Fast Self-Routing Permutation Switching on an Asymptotically Minimum Cost Network

Ching Yuh Jan and A. Yavuz Oruç, *Senior Member, IEEE*

Abstract— **Permutation switching is a key operation in many computer and communication systems. The well-known Beneš and Waksman permutation networks can be constructed with an asymptotically minimum number of switches, but the best routing algorithms for such networks need $O((\lg^4 n/\lg\lg n))$ time on an $n \lg n$-processor computer. Other networks that can be used for permutation switching are Batcher's sorting networks and Koppelman and Oruç's self-routing permutation network, but these networks require $O(n \lg^2 n)$ switches and $O(\lg^2 n)$ routing time. Using a new composite interconnection network model, this paper presents a self-routing permutation network with $O(n \lg n)$ switches and $O(\lg^2 n)$ routing time. More generally, it describes a permutation network with $O(kn^{1+1/k})$ cost and $O(k \lg n)$ routing time for any k: $1 \leq k \leq \lg n$. This improves Nassimi and Sahni's routing algorithm that gives $O(k \lg^3 n)$ routing time for the same cost expression. The only networks capable of permutation switching with $O(n \lg n)$ cost and $O(\lg n)$ routing time are the AKS sorting network and Upfal's packet routing network, but the constants hidden in the complexities of these networks are so large that they remain impractical until n gets very large.**

Index Terms— **Concentrator, cube network permutation network, radix permuter, radix sorting, self-routing network.**

I. Introduction

LOOSELY speaking, a permutation network is a switching structure with two finite sets of terminals, $X = \{x_1, x_2, \cdots, x_n\}$, called *inlets* and $y = \{y_1, y_2, \cdots, y_n\}$, called *outlets* such that, for each permutation map π between X and Y, inlet x_i can be connected to outlet $\pi(x_i), 1 \leq i \leq n$. In this paper, we consider the problem of designing a permutation network that has as close an optimal behavior as possible in three respects: It must have a) as mall a *cost* as possible, b) as mall a *depth* as possible, and c) as short a *routing time* as possible. The formal definitions of cost, depth and routing time will be given in Section II. Here, it suffices to say that the cost of a network is proportional to the number of devices encompassing $O(\lg n)$ or fewer binary logic gates,[1] its depth is proportional to the largest number of such devices that fall between an inlet and an outlet, and its routing time is the time it takes to determine the "behavior" of each such device, given a permutation π between X and Y.

Manuscript received September 13, 1991; revised November 2, 1992. This work is supported in part by the National Science Foundation under Grant CCR-8708864, and in part by the Minta Martin Fund of the School of Engineering at the University of Maryland.

The authors are with the Department of Electrical Engineering and the Institute for Advanced Computer Studies, University of Maryland, College Park, MD 20742-3025.

IEEE Log Number 9208758.

[1]This is an abuse of notation. What is meant here is that the number of logic gates in such devices grows at most at a rate of lg n.

Variations of the permutation network problem have been studied in the literature. For example, the problem of designing a permutation network with minimum cost was posed and long settled in an asymptotical sense by Beneš [4] who gave an explicit permutation network with $O(n \lg n)$ cost and $O(\lg n)^2$ depth. Other results followed this pioneering work to improve the constants, e.g., see [1], [8], [22]. The asymptotical cost optimality of Beneš' permutation network follows from a result of Shannon which states that any permutation network with n inlets must have $\Omega(n \lg n)$ binary switching devices [19]. Moreover, it was established in [10] that a permutation network with $O(n \lg n)$ cost must have $\Omega(\lg n)$ depth so that the Beneš network is both cost and depth optimal in an asymptotical sense.

Yet, the Beneš network has so far proven to have very poor routability: For an n-inlet Beneš network, the best known serial routing algorithm, called the looping scheme [15], [22], takes $O(n \lg n)$ time, and the best parallel routing algorithm takes $O(\lg^2 n)$ time but on an n-processor computer with $O(n^2)$ connections [13]. The complexity of the routing hardware can be reduced to $O(n \lg n)$ to match the cost of the network, but at the expense of increasing the routing time. In particular, the routing algorithm in [13] requires $O(\lg^4 n/\lg\lg n)$ time on a cube-connected or perfect shuffle computer with $O(n \lg n)$ processors.[3]

The excessive routing complexity of the Beneš network can be controlled by adjoining additional paths between the switches in the network. In fact, this approach was used successfully in [11] to obtain a self-routing permutation network with $O(n \lg^2 n)$ cost, $O(\lg^2 n)$ depth and $O(\lg^2 n)$ routing time. A more direct approach is to use Batcher's odd-even merge or bitonic sorting networks [3], [9] as they also have $O(n \lg^2 n)$ cost and $O(\lg^2 n)$ depth and $O(\lg^2 n)$ routing time. It should be noted that Stone's single-stage shuffle-exchange network implementation of Batcher's bitonic sorting gives a permutation network with $O(n)$ cost and $O(\lg^2 n)$ routing time. However, this network cannot be pipelined, and requires that the data be recirculated $O(\lg^2 n)$ times along with their addresses. This may be especially undesirable when the data size is much larger than the address and could increase the routing time beyond $O(\lg^2 n)$.

[2]All logarithms are in base 2 unless otherwise stated, and $lg\, n$ denotes $\log_2 n$. It is also assumed that n is a power of 2 unless otherwise stated.

[3]This time complexity is obtained by solving k from $n^{1+1/k} = n \lg n$, and substituting it in the time complexity expression $O(k \lg^3 n)$ that is given in [13].

We also note that a permutation network can be constructed by using certain bipartite graphs, called *expanders* [6], [17]. The AKS sorting network [1] and Upfal's packet routing permutation network [23] are based on such graphs. Even though both these networks have $O(n \lg n)$ cost and $O(n \lg n)$ routing time, the constants hidden in the O notations are very large, making these networks impractical unless n is extremely large. More recently, Leighton and Maggs [12] and Arora *et al.* [2] reported some results that seem to improve the constants involved in the complexities of these networks. However, their results depend on random generations of expander graphs with small degrees, and they state in [12] that they were working on finding a good time bound on random generations of expanders with small degrees.

The main result of this paper is the design of a permutation network with $O(n \lg n)$ cost, $O(\lg^2 n)$ depth and $O(\lg^2 n)$ routing time. More generally, we present a permutation network with $O(kn^{1+1/k})$ cost and $O(k \lg n)$ routing time for any $k; 1 \leq k \leq \lg n$. The significance of this network is that, when $k = \lg n$, its cost is asymptotically minimum, and yet it does not require routing hardware that is beyond its own cost. In addition, its routing time matches the routing time of the previously known best network designs. For other values of k, the permutation network of this paper outperforms the Beneš network with parallel routing schemes described in [13]. For the same cost expression, the time complexity of these parallel routing schemes is $O(k \lg^3 n)$ as compared to the $O(k \lg n)$ routing time of the permutation network given in this paper.

The key concept behind our permutation network is the notion of radix sorting, and a new composite network model. Under this new model we decompose the problem of permuting into two subproblems: a distribution problem and a concentration problem. The distribution problem is solved with a straightforward network that has $O(n)$ cost, $O(1)$ depth and $O(1)$ routing time. The concentration problem is trickier and is solved by sandwiching a cube network in between two sets of binary trees. The optimization of the cost, depth and routing time over the three components of this design then yields a concentrator with $O(n)$ cost, $O(\lg n)$ depth and routing time. Finally combining the distribution and concentration steps together into a recursive construction yields a permutation network with $O(n \lg n)$ cost, $O(\lg^2 n)$ depth and $O(\lg^2 n)$ routing time.

The rest of the paper is organized as follows. Section II states the basic notions and definitions needed in subsequent sections. Section III gives a brief overview of radix sorting and the structure of our permutation network. Sections IV and V describe the distributor and concentrator constructions used in this permutation network. Section VI determines the cost, depth and routing time complexities of this network, and the paper is concluded in Section VII.

II. Definitions

This section states some facts and definitions that are pertinent to a precise description of our results.

