Brief Contributions

An $O(\log_2 N)$ Depth Asymptotically Nonblocking Self-Routing Permutation Network

G.A. De Biase, C. Ferrone, and A. Massini

Abstract—A self-routing multi-log$N$ permutation network is presented and studied. This network has $3\log_2 N - 2$ depth and $N(\log_2^\gamma N)(3\log_2 N - 2)/2$ nodes, where N is the number of network inputs and γ a constant very close to 1. A parallel routing algorithm runs in $3\log_2 N - 2$ time on this network. The overall system (network and algorithm) can work in pipeline and it is asymptotically nonblocking in the sense that its blocking probability vanishes when N increases, hence the quasi-totality of the information synchronously arrives in $3\log_2 N - 2$ steps at the network outputs. This network presents very good fault tolerance, a modular architecture, and it is suitable for information exchange in very large scale parallel processors and communication systems.

Index Terms—Permutation networks, self-routing algorithm, blocking probability, stack of banyan networks.

I. Introduction

To construct a PRAM-like computing machine built by a very large number of processor elements (PEs), the device devoted to the exchange of information between processors and memories is a very critical point. In fact, efficient nonblocking permutation networks (e.g., necessary for the construction of a PRAM EREW machine) become more and more expensive when the number of their inputs (outputs) N increases (by using crossbar networks, which have $O(1)$ depth, topological complexity and cost increase with $O(N^2)$). For this reason various kinds of nonblocking permutation devices, built by means of blocking permutation networks with $O(N\log_2 N)$ topological complexity, have been extensively studied.

Among blocking permutation networks, multistage banyan networks are attractive for their moderate depth which guarantees that information reaches its destination in $\log_2 N$ steps, for the possibility to route information in pipeline by simple self-routing algorithms, and for their topological complexity which is $(N\log_2 N)/2$ nodes. They are considered a cost-effective alternative to crossbar networks for large N, but the fact that multistage banyan networks are blocking, compromises the above mentioned advantages when connection requests are simultaneously presented at all network inputs.

In some recent papers Lea [6], [7], Shyy and Lea [8], [10], and Melen [9] point out that vertical stacks of K banyan networks can be nonblocking or strictly nonblocking permutation networks under suitable conditions. These stacks (multi-log$N$ networks) maintain two important features of banyan networks:

1) the $\log_2 N$ or $O(\log_2 N)$ depth between inlet-outlet pairs, and
2) the self-routing capability.

Unfortunately these studies on multi-log$N$ networks do not provide routing algorithms with a time complexity comparable to the network depth.

A way to build routing algorithms on permutation networks is the probabilistic one [11], [12], [2], but, with this approach, some information cannot reach its destination, and therefore the probabilistic approach becomes valid if the amount of blocked information is negligible. Using this approach, in a recent work a quasi-nonblocking self-routing interconnection network with $\log_2 N$ depth was introduced [2]. In the present work a network, which is more efficient for very large N, is presented. It is a $3\log_2 N - 2$ depth multi-log$N$ network on which a probabilistic self-routing algorithm acts. The overall system (network and algorithm) is *asymptotically nonblocking* in the sense that its *blocking probability pb_N* vanishes when N increases.

