VII. Remarks

We have presented a unified view of fault-tolerant matrix triangularization procedures. The choice of weighted row checksums allows us to use such stable numerical techniques as pairwise pivotings and plane rotations. Our technique is very good for detecting and correcting one transient error, but is not easily extensible to the case of two or more errors.

References

[1] J. A. Abraham, "Fault tolerant techniques for highly parallel signal processing architectures," in *Highly Parallel Signal Processing Architectures, Proc. SPIE*, K. Bromley, Ed., vol. 614, 1986, pp. 49–65.

[2] H. M. Ahmed, J.-M. Delosme, and M. Morf, "Highly concurrent computing structures for matrix arithmetic and signal processing," *Computer*, vol. 15, pp. 65–82, Jan. 1982.

[3] A. Bojanczyk, R. P. Brent, and H. T. Kung, "Numerically stable solution of dense systems of linear equations using mesh-connected processors," *SIAM J. Sci. Statist. Comput.*, vol. 5, pp. 95–104, 1984.

[4] W. M. Gentleman and H. T. Kung, "Matrix triangularization by systolic arrays," in *Real Time Signal Processing IV, Proc. SPIE*, T. F. Tao, Ed., vol. 298, 1981, pp. 19–26.

[5] P. E. Gill, G. H. Golub, W. Murray, and M. A. Saunders, "Methods for modifying matrix factorizations," *Math. Comput.*, vol. 28, pp. 505–535, 1974.

[6] K-H. Huang and J. A. Abraham, "Algorithm-based fault tolerance for matrix operations," *IEEE Trans. Comput.*, vol. C-33, pp. 518–528, 1984.

[7] J-Y. Jou and J. A. Abraham, "Fault-tolerant matrix arithmetic and signal processing on highly concurrent computing structures," *Proc. IEEE*, vol. 74, pp. 732–741, 1986.

[8] H. T. Kung and M. S. Lam, "Fault-tolerant VLSI systolic arrays and two-level pipelining," in *Real Time Signal Processing VI, Proc. SPIE*, K. Bromley, Ed., vol. 431, 1983, pp. 143–158.

[9] H. T. Kung and C. E. Leiserson, "Algorithms for VLSI processor arrays," in *Introduction to VLSI Systems*, C. Mead and L. Conway, Eds. Reading, MA: Addison-Wesley, 1980, pp. 271–292.

[10] F. T. Luk, "Algorithm-based fault tolerance for parallel matrix equation solvers," in *Real Time Signal Processing VIII, Proc. SPIE*, W. J. Miceli and K. Bromley, Eds., vol. 564, 1985, pp. 49–53.

[11] ——, "A rotation method for computing the QR-decomposition," *SIAM J. Sci. Statist. Comput.*, vol. 7, pp. 452-459, 1986.

[12] F. T. Luk and H. Park, "Analysis of algorithm-based fault tolerance techniques," in *J. Parallel Distribut. Comput.*, vol. 5, pp. 172–184, 1988.

[13] J. M. Ortega, *Numerical Analysis, A Second Course*. New York: Academic, 1972.

[14] D. C. Sorensen, "Analysis of pairwise pivoting in Gaussian elimination," *IEEE Trans. Comput.*, vol. C-34, pp. 274–278, 1985.

[15] J. M. Speiser and H. J. Whitehouse, "A review of signal processing with systolic arrays," in *Real Time Signal Processing VI, Proc. SPIE*, K. Bromley, Ed., vol. 431, 1983, pp. 2–6.

[16] J. H. Wilkinson, *The Algebraic Eigenvalue Problem*. London, England: Oxford University Press, 1965.

Performance Analysis of Multistage Interconnection Networks with Hierarchical Requesting Model

WEN-TSUEN CHEN AND JANG-PING SHEU

Abstract—**This correspondence analyzes the performance of the multistage interconnection networks (MIN's) for interconnecting N processors or N processors to N commonly shared memory modules in a multiprocessor system. A general model, called hierarchical requesting model, has been proposed. The performance of the MIN's with respect to their memory bandwidth is analyzed and is compared to that of a crossbar under the proposed model. Based on the analytical results, we present a task allocation strategy to increase the memory bandwidth of the MIN's.**

Index Terms—**Acceptance probability, crossbar, delta networks, memory bandwidth, multistage interconnection networks, multiprocessor systems, task assignment.**

Manuscript received May 13, 1986; revised December 2, 1986. This work was supported by the National Science Council of the Republic of China under Grant NSC74-0611E007-03.