Definition 1: An (n, q)-network is a triple $(\mathcal{G}, \mathcal{F}, \mathcal{R})$ where a) $\mathcal{G}$ is an underlying graph with n distinguished nodes $x_0, x_1, \cdots, x_{n-1}$, called inlets, q distinguished nodes $y_0, y_1, \cdots, y_{q-1}$, called outlets, and finite set of nodes, called operation nodes; b) $\mathcal{F}$ is a set of well-defined computations with domain $\{x_0, x_1, \cdots, x_{n-1}\}$ and range $\{y_0, y_1, \cdots, y_{q-1}\}$; and c) $\mathcal{R}$ is an algorithm, called a routing scheme, that defines, for any computation $h \in \mathcal{F}$, the exact behavior of each operation node so that the tokens at the inlets are computed to the outlets as specified by h. ||

An operation node in a network can be as simple as a 2-inlet switch and as complex as the network itself. In fact, all the networks described in this paper are recursively defined so that they have operation nodes that are as complex in behavior as themselves.

The routing scheme of a network determines what each operation node must do in order to route the *tokens* to the outlets. Networks described in this paper have a particular routing scheme that is commonly referred to as *self-routing*.

Definition 2: A network $(\mathcal{G}, \mathcal{F}, \mathcal{R})$ is called self-routing if the routing scheme $\mathcal{R}$ is distributed over the operation nodes, i.e., each operation node has its own routing scheme, and its behavior is determined only by its tokens and its own routing scheme. ||

Remark 1: Even though this definition does not restrict the size of the tokens that are processed by operation nodes, all networks described in this paper process tokens with $O(\lg n)$ bits, where n is the number of inlets to the network. This includes both data and destination address.

To each network, we attach three performance parameters. All three parameters are defined recursively. The *cost* of a network is the sum of the costs of its constituent operation nodes. It is assumed that each operation node can be implemented using $O(\lg n)$ constant fanin logic gates for an n-inlet network. The *depth* of a network is the largest sum of the depths of its operation nodes that lie between an inlet and an outlet. It is assumed that the largest number of constant fanin logic gates between an input and outlet of an operation node in an n-inlet network is $O(\lg \lg n)$. The *routing time* of a network is the largest sum of the routing times of its components that lie between an inlet and an outlet, assuming that the network is self-routing.

For a network $(\mathcal{G}, \mathcal{F}, \mathcal{R})$, the computations in $\mathcal{F}$ define the functionality of the network. In particular we will, subsequently, encounter the following types of networks.

Definition 3: An (n, q)-network $(\mathcal{G}, \mathcal{F}, \mathcal{R})$ is called an n-inlet permutation network, or n-permuter, if $n = q$, and $\mathcal{F}$ is the set of all permutations according to each of which the tokens at the inlets of the network can be routed to its outlets.||

Definition 4: An (n, q)-network, where $q \leq n$, is called an (n, q)-concentrator, if the tokens at any k of the n inlets, $1 \leq k \leq q$, can be routed to some k *specified* outlets. ||

Remark 2: Every (n, q)-concentrator is an (n, q')-concentrator for any $q' \leq q$. This fact will be needed in Section V.

We note that the definition of a concentrator given here differs from a widely used definition of a concentrator which requires that the inlets be concentrated to outlets over vertex or link disjoint paths [6], [17]. The definition given here allows paths to be pipelined and shared by inlets (tokens).

Definition 5: An (n, fn)-distributor, where f is any positive integer, is a network with n inlets and f groups of n outlets, $O_0, O_1, \cdots, O_{f-1}$, such that, for any partition of the inlets into $r_0, r_1, \cdots, r_{k-1}$ inlets, where $r_0, r_1, \cdots, r_{k-1}. 1 \leq k \leq f$, are arbitrary integers satisfying $r_0+r_1+\cdots+r_{k-1} = n$, the tokens that belong to the set of r_i inlets can be routed to some r_i outlets in O_i for $i = 0, 1, \cdots, k-1$. ||

Definition 6: Let the inlets of an n-inlet network be labeled $0, 1, \cdots, n-1$. The rank of a token at inlet i is the number of tokens that occupy inlets that are less than i. An $(n. n)$-network is called an n-ranker, or n-inlet ranking circuit, if it returns the rank of the token at its ith inlet to its ith outlet for $0 \leq i \leq n-1$. The outlets that correspond to inlets without tokens are left unspecified. ||

Definition 7: An (n, n)-cube network, where n is a power of 2, is recursively formed by cascading two $(n/2. n/2)$-cube networks with a stage of $n/2(2, 2)$-switches such that one inlet of each switch is connected to one $(n/2. n/2)$-cube network, and its other inlet is connected to the other $(n/2, n/2)$-cube network. When the $(n/2, n/2)$-cube networks are recursively decomposed then an (n, n)-cube network consists of lg n stages each with $n/2(2, 2)$-switches. A detailed description of cube networks can be found in [11], [14], [16], [20]. ||

III. Radix Permuters

Given a set A of binary numbers, each with $b = \lg n$ bits, divide them into two sets A_0 and A_1 where A_0 is the set of numbers whose most significant bits are 0, and A_1 is the set of numbers whose most significant bits are 1. Repeat this process for the second bit, and let A_{00}, A_{01}, A_{10}, A_{11} denote the sets of numbers whose leftmost two bits are 00. 01. 10. 11 in that order, and iterate it for the remaining bits, each time dividing the numbers in each set into two other sets. If $A = \{0, 1, \cdots, n-1\}$ then the set $A_{a_{b-1}\, a_{b-2} \cdots a_{b-s}}$, where $a_{b-1}, a_{b-2}, \cdots, a_{b-s} \in \{0, 1\}, 1 \leq s \leq b$, consists of exactly 2^{b-s} numbers whose leftmost s bits match $a_{b-1}\, a_{b-2} \cdots a_{b-s}$. We call this set an *index set* of A of *degree* s, and call $a_{b-1}\, a_{b-2} \cdots a_{b-s}$ its *index*. In particular, $A_{a_{b-1} \cdots a_1\, a_0}$ is a singleton set containing the number $a_{b-1} \cdots a_1\, a_0$, and the numbers in A are sorted in ascending order into the sets $A_{00\cdots0}, A_{00\cdots1}, \cdots, A_{11\cdots1}$.

The most obvious application of radix sorting to permutation network design is obtained by cascading two groups of n binary trees back to back as shown in Fig. 1. The roots of the n binary trees in the first group correspond to the inlets of the network, and the roots of the n binary trees in the second group correspond to its outlets. The leaf nodes of the first group of binary trees are connected to the leaf nodes of the second group of n binary trees in such a way that each leaf node in each binary tree in the first group meets with a leaf node in a distinct binary tree in the second group. This is effectively equivalent to the radix sorting process outlined above where the leaf nodes of the first group of binary trees correspond to the index sets of A of degree lg n. More generally, the upper four nodes at the first level (i.e., the upper four child nodes of the root node) correspond to the index set A_0 and the lower four nodes correspond to the index set A_1, the four groups

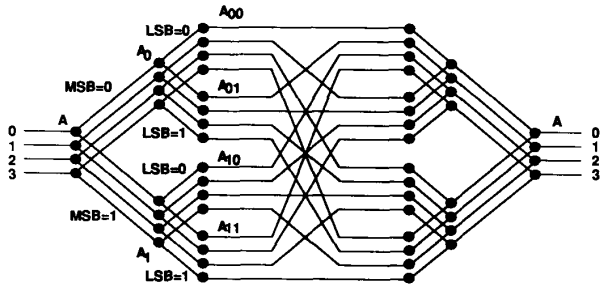


Fig. 1. A 4-inlet permutation network using radix sorting.

of four nodes at the second level correspond to the index sets A_{00}, A_{01}, A_{10}, A_{11}, and so on.

Routing on this network proceeds as follows: Each token that enters the network at an inlet has a destination address that identifies the outlet it wants to reach. Accordingly, routing a permutation over this network amounts to decoding the destination addresses of the tokens over the binary trees. The first set of binary trees are used to distribute the tokens to the right index sets. Once the tokens reach the leaf nodes of the first group of binary trees (i.e., the index sets A_{00}, A_{01}, A_{10}, A_{11}) then the second group of binary trees route these tokens to the desired outlets.

