

REPRINTED FROM:

PARALLEL COMPUTING

Parallel Computing 19 (1993) 1283–1301
North-Holland

PARCO 800

Fuzzy arithmetic on systolic arrays

Hassen Dhrif and Dilip Sarkar *

Department of Mathematics and Computer Science, University of Miami, Coral Gables, FL 33124, USA

Received 15 April 1991

Revised 30 March 1992, 18 May 1993



NORTH-HOLLAND – AMSTERDAM

PARCO 800

Fuzzy arithmetic on systolic arrays

Hassen Dhrif and Dilip Sarkar *

Department of Mathematics and Computer Science, University of Miami, Coral Gables, FL 33124, USA

Received 15 April 1991

Revised 30 March 1992, 18 May 1993

Abstract

This paper presents parallel algorithms for fuzzy arithmetic operations on systolic arrays. A large part of the arithmetic on fuzzy numbers is referred to as fuzzy convolutions, which take $O(n^2)$ time steps when implemented sequentially. In this paper, we present two parallel algorithms. The first algorithm runs on a linear array machine in $O(n)$ time, using $O(n)$ processors, and thus, achieving optimal time and cost. The second algorithm runs on a mesh-of-trees type of computer architecture in $L(\log n)$ time steps, using $O(n^2)$ processors. The latter, although not optimal achieved high speedup. Both can be adapted to handle larger linear fuzzy numbers of fuzzy numbers with higher dimension.

Keywords. Systolic arrays; fuzzy arithmetic; linear array of processors; mesh-of-trees of processors; speed-up analysis.

1. Introduction to fuzzy numbers

The notion of fuzzy numbers relies on what is known as the fuzzy theory. The original concept, developed in the mid-'60s by Lotfi Zadeh, a professor of computer science at the University of California, Berkeley, is that things in the real world do not fall into the neat, crisp categories defined by the traditional set theory.

This mathematics turns out to be very useful for controlling robots, machine tools and various electronic systems. A conventional air conditioner, for example, recognizes only two basic states: too hot or too cold. A fuzzy air conditioner, by contrast, would recognize that some room temperatures are closer to the human comfort zone than others. Its cooling system would begin to slow down gradually as the room temperature approached the desired setting. The result, a very comfortable room and a small electric bill.

Another way of putting fuzzy numbers into practice was established by AT&T Bell Laboratories, where researchers put an entire expert system on a computer chip and gave it a considerably faster speed than that of conventional expert systems [7]. While many researchers in the AI community have doubted that *fuzzy logic* (a logic system based on fuzzy variables instead of Boolean variables) actually makes a difference in practice, the AT&T chip seems to show fuzzy logic's power. It would be difficult, if not impossible for a more conventional expert system with many rules, a large memory and processor requirements to be squeezed onto a chip. Fuzzy logic is also currently used in designing automatic automobile transmission and anti-skid brake systems [8]. The idea is to make the transmission shift the way a person would when she or he subconsciously weighs different factors. The new transmission is controlled by a microprocessor that assigns values to about half a dozen variables, including the car's speed, the rate of acceleration, and the rate of change in opening

* *Corresponding author.* Email: sarkar@mthvax.cs.miami.edn

the throttle. Each value is then given a weight and the microprocessor calculates these weights and makes the decision on whether to shift or not. The anti-skid braking system uses similar principles. Here, the problem is to determine the maximum amount of braking pressure that can be applied without making the wheels lock. In [5], Kauffmann and Gupta presented other applications of fuzzy numbers in engineering and management science. Incorporation of fuzzy numbers in expert system design were also presented in [6].

When should we think fuzzy? If a system can be represented by a simple equation, it should probably be modeled by a straightforward control technique. However, if we are dealing with very complex parameters, that can be easily implemented as a series of IF-THEN-ELSE rules, then fuzzy logic is the way to go. In article [9], Bell presents with more detail the differences between traditional, proportional, integral and derivative control systems and their fuzzy associates. He also gives a list of market products using fuzzy theory.

1.1. Definitions and main properties

Fuzzy numbers are defined and described in [4]. An interval of confidence is a practical and logical process for treating uncertainty with whatever information that is available. This information can be *objective* (we are certain that the dimension has a position between the two measured data), or *subjective* (the information is obtained from experience or from the opinion of experts).

Another concept is worth mentioning before introducing fuzzy numbers. It is called the *level of presumption*. Let us assume, for example, that a certain event is to take place between June 10 and June 25. This is an interval of confidence. However, let us assume that this event is to occur on June 20, a possible date. If we wish, we may assign two levels of confidence to these two situations, 0 for June 10 and June 25 and 1 for June 20. These two levels of confidence are in fact levels of presumption, and we can present them by [0,1]. As a matter of fact we can extend those two intervals to a much larger set of interval such as

$$\forall a_1, b_1 \in R; \quad \text{and } \forall \alpha_1, \alpha_2 \in [0,1]; \quad (\alpha_1 < \alpha_2) \Rightarrow ([a_1^{(\alpha_2)}, a_2^{(\alpha_2)}] < [b_1^{(\alpha_1)}, b_2^{(\alpha_1)}]). \quad (1)$$

This means that if α_i increases, the interval of confidence never increases. This coupling between the level α_i of presumption and the interval of confidence at level α_i will be a way of defining the concept of an *uncertain number or fuzzy number*.

The concept of an uncertain number or a fuzzy number can be defined in R , R^+ , Z , N or in every referential set that is totally ordered or linearly ordered. This paper deals with discrete fuzzy numbers only, since continuous numbers carry more programming complications and are beyond the scope of this project.

Following [4], we introduce the concept of fuzzy numbers from the viewpoint of fuzzy set theory.

Let E be a referential set (for example, Z (the set of integers) or N (the set of natural numbers)). An ordinary subset A , of this referential set is defined by its characteristic function, μ such that:

$$\forall a \in E: \quad \mu_A(a) \in \{0,1\}, \quad (2)$$

which shows that an element of E belongs to, or does not belong to A according to the value of the characteristic function (1 or 0).

For the same referential set E a fuzzy subset A will be defined by its characteristic function, called the membership function, which takes its values in the interval $[0,1]$ instead of the binary set $\{0,1\}$.

$$\forall a \in E: \quad \mu_A(a) \in [0,1], \tag{3}$$

That is, the elements of E belong to A with a level located in $[0,1]$.

A fuzzy subset $A \subset Z$ is *convex* if,

$$\forall a \in Z \quad A_\alpha = \{a \mid \mu_A(a) \geq \alpha\}, \quad \alpha \in [0,1], \tag{4}$$

is convex; that is, if it is a closed interval of Z .

A fuzzy subset $A \subset Z$ is *normal* if and only if

$$\forall a \in Z \quad \bigvee_a \mu_A(a) = 1. \tag{5}$$

This means that the highest value of $\mu_A(a)$ is equal to 1. This maximum may or may not be unique.

Now a definition for uncertain or fuzzy numbers can be presented. A *fuzzy number* in Z is a fuzzy subset of Z that is convex and normal. Thus a fuzzy number, although, it may be considered as a generalization of the interval of confidence, is not a random variable. A random variable is an objective datum, whereas a fuzzy number is a subjective datum. It is a valuation, and not a measure.

Now that fuzzy numbers have been defined, one can describe some important operations on them. There is more than one way to define these operations. But the ones chosen, are best suitable for the discrete model to be programmed.