W.-T. Chen is with the Institute of Computer Science, National Tsing Hua University, Taiwan 30043, Republic of China.

J.-P. Sheu is with the Department of Electrical Engineering, National Central University, Taiwan 32054, Republic of China.

IEEE Log Number 8717771.

I. Introduction

In recent years, there have been many researchers attempting to construct multiprocessor systems with large computational power. The performance of a multiprocessor system depends significantly on the efficiency of its interconnection network (IN). Many types of IN's such as crossbar and multistage interconnection networks (MIN's), have been proposed in the past few years [5]–[8], [10]. For a crossbar network, it allows all possible one-to-one simultaneous connections between the processors and the memory modules. However, it needs $O(N^2)$ switches for connecting N processors to N commonly shared memory modules. On the other hand, various MIN's with $O(N \log N)$ switches have been proposed for applications with large values of N [6]–[8], [10]. The memory bandwidth (MBW) of both MIN's and crossbars has been analyzed by several authors [1], [2], [4], [7], [9]. The MBW is defined as the number of successful requests per memory cycle.

The MBW of an $N \times M$ crossbar in the case that N processors are equally likely to address any one of the M commonly shared memory modules is studied by Patel [7]. The case that each processor requests the ith commonly shared memory module with different probability P_i for $1 \leq i \leq M$ is studied by Du [4]. The effective MBW of the MIN's has also been analyzed by the other researchers in the case that requests are uniformly distributed on the memory modules [7], [9]. Bhuyan proposed a different model in which each processor has a favorite memory module [2]. The request probability of a processor to its favorite memory module is higher than those to nonfavorite ones. However, the processor requests connection to all the nonfavorite memory modules with equal probability. The case with equally likely requesting rate is shown to be a special case in [2].

II. The Hierarchical Requesting Model

In a multiprocessing environment, a job to be run on the system usually consists of a set of communicating tasks. To execute these tasks, the system should assign them to the processors with the minimum communication overhead. Hence, the task assignment procedure will assign those tasks that have a large amount of communication to the same processor or to a cluster of processors with low communication cost. It leads to that the probability of a processor communicating with other processors belonging to the same cluster is higher than those processors belonging to other clusters. To model such a system, a hierarchical requesting model is proposed. We assume that each processor has one of the commonly shared memory modules as its favorite memory module for storing the assigned tasks. Besides, the relations between the processors can be classified into an n-level hierarchy. Each processor has different fractions of requests to the memory modules belonging to a different level of subclusters. In general, the fraction of processors requesting connection to their favorite memory module is higher than those to nonfavorite ones. Furthermore, the fraction of requests connection to memory modules within the same cluster is higher than to those in other clusters.

For example, with a two-level hierarchy, a multiprocessor system with N pairs of processors and its favorite memory module can be partitioned into C clusters, and each cluster contains K pairs of processors and memory modules, when $N = C \times K$. Then a processor in a particular cluster C_i has three types of requests, namely, requests to its favorite memory module with fraction m_0, requests to each of the memory modules in cluster C_i with fraction

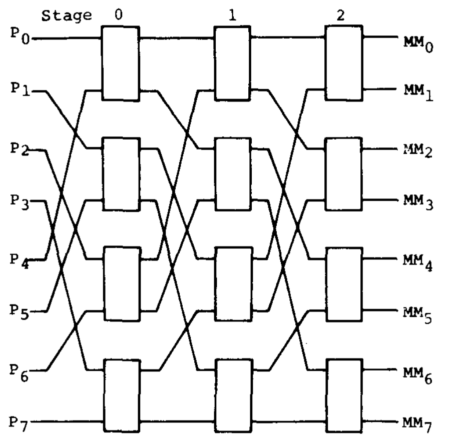

Fig. 1. An 8 × 8 delta network.

m_1, and requests to each of the memory modules in other clusters with fraction m_2. In general, $m_0 > m_1 > m_2$.

Assume $N = a^n$ and $M = b^n$. An $N \times M$ delta network [7] is an IN that connects N inputs to M outputs with n stages of $a \times b$ crossbar switches. In this correspondence, we restrict our analysis to $N \times N$ delta networks and an a-shuffle connection is used between each pair of adjacent stages, where "a" is the switching size. An 8 × 8 delta network with two-shuffle (perfect shuffle) interconnection stages is shown in Fig. 1. The analysis is based on the following assumptions.