The main idea of the radix sorting scheme outlined above is to keep separating the tokens into disjoint index sets until they find their destinations in the index sets $A_{00\cdots0}, A_{00\cdots1}, \cdots, A_{11\cdots1}$. To accomplish this, a path is made available between each inlet and all such index sets to which a token that enters the network at that inlet may potentially belong. However, since there are only n tokens propagating through the network at any given level in time, some of these paths are wasted. For example, in the network of Fig. 1, each inlet has a path to four index sets at the second level even though any token entering the network at that inlet can belong to only one of the index sets at that level. Call those inlets (outlets) in a given level that hold tokens *live* inlets (outlets). The fact that only some of the inlets or outlets are live at each level suggests that we may reduce the cost of the network construction given in Fig. 1, by "concentrating" the tokens in the live outlets. We formalize this idea in the form of a recursive network construction that is called a *radix permuter*, and shown in Fig. 2. Without loss of generality, we assume that n is a power of 2, and let $f = 2^s; 1 \leq s \leq \lg n$ be a positive integer which, obviously, divides n. The first stage of the network encompasses an (n, fn)-distributor, the second stage consists of $f(n, n/f)$-concentrators, and the third stage consists of fn/f-radix permuters. The parameter f is called the *fanout factor* of the permuter.

Each group of n outlets of the (n, fn)-distributor represents an index set of degree s which is identified by a distinct s-bit code. The distributor routes a token to an outlet in an index set if and only if the leftmost s bits in the destination address of that token match the degree of that index set. Thus, the distributor routes a total of $n/2^s = n/f$ tokens to each index set, and the remaining $n - n/f$ outlets in each index set just float.

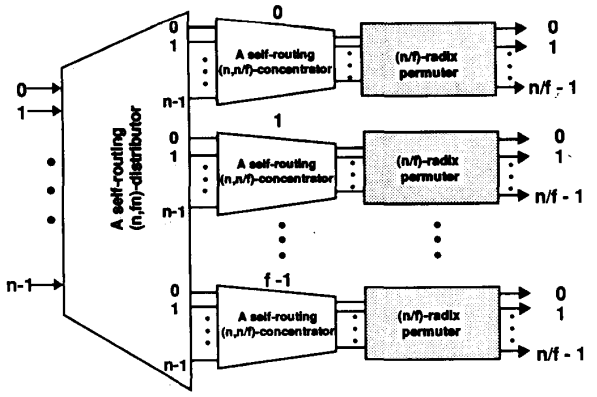


Fig. 2. An (n, n) radix permuter.

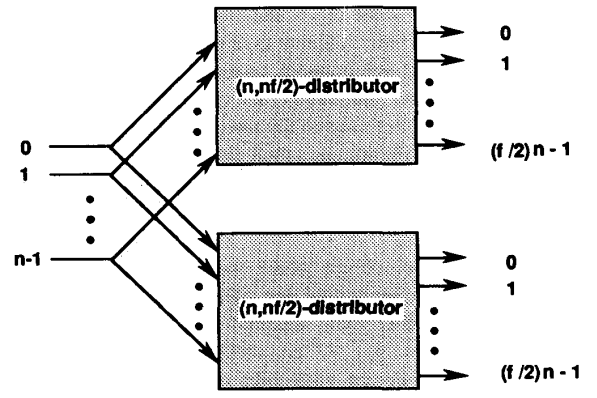


Fig. 3. An (n, fn) distribution network, $f = 2^2, s \geq 1$.

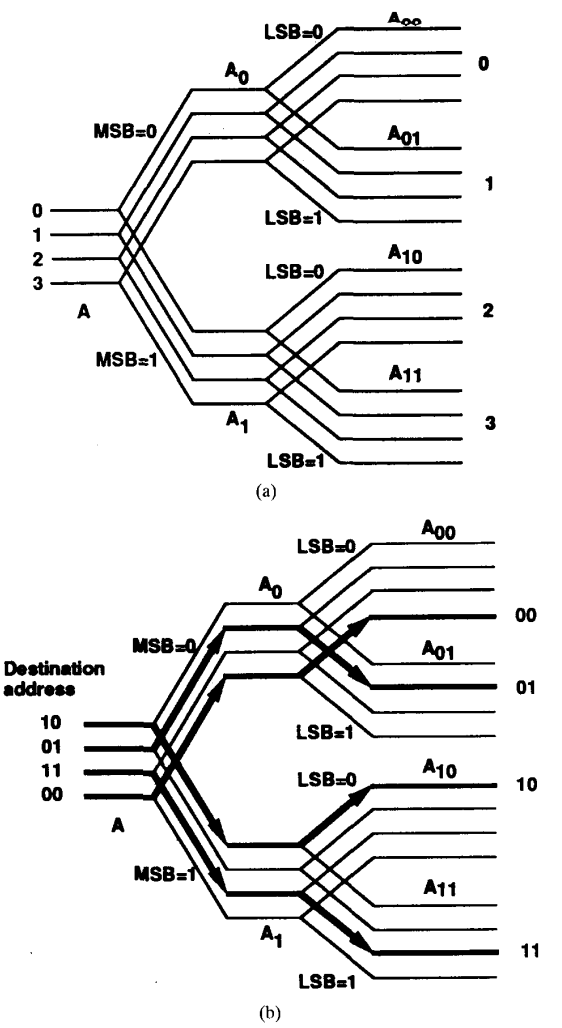
The concentrators in the second stage map the tokens at the live outlets in each index set onto the inlets of the permuter in the third stage with which it is cascaded. Once the tokens reach the inlets of the permuters on the right, they can then be permuted onto their final destinations as desired. Thus, the tokens entering the network at the inlets of the distributor on the left can be mapped onto the outlets of the radix permuters on the right in any one of $n!$ ways. Furthermore, since the distributor and concentrators in the construction are self-routing then the entire network can realize its permutations in a self-routing fashion as well. Hence, the following theorem.

Theorem 1: For all $n = 2^i$, where i is a positive integer, the network given in Fig. 2 is a self-routing permutation network.

To complete our construction of a self-routing permutation network, we next describe how to construct self-routing distributors and concentrators.

IV. A Self-Routing Distributor

As a self-routing distributor, we shall use the simple recursive network construction shown in Fig. 3. When fully decomposed, this network consists of 1×2 demultiplexers as shown in Fig. 4(a) for $n = 4$ and $f = 4$. The tokens are routed to their destinations through the distributor in a self-routing fashion by decoding the leftmost lg f bits of their destination addresses from left to right as illustrated in Fig.



(a)

(b)

Fig. 4. A fully decomposed (4, 16) distributor. (a) Distributor structure. (b) Illustration of distribution.

4(b) for $n = 4$ and $f = 4$. It is immediate that this node by node decoding scheme generalizes to any $n = 2^i, i \geq 1$ and $f = 2^s, s \geq 1$ and so this distributor is self-routing.

Let $C_{DIS}(n, fn)$ denote the number of 1×2 demultiplexers in a fully decomposed (n, fn)-distributor. Then, given that each 1×2 demultiplexer has unit cost, Fig. 3 reveals that

$$C_{DIS}(n, fn) = n + 2C_{DIS}(n, (f/2)n) \quad (1)$$

and the solution of this recurrence with the boundary condition $C_{DIS}(n, 2n) = n$ yields (observe that an $(n, 2n)$-distributor is just a set of n 1×2 demultiplexers)

$$C_{DIS}(n, fn) = n(f - 1). \quad (2)$$

Let $D_{DIS}(n, fn)$ denote the depth of this distributor. Then, given that each 1×2 demultiplexer has unit depth, it is immediate from Fig. 3 that

$$D_{DIS}(n, fn) = 1 + D_{DIS}(n, (f/2)n) \quad (3)$$

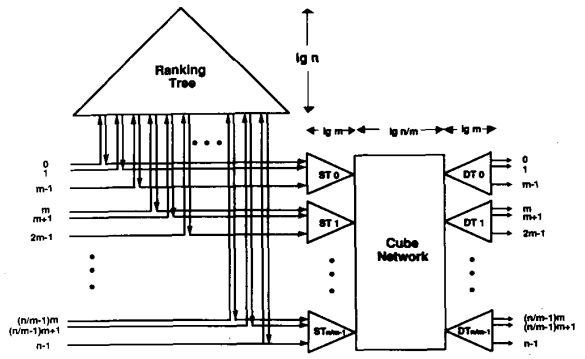


Fig. 5. A self-routing concentrator.

or,

$$D_{DIS}(n, fn) = \lg f = s. \tag{4}$$

Moreover, the distributor can be routed by first deciding the settings of the 1×2 demultiplexers in Fig. 3, and then recursively applying the same routing scheme to each of the two $(n, nf/2)$-distributors on the right. Let $T_{DIS}(n, fn)$ denote the routing time of an (n, fn)-distributor. Since each 1×2 demultiplexer can be set independently in unit time by examining a single bit in the destination address of the token it receives, we have

$$T_{DIS}(n, fn) = D_{DIS}(n, fn) = \lg f = s. \tag{5}$$

V. Construction of a Self-Routing Concentrator

The construction of a self-routing concentrator with a short routing time is trickier than that of a self-routing distributor. Previously reported concentrators have linear cost and logarithmic depth [6], [17], but very little is known about how to route such concentrators, let alone match their routing times with their depths. In what follows we give a concentrator construction that has a linear cost, logarithmic depth and routing time. This is the first known explicit construction of such a concentrator.

