International Journal of Fuzzy Logic and Intelligent Systems, vol. 1, no. 1, June 2001 pp. 119-124

Parallel Fuzzy Inference Method for Large Volumes of Satellite Images

Sang Gu Lee

Department of Computer Engineering, Hannam University
133 Ojung-Dong, Daeduk-Gu, Taejon 306-791, Korea

Abstract

In the pattern recognition on the large volumes of remote sensing satellite images, the inference time is much increased. In the case of the remote sensing data [5] having 4 wavebands, the 778 training patterns are learned. Each land cover pattern is classified by using 159,900 patterns including the trained patterns. For the fuzzy classification, the 778 fuzzy rules are generated. Each fuzzy rule has 4 fuzzy variables in the condition part. Therefore, high performance parallel fuzzy inference system is needed. In this paper, we propose a novel parallel fuzzy inference system on T3E parallel computer. In this, fuzzy rules are distributed and executed simultaneously. The ONE_TO_ALL algorithm is used to broadcast the fuzzy input to the all nodes. The results of the MIN/MAX operations are transferred to the output processor by the ALL_TO_ONE algorithm. By parallel processing of the fuzzy rules, the parallel fuzzy inference algorithm extracts match parallelism and achieves a good speed factor. This system can be used in a large expert system that has many inference variables in the condition and the consequent part.

Key Words : Fuzzy logic, Remote sensing data, Parallel fuzzy inference, Parallel computers

I. Introduction

Fuzzy theory is being applied to many fields such as an expert system, fuzzy knowledge-based system, fuzzy database system, a robot development, an industrial engineering, fuzzy computer development, pattern recognition, and fuzzy control. For fuzzy logic control, the performance of the system is dependent upon the methods of fuzzification, inference, and defuzzification. The inference method especially affects the performance of the system. It basically requires parallel characteristics and is possible in parallel processing. During the past decade, several specific hardware systems for fast fuzzy inference have been developed. A VLSI fuzzy controller with reconfigurable, cascadable architecture was developed by Watanabe et al. [14]. After the research of Togai and Watanabe [11, 12], faster and more elaborate fuzzy inference hardware systems have been developed [2, 9, 15]. The fuzzy logic controller with the optimized memory organization was proposed in [1]. An architecture of a 64-bit fuzzy inference processor was presented by Ungering [13]. In [3], parallel, high-speed PC fuzzy control for a highly adaptive fuzzy architecture was developed. In [4], a parallel fuzzy hardware system for fuzzy information processing was proposed. This architecture has SIMD structure, which consists of two parts; system control unit (Main Controller) and arithmetic units (Fuzzy Processing Elements). It provides various fuzzy set operations as well as various inference functions.

However, most of them are dedicated to be executed sequentially, rule by rule, although they have parallel processing manners in operations of Min/Max (AND/OR) computations. Therefore, it is necessary to distribute and parallelize fuzzy rules themselves in order to improve the inference speed.

In [6] and [7], as an implementation method of parallel fuzzy architecture, two types of architectures for parallel fuzzy inference are proposed. One is the parallel fuzzy architecture using Transputers. In this, only active rules are detected by index numbers of linguistic labels in the condition part. This system will be applied to build powerful architectures for control applications, like robot control, with time-critical sensor integration. Another is the cascadable parallel architecture for fuzzy information processing. In this, fuzzy inferences are processed in parallel with respect to antecedents and consequents. Each FPE (Fuzzy Processing Element) executes only the operations of the i-th antecedent and the i-th consequent. This parallel fuzzy inference architecture can be used in a system requiring a rapid inference time in real-time system and/or an expert system that has many inference variables in the condition and the consequent part.

In a large fuzzy database system or MIMO system that has large rules or large volumes of fuzzy data, however, the above two architectures do not fit so well in point of the memory capacities and throughput. Therefore, very high performance fuzzy inference system is essential for the large volumes of fuzzy data.