II. Asymptotically Nonblocking Networks

Let B_N be a permutation network of size N (with N inputs and N outputs) and let $I = \{i_n\}$ and $O = \{o_n\}$, $n = 1, \ldots, N$, be the sets of its inputs and outputs respectively. B_N realizes N simultaneous one-to-one connections between each i_n and each o_n. In other words, B_N performs a bijection $I \rightleftharpoons O$. The one-to-one mappings of I onto O are characterized by the set of all permutations $P = \{p_j\}$, $j = 1, \ldots, N!$, of the elements of I onto O. In nonblocking permutation networks all connection requests presented at the inputs i_n reach their destinations. If B_N is a blocking network, a certain number of requests cannot be honored. The ratio $pb_N = (r_{in} - r_{out})/r_{in}$, where r_{in} is the number of simultaneous connection requests (input) and r_{out} is the number of nonblocked requests (outputs), is the blocking probability of B_N [11], [12], [2] and represents the probability that a request at the generical input i_n cannot reach its destination o_n when N requests are simultaneously applied on the whole input set I. pb_N can depend on N (as an example, in banyan multistage networks pb_N increases when N increases [11], [12]). The quantity $\eta_N = 1 - pb_N$ is the probability that a request at an input i_n reaches its destination o_n when N requests are simultaneously applied on the whole input set, and it will be called *efficiency* (efficiency is a measure of the nonblocking capability of a network, and nonblocking networks have $\eta_N = 1$ for any N). In banyan multistage networks η_N decreases when N increases [11], [12].

Definition 1. *A blocking B_N with efficiency η_N is called asymptotically nonblocking if:*

$$\lim_{N\to\infty} \eta_N = 1 \tag{1}$$

It is evident that the above defined interconnection structures have great importance in very massive systems (multiprocessors or communication systems).

Let $S_N = \{B_{N_k}\}$, $k = 1, \ldots, K$, be a set of K identical and independent permutation networks B_{N_k} of size N, the inputs and the outputs of S_N belong to the set $I^* = \{I_k\} = \{i_{n,k}\}$, and $O^* = \{O_k\} = \{o_{n,k}\}$ ($n = 1, \ldots, N$; $k = 1, \ldots, K$), respectively. K permutations can act simultaneously on the set S_N, and K one-to-one mappings $I \rightleftharpoons O$ can be simultaneously performed (each mapping on each B_{N_k}). $P^* = \{p_k\}$, $k = 1, \ldots, K$ ($P^* \subset P$), is the set of K uncorrelated permutations p_k, belonging to the set P, which are simultaneously applied on B_{N_k} networks. If B_{N_k} are blocking networks (with blocking probability pb_N), the *overall* blocking probability pb_N^* of the whole set S_N can be defined as:

Definition 2. *The overall blocking probability pb_N^* of a set S_N is the probability that, if K requests are simultaneously presented each one at an input i_n of one B_{N_k} network, no connection request reaches its destination (when $N \times K$ requests are simultaneously*

Manuscript received Oct. 25, 1993; revised Oct. 24, 1994.
The authors are with the Dipartimento di Scienze dell'Informazione, Università di Roma la Sapienza, Via Salaria 113, 00198 Roma, Italy; e-mail: debiase%astrom.hepnet@csa4.lbl.gov.
IEEECS Log Number C95064.

applied at the whole input set I^* of S_N).

If connection requests at the inputs of each B_{N_k} are completely independent (uncorrelated permutations p_k simultaneously act on each B_{N_k}), pb_N^* is given by (see e.g., [4]):

$$pb_N^* = pb_N^K \tag{2}$$

If $0 < pb_N < 1$, one can write:

$$\lim_{k\to\infty} pb_N^K = 0 \tag{3}$$

From (2) and (3) there follows that:

$$\lim_{K\to\infty}\left(1 - pb_N^K\right) = \lim_{K\to\infty} \eta_N^* = 1 \tag{4}$$

where η_N^* is the overall efficiency of the set S_N.

DEFINITION 3. *The overall efficiency η_N^* of the set S_N is the probability that, if K requests are simultaneously presented each one at an input i_n of one B_{N_k} network, at least one connection request reaches its destination (when $N \times K$ requests are simultaneously applied at the whole input set I^* of S_N).*