This concentrator is obtained by sandwiching an n/m-inlet cube network, where m divides n, in-between two sets of binary trees, each with m leaf nodes, as shown in Fig. 5. Each of the binary trees on the left-hand side of the cube network is called a *selection tree* and each of those on the right is called a *distribution tree*. Another binary tree, called a *ranking tree*, is prestaged to this construction for ranking the live inlets (tokens) as was done in [5] and [10].

The n/m-inlet cube network consists of $\lg(n/m)$ stages of $n/2m$ 2×2 switches. Noting that each 2×2 switch can be formed by cascading two 1×2 demultiplexers with two 2×1 multiplexers, the cost of an n/m-inlet cube network is $(2n/m)\lg(n/m)$, and its depth is $2\lg(n/m)$. Each of the selection and distribution trees has m leaf nodes, where $m \leq n$ is some positive number whose value will be determined later in order to keep the cost of the concentrator linear and its depth logarithmic in its number of inlets. The nodes in the selection trees are 2×1 multiplexers, and those in the distribution trees are 1×2 demultiplexers, each with a single bit control inlet. Given that each 2×1 multiplexer and 1×2 demultiplexer has unit cost and unit depth, it is obvious that each selection (distribution) tree has $m - 1$ cost and $\lg m$ depth.

For n inlets, the ranking tree has $O(n)$ cost and $O(\lg n)$ depth, and is identical to the ranking circuit of Y-units described in [10].

With the assumptions above, let $C_{CON}(n : m)$ and $D_{CON}(n : m)$ denote the cost and depth of this concentrator.[4] Upon summing the costs of the constituent components of the concentrator, we find that

$$C_{CON}(n : m) = n - 1 + 2\frac{n}{m}(m - 1) + \frac{2n}{m}\lg\frac{n}{m} \tag{6}$$

$$\leq 3n + \frac{2n}{m}\lg\frac{n}{m}. \tag{7}$$

Similarly, we find that

$$D_{CON}(n : m) = \lg n + 2\lg m + 2\lg\frac{n}{m} \tag{8}$$

$$= 3\lg n. \tag{9}$$

For the particular case, $m = \lg n$, we obtain a concentrator with

$$C_{CON}(n : m) \leq 5n \tag{10}$$

$$D_{CON}(n : m) = 3\lg n. \tag{11}$$

The cost and depth bounds are satisfactory, but to obtain a logarithmic routing time we must yet prove that this concentrator can be routed in $O(\lg n)$ time, not just have $O(\lg n)$ depth.

To establish this, we first note that the concentration on this network proceed in two "phases": the first phase involves ranking the live inlets and can be done in $O(\lg n)$ time by the ranking circuit as described in [10]. In the second phase, the ranks of the live inlets are used to route the tokens at those inlets to consecutive outlets on the right-hand side starting with the topmost outlet. An example illustrating both phases is depicted in Fig. 6 for $n = 16$, and $m = 4$, where the tokens marked "L" on the left denote the live inlets.

To keep the routing time small, we will describe two parallel routing schemes. The first one allows exactly one token from each of the selection trees to start its trip towards its outlet with the provision that tokens from different selection trees may start in parallel. This ensures that no conflicts arise within a selection tree, but there can be conflicts once the tokens reach the inlets of the cube network since two or more tokens may be destined to outlets in the same distribution tree. In general, conflicts will occur any time two tokens arriving at a switch in the cube network must exit from the same outlet of that switch. The goal is to route the tokens through the cube network as fast as possible despite such conflicts.

In the second routing scheme, we introduce further parallelism by pipelining the tokens through the selection trees. In this case, conflicts may also occur at the nodes of the selection trees. We resolve such conflicts by letting the token at the

[4] This notation should not be confused with $C_{CON}(n, q)$, or $D_{CON}(n, q)$, where q denotes the number of outlets. The argument m in these expressions specifies a design parameter, not the number of outlets.

Time \ Nodes	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
0	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
1	0	1	3	5	7	8	-	-	-	-	-	-	-	-	-	-
2	-	-	-	5	-	8	0	1	3	7	-	-	-	-	-	-
3	-	2	4	5	-	8	0	-	-	-	1	3	7	-	-	-
4	-	-	-	5	-	-	0	2	4	8	1	-	-	3	-	7
5	-	-	-	5	-	9	0	2	-	-	1	8	4	-	-	-
6	-	-	-	-	-	-	0	2	5	9	-	-	-	1	8	4
7	-	-	-	6	-	-	0	-	-	-	2	9	5	-	-	-
8	-	-	-	-	-	-	0	-	6	-	-	-	-	2	9	5
9	-	-	-	-	-	-	-	-	-	-	0	-	6	-	-	-
10	-	-	-	-	-	-	-	-	-	-	-	-	-	0	-	6
11	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

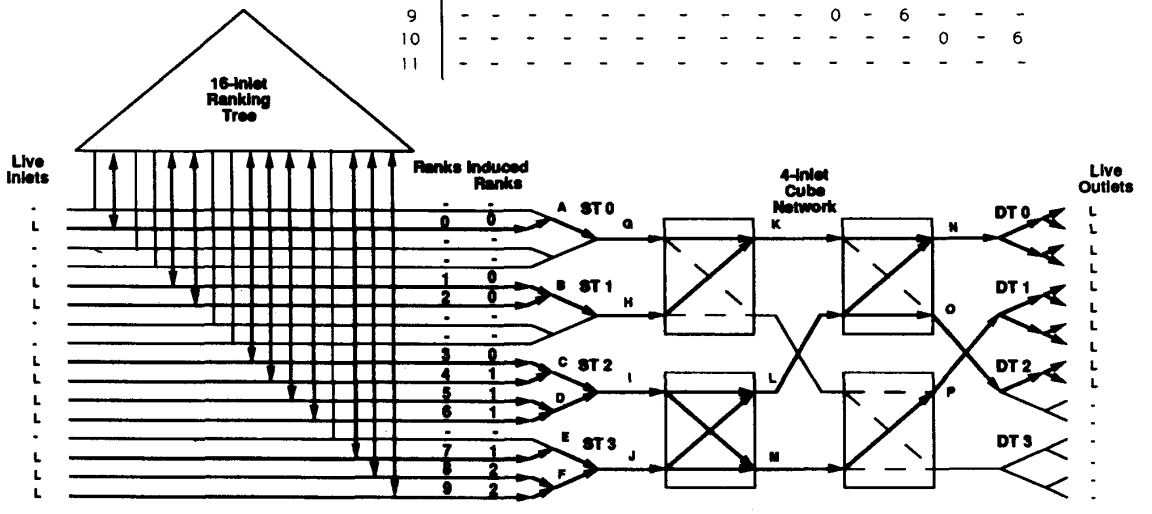


Fig. 6. Illustration of routing on the concentrator.

upper child node go first. As for the conflicts at the switches in the cube network, if two tokens arriving at a switch in the cube network want to exit at the same outlet then the lower token proceeds first. These two conventions will be referred to as the *upper child priority* and *lower inlet priority* routing schemes. In both cases, once a token reaches the root of a distribution tree it then uses the rightmost lg m bits of its rank to route itself to the right outlet. The decoding of the rank bits is done on a node by node basis as the token proceeds towards its outlet through the tree. If the current bit is 0 then the token moves over to the upper child node, if it is 1 it moves over to the lower child node. It should be obvious that tokens cannot run into any conflicts as they are routed through the distribution trees. It should also be obvious that with these conventions, tokens are self-routed through all components of the concentrator.