1) An IN operates in a synchronous mode. The connection requests of all processors are issued at the same time and each processor has an identical memory cycle time.

2) A multiprocessor system with $N = a^n$ processors and N shared memory modules can be logically partitioned into C clusters and each cluster contains K pairs of processors and memory modules, where $N = C \times K$. C is restricted to the power of "a," i.e., $C = a^i$ for some integer i. Each processor P_i, for $0 \le i \le N - 1$ has the memory module MM_i as its favorite memory module for storing the tasks that are assigned to P_i.

3) The processors are clustered in the ascending order. That is, processors $0, 1, \cdots, K - 1$ belong to the first cluster, processors K, $K + 1, \cdots, 2K - 1$ belong to the second cluster, and so on.

4) Each processor P_i generates random and independent connection requests. Each request has probability m_0 addressing to its favorite memory, probability m_1 to each other memory module in the same cluster with processor P_i, and probability m_2 to each memory module on the other clusters, where $m_0 + (K - 1)m_1 + (N - K)m_2 = 1$.

5) At the beginning of every memory cycle, each processor generates a new request with probability R_0. Thus, R_0 is also the average number of requests generated per memory cycle by each processor.

6) The requests which are blocked (not accepted) are ignored. That is, the requests issued at the next cycle are independent of those in previous cycle.

The last assumption is made to simplify the analysis. The simulation studies done by several researchers for similar problems have shown that the memory bandwidth is only slightly different if the last assumption is omitted [1], [2], [7]. Thus, the results of the analysis with this assumption provide a good estimate for comparing different networks.

III. Performance Analysis

In this section, we shall derive formulas for memory bandwidth and request acceptance probability for both delta networks and crossbars with the hierarchical requesting model.

A. Analysis of Crossbars

For an $N \times N$ crossbar, when requests of two or more processors are addressed to the same memory module, only one of them will be accepted and the rest will be rejected or blocked. The request rate to a memory module in a uniform case is given by [7]

$$p_e = 1 - (1 - R_0/N)^N. \tag{1}$$

With the hierarchical requesting model, the probability p_1 of processor P_i requesting a connection to its favorite memory module MM_i is equal to $R_0 \times m_0$. The probability of P_j requesting a connection to MM_i with $i \ne j$ is equal to $R_0 \times m_1$ provided that P_j and MM_i belong to the same cluster. Then the probability of at least one request to MM_i by those $K - 1$ processors within the same cluster is equal to

$$p_2 = 1 - (1 - R_0 \times m_1)^{K-1}.$$

We can easily derive the probability of at least one request to MM_i by other $N - K$ processors being equal to

$$p_3 = 1 - (1 - R_0 \times m_2)^{N-K}.$$

Hence, the total request rate to MM_i is

$$\begin{aligned} p_h &= 1 - (1 - p_1)(1 - p_2)(1 - p_3) \\ &= 1 - (1 - R_0 \times m_0)(1 - R_0 \times m_1)^{K-1}(1 - R_0 \times m_2)^{N-K}. \end{aligned} \tag{2}$$

The memory bandwidth MBW_h is thus equal to $N \times P_h$. The acceptance probability Pa that a connection request will be accepted is equal to p_h/R_0. If $m_1 = m_2$, then (2) can be reduced to

$$p_h = 1 - (1 - R_0 \times m_0)(1 - R_0 \times m_1)^{N-1}. \tag{3}$$

This equation is identical to the equation derived in [2] with the case that each processor has a higher rate of requests to its favorite memory module without clustering of processors. Note that (1) can also be obtained from (2) by substituting $m_0 = m_1 = m_2 = 1/N$.

B. Analysis of Delta Networks

The MBW of delta networks in the uniformly distributed requesting case is derived by Patel [7] with recursive computation of the rate of request at each stage. The request rate R_i on an output (input) line of stage $i - 1$ (i) is obtained by the following recursive formula:

$$R_{i+1} = 1 - (1 - R_i/a)^a \quad \text{where } 0 \le i \le n - 1. \tag{4}$$

Hence, $\text{MBW}_e = N \times R_n$.

