical axis. The final fuzzy output due to this single rule will be given by the equation:

fuzzy_output = min(α, p(i) of B) where i is from 1 to 8.

Consider the following illustration, where A and B are as shown in Figure 2a and b respectively. Suppose a fuzzy observation input A' (shown in Figure 2c) is provided. Then min computation between A and A' is given by min(p(i) of A, p(i) of A'), and will result in the fuzzy set shown in Figure 2d. The value of α in this case corresponds to the first level from the operation of max(0,0,0,0,1,1,0,0) = 1. The final fuzzy output is given by min(α,p(i) of B) (shown in Figure 2e).

As mentioned earlier, rules of inference are generally of the form:

Rule1: **if** A_1 **then** B_1
Rule2: else **if** A_2 **then** B_2
.
.
RuleN: else **if** A_N **then** B_N

Given the above rules, the fuzzy inference computation is given by the following algorithm:

1 For each of the rules (Rule1 through RuleN), compute α for the jth rule which is given by max(min(p(i) of A_j, p(i) of A')) where A_j is the jth rule and A' is an observation input. The fuzzy output B_j' is given by min(α of jth rule, p(i) of B_j) where B_j is the action part of the jth rule. The value of j is in the range of 1 to N.
2 Compute max(p(i) of B'_1 through B_N').

From the above illustration we can conclude that it is fairly straightforward to process a given set of fuzzy inference rules based on the existing inferencing algorithm.

NEW FUZZY COMPUTATION

To overcome the significant drawbacks in the algorithm described above, a new fuzzy computation scheme is introduced, although it slightly increases the complexity of computations for fuzzy processing (details of the drawbacks in the existing fuzzy computation scheme are described below). With the new computation scheme put forth, fuzzy expressions are easily expandable to multi-dimensional membership functions for more fruitful fuzzy expressions, while the existing fuzzy theory is based on one dimensional membership functions so that applications are limited. Also, architectures of fuzzy inference engines based on the new fuzzy scheme are proposed. The proposed fuzzy inference engine is reconfigurable depending on given fuzzy expressions or compositional rules of inference. In this sense, the engine has wider scope of applications. The proposed Very Large Scale Integration (VLSI) fuzzy inference engine is expected to be very powerful as a real-time expert system. The engine will have several Million Fuzzy Logic Inferences Per Second (MFLIPS) computation capability, thus the research on the new fuzzy inference engine is worthwhile, as well as indispensable, for real-time reasoning in some AI applications.

Conventional reasoning algorithm drawbacks

To understand the drawbacks in the conventional reasoning scheme, consider a single inference rule: 'if x = A **then** y = B.

Suppose:

$$A = \{x_1, x_2, \ldots, x_{10}\},$$
$$u(x) = [0,1],$$
$$A = 0.1/3 + 0.2/4 + 0.3/5 + 0.2/6 + 0.1/7,$$
$$B = \{y_1, y_2, \ldots, y_{10}\},$$

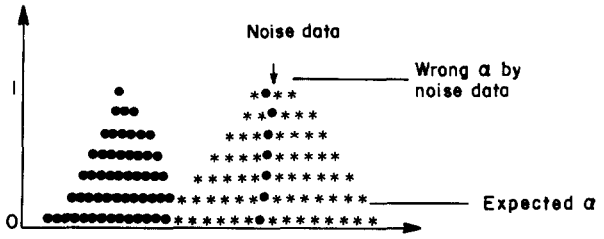

*Figure 3. Fuzzy matching with the presence of noise. ●●●: observation data; ***: antecedent data*

$u(y) = [0,1]$,
$B = 0.1/3+0.2/4+0.3/5+0.2/6+0.1/7$*

If the observation input x is A' where:

$$A' = 0.1/3+0.2/4+0.3/5+0.2/6+0.1/7$$

then the result of α will be 0.3 based on the conventional reasoning scheme. This is because α as given by the equation:

$$\alpha = max(min(uA(x_i), uA'(x_i))) \text{ where } i = 1,\ldots,10$$

becomes 0.3. This α should be 1.0 instead of 0.3 because the observation input A' and the antecedent A are exactly the same, i.e. the condition in 'x = A' is perfectly matched with the observation input A', which means that the conclusion 'y = B' should be 100% activated. To avoid such problems in the conventional reasoning scheme, the observation fuzzy input A' and the antecedent fuzzy set A' must always be normalized to 1 where the computation of normalizations can be very expensive. Otherwise, it is simply prohibited to use such unnormalized fuzzy sets.