We call the above two parallel schemes *semi-pipelined* and *fully-pipelines* routing schemes. We shall prove that in the semi-pipelined routing scheme, the conflicts can be resolved and all tokens can be routed in $O(\lg^2 n)$ time, and in the fully pipelined routing scheme, the same can be done in $O(\lg n)$ time. We first give an example to illustrate our conventions of resolving conflicts. The table on the righthand top corner of Fig. 6 indicates the flow of tokens through the selection trees and cube network in the fully pipelined case. (In the semi-pipelined scheme, the conflicts are resolved exactly the same way, except the timing is different.) The letters denote the various points in the overall network and the entries in the table indicate the locations of the tokens at a given time. For example, the token 0 is at point A at time 1, and at point G at time 2, at point K at time 9, etc.

A. Semi-Pipelined Routing

To prove that this procedure works and takes $O(\lg^2 n)$ time in the semi-pipelined case we need to establish some facts about rank patterns and the routing structure of the cube network. At the outset, we should emphasize that the ranks computed by the ranking tree determine the outlets of the concentrator to which the tokens at its live inlets are routed. As such, we will, occasionally, refer to these ranks as destination tags within the concentrator.

Definition 8: A *rank pattern* over the inlets of the selection trees in Fig. 5 is an ordered assignment of consecutive integer values $0, 1, 2, \cdots$ to the live inlets such that the topmost live inlet is assigned rank 0, the next topmost live inlet is assigned 1, etc. The ranks of the remaining inlets are marked with dashes (see Fig. 6). ||

Definition 9: An *induced rank pattern* of degree $\lg(n/m)$ is a rank pattern whose entries are obtained by retaining the leftmost $\lg(n/m)$ bits in the corresponding ranks. For example,

for the rank pattern shown in Fig. 6, the induced rank pattern of degree 2 is –0––00––0111–122. ||

Definition 10: A *sample induced rank pattern* of degree $\lg(n/m)$ is an ordered set of induced ranks whose ith element is selected from the group of induced ranks that belong to the ith selection tree. If each rank selected is associated with a live inlet then the sample induced rank pattern is called complete, and is otherwise called incomplete. For example, for the induced rank pattern –0––00––0111–122, 0001, 0012 and 0002 are all complete sample induced rank patterns, and –012 is an incomplete sample induced rank pattern. ||

As stated before, the ranks computed by the ranking tree specify the destination tags of the tokens. The induced rank patterns, retaining only $\lg(n/m)$ leftmost bits of these tags identify the actual distribution trees to which the tokens belong. Our goal is to route the tokens located at the live inlets of the selection trees to their destinations at the outlets of the distribution trees. To do this, we shall decompose the induced rank patterns into sample induced rank patterns in which the adjacent induced ranks differ by at most one. Each such sample induced rank pattern identifies a unique set of up to n/m live inlets, one from each selection tree. The tokens identified by the induced ranks in these sample induced ranks patterns are then routed set after set through the cube network onto the root nodes of the distribution trees. Once the tokens reach the root nodes of the distribution trees, they then get routed to their exact outlets by using the rightmost $\lg m$ bits in their destination tags.

We shall subsequently establish that any induced rank pattern can be decomposed into a set of sample rank patterns in which the adjacent induced ranks differ by at most one. First we show that any set of tokens identified by such a sample induced rank pattern can be self-routed through the cube network portion of the concentrator in a pipelined fashion in $O(m + \lg(n/m))$ time.

Theorem 2: The tokens associated with any sample induced rank pattern of degree $\lg(n/m)$ in which the induced ranks at adjacent positions differ by at most one can be self-routed through an n/m-inlet cube network in $O(m + \lg(n/m))$ time.

Proof: Let R_u and R_l be the induced ranks of two tokens that enter an arbitrary but fixed switch in stage i, $0 \leq i \leq (\lg(n/m)) - 1$, of the cube network at its upper and lower inlets, respectively. Let $r_{u,i}$ and $r_{l,i}$ denote the ith bits of R_u and R_l, respectively. Given that some sampled induced ranks may be incomplete, there are four cases to consider. If both R_u and R_l are "—"'s then neither inlet of the switch has a token, and so it may arbitrarily be set to either the identity or transpose state. If only R_u is a "—" then only the lower inlet has a token, and so the switch is set by $r_{l,i}$ and likewise if only R_l is a "—" then it is set by $r_{u,i}$. The last case is that neither R_u and R_l is a "—". In this case, if $r_{u,i} = 0$, and $r_{l,i} = 1$ then the switch is set to the identity, and if $r_{u,i} = 1$, and $r_{l,i} = 0$ then the switch is set to the transpose state. We further show that the other two cases, i.e., $r_{u,i} = r_{l,i} = 0$, or $r_{u,i} = r_{l,i} = 1$, are impossible except when $R_u = R_l$. First, for $i = 0$, by the hypothesis, the adjacent induced ranks differ by at most one, and hence, they must either be identical, or they must differ in their rightmost bit positions, i.e., $r_{u,i} \neq r_{l,i}$. Suppose, for some $i; 1 \leq i \leq (\lg(n/m)) - 1$, some two tokens with induced ranks R_u and R_l enter a switch in stage i. As argued in [10], this implies that the rightmost i bits of R_u and R_l, i.e., bits $0, 1, \cdots, i-1$, must be identical. Furthermore, the two tokens must have originated from some two inlets that belong to a set of 2^{i+1} consecutive inlets. Given that any two adjacent induced ranks may differ by at most one, this implies that $|R_u - R_l| \leq 2^{i+1} - 1$, and so R_u and R_l cannot differ in bit positions $(\lg(n/m)) - 1, (\lg(n/m)) - 2, \cdots, i+1$ either. Therefore, either $R_u = R_l$, or they must differ in bit position i as was to be shown. Thus, the only way that two tokens can conflict at a switch is when both have identical induced ranks. Consequently, a token with an induced rank in a sample induced rank pattern in which the adjacent induced ranks differ by at most one can be in conflict with no more than the number of tokens with that same induced rank. But this is bounded by the number of outlets of a distribution tree, which is m. Since the depth of the (n/m)-cube network is $\lg(n/m)$, and the routing time of each switch is constant, we conclude that tokens associated with any such sample induced rank pattern can be self-routed through the cube network in $O(m + \lg(n/m))$ steps. □

Given this fact, the only task that remains is to establish that any induced rank pattern of degree $\lg(n/m)$ can be decomposed into a set of sample induced rank patterns such that the adjacent entries in each sample induced rank pattern differ by at most one. The existence of such a decomposition is facilitated by the following results.

Proposition 1: The induced ranks associated with live inlets in any sample induced rank pattern are in nondecreasing order.

Proof: It is obvious. □

Proposition 2: In any distribution of tokens to the inlets of the selection trees in Fig. 5, every set of m tokens with the same induced rank, except that set of tokens with the largest induced rank, has cardinality m.

Proof: It follows from Proposition 1. □

Proposition 3: Any two induced ranks that belong to any $k, 1 \leq k \leq n/m$, consecutive selection trees, each with m inlets, can differ by at most k.

Proof: Suppose that in some k consecutive selection trees there are two induced ranks that differ by $k+1$ or more. Then these k selection trees must have more than mk tokens which is impossible. □

Theorem 3: Given any induced rank pattern, the induced ranks associated with the topmost tokens in any two adjacent selection trees differ by at most one.

Proof: By Proposition 3, any two induced ranks in any two adjacent selection trees ST_i and ST_{i+1} can at most differ by two. If they differ by zero or one then there is nothing to prove. On the other hand, if they differ by two then it is easy to see that the two selection trees must contain three consecutive induced ranks R_x, $R_x + 1$, $R_x + 2$. However, by Proposition 1, the induced rank of the topmost token of ST_i must be R_x. Also, given that $R_x + 1$ is not the largest induced rank, by Proposition 2, there must be m tokens with induced rank $R_x + 1$ distributed over ST_i and ST_{i+1}. Furthermore, since induced ranks are assigned to tokens in nondecreasing

order, the induced rank of the topmost token in ST_{i+1} must be $R_x + 1$, and the statement follows. □

Proposition 4: Given two adjacent selection trees ST_i and ST_{i+1} suppose that ST_i has tokens with induced ranks R_x and $R_x + 1$, and ST_{i+1} has tokens with induced ranks $R_x + 1$ and $R_x + 2$. Let $N_{x,i}$ be the number of tokens in ST_i with induced rank R_x, and let $N_{x+1,i}$ and $N_{x+1,i+1}$ be the numbers of tokens with induced rank R_x+1 in ST_i and ST_{i+1}, respectively. Then $N_{x,i} \leq N_{x+1,i+1}$.