Equation (4) shows that for any size N, when K (the number of B_{N_k} networks) goes to ∞, the overall efficiency η_N^* of the set S_N goes to 1 under the sole condition that $0 < pb_N < 1$. According to Definition 3, the condition expressed by (1), which states that a permutation network is asymptotically nonblocking when the size N increases, can be generalized for a set S_N:

$$\lim_{N\to\infty} \eta_N^* = 1$$

THEOREM 1. *Let S_N be a set of identical and independent blocking networks B_{N_k} with blocking probability pb_N, and let K_N, depending on N, be the number of B_{N_k} networks of the set S_N. The set S_N is asymptotically nonblocking if all permutations p_k presented at B_{N_k} networks are uncorrelated, and if:*

$$K_N = -\left\lceil \frac{C(N)}{\ln pb_N} \right\rceil \tag{5}$$

where $0 < pb_N < 1$ for any N, and C(N) is any function for which:

$$\lim_{N\to\infty} C(N) = \infty \tag{6}$$

PROOF. The overall efficiency of a set S_N is $\eta_N^* = 1 - pb_N^*$, and $\lim_{N\to\infty} \eta_N^* = 1$ when $\lim_{N\to\infty} pb_N^* = 0$. There follows from (2) that: $\lim_{N\to\infty} pb_N^* = \lim_{N\to\infty} pb_N^{K_N} = 0$ which is true if:

$$\lim_{N\to\infty} K_N \ln pb_N = -\infty \tag{7}$$

It is easy to see that, with the substitution $K_N = -C(N)/\ln pb_N$, (7) is always verified if $\lim_{N\to\infty} C(N) = \infty$. The ceiling in (5) is necessary to guarantee integer K_N values. □

Theorem 1 states that a set of blocking networks S_N is asymptotically nonblocking if K_N increases according to (5).

III. AN ASYMPTOTICALLY NONBLOCKING SELF-ROUTING PERMUTATION DEVICE

Based on the results given in Section II, a new self-routing asymptotically nonblocking permutation device, based on stacks of K_N blocking networks, can be carried out if the following assumptions hold:

- the permutation p_j (input of the whole device) is transformed into a set P^* of K_N uncorrelated permutations p_k,
- each permutation p_k of the set P^* acts independently on one B_{N_k} network of the set S_N.

Finally, every output of the device is obtained by the logical union of the corresponding outputs o_n of all B_{N_k} networks (see Definition 3).