The other significant drawback lies in noise tolerancy. If a fuzzy observation input contains unfiltered noise data, the value of α can be incorrectly computed by the presence of a single noise data. Referring to the case shown in Figure 3, where the shaded regions of '***' and '●●' represent the antecedent and observation data respectively, the value of α computed will be equal to 1. The correct α should be close to 0.

New fuzzy reasoning scheme

To introduce the new fuzzy computation algorithm we have to first define an important term, 'similarity', which is denoted by σ. Simlarity is a measure of resemblance between two fuzzy sets. The magnitude of σ is between zero and unity. Consider the inference rule '**if** x = A_1 **then** y = B_1'. Let A' be an observation fuzzy input and B' be the final fuzzy output. Similarity σ between A' and A_1 is applied to obtain uB' where:

$$uB' = \sigma \times uB_1(y_i) \text{ for i} = 1,\ldots,M$$

The similarity between A' and A_1 can be computed from the following equations:

*The first equation denotes that there are ten members in the membership; the second equation implies that probability is in a range of 0 to 1; in the third equation, 0.1/3 (grade/member) means that the probability of member 3 is 0.1. For members not specified, a description of 0/member is assumed for simplicity.

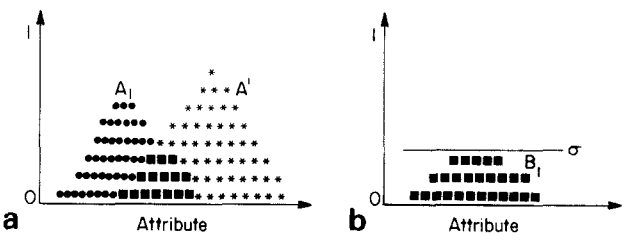

Figure 4. (a) Matching of the **if** *part 'uA$_1$(x_i)' and 'uA'(x_i)'. ●●●: A1; ***: A', ■■■: difference d between fuzzy set A_1 and A', (b) Activation of the* **then** *part of 'uB$_1$(y_i)' of the rule. ■■■: difference d*

$$\sigma = 1 - \left[\frac{d}{a+a'}\right]$$

$$d = \sum_{i=1}^{M} [max(uA_1(x_i), uA'(x_i)) - min(uA_1(x_i), uA'(x_i))]$$

$$a = \sum_{i=1}^{M} uA_1(x_i)$$

$$a' = \sum_{i=1}^{M} uA'(x_i)$$

To better understand the above equations, refer to Figure 4a and b. The shaded areas ('●●' and '**') in Figure 4a are the difference d between the fuzzy set A_1 and the observation input A'. The difference d corresponds to the result of an exclusive-OR operation in set theory. Thus, the relative difference is given by d/($a_1 + a'$), where a_1 and a' are $\Sigma uA_1(x_i)$ and $\Sigma uA'(x_i)$ respectively. The significant advantage of the proposed fuzzy scheme lies in the fact that all the fuzzy sets used such as A', A_1 and B_1 are not necessarily normalized, while all the fuzzy sets in conventional fuzzy theory have to be normalized to 1. Remember that A_1, A' and B_1 sets in Figure 4a and b are not acceptable by the conventional theory, since they are not normalized. The second advantage of similarity lies in the fact that similarity computation can eliminate the undesired noise data effects (discussed above — see Figure 3).

Consider the following set of inference rules:

Rule#1: **if** A_1 **then** B_1
Rule#2: else **if** A_2 **then** B_2
Rule#3: else **if** A_3 **then** B_3
.
.
Rule#N: else **if** A_N **then** B_N

Given an observation data A', the following equations can be used to infer from the rules above:

σ_j is the similarity in the jth rule between A_j and A'.

$$\sigma_j = \left[1 - \frac{d_j}{(a_j + a')}\right]$$

$$d_j = \sum_{i=1}^{M} [max(uA_j(x_i), uA'(x_i)) - min(uA_j(x_i), uA'(x_i))]$$ (where M is the number of members).

$$a_j = \sum_{i=1}^{M} uA_j(x_i)$$

$$a' = \sum_{i=1}^{M} uA'(x_i)$$

$$uB'_j(Y) = \sigma_j \times uB_j(Y)$$

$$uB' = max(uB'_j(Y)) \text{ for } j = 1, \ldots, N$$

To understand complex expressions, consider the following set of rules:

Rule#1: **if** ($w = A_1$ and x - B_1) *or* $y = C_1$ **then** $z = D_1$,
Rule#2: else **if** $w = A_2$ *or* $x = B_2$ *or* $y = C_2$ **then** $z = D_2$,
Rule#3: else **if** $w = A_3$ and $x = B_3$ and $y = C_3$ **then** $z = D_3$.