Proof: Since $R_x + 1$ is the middle induced rank, by Proposition 2, $N_{x+1,i} + N_{x+1,i+1} = m$. Also, $N_{x+1,i} + N_{x,i} \leq m$, and the statement follows. □

Theorem 4: Given any induced rank pattern of degree $\lg(n/m)$. If we remove the induced rank associated with the topmost token from each of the selection trees, then the remaining induced ranks form an induced rank pattern in which the induced ranks of the topmost tokens in any two adjacent selection trees differ by at most one.

Proof: It follows from the previous two results. □

These results establish that any induced rank pattern of degree $\lg(n/m)$ can be decomposed into a sequence of sample induced rank patterns in which two adjacent induced ranks differ by at most one. By Theorem 2, each of these sample induced rank patterns can be self-routed through the cube network in Fig. 5 in $O(m + \lg(n/m))$ steps. Moreover, the tokens associated with the induced ranks in each sample induced rank pattern can be routed to the roots of their respective selection trees in $\lg\ m$ steps by using the upper child priority routing scheme. (It is easy to see that this scheme ensures that the topmost (smallest) token reaches the root.) It will take additional $\lg\ m$ steps to route the group of tokens associated with each sample pattern through the distribution trees to their outlets. This can also be done on a self-routing basis, this time using the rightmost $\lg m$ bits of the ranks in the sample induced rank pattern. Finally, since each selection tree has at most m tokens, the number of sample induced rank patterns in a decomposition of any induced rank pattern is at most m. Thus, assuming that the routing is done in a semi-pipelined fashion (i.e., only the routing of tokens in the cube network is pipelined) and combining and ranking and concentration steps described we have established the following theorem.

Theorem 5: Under the semi-pipelined routing scheme, the network in Fig. 5 can concentrate the tokens with any rank pattern on a self-routing basis in $O(\lg n + m(\lg m + \lg(n/m) + m))$ time. □

In particular, we have the following corollary.

Corollary 1: When $m = \lg n$, the network given in Fig. 5 has $O(n)$ switches and it can concentrate the tokens with any rank pattern on a self-routing basis in $O(\lg^2 n)$ time. □

B. Fully Pipelined Routing

The routing time can be reduced to $O(\lg n)$ by using the fully pipelined routing scheme defined earlier. In this case, conflicts may occur among tokens with nonidentical ranks in the selection trees as well as tokens with identical induced ranks at the switches in the cube network. We resolve the conflicts in the selection trees by using the upper child priority scheme, and those in the cube network by using the lower inlet priority routing scheme. In order to show that this results in $O(\lg n)$ routing time, it suffices to prove that any conflicts in the cube network are limited to those tokens with identical induced ranks. This is because, if we have only such conflicts then a token need not wait any more than m time units to reach its destination (recall that there are at most m tokens with the same induced rank), not counting the depth of the path it travels on. Since the depth from the inlets of the selection trees to the outlets of the distribution trees is $2\lg m + 2\lg(n/m)$, and the routing proceeds in a fully pipelined fashion, the last token within any group of tokens with the same induced rank should reach its destination in $2\lg m + 2\lg(n/m) + m$ time. Letting $m = \lg n$ then yields the desired $O(\lg n)$ time.

What remains to be shown is that no two tokens with distinct induced ranks can be in conflict at the switches in the cube network. The following results establish this claim.

Proposition 5: If any two inlets I_x and I_y of the cube network meet over any two paths through the network at a switch in stage $i, 0 \leq i \leq (\lg(n/m)) - 1$, then $|I_x - I_y| < 2^{i+1}$.

Proof: By the structure of the cube network, any switch in stage i can be reached by no more than 2^{i+1} consecutive inlets. Therefore, for any two inlets I_x, I_y that are in this set of consecutive inlets, it must be that $|I_x - I_y| < 2^{i+1}$. □

Proposition 6: Given any two tokens with distinct induced ranks R_x and R_y and located at two inlets I_x and I_y of the cube network, respectively, if $|R_x - R_y| \leq |I_x - I_y|$ then the two tokens can never be in conflict at any switch through the network.

Proof: Suppose that a token with induced rank R_x is in conflict with a token with induced rank R_y at a switch in stage i. Since both tokens are routed to the same switch, the rightmost i bits of R_x and R_y, i.e., bits $0, 1, \cdots, i-1$, must be identical. Now, by the hypothesis, $|R_x - R_y| \leq |I_x - I_y|$ and so by Proposition 5, $|R_x - R_y| < 2^{i+1}$. This means that R_x and R_y cannot differ in any of the bit positions $i+1, i+2, \cdots, \lg(n/m) - 1$. Given these and the fact that $R_x \neq R_y$, R_x, and R_y must differ in bit position i, and since the switches in stage i are set by bit i the statement follows. □

Theorem 6: If all the tokens in a selection tree have the same induced rank R_x then none of those tokens can be in conflict with a token whose induced rank is other than R_x as they are routed through the cube network.

Proof: Let a token T_x with induced rank R_x be located at inlet I_x of the cube network (i.e., T_x belongs to selection tree ST_{I_x}), and consider another token T_y with induced rank R_y that is located at inlet I_y. Since all the tokens that belong to the selection tree ST_{I_x} have the same induced rank, the number of live tokens that belong to the selection trees in-between ST_{I_x} and ST_{I_y}, (excluding ST_{I_x} and ST_{I_y}), must be less than or equal to $(|R_x - R_y| - 1)m$. Since the number of tokens (i.e., live inlets) that belong to these selection trees must be less than or equal to the total number of inlets they have, we must have $(|I_y - I_x|) - 1)m \geq (|R_y - R_x| - 1)m$, or $|I_y - I_x| \geq |R_y - R_x|$. Hence, the statement follows from Proposition 6. □

This theorem proves that if the induced ranks of all tokens in a selection tree are the same then these tokens cannot be in conflict with any tokens as they are routed through the cube network except with those whose induced ranks are the same as their own. We next consider the case when tokens in a selection tree may have different induced ranks.

Lemma 1: Suppose that, in routing the tokens through the cube network, if any time two tokens entering a switch conflict, the token at the lower inlet goes first. Then if any two tokens T_x and T_y initially located at any two inlets I_x and I_y, where $I_x < I_y$, conflict at a switch then token T_y always exits that switch first.

Proof: It is immediate from the structure of the cube network that every pair of its tokens meet at most once at some predetermined switch. Let T_x and T_y be two tokens that enter a switch at its upper and lower inlets, respectively. Because of the recurring structure of the cube network, T_x and T_y must have originated from some two inlets I_x and I_y where $I_x < I_y$. Combining this fact with the hypothesis proves the statement. □

Lemma 2: Given two selection trees ST_i and $ST_{i+k-1}, 2 \leq k \leq (\lg n/m) - i, 0 \leq i \leq (\lg n/m) - 2$, suppose that ST_i has tokens with induced ranks R_x and $R_x + 1$ and ST_{i+k-1} has tokens with induced ranks $R_x + k - 1$ and $R_x + k$. Let N_x be the number of tokens with induced rank R_x in selection tree ST_i, and N_{x+k-1} be the number of tokens with induced rank $R_x + k - 1$ in selection tree ST_{i+k-1}. Then $N_x \leq N_{x+k-1}$.

Proof: This is an immediate generalization of Proposition 4. □

Theorem 7: Suppose that all tokens with an induced rank R_x are distributed over a set of selection trees such that the topmost selection tree in the set also contains tokens with induced rank $R_x - 1$ and the bottommost selection tree in the set also contains tokens with $R_x + 1$ in addition to R_x. If the conflicts between the tokens in the selection trees are resolved using the upper child priority routing scheme, and the conflicts between the tokens with the same induced rank at the switches of the cube network are resolved using the lower inlet priority routing scheme than all those tokens with induced rank R_x that do belong to the topmost or bottommost selection tree will never be in conflict with any tokens with induced ranks other than R_x as they are routed through the cube network.