To meet this challenge, in this paper, we propose a novel parallel fuzzy inference system on parallel

Manuscript received March 15, 2001; revised May 1, 2001. This work was supported by grant No. 2000-1-30 300 -007-2 from the Basic Research Program of the Korea Science & Engineering Foundation.

computer. The parallel computer is an ideal parallel architecture because of its powerful interconnection feature. Additionally, regularity, symmetry, logarithmic connectivity, partitionability, and simple routing makes the system superior to some of the other networks. Also, important communication operations such as broadcasting and personalized communication can be supported efficiently in the parallel computer.

Hence, the Cray T3E is a choice for the architecture for large-scale parallel fuzzy inference machine. This can reduce the total amount cost by reducing inference time with a hardware generality and a simple algorithm. It can be used in a large expert system including fuzzy database and a complex system that is required for very large rules. By parallel processing of the fuzzy rules on parallel computer, the parallel fuzzy inference algorithm extracts match parallelism and achieves a good speed factor without substantially degrading the performance of the algorithm.

II. Fuzzy control system with large rules

Fuzzy logic is effective to express uncertain and ambiguous phenomena in the real world since it is much similar to human thinking structure or features of natural language than existing logic structures. So the fuzzy logic has been researched in many fields such as control, expert system, database system, several industrial applications, and academic fields.

Fuzzy control rules stored in a control systems knowledge database are expressed by IF-THEN format as the state evaluation type of the fuzzy control rules. In the case of having many input and output variables, it is good to parallelize and to inference many fuzzy rules simultaneously.

In fuzzy computing, we can summarize the comparisons of architectures, features and application areas in parallel fuzzy inference systems as following as Table 1 [6, 7, 8].

Table 1. Comparisons of parallel fuzzy inference systems

Architecture system	Hypercube	Transputers	FPGA
Degree of parallelism	Fuzzy Rule Partitioned rules	Fuzzy Rule	Partial fuzzy rule
Needed number of processors	N (= 2n)	N (= 2n)	Maximum of the number of variables in IF and THEN parts
Inferenced rules	All rules	Only active rules	All rules
Size level	Parallel computer	Board level	Chip level
Merits	Large rules (above 1024 rules)	Real-time inference	Many variables, MIMO system
Applications	Expert system, database system	Robot control, image processing	Expert system

In fuzzy inference of the complicated control system such as an large expert system, if the number of variables in the condition part and/or consequent part increase, inference time is also increase. Therefore, it is necessary to develop a new method of parallelizing fuzzy inference. The parallel fuzzy inference system using parallel computer is well suited. This system is a parallel machine with independent processing elements that execute fuzzy inference programs concurrently and pass messages over a regular communication structure.

III. Parallel fuzzy inferencing

3.1 Distribution of the rules

The number of rules required in the inferencing is increased, as characteristics of the control objects became complex. A parallel search in the inference part makes the controller faster rather than sequential search to execute a fuzzy control for an object. If the inferencing part is processed in a sequential manner, there exists a disadvantage that a result of real-time inferencing cannot be obtained due to an increase of inference time.

The inferred rules are generated to control correctly the control objects and execute the parallel fuzzy inference. The generated rules should be stored distributedly in each node of the parallel computer. To distribute the rules into each node, m modulo function is introduced. For the n-dimensional cube, the cube is consisted of m (=2n) numbers of nodes. These nodes are from P0 to Pm-1. All rules have own addresses in the m nodes. If local memory of each node is called to memory bank that hereinafter is mentioned by MB, MBi expresses the MB of the i-th node. Using the m modulo function, 0th result of the m modulo function is stored in MB0 of the P0 node, 1st one in MB1 of the P1 node, and (m-1)-th one in MBm-1 of the Pm-1 node.

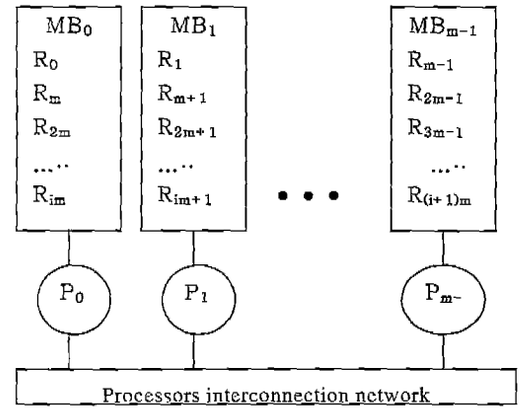

Fig. 1. Distribution of fuzzy rules

Fig. 1 shows the distribution of inference rules. The rules are stored distributedly in the MBs of many nodes, so this kind of distributed storage has two advantages for