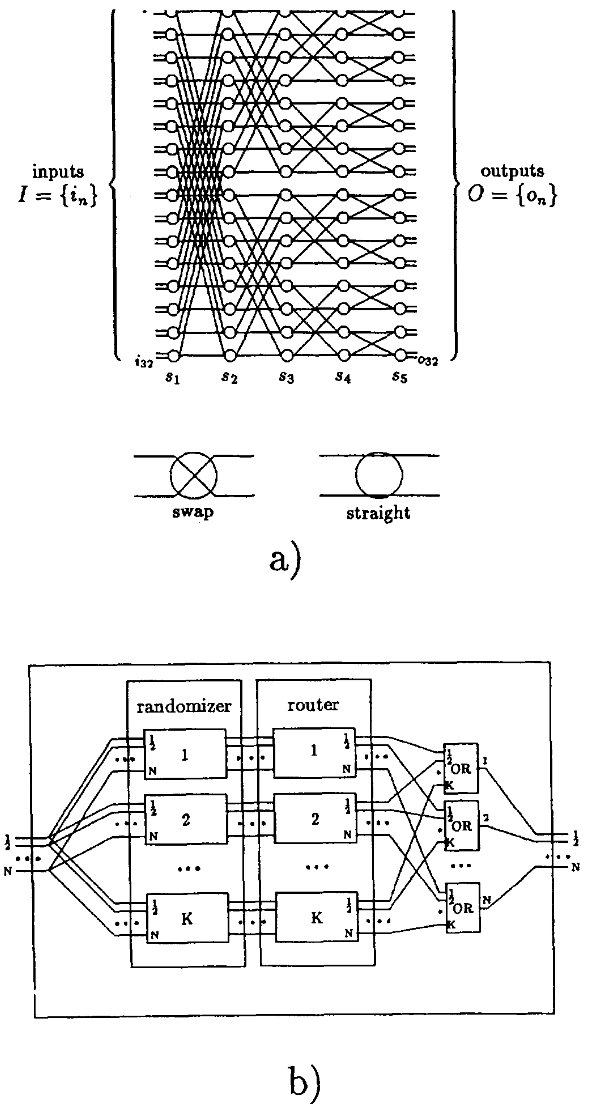

Fig. 1. (a) A popular banyan network: the butterfly. This network is plotted for $N = 32$ and $H = 5$. Below: Its 2×2 node with its permitted states. (b) The permutation device.

Now, let $B_{N_k} = \{s_h\} (h = 1, \ldots, H)$ be a multistage banyan network of size N, consisting in $H = \log_2 N$ cascaded stages s_h, each made by $N/2$ two-state 2×2 nodes (see Fig. 1a), and let $S_N = B_{N_k}$ be a vertical stack of K_N independent banyan networks B_{N_k}. The self-routing permutation device consists of two parts: The first part (randomizer) is devoted to transforming the input permutation p_j into a set P^* of K_N

uncorrelated permutations p_k, while the second one (router) addresses connection requests towards their destinations. The output ports o_n of the whole device are obtained by the logical OR of the corresponding output ports of all B_{N_k} (see Fig. 1b). To easily obtain self-routing capability, this permutation device is built by stacks, the planes of which are butterfly networks, as shown in Fig. 2. As one can see, in all planes the output nodes of a network are in common with the input nodes of the subsequent network, for this reason the device depth is $3\log_2 N - 2$ stages.

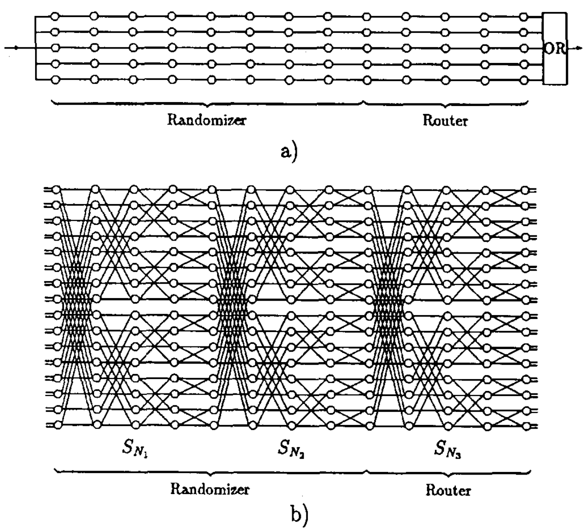

Fig. 2. (a) Vertical section of the permutation device for $N = 2^5$. The corresponding outputs of each plane are ORed. (b) A device plane consisting of three butterfly networks. The output nodes of a butterfly are in common with the input nodes of the subsequent butterfly.

A. The Randomizer

To generate the set P^* of K_N uncorrelated permutations p_k, a way similar to that discussed in [3] can be efficiently used. In that work it is pointed out that two cascaded banyan networks, which are isomorphic to a Benes network, are effective in generating random permutations. Hence, two cascaded stacks (S_{N_1} and S_{N_2}, see Fig. 2) act as a randomizer.

At the inputs of each plane of the first stack, K_N copies of the same permutation p_j are presented simultaneously. In each plane of each stack (constructed by butterfly networks) the nodes are set, at each time T, on a randomly chosen status (swap or straight). In this way, on each B_{N_k}, N one-to-one connections between any input i_n and any output o_n are always obtained, and each connection request at an input i_n always reaches a random chosen output. Hence the permutation p_j is split into a set P^* of K_N uncorrelated permutations p_k [3].