1.2. Convolutions of fuzzy numbers

All operations on fuzzy numbers such as addition, subtraction, multiplication, division, minimum, and maximum are called max-min convolutions. They all involve elements of Z (eventually Z^+ only). The following is a list of the max-min convolutions of concern in this paper. For examples, see *Figs. 1*, and *2*. The symbols \wedge and \vee denote the minimum and the maximum operations, respectively.

$$\forall a, b, c \in R:$$

$$\mu_{A(+)B}(c) = \bigvee_{c=a+b} (\mu_A(a) \wedge \mu_B(b)). \tag{6}$$

$$\mu_{A(-)B}(c) = \bigvee_{c=a-b} (\mu_A(a) \wedge \mu_B(b)). \tag{7}$$

$$\mu_{A(\cdot)B}(c) = \bigvee_{c=a \cdot b} (\mu_A(a) \wedge \mu_B(b)). \tag{8}$$

$$\mu_{A(/)B}(c) = \bigvee_{c=a/b} (\mu_A(a) \wedge \mu_B(b)). \tag{9}$$

$$\mu_{A(\wedge)B}(c) = \bigvee_{c=a \wedge b} (\mu_A(a) \wedge \mu_B(b)). \tag{10}$$

$$\mu_{A(\vee)B}(c) = \bigvee_{c=a \vee b} (\mu_A(a) \wedge \mu_B(b)). \tag{11}$$

What we have defined is fuzzy numbers in one dimension, but these can be extended to two or higher dimensions. Again the domain of values is restricted to a discrete ordered set such as Z or Z^+ .

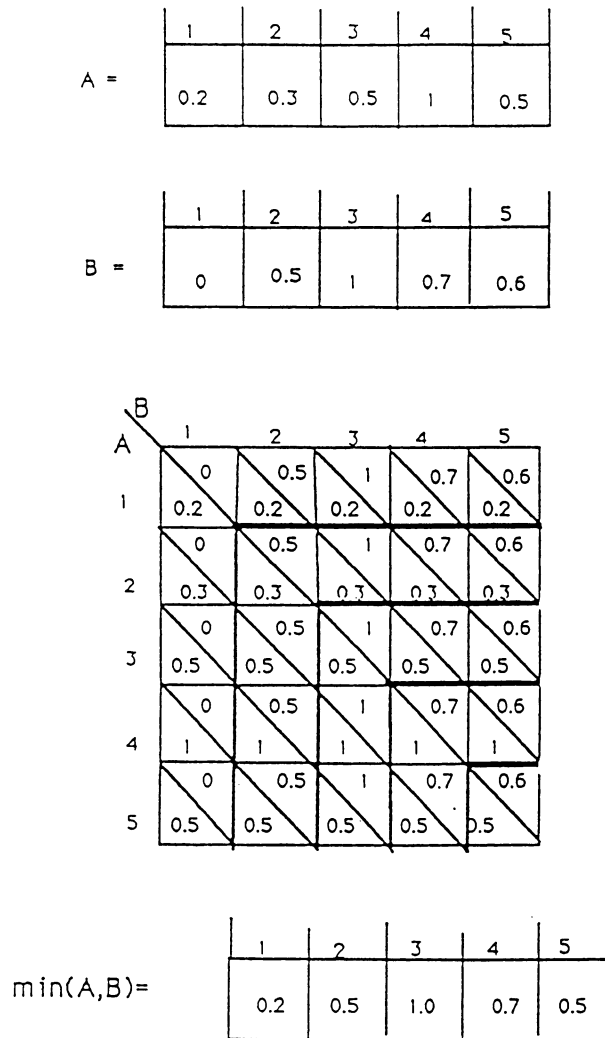


Fig. 1. The minimum convolution of 2 fuzzy numbers.

Section 2 presents the sequential algorithms for the minimum and addition of two linear fuzzy numbers a and b . Section 3 presents the data dependency for each of the algorithms in Section 2. Section 4 presents mapping of the operations in Section 2 into linear systolic arrays. Section 5 presents the algorithms for the same operations on a novel class of modified-mesh-of-trees.

2. The sequential algorithms

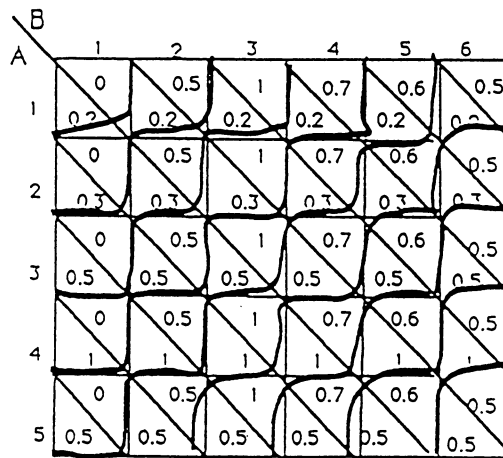
For completeness and better understanding, before introducing parallel algorithms, the corresponding sequential algorithms are presented. The algorithms are self explanatory and given in Figs. 3 and 4. Each algorithm requires a memory space of size $m + n$, where $m = |a|$ and $n = |b|$, and takes $O(n^2)$ time steps to be executed. Only fuzzy minimum and addition

A =

1	2	3	4	5
0.2	0.3	0.5	1	0.5

B =

1	2	3	4	5	6
0	0.5	1	0.7	0.6	0.5



A + B =

2	3	4	5	6	7	8	9	10	11
0.0	0.2	0.3	0.5	0.5	1.0	0.7	0.6	0.5	0.5

Fig. 2. The addition convolution of 2 fuzzy numbers.

are presented, since maximum and subtraction differ from the previous two only in their computation order.

3. The data dependency

Let M be a temporary variable defined such that

$$M_{i,j} = \vee(a_i, b_j).$$

ear
in
ys.
sh-

he
nd
a|
on

Algorithm SeqFuzzymin(a, b, c)

Input The 2 fuzzynumbers a and b are represented as 2 linear lists $a[1...n]$ and $b[1...n]$.

Output The fuzzynumber c as the minimum of a and b , represented as $c[1...n]$.

1. For $i := 1$ To n Do
2. $c_i := 0$;
3. For $j := i$ To n Do
4. $temp := \wedge(a_j, b_i)$
5. If $temp > c_i$ Then
6. $c_i := temp$
7. End.

Fig. 3. The sequential algorithm for fuzzy minimum.

A second index is added to each entry in c to illustrate the cohesion between the elements involved in computing the final value of that entry. The entry $c_{i,j}$ is defined in terms of the following recurrence equations for the two fuzzy operations considered:

Algorithm SeqFuzzyadd(a, b, c)

Input The 2 fuzzynumbers a and b represented as 2 linear lists $a[1...m]$ and $b[1...n]$.

Output The fuzzynumber c as the sum of a and b , represented as $c[1...m+n-1]$.