process of the inferencing performance. The first advantage is that it is possible to execute the parallel processing of a process to find the correct rules in the fuzzy inferencing using structured features of the computer system with the general parallel computer structure. Each node stores only R/m numbers of rules among the total R rules. Once the rules are distributed, additional rules can be easily stored to proper MBi by using the modulo function. The second one is that it is efficient in the parallel processing of Min operations for fuzzy inference.

3.2 Broadcasting algorithm and parallel Min operation

In the status of the distributed storage of the rules, if the fuzzy input values are come to P0 (input node) as an input vector X, X is routed from P0 to all the other nodes through the network by ONE_TO_ALL broadcasting algorithm. This algorithm is shown in Fig. 2. The input vector X is composed of input fuzzy data. Each node receives the input vector X with the same value, and routes the input vector X to adjacent next node using store-and-forward routing method until the broadcasting algorithm is finished.

In the broadcasting algorithm of this system, the input vector X can be routed from P0 to all nodes within log m steps. In the general n-dimensional cube, an address of each node can be expressed by a bit string consisting of n bits. The ONE_TO_ALL broadcasting algorithm is masking the bit string by using AND operation. XOR operation is used to select a next destination node to route the input vector X or a source node for the input vector X to be routed. For fuzzy inference, parallel Min operation between fuzzy input data and membership functions in the condition part is processed.

3.3 Max operation and defuzzification

The Min and Max operations can not be processed simultaneously since the Min operation can be processed parallelly in each node using control rules that is found in the MB of the node by referring input vector X in the node. However, in general, the Max operation can be processed by using all results of the Min operations of each node after finishing the Min operations, and so the parallel processing of the Max operation is impossible.

However, parallel processing of a Max operation in this research is partially possible since the Max operation is partially executed step by step during routing a vector Y to Pm-1 node. This output node is directly connected to a controlled object and is only for defuzzification operation. The vector Y is the result of Min operations of each node of which parallel computer is consisted. The ALL_TO_ONE algorithm is also finished within log m steps similar as the ONE_TO_ALL broadcasting algorithm. Fig. 3 represents the ALL_TO_ONE algorithm. The ALL_TO_ONE algorithm is processed simultaneously in all nodes. After collecting in Pm-1 the results of the Min operation, the final Max operation is performed.