Proof: We consider two cases. First consider any token T_x with induced rank R_x in the bottommost selection tree. Let T_y be any token with some arbitrary but fixed induced rank $R_y \neq R_x$. Suppose that T_x and T_y are located at inlets I_x and I_y, respectively. If $R_y < R_x$ then it it can be shown as in the proof of Theorem 6 that $R_x - R_y \leq |I_x - I_y|$, and hence, by Proposition 6, T_x cannot be in conflict with T_y at any switch in the cube network. Now suppose $R_y > R_x$, and assume that $I_y - I_x = k - 1$, for some k; $2 \leq k \leq (\lg(n/m)) - 1$. If $R_y - R_x \leq k - 1$ (recall that by Proposition 3, $R_y - R_x \leq k$), then, by Proposition 6, T_x cannot be in conflict with T_y. So, suppose that $R_y - R_x = k$. Then it is easy to see that R_x is the smaller induced rank in ST_{I_x} and $R_y = R_x + k$ is the larger induced rank in ST_{I_y}. Therefore, by Lemma 2, the number of tokens with induced rank R_x in ST_{i_x} is less than or equal to the number of tokens with induced rank $R_y - 1 = R_x + k - 1$ in $ST_{I_y}, k = 2, 3, \cdots, \lg(n/m) - 1$. Now, since the conflicts at the switches in the cube network are resolved by the lower inlet priority routing scheme and ST_{I_x} is the bottommost selection tree containing tokens with rank R_x, Lemma 1 ensures that T_x never has to wait for any token with induced rank R_x and that belongs to any other selection trees. In addition, involving the upper child priority routing scheme for the tokens in the selection trees ensures that all tokens with induced rank R_y should enter the cube network after all the tokens with induced rank $R_y - 1$ in selection tree ST_{I_y}. Thus, given that the number of tokens with induced rank R_x in ST_{I_x} is less than or equal to the number of tokens with induced rank $R_y - 1$ in ST_{I_y}, and those tokens in ST_{I_x} with induced rank R_x can never meet with a token with induced rank R_y at a switch in the cube network, and therefore, they cannot be in conflict with any such token as was to be proved. The second case, i.e., when the tokens with induced rank R_x belong to the topmost selection tree among all those containing such tokens, is argued similarly and hence the statement follows. □

Combining Theorems 6 and 7, we have the following theorem.

Theorem 8: If, in routing tokens through the network in Fig. 5, the upper child priority routing scheme is used to resolve conflicts in the selection trees and the lower inlet priority routing scheme is used to resolve conflicts in the cube network then any two tokens with distinct induced ranks can never be in conflict at any switch in the cube network. □

Remark 3: The main point of Theorem 8 is that the conflicts among tokens are limited to those of a given induced rank. Therefore, if the tokens are routed using the fully pipelined scheme, the last token should exit the cube network in no more than $m + \lg(n/m)$ steps once it reaches an inlet of the cube network. Combining this with the ranking, selection and distribution steps,it takes no longer than $2\lg n + 2\lg m + 2\lg(n/m) + m$ time to route all live inlets to their final destinations at the outlets of the distribution trees. □

We have thus proved our main result.

Theorem 9: Using the fully pipelined routing scheme, the network in Fig. 5 can concentrate any pattern of live inlets on a self-routing basis in $2\lg n + 2\lg m + 2\lg(n/m) + m$ time.□

In particular, combining this result with (10) and (11) we have the following corollary.

Corollary 2: When $m = \lg n$, the network given in Fig. 5 has $\leq 5n$ cost, $3\lg n$ depth, and it can concentrate any pattern of live inlets in $\leq 5\lg n$ time. □

Remark 4: Since this network is a (n, n)-concentrator, in lieu of Remark 2, we have constructed an $(n, \gamma n)$-concentrator with $\leq 5n$ cost, $3\lg n$ depth, and $\leq 4\lg n$ routing time for any $\gamma; 0 < \gamma \leq 1$. □

VI. Performance Analysis

Given the preceding results in this section, we determine the cost, depth and routing time of the radix permuter described in Section III. Earlier we established that

$$C_{DIS}(n, fn) = n(f - 1) \tag{12}$$

$$D_{DIS}(n, fn) = \lg f = s \tag{13}$$

$$T_{DIS}(n, fn) = \lg f = s \tag{14}$$

and[5]

$$C_{CON}(n, \gamma n) \leq 5n \tag{15}$$

$$D_{CON}(n, \gamma n) \leq 3 \lg n \tag{16}$$

$$T_{CON}(n, \gamma n) \leq 5 \lg n \tag{17}$$

for any $f; 2 \leq f \leq n$, and $\gamma; 0 < \gamma \leq 1$.

Let $C_{RP}(n), D_{RP}(n)$, and $T_{RP}(n)$ denote the cost, depth and routing time of the radix permuter shown in Fig. 2 in that order. Using these facts, we establish the following theorem.

Theorem 10:

$$C_{RP}(n) = O(kn^{1+1/k}) \tag{18}$$

$$D_{RP}(n) = O(k \lg n) \tag{19}$$

$$T_{RP}(n) = O(k \lg n) \tag{20}$$

for each $k; 1 \leq k \leq \lg n$.

Proof: From the construction of $RP(n)$ (refer to Fig. 2), for any $f = 2^s, 1 \leq s \leq \lg n$,

$$C_{RP}(n) = C_{DIS}(n, fn) + fC_{CON}(n, n/f) + fC_{RP}\left(\frac{n}{f}\right).$$

Upon replacing $C_{DIS}(n, fn)$ with the right-hand side of the first expression in (12), and noting, from (15), that $C_{CON}(n, n/f) \leq 5n$, we have

$$\begin{aligned} C_{RP}(n) &\leq (f-1)n + 5fn + fC_{RP}\left(\frac{n}{f}\right) \\ &\leq 6fn + fC_{RP}\left(\frac{n}{f}\right). \end{aligned}$$

It is easy to show that the solution of this recurrence after k iterations, and under the boundary condition $n/f^k = 1$ and $C_{RP}(1) = 1$ yields

$$C_{RP}(n) \leq 6nfk + f^k = 6kn^{1+1/k} + n, \tag{21}$$

for any $k; 1 \leq k \leq \lg n$. Likewise, from the construction of $RP(n)$

$$D_{RP}(n) = D_{DIS}(n, fn) + D_{CON}(n, n/f) + D_{RP}\left(\frac{n}{f}\right).$$

Upon substituting $D_{DIS}(n, fn)$ and $D_{CON}(n, n/f)$ from (13) and (16) we have

$$D_{RP}(n) \leq s + 3 \lg n + D_{RR}\left(\frac{n}{f}\right).$$

It is easy to show that the solution of this recurrence after k iterations, and with the boundary condition $n/f^k = 1$ (i.e., $\lg n = ks$) and $D_{RP}(1) = 1$ yields

$$\begin{aligned} D_{RP}(n) &\leq ks + 3 \lg n + 3 \lg \frac{n}{f} + \cdots + 3 \lg \frac{n}{f^{k-2}} + 1 \\ &\leq \lg n + 3k \lg n + 1 = (3k+1) \lg n + 1 \\ &\leq 4k \lg n + 1. \end{aligned} \tag{22}$$

Since the routing times of the distributor and concentrator components of the network are of the same order as their depths, (20) also follows. □

In particular, we have the next corollary.

[5] $T_{CON}(n, \gamma n)$ denote the routing time of an $(n, \gamma n)$-concentrator.

Corollary 3: If $f = 2$ (i.e., $k = \lg n$) then upon recursively decomposing the radix permuter in Fig. 2 $\lg n$ times we obtain a self-routing permutation network with $O(n \lg n)$ cost and $O(\lg^2 n)$ depth, and $O(\lg^2 n)$ routing time.

VII. Concluding Remarks

The paper presented a permutation network with $O(n \lg n)$ cost, $O(\lg^2 n)$ depth, and $O(\lg^2 n)$ routing time. More generally, the paper has obtained a permutation network with $O(kn^{1+1/k})$ cost, and $O(k \lg n)$ routing time for any $k, 1 \leq k \leq \lg n$. This improves the routing algorithm in [13] that gives $O(k \lg^3 n)$ routing time for the same cost expression.

The key component of this network is a linear cost self-routing concentrator that is obtained by scaling down an n-inlet cube network to an $n/\lg n$-inlet cube network, and sandwiching it in between two sets of $O(\lg n)$-node binary trees. While linear cost concentrators have been known for some time [6], [17], this is the first such concentrator with $O(\lg n)$ routing time. We must also contrast this concentrator with those given in [5] and [10]. If the prefix sum is carried out in a bit-serial fashion as described in [5], then the concentrators in [5] and [10] give $O(n \lg n)$ cost and $O(\lg n)$ routing time, both at bit-level. Using a bit-serial prefix sum to compute the ranks in our construction gives a concentrator with $O(n)$ bit-level cost and $O(\lg^2 n)$ bit-level routing time.