σ_{i1} is similarity of A_i and A' in the ith rule.
σ_{i2} is similarity of B_i and B' in the ith rule.
σ_{i3} is similarity of C_i and C' in the ith rule.

For rule#1, $uD'_1 = max(min(\sigma_{11},\sigma_{12}),\sigma_{13}) \times uD_1$
For rule#2, $uD'_2 = max(\sigma_{21},\sigma_{22},\sigma_{23}) \times uD_2$
For rule#3, $uD'_3 = min(\sigma_{31},\sigma_{32},\sigma_{33}) \times uD_3$

The final uD' will be given by:

$$uD' = max(uD'_1, uD'_2, uD'_3)$$

Note that the *and* and *or* operators in the antecedent of complex expressions do not require any special additional hardware to be implemented. The *and* and *or* operators correspond to min and max operations respectively, therefore expressions of arbitrary complexity can be modified to equivalent expressions with just *and* and *or* logical operators, which in turn can be realized in hardware with the basic min and max modules.

REASONING INVOLVING MULTI-DIMENSIONAL MEMBERSHIP FUNCTION

The conventional fuzzy theory contstrains users to use only a one-dimensional membership function. The constraint was not significant in linguistic applications, but general fuzzy inference applications have to deal with more sophisticated membership functions. In many practical applications, two or more interdependent variables have to be monitored simultaneously to determine the final decision, e.g. when fuzzy inference rules are applied to pattern recognition, at least two-dimensional membership functions have to be used. To represent a two-dimensional membership function, two different attributes and probability (membership function) have to be used, e.g. to express a small hill in Figure 5, the fuzzy set will be:

$$\begin{aligned} u(X,Y) &= u/x/y \\ &= 1/4/3+1/5/3+1/6/3+1/3/4+2/4/4+ \\ &\quad 3/5/4+2/6/4+1/7/4 \\ &\quad +1/3/5+3/4/5+4/5/5+3/6/5+1/7/5+ \\ &\quad 1/3/6+2/4/6+3/5/6 \\ &\quad +2/6/6+1/7/6+1/4/7+1/5/7+1/6/7 \end{aligned}$$

where u = [0,8], X = {1,2,3,4,5,6,7,8,9,10}, and Y = {1,2,3,4,5,6,7,8,9,10}.

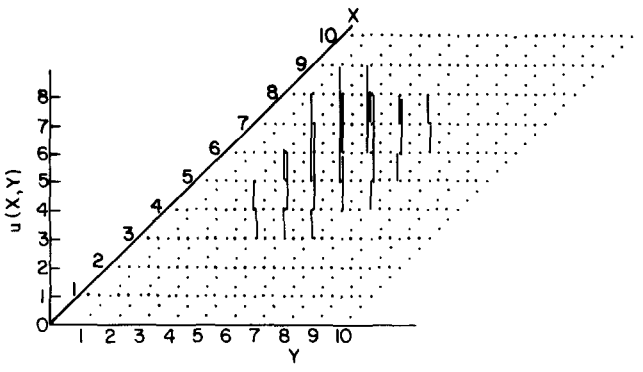

Figure 5. Two dimensional membership function

Similarly, to represent a three-dimensional membership function, three different attributes and probability (membership function) have to be used. Thus, the membership function u(X,Y,Z) can be applied to three-dimensional space problems. The complexity of the inference engine increases as the number of attributes increases.

FUZZY INFERENCE ENGINE ARCHITECTURES

There are basically two types of architecture for sequential inference engines: serial; and parallel. In serial systems, each rule is processed independently one after another. The valid fuzzy output is obtained when all the inference rules have been processed. For parallel engines, all the rules are processed at the same time, and the final conclusion is produced by introducing a max operation on the conclusions of all the rules. The decision whether to use Random Access Memory (RAM) or Read Only Memory (ROM) to store the rules and whether to load the rules from within the chip or externally depends on the application and the availability of silicon space within the processor. Two alternative architectures for realizing fuzzy inference engines on hardware are described below.