1. For $i := 1$ To $n-1$ Do
2. For $j := 1$ To $\lfloor \frac{i}{2} \rfloor$ Do
3. $k := 2j - 1$;
4. $temp := \vee(\wedge(a_k, b_{i-k-1}), \wedge(a_{k+1}, b_{i-k}))$;
5. If $temp > c_i$ Then
6. $c_i := temp$
7. For $i := n$ To m Do
8. For $j := 1$ To $\lfloor \frac{m}{2} \rfloor$ Do
9. $k := 2j - 1$;
10. $temp := \vee(\wedge(a_k, b_{i-k-1}), \wedge(a_{k+1}, b_{i-k}))$;
11. If $temp > c_i$ Then
12. $c_i := temp$
13. For $i := m+1$ To $m+n-1$ Do
14. For $j := 1$ To $\lfloor \frac{i-n}{2} \rfloor$ Do
15. $k := 2j - 1$;
16. $temp := \vee(\wedge(a_k, b_{i-k-1}), \wedge(a_{k+1}, b_{i-k}))$;
17. If $temp > c_i$ Then
18. $c_i := temp$
19. End.

Fig. 4. The sequential algorithm for fuzzy addition.

Fuzzy minimum:

$$c_{i,j} = \wedge (c_{i,j-1}, M_{i,j}, M_{j,i}), \text{ for } 1 \leq i \leq j \leq n$$

$$c_i = c_{i,n}$$

$$c_{i,i-1} = \infty$$

Fuzzy addition:

$$c_{i,j} = \wedge (c_{i,j-1}, M_{k,i-k-1}, M_{k+1,i-k})$$

$$k = 2j - 1, \alpha < j \leq \left\lceil \frac{m}{2} \right\rceil, 1 \leq i < m + n$$

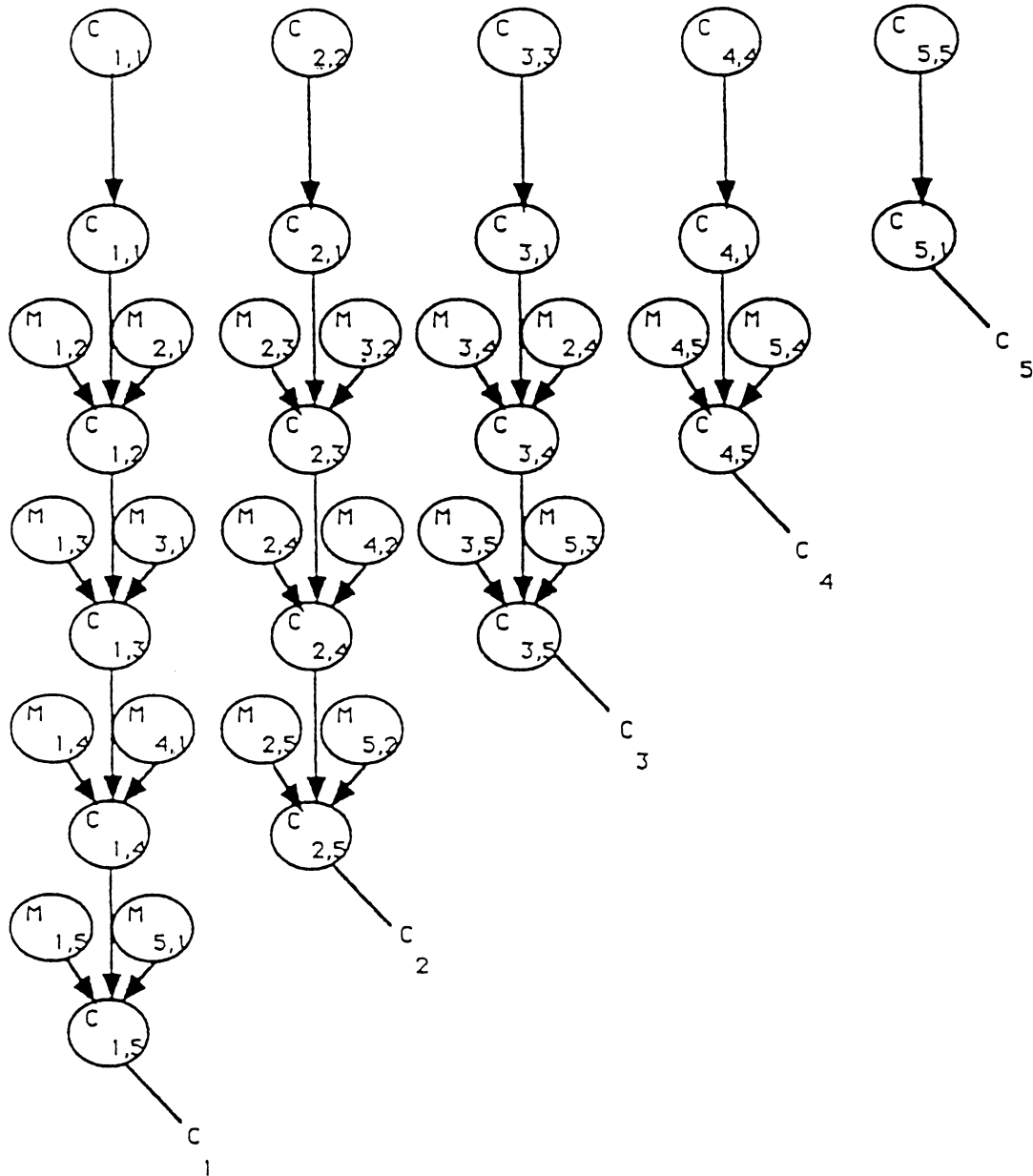


Fig. 5. Data dependency for fuzzy minimum.

$$C_{i,\alpha} = \infty$$

$$\alpha = \begin{cases} \left\lfloor \frac{m-i}{2} \right\rfloor & \text{if } 1 \leq i \leq m \\ 0 & \text{if } m \leq i \leq n \\ \left\lfloor \frac{i-n}{2} \right\rfloor & \text{if } n \leq i \leq m+n \end{cases}$$

The development of the parallel algorithms in the next two sections relies on the recurrence equations defined above. Refer to Fig. 5 for fuzzy minimum data dependency diagram, and similarly Fig. 6 presents the data dependency diagram for fuzzy addition.