In the following, we shall derive the memory bandwidth of an $N \times N$ delta network with the hierarchical requesting model. Similar to Patel's approach, we shall derive a recursive formula for the request rate R_i. Let m_{jk} represent the probability of processor P_j requesting connection to memory module MM_k when there is a request generated by the processor P_j. Assume that processor P_j generates a request with tag $(d_0 d_1 \cdots d_{n-1})_a$. Let $q_j[x_{i-1}/x_0, x_1, \cdots, x_{i-2}]$ denote the conditional probability that d_{i-1} equals to x_{i-1} given the first $i - 1$ tag digits $(d_0 d_1 \cdots d_{i-2})$ equal to $(x_0 x_1 \cdots x_{i-2})$. Note that the summation of $q_j[x_{i-1}/x_0, x_1, \cdots, x_{i-2}]$ over all $0 \le x_{i-1} \le a - 1$ is equal to 1.

The conditional probability $q_j[x_{i-1}/x_0, x_1, \cdots, x_{i-2}]$ can be computed as follows. The probability of a request generated by the processor P_j connecting to the $(x_0 + 1)$th output of a switch in the first stage, i.e., $d_0 = x_0$, is equal to

$$q_j[x_0] = \sum_{k=C_0}^{B_0} m_{jk}$$

where $C_0 = x_0 \times a^{n-1}$, and $B_0 = C_0 + a^{n-1} - 1$. Then the

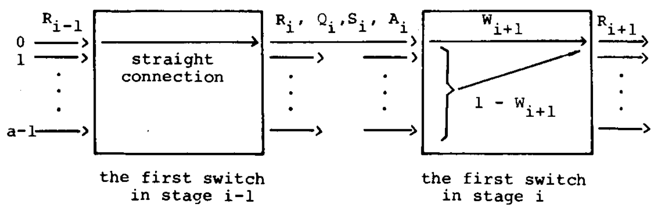

Fig. 2. Request probability at two adjacent stages of a delta network.

conditional probability for $d_1 = x_1$ given $d_0 = x_0$ is equal to

$$q_j[x_1/x_0] = \sum_{k=C_1}^{B_1} m_{jk} \Big/ \sum_{k=C_0}^{B_0} m_{jk}$$

where $C_1 = C_0 + x_1 \times a^{n-2}$ and $B_1 = C_1 + a^{n-2} - 1$. In general, the conditional probability $q_j[x_{i-1}/x_0, x_1, \cdots, x_{i-2}]$, for $2 \le i \le n$, is equal to

$$\sum_{k=C_{i-1}}^{B_{i-1}} m_{jk} \Big/ \sum_{k=C_{i-2}}^{B_{i-2}} m_{jk} \tag{5}$$

where $C_0 = x_0 \times a^{n-1}$, $C_{i-1} = C_{i-2} + x_{i-1} \times a^{n-i}$, and $B_{i-1} = C_{i-1} + a^{n-i} - 1$.

For example, assume that an 8×8 delta network is partitioned into four clusters and each cluster contains two pairs of processors and memory modules. Let processor P_0 with probability m_{0k} request connection to memory module MM_k. According to assumptions 3) and 4), $m_{00} = m_0$, $m_{01} = m_1$, and $m_{02} = m_{03} = \cdots = m_{07} = m_2$. If processor P_0 generates a request with destination $(d_0d_1d_2)_2$, then the probability of this request connecting to the upper output line of a switch in the first stage is equal to $q_0[0] = m_0 + m_1 + 2m_2$, and the probability of this request connecting to the lower output line is equal to $q_0[1] = 4m_2$. The conditional probability $q_0[1/1]$ is equal to $2m_2/4m_2 = 1/2$.

In the following, we present two useful properties for computing the conditional probability $q_j[x_{i-1}/x_0, x_1, \cdots, x_{i-2}]$.

Property 1: If a processor P_j generates a request with destination $(d_0d_1 \cdots d_{n-1})_a$, and subscript j is expressed in radix-a system as $(s_0s_1 \cdots s_{n-1})_a$, then the conditional probability $q_j[x_{i-1}/x_0, \cdots, x_{i-2}]$ is equal to $1/a$ as long as there exists a k such that $x_k \ne s_k$ for $0 \le k \le i-2$, where $0 \le x_k \le a-1$ and $2 \le i \le n$.