Serial engine

Figure 6 shows the data path for a serial engine. The memory required for storing all the fuzzy sets can be located within the chip or it can be external to the chip. The fuzzy sets are repeatedly fed into the processor until all the rules have been applied. The advantage of such a system is its smaller area: this is particularly useful if we have a large number of rules for the expert system and the engine has to be small. The register labelled Reg_A in Figure 6 retains the value of similarity after the fuzzy set that represents the antecedent part of the rule has been processed. This value is denoted by σ, as described before. A shift register with storage equal to the number of bits for one fuzzy set is needed to accumulate the activation of all the rules. Retrieving the valid output of the inference engine can be done after the last rule has been applied. Assuming that a single chip consists of a single serial engine, it is easy to enhance the speed by using more than one chip. As shown in Figure 7 , if we have two fuzzy engines, each engine can process half the number of fuzzy sets. An

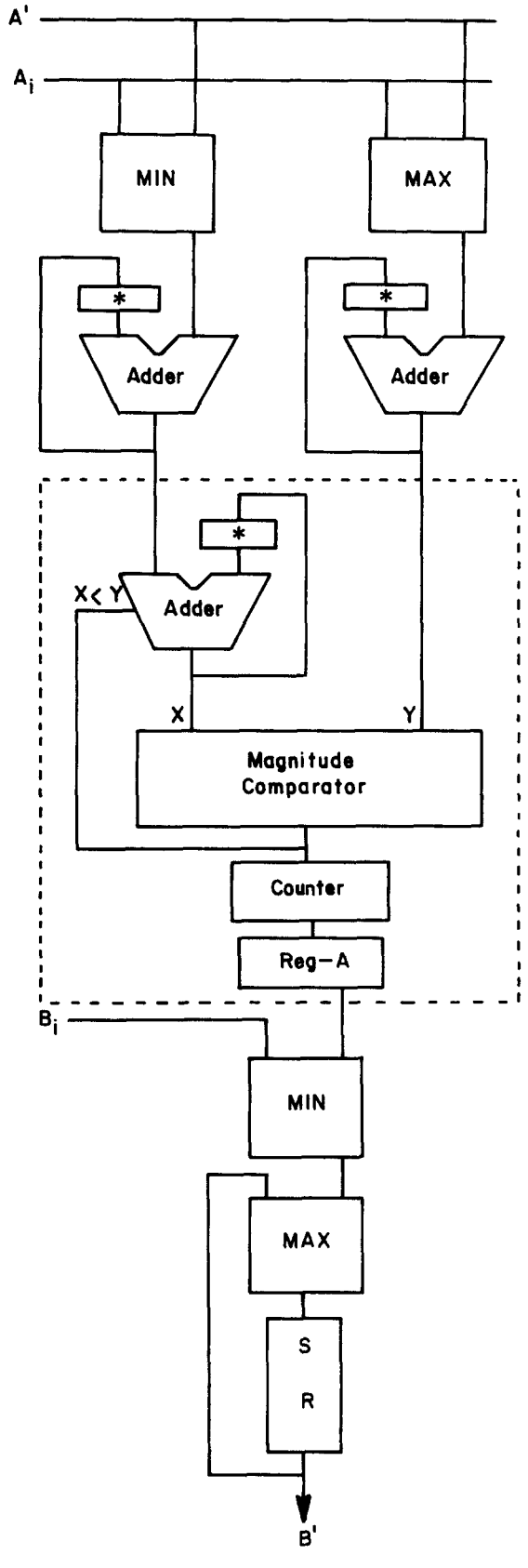

*Figure 6. Serial engine. *: register; SR: shift register; ---: divider*

additional max operation is needed externally to compute the final decision.

Based on the new computation scheme, a divider is required to perform division between two numbers. To achieve this, a divider has been used that is composed of an adder, a magnitude comparator and a synchronous counter, e.g. if we have fuzzy sets with 8 levels of grade and 16 members in the membership, we would need a 7-bit adder, 7-bit magnitude comparator and a 3-bit counter to implement the divider (note that accurate division is not necessary since the probability levels have been discretized to grade levels. This is analogous to performing a long division and discarding the remainder).