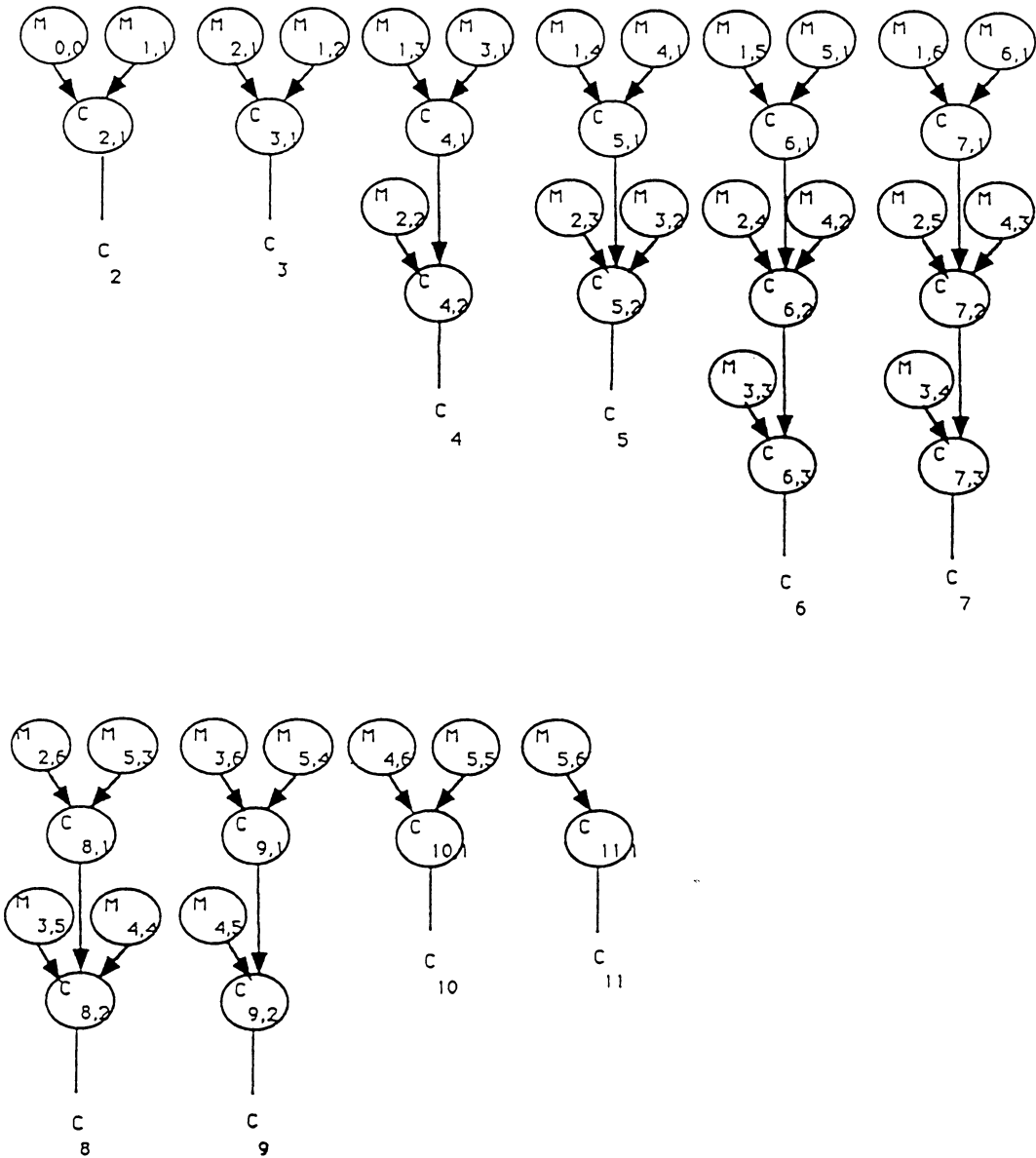


Fig. 6. Data dependency for fuzzy addition.

4. The linear array algorithms

This section presents the linear array algorithms for computing the convolutions introduced in Section 1.2. The rest of this section proceeds by presenting the computation model to be used (hardware), both an informal and a formal description of the algorithms, their RCT diagrams (the term will be defined later) and finally a complexity analysis.

4.1. The computation model

The computation model is the linear array depicted in Fig. 7. The terms cell and processor will be used but they are synonymous.

Each cell has its own memory and control unit, and all cells are synchronized. Every processor knows its number which is contained in a local variable *pid*. Data passes through from left to right. Each cell has its own memory and communicates by a message passing protocol with its neighbors. Cell *k* sends a word of data to cell *k + 1* using a statement of the form

send($P_k, P_{k+1}, \text{data}$)

Similarly, cell *k* receives a word of data from cell *k - 1* using a statement of the form

receive($P_k, P_{k-1}, \text{data}$)

The host is referred to as either processor P_0 or P_{n+1} . The machine operates in an asynchronous MIMD mode. Communications are by rendez-vous, and there is no overlap between computations and communications. This communication scheme will be assumed for during all subsequent systolic architectures presented in the next sections.

4.2. The algorithms

Figures 8 and 9 contain Pascal-like descriptions of the algorithms. Each algorithm is divided into 3 stages: data-entry, data-computation and data-output. First, the two fuzzy numbers *a* and *b* have to be transmitted to all processors; data will then be pipelined from left to right. Each processor will save only the elements to be used in computing the entry in *c* assigned to it. Once every processor holds the appropriate data, it starts the data-computation and eventually the data-output procedures. Note that, the data-entry phase of the algorithm can be merged into the data-computation phase, if multiple data channels are provided by the hardware of the PEs. That way, one channel will be reserved for the partial results, and two others will serve as data pipelines of *a* and *b*. The following is a more detailed discussion about the three stages of the algorithm mentioned earlier for the two operations, fuzzy minimum and addition. Assume that $|a| = m$ and $|b| = n$, and that $n = m$ when computing the minimum and $m > n$ when computing the sum.

Fuzzy minimum: At time $t = i$, ($1 \leq i \leq n + j - 2$), processor P_j , ($1 \leq j \leq n$), is used to receive and transmit (a_k, b_k) such that, $k = i + j - 1$ when $j \leq i$ and processor P_j is idle when $j > i$.

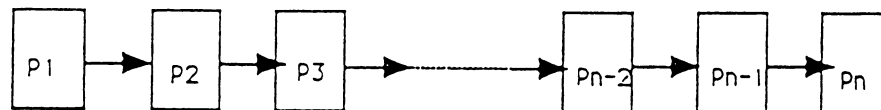


Fig. 7. The linear array model.

Algorithm Linfuzzyminimum;
 (*This algorithm is to be executed in P_j *)

1. $i := \text{clock};$
2. $j := \text{pid};$
3. $t_0 := n + j - 1;$
4. $i' := i - t_0;$
5. If $i > j$
6. Then
7. For $i = j$ To $n + j - 2$ Do
8. $k := i + j - 1$
9. receive (P_j, P_{j-1}, a_k, b_k)
10. Endfor;
11. For $i' = 1$ To j Do
12. $k := i'$
13. $l := j$
14. $c_{k,k-1} := \infty;$
15. If $i \neq t_0$
16. Then
17. receive ($P_j, P_{j-1}, c_{k,l-1}$);
18. $c_{k,l} := \vee(c_{k,l-1}, \wedge(a_k, b_l), \wedge(a_l, b_k));$
19. If $j \neq n$
20. Then
21. send ($P_j, P_{j+1}, c_{k,l}$);
22. Else
23. send (P_j, host, c_k);
24. Endfor
25. End.

Fig. 8. The linear array algorithm for fuzzy minimum.

The variable $t_{0,j}$ is the time P_j uses to finish the data entry phase. At $t = i$, ($n + j - 1 \leq i \leq n + 2j - 1$), processor P_j , ($1 \leq j \leq n$), is:

Computing Receiving Transmitting Time

$$\begin{cases} c_{k,l} \\ k = i' \\ l = j \end{cases} \quad \begin{cases} c_{k',l'} \\ k' = k \\ l' = l - 1 \end{cases} \quad \begin{cases} c_{k'',l''} \\ k'' = k \\ l'' = l \end{cases} \quad 1 \leq i' \leq j$$

$$t_{0,j} \quad n + j - 1$$

$$i' = i - t_0$$

Note that at $n = n + 2j - 2$, processor P_j is not receiving any data. At $t = i'$, ($1 \leq i' \leq n$), processor P_n is transmitting $c_{i'}$ to the host. No partial results need to be saved since computations are pipelined.