Property 1 means that if a connection request generated by any processor does not keep straight connection in a switch at stage i, then at the subsequent stages the request will result in connection to each of the outputs of a switch with probability $1/a$. The request in the ith input line of a switch which requests a connection to the ith output line of the switch is called a straight connection. For example, in Fig. 1 the subscript of P_0 can be expressed in binary as $(000)_2$. By Property 1, all of the conditional probabilities $q_0[0/1]$, $q_0[1/1]$, $q_0[0/01]$, $q_0[1/01]$, $q_0[0/10]$, $q_0[1/10]$, $q_0[0/11]$, and $q_0[1/11]$ are equal to 1/2.

Property 2: Assume that a processor P_j generates a request with destination tag $(d_0d_1 \cdots d_{n-1})_a$, and let j be expressed in radix-a system as $(s_0s_1 \cdots s_{n-1})_a$. If the conditional probability $q_j[s_{i-1}/s_0, \cdots, s_{i-2}]$ is equal to Q_{i-1}, then the conditional probability $q_j[x_{i-1}/s_0, \cdots, s_{i-2}]$ is equal to $(1 - Q_{i-1})/(a-1)$ for all $x_{i-1} \ne s_{i-1}$, where $2 \le i \le n$.

Property 2 means that if a request keeps straight connection all the way from the first stage to the ith stage, then this request has equal probability of connecting to all nonstraight outputs of the ith stage.

Based on Properties 1 and 2, we have that each input line of a switch in stage i has a probability S_i straightly connecting to stage $i+1$ and $(1 - S_i)/(a-1)$ to each of the other output lines. Therefore, any output line in a switch in the first stage has the same output rate

$$R_1 = 1 - (1 - R_0 \times S_0)\left(1 - R_0 \frac{1-S_0}{a-1}\right)^{a-1}.$$

Its derivation is similar to that for formula (3). In the second stage, the ith input of a switch has the probability S_1 connecting to the ith output and $(1 - S_1)/(a-1)$ to each of the other output lines. Hence, all the outputs in the second stage also have the same output rate R_2. With given S_i, then we have the following recursive formula [2]:

$$R_{i+1} = 1 - (1 - R_i \times S_i)\left(1 - R_i \frac{1-S_i}{a-1}\right)^{a-1}, \quad \text{for } 0 \le i \le n-1. \tag{6}$$

Because all output lines in a stage $i - 1$ have the same output rate R_i, we can obtain R_i by considering only the output rate at the first output line of the first switch in each stage i, for $0 \le i \le n-1$. S_i in formula (6) can be considered as the probability for the first input line of the first switch in stage i straightly connecting to the first output line of the first switch in stage i. We shall derive S_i by first defining A_i as the fraction of R_i that is contributed originally by requests at the first input line of the first stage, i.e., from P_0. So, the fraction of R_i that is contributed by requests from the other input lines of the first stage, i.e., from $P_1, P_2, \cdots, P_{N-1}$ is $1 - A_i$. According to Property 2, Q_i is the conditional probability that a request will straightly connect in stage i given that the request comes from P_0. The request that came from P_0 has the probability $(1 - Q_i)/(a-1)$ to each of the other output lines of the first switch in stage i.

By Property 1, each of the other processors P_j, for $1 \le j \le N-1$, in the first input line of stage i have probability $1/a$ connecting to each output line of the first switch in stage i. Hence, we can obtain

$$S_i = A_i \times Q_i + (1 - A_i)/a. \tag{7}$$

In order to find A_i, let W_{i+1} be the fraction of R_{i+1} that is contributed by requests which come from straight connection in stage i. Hence, $1 - W_{i+1}$ is the fraction of R_{i+1} that is contributed by the inputs without straight connection in stage i as shown in Fig. 2. Then W_{i+1} can be derived by the following formula [2]:

$$W_{i+1} = \frac{1}{R_{i+1}} \frac{S_i(a-1)}{a(1-S_i)} \left\{1 - \left(1 - R_i \frac{1-S_i}{a-1}\right)^a\right\}. \tag{8}$$