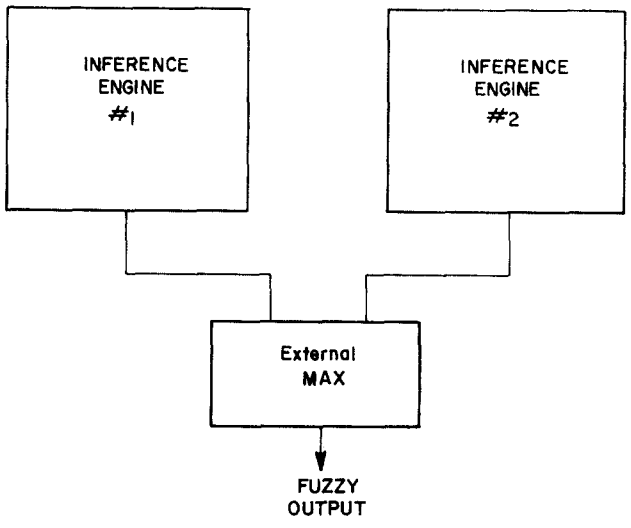

Figure 7. Inference engine using two processors

Two different reset signals are needed to synchronize the inference engine: the first reset signal resets the engine up to the point of Reg_A in Figure 7, i.e. after the value of σ has been computed for each rule; a second global reset signal is needed to refresh the whole system prior to the start of a new fuzzy logic inference.

Parallel engine

An alternative to the serial architecture is the parallel engine. In this case, all the rules are processed in parallel, thus enhancing the speed. Figure 8 shows the data path of a single rule processor in a parallel engine. Since all the rules are processed at the same time, the shift register (shown in Figure 7) is not necessary in this case. The data path of Figure 8 is repeated as many times as the number of rules to be handled. Final output is then produced by a max operation which can either be external to, or locally within the chip. A global reset signal prepares the engine for each new fuzzy logic inference. Even though this architecture consumes more silicon area, it is generally faster and is suitable for real-time expert systems. To enhance the speed even further, each fuzzy set of a rule can be subdivided into portions that are processed in parallel, e.g. to double the speed, two data paths of Figure 8 with an additional max operation will be used to process a single rule. For inference engines with too many rules, more than one chip can be used. As is the case with serial engines (see Figure 7), an external max operation is needed. The main limitations to the number of rules that can be realized on a single chip are the die area and the number of input and output pins per chip.

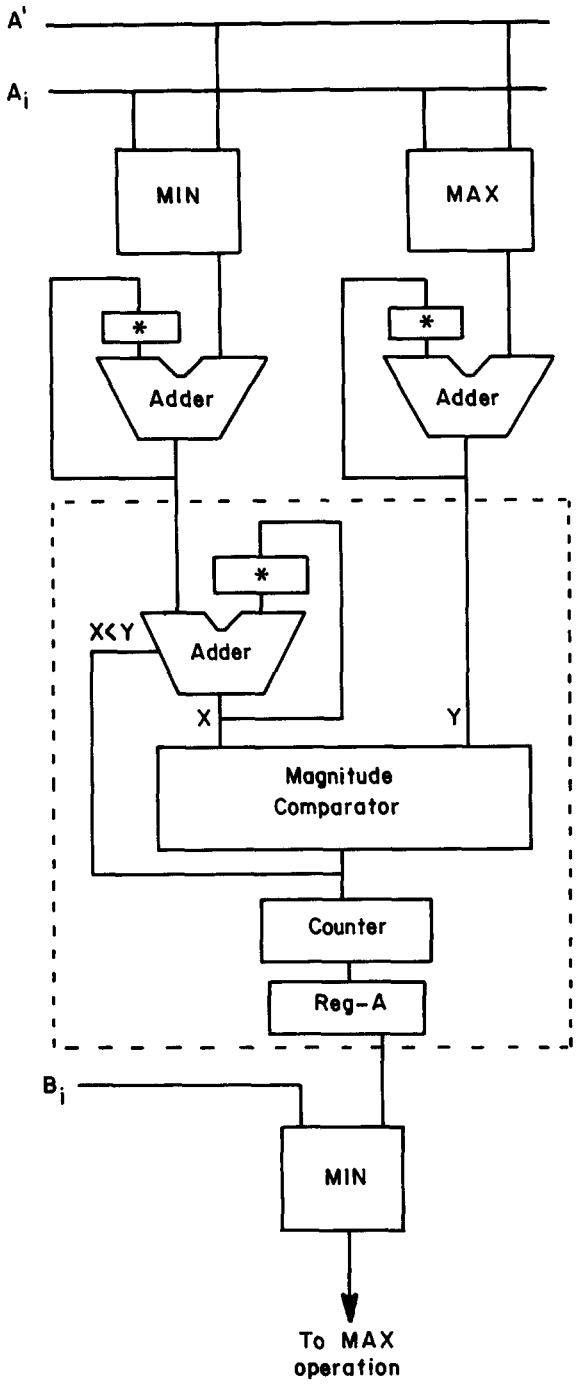

*Figure 8. Parallel engine. *: register; ----: divider*

RECONFIGURABLE ARCHITECTURE FOR GENERAL PURPOSE FUZZY PROCESSING

The overall fuzzy inference engine can be implemented using a few basic modules: min and max modules are the most commonly used. A general purpose module which can perform both min and max operations with a mode input that determines the mode of operation (either min or max) can be used. However, in order to optimize designs, it is preferable to have specific modules that perform only min or max operation. Other modules that are also used are shift registers, counters, RAM and ROM. With the computation scheme described above, modules for performing addition and magnitude comparison are also required.