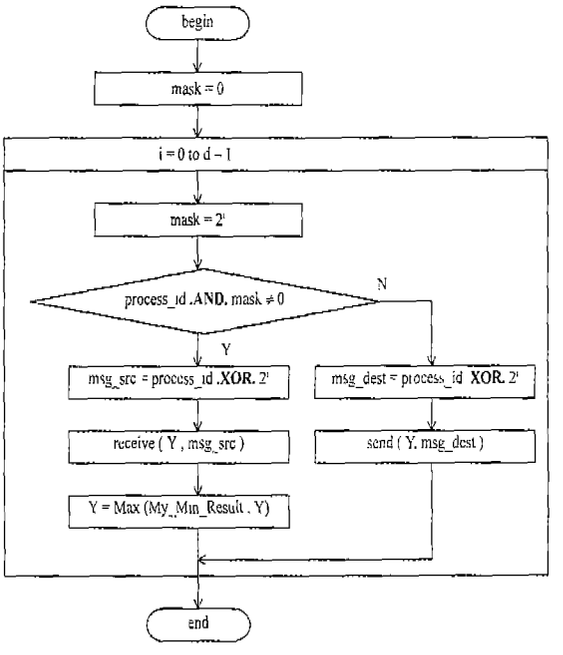

Fig. 2. ONE_TO_ALL broadcasting algorithm

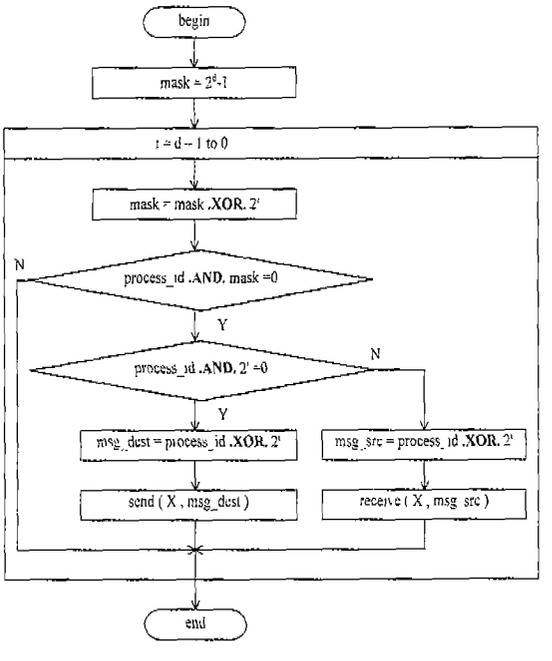

Fig. 3. ALL_TO_ONE algorithm

In the proposed system, the input and the output nodes are separated. The purpose of it is that P0 receives the input and Pm-1 executes a role of output part of the fuzzy control for objects. The isolation of the input and output node is to get load balancing of processors and reduce unnecessary routing and waiting. It also gives a new possibility of pipelining in the proposed architecture.

Fig. 4 shows the sequences for the input vector X and

vector Y of the Min operation result to be routed between nodes by the above two algorithms and ones for the output vector Z of the defuzzification result in Pm-1 to be output. In the Pm-1 node, the Max operation is executed with a Min operation result of Pm-1 and vector Ys of Min operation results which are routed from all nodes except Pm-1 node itself. The defuzzification operation is executed with the results of the Max operations and uses a method of center of gravity.

The overall processes of our parallel fuzzy inference system are expressed as follows:

1. P0 node (input node) receives the input vector X.
2. The input vector X is routed to all nodes by using the ONE_TO_ALL algorithm.
3. In each node, Min operations are performed between the input vector X and the input fuzzy variable. The Min operation results are stored in an intermediate result vector Y.
4. The vector Y in each node is routed to Pm-1 by using ALL_TO_ONE algorithm. At that time, Max sub-operations are partially performed during routing to each node.
5. In the Pm-1 node (output node), the Max operations are processed with the Min operation result of itself and the value of the intermediate result vector Y.
6. In the Pm-1 node, the defuzzification is executed by using the Max operation results. The output control vector Z, the defuzzification result, is output to the controlled object.