B. Information Routing

At the output of the randomizer the conditions for a correct application of (2) are verified. Requests are routed to their destinations by a third stack S_{N_3} (consisting of K_N butterfly networks too) on which runs the same simple distributed algorithm presented in [12] which works in parallel on all planes and on all nodes stage-by-stage, namely:

- on each node of stage h, each request is routed following its binary destination address, namely: Nodes on stage h are set in a way that the requests are routed to the upper or lower node terminal if the hth most significant bit of the destination address is 0 or 1, respectively,
- if on a node two requests claim simultaneously two different node states (conflict occurrence), the state of the node is randomly chosen, and only one request continues along its correct path.

C. Number of Planes

Equation 5 gives the number of planes K_N of the routing stack and, consequently, of the randomizer. The values of efficiency η_N, for $N = 2^1, 2^2, 2^3, \ldots$ of a banyan multistage network, under permutation requests, are recursively given with good accuracy by the model presented in [12] by Szymansky and Hamacher:

$$\eta_{N_{h+1}} = \sum_{i=1}^{2} \binom{2}{i} \eta_{N_h}^{i} \left(1 - \eta_{N_h}\right)^{2-i} \cdot \sum_{t=1}^{i} \binom{i}{t} \frac{\binom{2^{H-h}}{t}\binom{2^{H-h}}{i-t}}{\binom{2^{H-h+1}}{t}\binom{2^{H-h+1}}{i-t}} \tag{8}$$

starting from: $\eta_{N_1} = \frac{3 \cdot 2^{H-1} - 1}{2(2^H - 1)}$. In (8) $h = 1, \ldots, H - 1$, and $H = \log_2 N$ is the number of network stages. This model can be used to compute, by (2), the blocking probability of the router planes, in fact pb_N values can be obtained by (8) by the substitution $pb_N = 1 - \eta_N$.

When in an interval (N_a, N_b), K_N values are given by a function $f(N) \geq -C(N)/\ln pb_N$, Theorem 1 guarantees that the efficiency η_N^* of the set S_N increases with N for all values of N belonging to the same interval. Among these functions, $K_N = \log_2^{\gamma} N$ is a good compromise between network complexity and efficiency increase. In this case the topological complexity of the presented network is $N(\log_2^{\gamma} N)(3\log_2 V - 2)/2$ nodes and, also using γ values very close to 1 ($\gamma = 1.05, 1.10, \ldots$), the network efficiency quickly increases in a wide interval of N, as shown in Fig. 3a.

IV. SIMULATIONS

The behavior of the device has been examined by numerical simulation under the assumption of permutation requests [11]. Simulations give the values of the device efficiency η_N^S versus N when a suitable function $K_N = f(N)$ is chosen. The numerical program simulates the behavior of all stacks. For each N it utilizes, as input of the whole device, a randomly chosen permutation p_j. The randomization of requests is obtained only by setting all the nodes of each plane of the randomizer on randomly chosen states. The routing of requests is obtained on the routing stack by the simple distributed algorithm presented in Section III.B. Because the rapid increase with N of the number of permutations p_j ($N!$) generates very large computation times, to obtain the values of η_N^S a number of attempts, sufficient to reach at least the 99% confidence level, has been executed in the interval $2^3 \leq N \leq 2^{13}$.

Simulated values of the device efficiency η_N^S compared with η_N^* computed values, when $K_N = \log_2 N$ and $K_N = \log_2 N - 1$, are presented in Fig. 3b. The blocking probability of the component banyan networks are computed by (8), and the efficiency of the whole device by (2). The model of Szymansky and Hamacher generates slightly overestimated pb_N values (see Fig. 3 in [11] and Fig. 4 in [12]), and consequently the simulated efficiency values η_N^S of the permutation device are slightly greater than the computed ones, η_N^* (about 0.25% in the considered interval of N).