Algorithm Linfuzzyaddition;

(*This algorithm is to be executed in P_j *)

1. $i := \text{clock}$
2. $j := \text{pid}$
3. $t_0 := n + j - 1$
4. $t_1 := t_0 + n - m + 2j - 1$
5. $t_2 := t_1 + m - 2j$
6. $t_3 := t_2 + n + 2j$
7. $i' := i - t_0$
8. If $j > i$
9. Then
10. For $i = j$ To $n + j - 2$ Do
11. $k := i + j - 1$
12. receive (P_j, P_{j-1}, a_k, b_k)
13. Endfor;
14. For $i' = 1$ To $n - m + 2j - 1$ Do
15. $k := n - i' + 1$
16. $l := j$
17. If $1 \leq k \leq m$
18. Then
19. $c_{k, \lfloor \frac{m-i}{2} \rfloor} := \infty$;
20. If $m \leq k \leq n$
21. Then
22. $c_{k,0} := \infty$;
23. If $i \neq (t_1 - 1, t_1)$
24. Then
25. receive ($P_j, P_{j-1}, c_{k,l-1}$);
26. $c_{k,l} := \vee(c_{k,l-1}, \wedge(a_k, b_l), \wedge(a_l, b_k))$;
27. If $j < \lceil \frac{m}{2} \rceil$
28. Then
29. send ($P_j, P_{j+1}, c_{k,l}$)
30. Else
31. send (P_j, host, c_k)
32. Endfor
33. $i'' := i - t_2$
34. For $i'' = 1$ To $2j$ Do
35. $k := n + i''$
36. $l := j$

Fig. 9. The linear array algorithm for fuzzy addition.

$i \leq n$

$\leq n$),
since

time	P_6			P_7			P_8			P_9		
	R	C	T	R	C	T	R	C	T	R	C	T
1												
2												
3												
4												
5												
6	a_1, b_1		a_1, b_1									
7	a_2, b_2		a_2, b_2	a_1, b_1		a_1, b_1						
8	a_3, b_3		a_3, b_3	a_2, b_2		a_2, b_2	a_1, b_1		a_1, b_1			
9	a_4, b_4		a_4, b_4	a_3, b_3		a_3, b_3	a_2, b_2		a_2, b_2	a_1, b_1		
10	a_5, b_5		a_5, b_5	a_4, b_4		a_4, b_4	a_3, b_3		a_3, b_3	a_2, b_2		
11	a_6, b_6		a_6, b_6	a_5, b_5		a_5, b_5	a_4, b_4		a_4, b_4	a_3, b_3		
12	a_7, b_7		a_7, b_7	a_6, b_6		a_6, b_6	a_5, b_5		a_5, b_5	a_4, b_4		
13	a_8, b_8		a_8, b_8	a_7, b_7		a_7, b_7	a_6, b_6		a_6, b_6	a_5, b_5		
14	a_9, b_9		a_9, b_9	a_8, b_8		a_8, b_8	a_7, b_7		a_7, b_7	a_6, b_6		
15	$c_{1.5}$	$c_{1.6}$	$c_{1.6}$	a_9, b_9		a_9, b_9	a_8, b_8		a_8, b_8	a_7, b_7		
16	$c_{2.5}$	$c_{2.6}$	$c_{2.6}$	$c_{1.6}$	$c_{1.7}$	$c_{1.7}$	a_9, b_9		a_9, b_9	a_8, b_8		
17	$c_{3.5}$	$c_{3.6}$	$c_{3.6}$	$c_{2.6}$	$c_{2.7}$	$c_{2.7}$	$c_{1.7}$	$c_{1.8}$	$c_{1.8}$	a_9, b_9		
18	$c_{4.5}$	$c_{4.6}$	$c_{4.6}$	$c_{3.6}$	$c_{3.7}$	$c_{3.7}$	$c_{2.7}$	$c_{2.8}$	$c_{2.8}$	$c_{1.8}$	$c_{1.9}$	c_1
19	$c_{5.5}$	$c_{5.6}$	$c_{5.6}$	$c_{4.6}$	$c_{4.7}$	$c_{4.7}$	$c_{3.7}$	$c_{3.8}$	$c_{3.8}$	$c_{2.8}$	$c_{2.9}$	c_2
20		$c_{6.6}$	$c_{6.6}$	$c_{5.6}$	$c_{5.7}$	$c_{5.7}$	$c_{4.7}$	$c_{4.8}$	$c_{4.8}$	$c_{3.8}$	$c_{3.9}$	c_3
21				$c_{6.6}$	$c_{6.7}$	$c_{6.7}$	$c_{5.7}$	$c_{5.8}$	$c_{5.8}$	$c_{4.8}$	$c_{4.9}$	c_4
22					$c_{7.7}$	$c_{7.7}$	$c_{6.7}$	$c_{6.8}$	$c_{6.8}$	$c_{5.8}$	$c_{5.9}$	c_5
23							$c_{7.7}$	$c_{7.8}$	$c_{7.8}$	$c_{6.8}$	$c_{6.9}$	c_6
24								$c_{8.8}$	$c_{8.8}$	$c_{7.8}$	$c_{7.9}$	c_7
25										$c_{8.8}$	$c_{8.9}$	c_8
26											$c_{9.9}$	c_9

Fig. 10. The RCT diagram for fuzzy minimum.

4.4. The complexity analysis

The complexity analysis is based on two factors: the computation time T (i.e. the number of parallel steps between the first impute element to the array and the last output element into the host machine) and the number of processors P . In all algorithms, procedure data-input takes n time steps and no time is needed for data-output. For fuzzy minimum (maximum), procedure data-computation takes $2n - 1$ time steps and n processors are required. Meanwhile, for fuzzy addition (subtraction), Procedure data-computation takes $2m + n - 1$ time steps, but only $\lfloor m/2 \rfloor$ processors are used. Hence,

$$\begin{aligned} \text{Minimum: } & T = 3n - 1, & P &= n. \\ \text{Addition: } & T = 3m + n - 1, & P &= \lfloor m/2 \rfloor. \end{aligned}$$