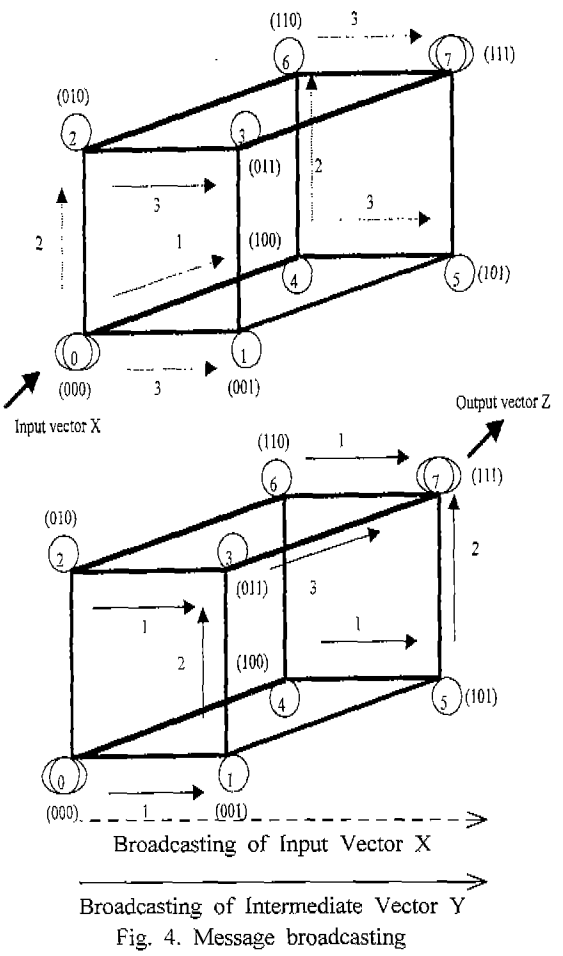

Fig. 4. Message broadcasting

IV. Performance evaluation

We use the Cray T3E parallel computer for the experiments of the parallel fuzzy inference system. The performance of the proposed parallel fuzzy inference system which uses ONE_TO_ALL and ALL_TO_ONE algorithms were compared on a Cray T3E computer system. The T3E consists of 128 nodes, each of DEC alpha 21164 RISC processor and 128 Mbyte of memory. The processor communication network has a 3-D Torus organization. By modulo-m function the entire fuzzy rules are distributed over the T3E such that each memory bank has equal size of the partitioned rules.

To transfer the data between processors, in T3E, there are 3 message passing libraries such as MPI(Message Passing Interface), PVM(Parallel Virtual Machine) and SHMEM(Shared Memory). In this paper, we use the MPI.

In the pattern recognition on the large volumes of remote sensing satellite images, the inference time is much increased. In the case of the remote sensing data [5] having 4 wavebands, the 778 training patterns are learned. Each land cover pattern is classified by using 159,900 patterns including the trained patterns. For the fuzzy classification, the 778 fuzzy rules are generated. Each fuzzy rule has 4 fuzzy variables in the condition part. Therefore, high performance parallel fuzzy inference system is needed. For the evaluation of the parallel fuzzy inference for the remote sensing satellite images on parallel computer, we designed a fuzzy expert system that has 4 variables in the antecedent part and 1 variable in the consequent part. The number of the linguistic terms in each variable is 12. In this system, we select only 778 rules. We use FORTRAN(CF90) language for the parallel program for each CPU. We have considered three cases of having CPUs(16, 32 and 64 nodes) and tested. The data transfer functions are SEND(data, N, B) and RECV(data, N, B).

NQS script that runs the fuzzy inference Fortran program by 16 nodes is as followings.

```
        < NQS script >
#!/bin/c f
#QSUB r fuzzy
#QSUB 1 mpp_p=16
#QSUB IT 60
#QSUB ro re eo o fuzzy.out
.....
```

```
ja
....
....

module load mpt
f90 o fuzzy.exe fuzzy.f
# -- run the executable on 16 processors
mpprun n16 ./fuzzy.exe
```

In our experiments, we measured the times to reach the final computations and gained the results in the case of 16, 32 and 64 processors, respectively. Table 2 shows the speedups of the three cases.

Table 2. Speedup

No. of processors	Speedup
16	11.8
32	17.3
64	28.7

As a result of experiments, maximum inference speed including defuzzification process is 600 MFLIPS (Mega Fuzzy Logic Inference per Second) in the case of 32 CPUs. Speedup factors are 11.8, 17.3 and 28.7 for the case of 16, 32 and 64 CPUs, respectively.

In this system, especially, each node processes exactly R/p (total rules / number of processors) elements. Therefore, computational load is well distributed among the nodes, which contributes to the speedup of the algorithm. The delay incurred by inter-processor message communication is due to two components: the message setup time and the actual transfer time. In most of the commercially available message passing multiprocessors, TSetup >> TB, where TB is the one-byte transfer time.

V. Conclusions

When a control system requiring complex objects is configured for the fuzzy inferencing, the number of fuzzy rules is much increased due to the increase of necessary input and output variables that should be considered into.

In this paper, we have proposed a parallel fuzzy inference system for the large volumes of remote sensing data on parallel computer. In this, fuzzy rules are distributed to local memories in each node by using modulo function, and several fuzzy rules are executed simultaneously. The ONE_TO_ALL algorithm is used to broadcast the fuzzy input data to the all nodes. The results of the MIN/MAX operations are transferred to the output processor by the ALL_TO_ONE algorithm.

As the number of fuzzy rules grows, the speedup of the system grows faster than that of having small rules. That is well suitable to the large volumes of data or large expert systems concerning with fuzzy data.