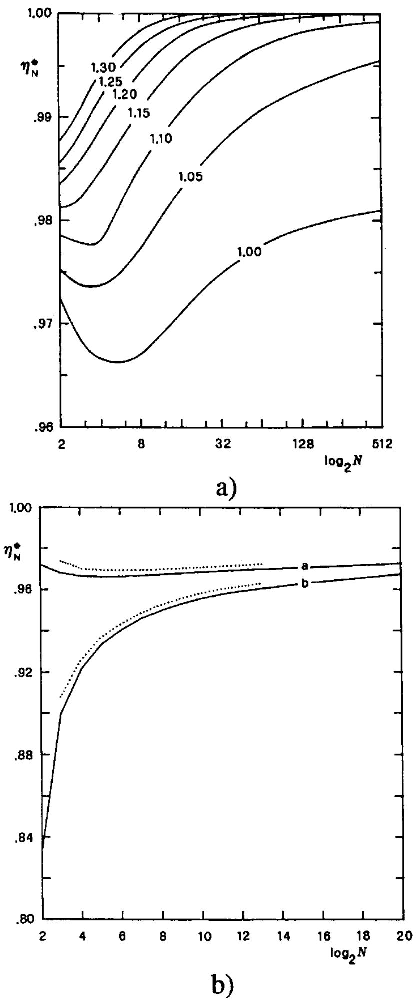

Fig. 3. (a) Efficiencies η_N^* of the permutation device versus $\log_2 N$; $K_N = \log_2^\gamma N$ (γ = 1.00, 1.05, 1.10, 1.15, 1.20, 1.25, 1.30). (b) Computed efficiency values, η_N^*, versus $\log_2 N$: plot a is $K_N = \log_2 N$, plot be is $K_N = \log_2 N - 1$. The dotted line represents efficiency values η_N^S obtained by numerical simulation.

V. Concluding Remarks

The architecture presented is inherently fault tolerant, in fact it consists of three vertical stacks each of K_N banyan networks which implement many physical paths for each logical path. Faults on every network of each stack act on the overall performance of the structure. In absence of faults the overall efficiency of M cascaded stacks S_{N_j} is given by: $\eta_N^* = \Pi_{j=1}^M \eta_{N_j}^* = \Pi_{j=1}^M \eta_{N_j}^{K_N}$. In the presented device $\eta_N^* = \eta_{N_3}^*$, because $\eta_{N_1}^* = \eta_{N_2}^* = 1$ in the randomizer, while $\eta_{N_3}^*$ is given by (2) and (8).

If in presence of faults the efficiency of each component network decreases by a quantity $\Delta\eta_N$, the degradation of the efficiency of the whole device $\Delta\eta_N^*$ can be computed by:

$$\eta_N^* - \Delta\eta_N^* = \prod_{j=1}^{3}\left(\eta_{N_j} - \Delta\eta_{N_j}\right)^{K_N} \tag{9}$$

Now, by using (9) the degradation of the efficiency of the whole permutation device can be examined versus the efficiency degradation of component networks. In Fig. 4, $\Delta\eta_N^*$ values versus the size N, for several γ values, are plotted. A small (10%) and a strong (40%) value for the efficiency degradation of all component networks is considered. The efficiency values of component banyan networks, in the absence of faults, are assumed $\eta_N = 1$ for the randomizer, while η_N values are computed by (8) for the router. As one can see from the figures, also with γ values very close to 1, fault tolerance quickly increases with N.