time	P_5			P_6			P_7			P_8		
	R	C	T	R	C	T	R	C	T	R	C	T
24	a_1, b_1		a_1, b_1	a_2, b_2		a_2, b_2	a_3, b_3		a_3, b_3	a_4, b_4		
25	$c_{20,4}$	$c_{20,5}$	$c_{20,5}$	a_1, b_1		a_1, b_1	a_2, b_2		a_2, b_2	a_3, b_3		
26	$c_{19,4}$	$c_{19,5}$	$c_{19,5}$	$c_{20,5}$	$c_{20,6}$	$c_{20,6}$	a_1, b_1		a_1, b_1	a_2, b_2		
27	$c_{18,4}$	$c_{18,5}$	$c_{18,5}$	$c_{19,5}$	$c_{19,6}$	$c_{19,6}$	$c_{20,6}$	$c_{20,7}$	$c_{20,7}$	a_1, b_1		
28	$c_{17,4}$	$c_{17,5}$	$c_{17,5}$	$c_{18,5}$	$c_{18,6}$	$c_{18,6}$	$c_{19,6}$	$c_{19,7}$	$c_{19,7}$	$c_{20,7}$	$c_{20,8}$	c_{20}
29	$c_{16,4}$	$c_{16,5}$	$c_{16,5}$	$c_{17,5}$	$c_{17,6}$	$c_{17,6}$	$c_{18,6}$	$c_{18,7}$	$c_{18,7}$	$c_{19,7}$	$c_{19,8}$	c_{19}
30	$c_{15,4}$	$c_{15,5}$	$c_{15,5}$	$c_{16,5}$	$c_{16,6}$	$c_{16,6}$	$c_{17,6}$	$c_{17,7}$	$c_{17,7}$	$c_{18,7}$	$c_{18,8}$	c_{18}
31	$c_{14,4}$	$c_{14,5}$	$c_{14,5}$	$c_{15,5}$	$c_{15,6}$	$c_{15,6}$	$c_{16,6}$	$c_{16,7}$	$c_{16,7}$	$c_{17,7}$	$c_{17,8}$	c_{17}
32	$c_{13,4}$	$c_{13,5}$	$c_{13,5}$	$c_{14,5}$	$c_{14,6}$	$c_{14,6}$	$c_{15,6}$	$c_{15,7}$	$c_{15,7}$	$c_{16,7}$	$c_{16,8}$	c_{16}
33	$c_{12,4}$	$c_{12,5}$	$c_{12,5}$	$c_{13,5}$	$c_{13,6}$	$c_{13,6}$	$c_{14,6}$	$c_{14,7}$	$c_{14,7}$	$c_{15,7}$	$c_{15,8}$	c_{15}
34	$c_{11,4}$	$c_{11,5}$	$c_{11,5}$	$c_{12,5}$	$c_{12,6}$	$c_{12,6}$	$c_{13,6}$	$c_{13,7}$	$c_{13,7}$	$c_{14,7}$	$c_{14,8}$	c_{14}
35	$c_{10,4}$	$c_{10,5}$	$c_{10,5}$	$c_{11,5}$	$c_{11,6}$	$c_{11,6}$	$c_{12,6}$	$c_{12,7}$	$c_{12,7}$	$c_{13,7}$	$c_{13,8}$	c_{13}
36	$c_{9,4}$	$c_{9,5}$	$c_{9,5}$	$c_{10,5}$	$c_{10,6}$	$c_{10,6}$	$c_{11,6}$	$c_{11,7}$	$c_{11,7}$	$c_{12,7}$	$c_{12,8}$	c_{12}
37	$c_{8,4}$	$c_{8,5}$	$c_{8,5}$	$c_{9,5}$	$c_{9,6}$	$c_{9,6}$	$c_{10,6}$	$c_{10,7}$	$c_{10,7}$	$c_{11,7}$	$c_{11,8}$	c_{11}
38		$c_{7,5}$	$c_{7,5}$	$c_{8,5}$	$c_{8,6}$	$c_{8,6}$	$c_{9,6}$	$c_{9,7}$	$c_{9,7}$	$c_{10,7}$	$c_{10,8}$	c_{10}
39				$c_{7,5}$	$c_{7,6}$	$c_{7,6}$	$c_{8,6}$	$c_{8,7}$	$c_{8,7}$	$c_{9,7}$	$c_{9,8}$	c_9
40					$c_{6,6}$	$c_{6,6}$	$c_{7,6}$	$c_{7,7}$	$c_{7,7}$	$c_{8,7}$	$c_{8,8}$	c_8
41					$c_{5,6}$	$c_{5,6}$	$c_{6,6}$	$c_{6,7}$	$c_{6,7}$	$c_{7,7}$	$c_{7,8}$	c_7
42								$c_{5,7}$	$c_{5,7}$	$c_{6,7}$	$c_{6,8}$	c_6
43								$c_{4,7}$	$c_{4,7}$	$c_{5,7}$	$c_{5,8}$	c_5
44								$c_{3,7}$	$c_{3,7}$	$c_{4,7}$	$c_{4,8}$	c_4
45	$c_{21,4}$	$c_{21,5}$	$c_{21,5}$							$c_{3,7}$	$c_{3,8}$	c_3
46	$c_{22,4}$	$c_{22,5}$	$c_{22,5}$	$c_{21,5}$	$c_{21,6}$	$c_{21,6}$					$c_{2,8}$	c_2
47	$c_{23,4}$	$c_{23,5}$	$c_{23,5}$	$c_{22,5}$	$c_{22,6}$	$c_{22,6}$	$c_{21,6}$	$c_{21,7}$	$c_{21,7}$		$c_{1,8}$	c_1
48	$c_{24,4}$	$c_{24,5}$	$c_{23,5}$	$c_{23,5}$	$c_{23,6}$	$c_{22,6}$	$c_{22,6}$	$c_{22,7}$	$c_{21,7}$	$c_{21,7}$	$c_{21,8}$	c_{21}
49	$c_{25,4}$	$c_{25,5}$	$c_{25,5}$	$c_{24,5}$	$c_{24,6}$	$c_{24,6}$	$c_{23,6}$	$c_{23,7}$	$c_{23,7}$	$c_{22,7}$	$c_{22,8}$	c_{22}
50	$c_{26,4}$	$c_{26,5}$	$c_{26,5}$	$c_{25,5}$	$c_{25,6}$	$c_{25,6}$	$c_{24,6}$	$c_{24,7}$	$c_{24,7}$	$c_{23,7}$	$c_{23,8}$	c_{23}
51	$c_{27,4}$	$c_{27,5}$	$c_{27,5}$	$c_{26,5}$	$c_{26,6}$	$c_{24,6}$	$c_{25,6}$	$c_{25,7}$	$c_{23,7}$	$c_{24,7}$	$c_{24,8}$	c_{24}
52	$c_{28,4}$	$c_{28,5}$	$c_{28,5}$	$c_{27,5}$	$c_{27,6}$	$c_{27,6}$	$c_{26,6}$	$c_{26,7}$	$c_{26,7}$	$c_{25,7}$	$c_{25,8}$	c_{25}
53	$c_{29,4}$	$c_{29,5}$	$c_{29,5}$	$c_{28,5}$	$c_{28,6}$	$c_{28,6}$	$c_{27,6}$	$c_{27,7}$	$c_{27,7}$	$c_{26,7}$	$c_{26,8}$	c_{26}
54	$c_{30,4}$	$c_{30,5}$	$c_{30,5}$	$c_{29,5}$	$c_{29,6}$	$c_{29,6}$	$c_{28,6}$	$c_{28,7}$	$c_{28,7}$	$c_{27,7}$	$c_{27,8}$	c_{27}
55	$c_{31,4}$	$c_{31,5}$	$c_{31,5}$	$c_{30,5}$	$c_{20,6}$	$c_{20,6}$	$c_{29,6}$	$c_{29,7}$	$c_{29,7}$	$c_{28,7}$	$c_{28,8}$	c_{28}
56				$c_{31,5}$	$c_{31,6}$	$c_{31,6}$	$c_{30,6}$	$c_{30,7}$	$c_{30,7}$	$c_{29,7}$	$c_{29,8}$	c_{29}
57					$c_{32,6}$	$c_{32,6}$	$c_{31,6}$	$c_{31,7}$	$c_{31,7}$	$c_{30,7}$	$c_{30,8}$	c_{30}
58					$c_{33,6}$	$c_{33,6}$	$c_{32,6}$	$c_{32,7}$	$c_{32,7}$	$c_{31,7}$	$c_{31,8}$	c_{31}
59							$c_{33,6}$	$c_{33,7}$	$c_{33,7}$	$c_{32,7}$	$c_{32,8}$	c_{32}

Fig. 11. The RCT diagram for fuzzy addition.

5. The modified-mesh-of-trees algorithms

This section provides a mapping of the same sequential algorithms presented in Section 2 on a modified-mesh-of-trees type of systolic architecture. As in Section 4, this section includes the computation model, the algorithm design steps, and the complexity analysis.