Thus, the fraction of R_{i+1} that is originally contributed by P_0, i.e., A_{i+1}, can be computed as follows:

$$A_{i+1} = \frac{A_i \times Q_i}{S_i} W_{i+1}. \tag{9}$$

With given values of N, C, K, a, R_0, m_0, m_1, and m_2, then R_i, Q_i, A_i, W_i, and S_i can be computed recursively for $1 \le i \le n$ with formulas (6)–(9). Note that in the first stage,

$$A_0 = 1,$$

and

$$Q_0 = \sum_{k=0}^{N/a-1} m_{0k}. \tag{10}$$

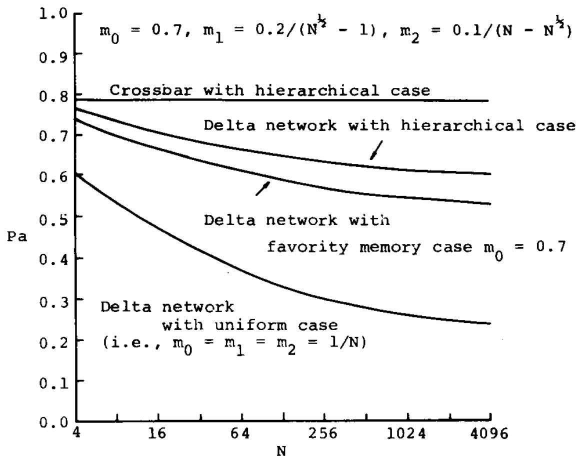

Fig. 3. Acceptance probability versus N.

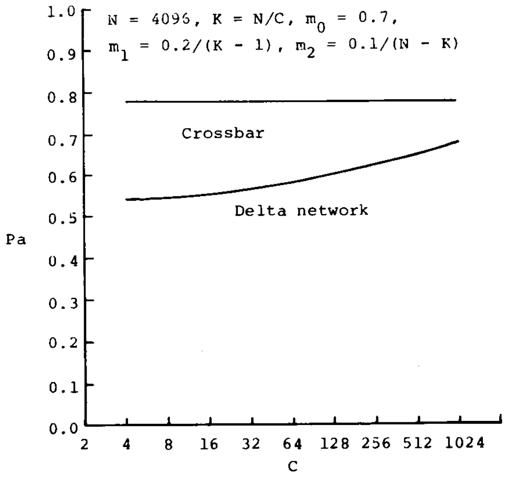

Fig. 4. Acceptance probability versus C.

The conditional probability Q_i can be computed as follows:

$$Q_i = \sum_{k=0}^{B_i} m_{0k} \Bigg/ \sum_{k=0}^{B_{i-1}} m_{0k} \tag{11}$$

where $B_i = a^{n-i-1} - 1$, $m_{00} = m_0$, $m_{0e} = m_1$, and $m_{0f} = m_2$ for $1 \leq e \leq K - 1$ and $K \leq f \leq N - 1$. The memory bandwidth MBW_h is equal to $N \times R_n$ and the acceptance probability Pa is equal to R_n / R_0.

Although the above analysis is performed for the processor-to-memory multiprocessor system, the results can also be applied to the processor-to-processor system in which a processor and a local memory are combined together to form a processing element. Hence, a favorite memory request of any processor refers to its own local memory. There is no traffic in the IN for a favorite memory request. As a result, the network traffic in processor-to-processor systems is less than that in the processor-to-memory system. So, the formulas (6)-(11) can be used to evaluate the MBW of MIN's in the processor-to-processor system with $m_0 = 0$.

In addition, the MBW of a multiprocessor system with any level of hierarchical requesting case can also be evaluated by the formulas (6)-(11). For example, with a three-level hierarchy, assume that the 8×8 delta network as shown in Fig. 1 is partitioned into two clusters, and each cluster contains two subclusters in which each subcluster has two pairs of processors and memory modules. Then processor P_0 has the probability m_0 of requesting connection to MM_0, m_1 requesting connection to MM_1, m_2 requesting connection to MM_2 and MM_3, and m_3 requesting connection to MM_4, MM_5, MM_6, and MM_7, where $m_0 + m_1 + 2m_2 + 4m_3 = 1$. Then the request rate R_3 can be obtained from formulas (6)-(11) with given values of R_0, m_0, m_1, m_2, and m_3.

IV. Numerical Results and Task Allocation Strategy

In this section, we give some values of N, C, K, a, m_0, m_1, and m_2 to evaluate the performance of delta networks and crossbars. With a two-level hierarchy, let $a = 2$, $C = K = N^{1/2}$, $m_0 = 0.7$, $m_1 = 0.2/(K - 1)$, and $m_2 = 0.1/(N - K)$. When the request generation rate of each processor is $R_0 = 1$, then the acceptance probability Pa for a delta network with various sizes of N is plotted in Fig. 3. Acceptance probability for the delta networks with uniform requesting case and the crossbar networks with the hierarchical requesting case is also plotted in Fig. 3. It shows that the acceptance probability of a delta network with the hierarchical requesting case is much better than the delta network with the uniform requesting case in various network sizes. It also shows that in the hierarchical requesting case, the delta networks perform close to the crossbar networks. In the favorite memory case [2], the acceptance probability for delta networks with $m_0 = 0.7$ is also plotted in Fig. 3. The acceptance probability with the favorite memory case is smaller than that of the hierarchical requesting case.