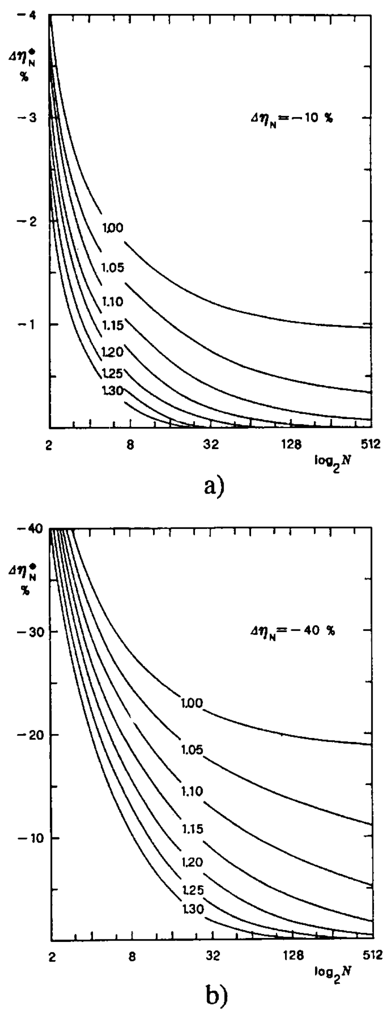

Fig. 4. Efficiency degradation of the whole permutation device, $\Delta\eta_N^*$, versus $\log_2 N$ for several γ values. The degradatioon of the efficiency of all component banyan networks is a) $\Delta\eta_N$ = 10% and b) $\Delta\eta_N$ = 40%.

The multistage structure of the permutation device, the distributed self-routing algorithm, and the forward direction of the information flux, guarantee that the information wavefront synchronously passes through the network stages in $3\log_2 N - 2$ steps. Then the system can work in pipeline, and information can be presented at the device inputs at each time interval T, where T is the stage-to-stage propagation time.

The behavior of the efficiency of the device has been examined under the assumption of permutation request patterns. The efficiency values, closer and closer to 1 for large N, guarantee that, also under the assumption of random request patterns [12], the behavior of the permutation device is very close to that of nonblocking networks [2]. In the presented permutation device the two most important features of banyan networks are maintained: moderate depth ($3\log_2 N - 2$ stages) and simple request routing (obtainable by a self-routing distributed algorithm, which permits pipelined operations). This permutation network becomes asymptotically nonblocking when a suitable increase with N of the number of planes, K_N, is chosen (see Theorem 1). When the planes are banyan networks, this behavior is already possible with $K_N = \log_2 N$. In this case the topological complexity (number of nodes) is $O(\log_2^2 N)$, which is the same as the Koppelman-Oruç and Batcher networks [5], [1], that are nonblocking but have a worse depth ($O(\log_2^2 N)$ instead of $O(\log_2 N)$). With a very little increase in topological complexity, the desired efficiency of this permutation device can be quickly reached for any N. The behavior of efficiency and fault tolerance clearly highlights that this network becomes more and more advantageous as device size increases.

REFERENCES

[1] K.E. Batcher, "Sorting networks and their application," *Proc. Spring Joint Computer Conf.*, pp. 307-314, 1968.

[2] G.A. De Biase, C. Ferrone, and A. Massini, "A quasi-nonblocking self-routing network which routes packets in $\log_2 N$ time," *Proc. IEEE INFOCOM 93*, pp. 1,375-1,381, 1993.

[3] R L. Cruz, "The statistical data fork: A class of broad-band multichannel switches," *IEEE Trans. Communications*, vol. 40, pp. 1,625-1,634, 1992

[4] D.V. Huntsberger, *Statistical Inference*. Boston: Allyn and Bacon Inc., 1967.

[5] D.M. Koppelman and A.Y. Oruc, "A self-routing permutation network," *J. Parallel and Distributed Computing*, vol. 10, pp. 140-151, 1990.

[6] C.T. Lea, "Bipartite graph design principle for photonic switching systems," *IEEE Trans. Communications*, vol. 38, pp. 529-538, 1990.

[7] C.T. Lea, "Multi-$\log_2 N$ networks and their applications in high speed electronic and photonic switching systems," *IEEE Trans. Communications*, vol. 38, pp. 1,740-1,749, 1990.

[8] C.T. Lea and D.J. Shyy, "Tradeoff of horizontal decomposition versus vertical stacking in rearrangeable nonblocking networks," *IEEE Trans. Communications*, vol. 39, pp. 899-904, 1991.

[9] R. Melen, "A general class of rearrangeable interconnection networks," *IEEE Trans. Communications*, vol. 39, pp. 1,737-1,739, 1991.