5.1. 5
 TI
 mesh
 comp
 direc
 data
 data
 corre
 maxi
 conr
 num
 over
 L
 let's
 L
 to 0

5.1. The computation model

The computation model to be considered in this section is a special combination of the mesh and tree systolic arrays. As depicted in Fig. 12, each row and column contains a complete tree of processors. The diagonal processors (illustrated with diamond-shapes) are directly connected to the host computer. This direct connections support data-entry and data-output for each row and column of processors to proceed simultaneously, that is, once data is received by the diagonal processor, it follows the tree of processors along both the corresponding row and column until it reaches all of the PEs in the trees. This takes a maximum of $\lceil \log(n/2) - i + 1 \rceil$ time steps, i is the number of row and column of processor connected to the i th diagonal processor, n is the size of each of the fuzzy operands. The number of rows and columns is $n/2$. As will be shown later, this extra time will not affect the overall logarithmic time complexity of the algorithms.

Let $f(i)$ be the number of processors in either row i or column i . Before computing $f(i)$, let's define a couple of other functions:

Let $\beta(j)$ be equal to the number of 1s in the binary representation of j , and $\alpha(j)$ be equal to 0, if j is a power of 2, and be equal to 1 otherwise. Then

$$f(j) = j + \left\lfloor \frac{j}{2} \right\rfloor + \left\lfloor \frac{j}{4} \right\rfloor + \dots + 2 + 1$$

$$= 2j - \beta(j) + \alpha(j) \lceil \log j \rceil$$

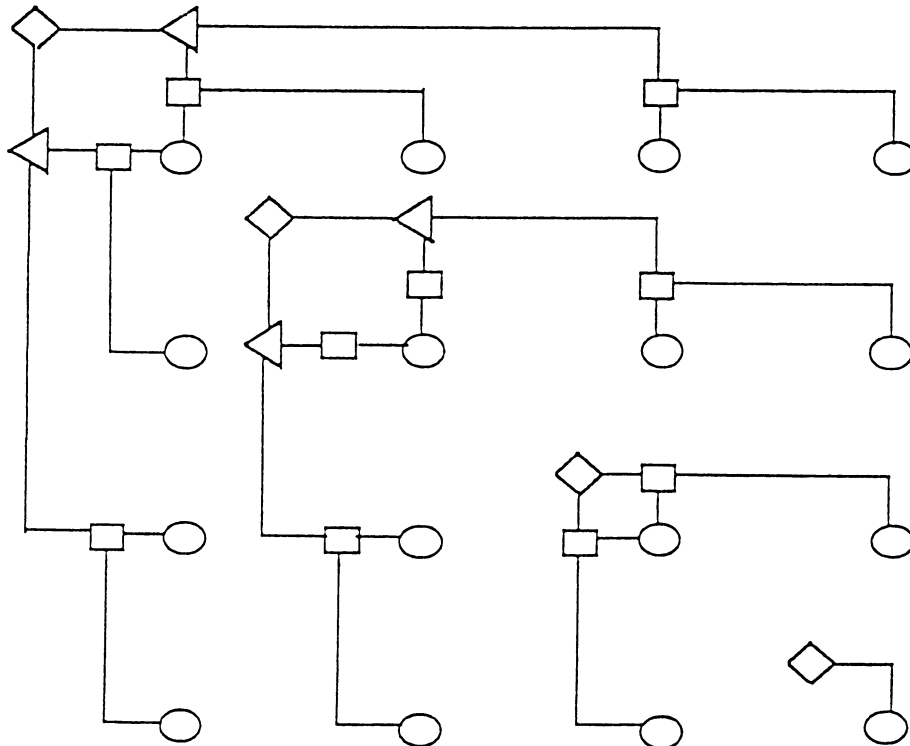


Fig. 12. A modified-mesh-of-trees systolic array of size 4.

	T
8	C20
8	C19
8	C18
8	C17
8	C16
8	C15
8	C14
8	C13
8	C12
8	C11
8	C10
	C9
	C8
	C7
	C6
	C5
	C4
	C3
	C2
	C1
C21	
C22	
C23	
C24	
C25	
C26	
C27	
C28	
C29	
C30	
C31	
C32	

ion 2
ludes

Therefore the total number of row processors P_{row} is:

$$P_{row} = \sum_{j=1}^{n/2} f(j).$$

Since the row j and column j have one processor in common, the total number of column processor P_{col} is:

$$P_{col} = P_{row} - \frac{n}{2}.$$

The total number of diagonal processors is:

$$P_{diag} = \frac{n}{2}.$$

Thus, the total number of processors P is:

$$P = P_{row} + P_{col} + P_{diag}.$$

Each processor is identified with 3 indices: i, j and k . Index i is the row number, j is the column number and k is the level number in the tree. As in the linear array model, processors communicate through the send and receive functions described in Section 4.1.

5.2. The algorithm

As described earlier, the data-entry phase of the algorithms is initiated through the diagonal PEs. The same phase can be integrated into the data-computation phase by adding more data channels to each processor. The following is a description of the data-computation state, for the minimum operation. Figure 13 describes the correspondence between the processors of the row and the entries in c they compute.

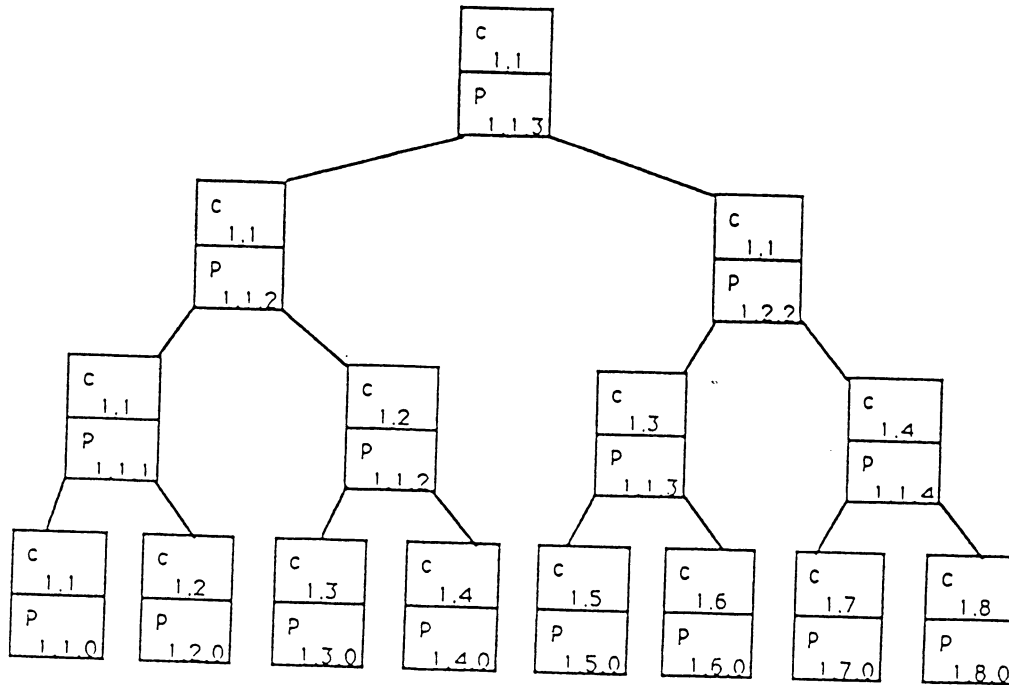


Fig. 13. Correspondence between processors and c entries in the first row of a modified-mesh-of-trees array of size.