This parallel fuzzy inference system can be used in a system requiring a rapid inference time in real-time and/or a large expert system that has many inference variables in the condition and the consequent part. Especially, this architecture system is very efficient to MIMO and large scale fuzzy database systems. The advantage of our fuzzy inferencing system is that reconfiguration of hardware structure is not necessary and hardware independent parallel fuzzy inferencing is possible due to the parallel processing of fuzzy rules. There is also a new possibility of 3-stage pipeline processing in the proposed architecture since the input and output nodes are isolated.

References

[1] T. Chiueh, "Optimization of fuzzy logic inference architecture", *IEEE Computer Magazine*, pp. 67-71, May 1992.

[2] R. Corder, "A High-speed fuzzy processor", *Proc. 3rd IFSA Congress*, pp. 379-381, 1989.

[3] A. Jaramillo-Botero, "Parallel, high-speed PC fuzzy control", *IEEE Micro*, p. 63, Dec. 1995.

[4] Y. D. Kim and H. Lee-Kwang, "High speed flexible fuzzy hardware for fuzzy information processing", *IEEE Trans. on Syst., Man, Cybern.*, vol. 27, no. 1, pp. 45-56, Jan. 1997.

[5] S. G. Lee and J. G. Han, "A Neuro-fuzzy Classifier for Land Cover Classification", 1999 *IEEE Int. Fuzzy System Conference*, vol. 2, pp. 1063-1068, 1999.

[6] S. G. Lee and K. Akizuki, "A parallel inference method of fuzzy rules using Transputers", *T. IEE Japan*, vol. 118-C, no. 7/8, pp. 1176-1182, 1998.

[7] S. G. Lee and K. Akizuki, "Design of an effective parallel architecture for fuzzy information processing", *T. IEE Japan*, vol. 118-C, no. 7/8, pp. 1190-1195, 1998.

[8] S. G. Lee, "Parallel inference method of fuzzy rules", *Journ. of Electrical Engineering and Information Science*, vol. 4, no. 3, pp. 357-363, Jun. 1999.

[9] M. Sasaki, F. Ueno, and T. Inoue, "7.5 MFLPS fuzzy microprocessor using SIMD and logic-in-memory structure", *Proc. IEEE Int.* Conf. on Fuzzy Systems, pp. 527-534, 1992.

[10] M. Sugeno and M. Nishida, "Fuzzy control of model car", *Int. Journ. Man Mach. Studies*, vol. 16, pp. 103-113, 1985.

[11] M. Togai and H. Watanabe, "Expert system on a chip: An engine for real-time approximate reasoning", *IEEE Expert*, vol. 1, pp. 55-62, Aug. 1986.

[12] M. Togai and H. Watanabe, "A VLSI implementation of fuzzy inference engine: Toward an expert system on chip", *Information Sciences*, vol. 38, pp. 147-163, 1986

[13] A. P. Ungering and K. Goser, "Architecture of a 64-bit fuzzy inference processor", *Proc. FUZZ-IEEE WC-CI*, vol. 26, no. 6, pp. 1776-1780, 1994,

[14] H. Watanabe, W. D. Dettloff, and K. E. Yount, "A VLSI fuzzy logic controller with reconfigurable architecture", *IEEE J. of Solid State Circuits*, vol. 25, pp. 376-382, Apr. 1990.

[15] H. Watanabe, "RISC approach to design of fuzzy processor architecture", *Proc. IEEE Int. Conf. on Fuzzy Systems*, pp. 431-441, 1992.

Sang Gu Lee

was born in Seoul, Korea on September 22, 1955. He received the B.S. degree in electronics engineering from Seoul National University, Seoul, Korea in 1978, and the M.S. degree in Computer Science from KAIST, Korea in 1981.

He received the Ph.D. degree in Electrical Electronics and Computer Engineering from Waseda University, Tokyo, Japan. From 1996 to 2000, he was a member of international program committee in IASTED. Since 1983, he has been a professor in division of Computer Engineering, Hannam University, Taejon, Korea. His current research interests are in parallel processing, parallel architecture and parallel neuro-fuzzy computing.

Phone : +82-42-629-7551
Fax : +82-42-487-9335
E-mail : sglee@eve.hannam.ac.kr