[10] D.J. Shyy and C.T. Lea, "$\text{Log}_2(N,m,p)$ strictly nonblocking networks," *IEEE Trans. Communications*, vol. 39, pp. 1,502-1,510, 1991.

[11] T.H. Szymansky and V.C. Hamacher, "On the permutation capability of multistage interconnection networks," *IEEE Trans. Computers*, vol. 36, pp. 810-822, 1987.

[12] T.H. Szymansky and V.C. Hamacher, "On the universality of multipath multistage interconnection networks," *J. Parallel and Distributed Computing*, vol. 7, pp. 541-569, 1989.

Contention-Free 2D-Mesh Cluster Allocation in Hypercubes

Stephen W. Turner, Lionel M. Ni, and Betty H.C. Cheng

***Abstract*—Traditionally, each job in a hypercube multiprocessor is allocated with a subcube so that communication interference among jobs may be avoided. Although the hypercube is a powerful processor topology, the 2D mesh is a more popular application topology. This paper presents a 2D-mesh cluster allocation strategy for hypercubes. The proposed auxiliary free list processor allocation strategy can efficiently allocate 2D-mesh clusters without size constraints, can reduce average job turnaround time compared with that based on subcube allocation strategies, and can guarantee no communication interference among allocated clusters when the underlying hypercube implements deadlock-free E-cube routing. The proposed auxiliary free list strategy can be easily implemented on hypercube multicomputers to increase processor utilization.**

***Index Terms*—Hypercube, processor allocation, 2Dimensional mesh, job turnaround time, message routing.**

I. INTRODUCTION

The problem of subcube allocation has been studied extensively to maximize processor utilization and minimize system fragmentation in hypercubes. Several strategies have been proposed and implemented for subcube allocation, including the buddy strategy [1], the gray code (GC) strategy [2], the modified buddy strategy [3], the free list strategy [4], and the tree collapsing strategy [5]. Of these approaches, only the free list and tree collapsing strategies have been shown to perform optimally, since they provide perfect subcube recognition.

For hypercube machines, such as the nCUBE-2 and the newly announced nCUBE-3, the restriction of allocating subcubes causes low processor utilization. Although the hypercube is a powerful network topology [6], 2D and 3-D meshes are more popular application topologies. For example, grid domain decomposition for solving partial differential equations is an application that can easily be implemented on 2D and 3-D meshes. In addition, 2D and 3-D meshes can allocate exactly (or close to) the number of processors requested. For example, if the optimal number of processors for a task is 600, then the smallest subcube that can be allocated is 1,024 processors, resulting in a waste of 424 processors, while a 2D mesh may allocate a 20×30 cluster.

Consider the 4-dimension cube shown in Fig. 1, in which one job is allocated a 2×5 mesh, and another job is allocated a 2×3 mesh. With a restriction to subcube allocation, both jobs cannot be simultaneously executed, even though the total number of processors, 16, is sufficient. If the restriction to subcube allocation is removed, both clusters may be allocated in the 4-cube. However, a closer look reveals that communication from node 0100 to node 1010 in the 2×5 cluster will potentially cause link contention with communication between nodes 0110 and 0010 in the 2×3 cluster, if the popular deadlock-free E-cube routing is used [7]. This contention potentially results in *intercluster communication interference*, which should be minimized. Many known processor allocation strategies have been developed to guarantee contention-free cluster allocation, such as those subcube allocation strategies for hypercubes, the strategy used in the Intel Touchstone (2D mesh topology), and the one used in

Manuscript received June 10 1993; revised May 17, 1994.

S.W. Turner is with Ford Systems Integration Center, Allen Park, MI 48101.

L.M. Ni, and B.H.C. Cheng are with the Department of Computer Science, Michigan State University, East Lansing, MI 48824-1027;

e-mail: ni@cps.msu.edu.

IEEECS Log Number C95082.