At ti
Com

Rece

Tran

Bot
the
it re
pha
alg

5.3.

add
arc

Algorithm Mesh-of-treesfuzzyminimum;

(*This algorithm is to be executed in $P_{i,j,k}$ *)

1. $(i, j, k) := pid;$
2. $t := time;$
3. If $t = 0$ (*Data-entry*)
4. Then
5. receive $(P_{i,j,k}, P_{i,j+2^k,k+1}, a_i, b_{2j-1}, b_{2j});$
6. $c_{i,j} := \wedge(M_{i,2j-1}, M_{i,2j});$
7. Else (*Data-computation*)
8. receive $(P_{i,j,k}, P_{i,2j-1,k-1}, c_{i,2j-1});$
9. receive $(P_{i,j,k}, P_{i,2j,k-1}, c_{i,2j});$
10. $c_{i,j} := \vee(c_{i,2j-1}, c_{i,2j});$
11. send $(P_{i,j,k}, P_{i, \lceil \frac{j}{2} \rceil, k+1}, c_{i,j});$
12. End.

Fig. 14. The algorithm for minimum run by each processor in the mesh-of-trees.

At time k , processor $P_{i,j,k}$ is:

Computing

$$\begin{aligned}
 c_{i,j} &= M_{i,2j-1} \vee M_{i,2j} && \text{if } k = 0 \\
 c_{i,j} &= c_{i,j} \wedge c_{i,j+2^{k-1}} && \text{if } k > 0 \\
 c_{i,j} &= c_{i,j} && \text{if } i = j \text{ and } (i \bmod k) = 1
 \end{aligned}$$

Receiving

$$\begin{aligned}
 c_{i,2j-1} &\text{ from } P_{i,2j-1,k-1} \\
 c_{i,2j} &\text{ from } P_{i,2j,k-1}
 \end{aligned}$$

Transmitting

$$\begin{aligned}
 c_{i,j} &\text{ to } P_{k, \lceil j/2 \rceil, k+1} \\
 1 &\leq i \leq \left\lceil \frac{n}{2} \right\rceil \\
 i &\leq j \leq \frac{n}{2} - 2^k \\
 0 &\leq k \leq \left\lceil \log \left(\frac{n}{2} - i + 1 \right) \right\rceil
 \end{aligned}$$

Both the i th row and the i th column execute simultaneously. They pass their final values to the diagonal processor $P_{i, i, \log(n/2)+1}$, which in turn computes c_i by comparing the two entries it received. Once c_i has been computed it will be sent to the host computer. This data output phase does not add to the overall time complexity of the algorithm. A formal coding of the algorithm is presented in Fig. 14.

5.3. Analysis

The following are the complexity factors for the two fuzzy operations, minimum and addition, when the corresponding algorithms are mapped onto a mesh-of-trees systolic architecture.

column

i is the model, 4.1.

gh the adding station on the

]

f size.

$$\text{Minimum: } T = 3 \log n, \quad P = \frac{n^2}{2} - n - 1.$$

$$\text{Addition: } T = 3 \log(m + n), \quad P = \frac{m^2}{4} - m.$$

6. Expandability

Expanding the algorithms described in both Sections 4 and 5 mean either increasing the values of m and n , or raising the dimension of the two fuzzy operators, or both. The first problem can be solved using some of the partitioning techniques proposed in [1,3]. The second problem, however, depends on the speed-up requested and the hardware available. For instance, a minimum of two fuzzy numbers of dimension 2 can be computed in $O(n)$ time, using a linear array of $O(n^2)$ processors. That is, each n sequence of n processors will compute one row of c , and then the results will be pipelined just as in the linear array algorithm for numbers of dimension 1. On the other hand, if one is willing to give up time and speed for the cost of hardware, a second approach can be used. Each processor in the linear array of size n will be computing one row of c . This will take $O(n^2)$ time steps. The same techniques may be used for the mesh-of-trees systolic array.

7. Conclusion

Many applications require such operations on fuzzy numbers as minimum, maximum, addition, and subtraction etc. A fuzzy number is defined by a convex and normal fuzzy subset. Thus, to represent a fuzzy number we need as many values as the number of elements in the fuzzy subset. Hence, the size of a fuzzy number can be considered as the size of corresponding fuzzy subset. Any binary operation on two fuzzy numbers of sizes n and m requires $(n \times m)$ conventional arithmetic operations. Thus, the use of fuzzy numbers in such systems as artificial expert systems make them slow. In this paper, we have presented $O(n + m)$ and $O(\log n + \log m)$ time systolic algorithms for fast parallel computation of some fuzzy binary operations.

The choice of network architectures used previously was to highlight the fact that systolic arrays can rely on parallel or pipelined processing or both. The linear array algorithm takes advantage of the recurrences between different instances of the same entry and uses extensive pipelining to achieve a linear speed-up which is optimal in this case, given the number of processors used. On the other hand, the mesh-of-trees algorithm takes advantage of the independence between different instances of the same entry, and thus relies on the parallel execution of those partial results. It achieves a higher speed-up than the former algorithm, but requires more PEs to be used for a shorter period of time.

Performance study of these parallel algorithms on actual systolic arrays would make this work more complete. However, since we have no access to such a system, we shall conclude this section after discussing the expected speed-up on a linear systolic array. As it can be seen from Fig. 2, any sequential algorithm will require $(n \times m)$ time steps for addition of two fuzzy numbers of sizes n and m . From the end of Section 4, it is clear that the same computation on a linear systolic array of size $(m/2)$ will take only $(3m + n - 1)$ time steps. Thus, when $n = m$, a speedup of $(n/4)$ is obtained and the resulting processor efficiency is about 50 percent. We hope to implement these algorithms when systolic arrays are available to us.

References

- [1] D.I. Moldovan and J.A.B. Fortes, Partitioning and mapping alg. into fixed size systolic arrays, *IEEE Trans. Comput.* C-35(1) (Jan. 1986).
- [2] D. Sarkar and A. Mukherjee, Design of optimal systolic algorithms for the transitive closure problem, *IEEE Trans. Comput.* 41(4) (April 1992).
- [3] J.J. Navarro, J.M. Llaberia and M. Valero, Partitioning: An essential step in mapping algorithms into systolic array processors, *Computer* (Jul. 87).
- [4] A. Kaufmann and M.M. Gupta, *Introduction to Fuzzy Arithmetic* (Van Nostrand Reinhold, New York, 1985).
- [5] A. Kaufmann and M.M. Gupta, *Fuzzy Mathematical Models in Engineering and Management Science* (Elsevier Science, Amsterdam, 1988).
- [6] K.S. Leung and W. Lam, Fuzzy concepts in expert systems, *Computer* (Sep. 1988).
- [7] AT&T research use fuzzy logic to put an expert system on a computer chip, *Res. Development* (Mar. 1986).
- [8] E.L. Andrews, New auto systems use fuzzy logic, *Wall Street J.* (Jan. 1989).
- [9] G. Bell, The future of high performance computers in science and engineering, *Comm. ACM* 32(9) (Sep. 1990).

ng the
e first
l. The
ilable.
time,
s will
array
e and
linear
same

num,
bset.
n the
ond-
quires
ns as
and
inary

stolic
takes
nsive
er of
the
allel
thm,

this
lude
seen
uzzy
tion
hen
t 50