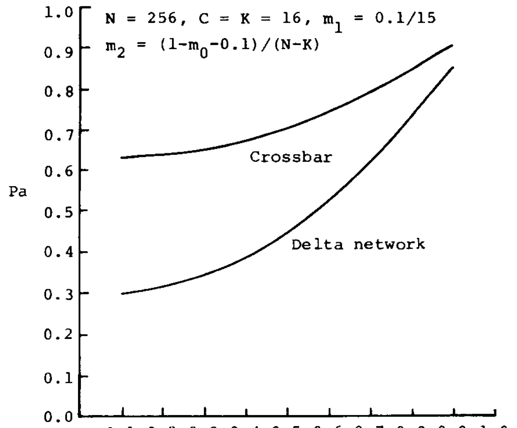

Fig. 5. Acceptance probability versus m_0.

With fixed values of $N = 4096$, $a = 2$, $m_0 = 0.7$, $m_1 = 0.2/(K - 1)$, $m_2 = 0.1/(N - K)$, and $R_0 = 1$, Fig. 4 shows that the acceptance probability Pa of a delta network is increasing as the number of clusters C increases. On the other hand, it shows that the performance of a crossbar remains the same with any size of C. This implies that the performance of a delta network will be close to a crossbar provided that each cluster of the subnetwork is small and communications between subnetworks are rare.

Fig. 5 shows that the acceptance probability Pa of a delta network is quickly increasing as the fraction of the favorite memory module request increases. Fig. 6 also shows that high fraction of intracluster request in a delta network has a high probability to accept a request. Both Figs. 5 and 6 show that the performance of a crossbar increases slowly as the fractions of m_0 and m_1 increase. As a result, the delta network will perform as well as the crossbar when the fractions of m_0 and m_1 are high enough.

Based on the analytical results, we can propose an effective procedure for task assignment to increase the memory bandwidth of the MIN's. Assume that there are T tasks to be assigned to N processors which are interconnected by an $N \times N$ delta network with $a \times a$ switches. The assignment policy is to recursively partition a number of tasks into "a" groups with equal size and keep the tasks with high intertask communication in the same group. In a group of the first partition, the T tasks are partitioned into "a" parts with

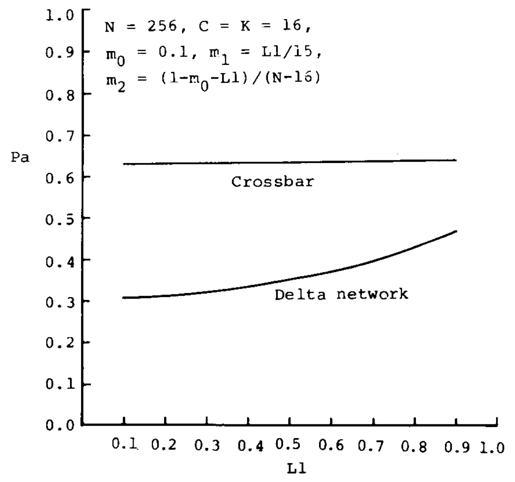

Fig. 6. Acceptance probability versus intracluster request probability $L1$.

equal size and each part is assigned to N/a processors with contiguous addresses. The tasks in each part are partitioned into "a" subparts again. Each subpart is assigned to N/a^2 processors with contiguous addresses of the previous assigned processors. Repeat the partition and assignment procedure until the number of subparts is equal to N or each subpart contains only one task. Although the optimal a-way partitioning problem is NP-complete, however, some effective heuristics that have been proposed in [3] can be used.

V. Conclusions

In this correspondence, we have proposed a general hierarchical requesting model. Any level of hierarchical requesting between inputs and outputs of an MIN can be evaluated with our analytical results. The performance of delta networks and a crossbar under the two-level hierarchy is compared. We have shown that the performance of delta networks is close to that of crossbars if fractions of the favorite memory request and intracluster request are high enough. A task assignment strategy to increase the MBW of the MIN's is also suggested.