Using these bit-level complexities, our radix permuter has $O(n \lg n)$ bit-level cost and $O(\lg^3 n)$ bit-level routing time whereas the permutation network given in [10] has $O(n \lg^2 n)$ bit-level cost and $O(\lg^2 n)$ bit-level routing time by taking into account Cormen's bit-serial prefix ranking circuit. It should also be pointed out that, when operated in a bit-serial fashion, Batcher sorters also give $O(\lg^2 n)$ bit-level routing time at $O(n \lg^2 n)$ bit-level cost.

As for future work, it still remains open whether or not a permutation network can be constructed with $O(n \lg n)$ cost and $O(\lg n)$ routing time where the constants hidden in the O notations are reasonably small, e.g., ≤ 10. The recent network constructions given in [2] and [12] seem to provide progress in this direction, but the fact that they must randomly generate the expanders in their networks to secure small constants makes these constructions impractical. Another direction that is worth pursuing is to determine whether the network given here can route as efficiently in the presence of faults. This problem was considered in [18] for fixed topology networks, and more recently in [2] for nonblocking networks. It will be worthwhile to develop fault tolerant routing schemes for the permutation network of this paper as well.

Appendix

(n, q)-network	A network with n inlets and q outlets.
$C_{DIS}(n, fn)$	Cost of an (n, fn)-distributor.
$D_{DIS}(n, fn)$	Depth of an (n, fn)-distributor.
$T_{DIS}(n, fn)$	Routing time of an (n, fn)-distributor.
$C_{CON}(n : m)$	Cost of an n-inlet concentrator with parameter m.

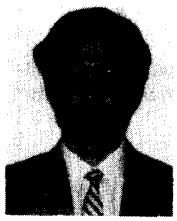
$D_{CON}(n:m)$	Depth of an n-inlet concentrator with parameter m.
$T_{CON}(n:m)$	Routing time of an n-inlet concentrator with parameter m.
$C_{CON}(n, \gamma n)$	Cost of an $(n, \gamma n)$-concentrator.
$D_{CON}(n, \gamma n)$	Depth of an $(n, \gamma n)$-concentrator.
$T_{CON}(n, \gamma n)$	Routing time of an $(n, \gamma n)$-concentrator.
$C_{RP}(n)$	Cost of an n-inlet radix permuter.
$D_{RP}(n)$	Depth of an n-inlet radix permuter.
$T_{RP}(n)$	Routing time of an n-inlet radix permuter.

ACKNOWLEDGMENT

The authors wish to thank the anonymous referees for their constructive suggestions, and D. Koppelman of Louisiana State University for his comments which improved the presentation of the paper.

REFERENCES

[1] M. Ajtai, J. Komlos, and E. Szemeredi, "An $O(n \lg n)$ sorting network," in *Proc. 15th ACM Symp. Theory Comput.*, 1983, pp. 1–9.
[2] A. Arora, T. Leighton, and B. Maggs, "On-line algorithms for path selection in a nonblocking network," in *Proc. 22nd ACM Symp. Theory Comput.*, May 1990, pp. 149–158.
[3] K. E. Batcher, "Sorting networks and their applications," in *Proc. 1968 Spring Joint Comput. Conf.*, pp. 307–314.
[4] V. E. Beneš, *Mathematical Theory of Connecting Networks and Telephone Traffic.* New York: Academic, 1965.
[5] T. H. Cormen, "Concentrator switches for routing messages in parallel computers," M.S. thesis, Massachusetts Inst. Technol., Cambridge, 1986.
[6] O. Gabber and Z. Galil, "Explicit constructions of linear-sized superconcentrators," *J. Comput. Syst. Sci.*, vol. 22, pp. 407–420, 1981.
[7] L. J. Goldstein and S. W. Leibholz, "On the synthesis of signal switching networks with transient blocking," *IEEE Trans. Comput.*, vol. EC-16, pp. 637–641, Oct. 1967.
[8] A. E. Joel Jr., "On permutation switching networks," *Bell Syst. Tech. J.*, vol. 47, no. 5, pp. 813–822, May–June 1968.
[9] D. E. Knuth, *The Art of Computer Programming*, vol. 3. Reading, MA: Addison-Wesley, 1973.
[10] D. Koppelman and A. Y. Oruç, "Time-space tradeoffs in parallel communications," in *Proc. Allerton Conf.*, Urbana, IL, 1989.
[11] D. Koppelman and A. Yavuz Oruç, "A self-routing permutation network," *J. Parallel Distributed Comput.*, no. 10, pp. 140–151, 1990.
[12] T. Leighton and B. Maggs, "Expanders might be practical: Fast algorithms for routing around faults in a multibutterfly," in *Proc. 30th IEEE Symp. Found. Comput. Sci.*, Nov. 1989, pp. 384–389.
[13] D. Nassimi and S. Sahni, "Parallel algorithms to set up the Benes permutation network," *IEEE Trans. Comput.*, vol. C-31, no. 2, pp. 148–154, Feb. 1982.
[14] A. Yavuz Oruç and M. Mittal, "Setup algorithms for cube-connected parallel computers using recursive karnaugh maps," *IEEE Trans. Comput.*, vol. 40, pp. 217–221, Feb. 1991.
[15] D. C. Opferman and N. T. Tsao-Wu, "On a class of rearrangeable switching networks," *Bell System Tech. J.*, pp. 1579–1600, May–June 1971.
[16] M. C. Pease, "The indirect binary n-cube microprocessor array," *IEEE Trans. Comput.*, vol. C-26, pp. 458–473, May 1976.
[17] M. S. Pinsker, "On the complexity of a concentrator," in *Proc. 7th Int. Teletraffic Congress*, Stockholm, 1973, pp. 318/1–318/4.
[18] D. K. Pradhan and S. M. Reddy, "A fault-tolerant distributed processor architecture," *IEEE Trans. Comput.*, vol. C-31, no. 9, pp. 863–870, Sept. 1982.
[19] C. Shannon, "Memory requirements in a telephone exchange," *Bell System Tech. J.*, vol. 29, pp. 343–349, 1950.
[20] H. J. Siegel and S. D. Smith, "Study of multistage SIMD interconnection networks, in *Proc. 5th Ann. Symp. Comp. Arch.*, Apr. 1978, pp. 223–229.
[21] H. S. Stone, "Parallel processing with the perfect shuffle," *IEEE Trans. Comput.*, vol. C-20, pp. 153–161, Feb. 1971.
[22] A. Waksman, "A permutation network," *J. ACM*, vol. 15, no. 1, pp. 159–163, Jan. 1968.
[23] E. Upfal, "An $O(\log N)$ deterministic packet routing scheme," in *Proc. 21st ACM Symp. Theory Comput.*, May 1989, pp. 241–250.



Ching Yuh Jan received the B.Sc. degree in electrical engineering from the National Taiwan University, Taipe, Taiwan, in June 1982, and the M.Sc and Ph.D. degrees in electrical engineering from the University of Maryland, College Park, in May 1988 and August 1992, respectively.

He was a Visiting Assistant Professor in the Department of Electrical and Computer Engineering at the California State University, Fresno, CA, between August 1991 and August 1992. He also served as a Teaching Assistant in the Department of Computer Science at the University of Maryland from January 1986 to May 1991. His research interests include parallel computer architectures, interconnection networks, and VLSI system design.

A. Yavuz Oruç (S'81–M'83–SM'92) received the B.Sc. degree in electrical engineering from the Middle East Technical University, Ankara, Turkey, in 1976, the M.Sc. degree in electronics from the University of Wales, Cardiff, United Kingdom, in 1978, and the Ph.D degree in electrical engineering from Syracuse University, Syracuse, NY, in 1983.

Since January 1988, he has been an Associate Professor in the Department of Electrical Engineering at the University of Maryland, College Park, Maryland. Prior to joining the University of Maryland, he was on the faculty of the Department of Electrical, Computer and Systems Engineering at Rensselaer Polytechnic Institute, Troy, NY. His research interests include parallel computer and communication systems, and interconnection networks.

Dr. Oruç is a Member of IEEE Computer and Information Theory Societies.