A proper design environment requires the support of a good module library. For designing application-specific VLSI systems, by limiting the domain of applications it is possible to create a design environment such that all the proper resources are available to support our design acivities. In the case of inference engines, the module library should as much as possible cover a complete range of modules that will be used like min, max and counters. Fortunately, from the standpoint of practicality in real world applications, it is not necessary to have a big library with a huge number of modules. Other modules that are structurally regular like RAM, ROM and shift registers can be automatically generated.

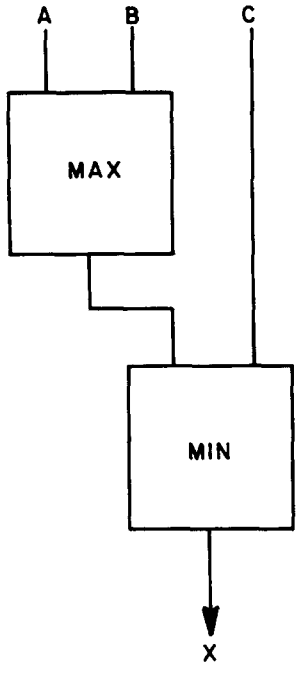

Figure 9 (A) or *(B)* and *C expression*

Depending on the expression in the inference rules, the architecture of the inference engine may have to be reconfigured. So far, the architecture described takes into account rules with single variable antecedent. When the antecedent contains more than one variable, such as 'x = A and y = B', 'x = A or y = B', 'x = A or y = B and z = C', etc., we need to modify the existing inference engine to handle the multiple variables. As it turns out, the logical operators *and* and *or* correspond to min and max operation respectively. As shown in Figure 9, for an expression with antecedent of the form '(x = A or y = B) and z = C', the output labelled X becomes the input instead of A_i in Figures 6 and 8.

CONCLUSION

A reliable approach to approximate reasoning in expert systems has been presented here. With this improvement, it is not necessary for the fuzzy data to be normalized, which is not the case in conventional fuzzy reasoning schemes. Using this improved scheme, the presence of noise does not affect the correctness of the decision. This is achieved by computing the similarity factor σ between two fuzzy sets. Similarity is a measure of resemblance between two fuzzy sets, and its value deter-

mines the level of activation of the conclusion part of each rule. The use of multi-dimensional membership fuzzy logic for certain real-time expert systems has also been introduced. To implement the fuzzy inference engine, two different architectures using essentially the same components were presented. The architectures described are also easily reconfigurable to handle complex fuzzy expressions for more fruitful real-time applications.

REFERENCES

1 **Ito, N and Shimizu, H** 'Data-flow based execution mechanisms of parallel and concurrent Prolog' *New Gener. Comput.* Vol 3 (1985) pp 15–41

2 **Onai, R, Shimizu, H, Masuda, K, Matsumoto, A and Aso, M** 'Architecture and evaluation of a reduction-based parallel inference machine: PIM–R' *Lecture Notes in Computer Science* Springer-Verlag, FRG (1986)

3 **Yokota, H and Itoh, H** 'A model and an architecture for a relational knowledge base' *Proc. 13th Int. Symposium on Comput. Architecture* (1986) pp 2–9

4 **Lusk, E L and Overbeek, R A** *The Automated Reasoning System ITP* Argonne National Laboratory, Argonne, IL, USA (1984)

5 **Togai, M and Watanabe, H** 'Expert system on a chip: an engine for real-time approximate reasoning' *IEEE Expert* Vol 1 No 3 (Autumn 1986) pp 55–62

6 **Takefuji, Y and Avampato, J** 'Paramodulation Inference Engine' Internal Report, CAISR, Case Western Reserve University, USA

7 **Zadeh, L A** 'Outline of a new approach to the analysis of complex systems and decision processes' *IEEE Trans. Syst. Man & Cybern.* Vol 3 (1973) pp 28–45