References

[1] D. P. Bhandarkar, "Analysis of memory interference in multiprocessors," *IEEE Trans. Comput.*, vol. C-24, pp. 897–908, Sept. 1975.
[2] L. N. Bhuyan, "An analysis of processor-memory interconnection networks," *IEEE Trans. Comput.*, vol. C-34, pp. 279–283, Mar. 1985.
[3] W. T. Chen and J. P. Sheu, "Task assignment in loosely-coupled multiprocessor systems," *J. Chinese Instit. Eng.*, vol. 10, pp. 721–726, Nov. 1987.
[4] H. C. Du, "On the performance of synchronous multiprocessors," *IEEE Trans. Comput.*, vol. C-34, pp. 462–466, May 1985.
[5] T. Y. Feng, "A survey of interconnection networks," *IEEE Computer*, pp. 12–27, Dec. 1981.
[6] D. H. Lawrie, "Access and alignment of data in an array processor," *IEEE Trans. Comput.*, vol. C-25, pp. 1145–1155, Dec. 1975.
[7] J. H. Patel, "Performance of processor-memory interconnections for multiprocessors," *IEEE Trans. Comput.*, vol. C-30, pp. 771–780, Oct. 1981.
[8] M. C. Pease, "The indirect binary n-cube microprocessor array," *IEEE Trans. Comput.*, vol. C-26, pp. 458–473, May 1977.
[9] S. Thanawastien and V. P. Nelson, "Interference analysis of shuffle/exchange networks," *IEEE Trans. Comput.*, vol. C-30, pp. 545–556, Aug. 1981.
[10] C. L. Wu and T. Y. Feng, "The reverse-exchange interconnection networks," *IEEE Trans. Comput.*, vol. C-29, pp. 801–811, Sept. 1980.

Relationship Between P-Valued Majority Functions and P-Valued Threshold Functions

YOSHINORI YAMAMOTO AND SHIRO FUJITA

Abstract—**In a previous paper, the authors defined a new class of multiple-valued logic functions, called multiple-valued majority functions. This correspondence clarifies the distinction of multiple-valued majority functions from multiple-valued threshold functions through the difference between a number function and an inner product of an input vector and a weight vector.**

Index Terms—**Intersection of majority functions and threshold functions, multiple-valued majority function, multiple-valued threshold functions, number function.**

I. Introduction

Ternary threshold functions were first defined by Hanson in 1963 [1]. Since then, many studies have been done in this field. An output value of a ternary threshold function is determined according to the magnitude of an inner product of an input vector and a weight vector. This indicates that ternary threshold functions are an extension of well-known binary threshold functions to ternary logic. It should be noted, however, that ternary threshold functions do not include a ternary OR (i.e., a maximum of inputs) and a ternary AND (i.e., a minimum of inputs). Also, no function which has a meaning as that of binary voter is included by ternary threshold functions. Due to these facts, it was pointed out in [2] that there may exist another way of defining ternary "threshold functions."

Yamamoto and Fujita [3] have proposed a new class of ternary logic functions. The proposed ternary logic functions represent a rule of one type of decision making by means of three kinds of notion. From this reason, this new class of ternary logic functions was denominated ternary majority functions. After the first definition, several studies have been published on ternary majority functions mainly in the International Symposium on Multiple-Valued Logic and the *Trans. of IECE Japan*. Subjects of these works are mathematical aspects [3]–[6], an extension to $P(\geq 4)$-valued logic [4], [7], a testing and realization [4], [6], [8], a synthesis method of self-checking multiple-valued majority elements [9], an application to a social decision [10], and an enumeration [11]. Ternary majority functions include a ternary OR and a ternary AND as a special case. This fact strongly suggests to us that ternary majority functions differ in the definition from ternary threshold functions. Misunderstanding is seen, however, on the relation between these two classes. For example, a paper [11] concerning ternary majority functions is cited as a reference of ternary threshold functions in [12]. The authors consider that detailed discussions should be made on the relation between these two classes, because no study on this subject has been published so far in the English language.

Manuscript received May 14, 1986; revised October 10, 1986.
Y. Yamamoto is with the Department of Management, Takasaki City University of Economics, Takasaki 370, Japan.
S. Fujita is with the Faculty of Engineering, Okayama University of Science, Okayama 700, Japan.
IEEE Log Number 8717772.