High Performance Concentrators and Superconcentrators Using Multiplexing Schemes

Minze V. Chien, *Member, IEEE,* and A. Yavuz Oruç, *Senior Member, IEEE*

Abstract— **Concentrators are used to interface and combine together low speed communication channels onto higher speed transmission links to alleviate transmission costs. They are also used to construct more powerful switching fabrics such as permutation and broadcast networks. Using an adaptive binary sorting network model, this paper constructs new concentrators and superconcentrators. Unlike the previously reported concentrators and superconcentrators, these new constructions are fast, and can easily be implemented using simple switching devices. More specifically, for n inputs, they can be constructed with $O(n \lg \lg n)$ constant fanin bit-level multiplexers and demultiplexers, and can be routed in $O(\lg^2 n)$ bit-level time.**

I. Introduction

In communication networks, signals are transmitted over long-haul channels such as coaxial cables, optical trunks, or remote microwave carriers when the distance between callers and receivers exceed a few tens of miles. To reduce channel costs, and achieve higher efficiency and utilization, while containing congestion, the callers within each local area are often allocated a set of channels, each of which can be used by any of the callers to reach any of the receivers outside that area. As the number of callers within a given area is, in general, much larger than the number of allocated channels, one must use a switching network to multiplex the callers into the channels. Such a switching network is often called a *concentrator* [21], [25].

Formally, an (n, m)-concentrator, where $m \leq n$, is a switching network with n inputs (for callers), and m outputs (for channels) that can connect any k of its inputs to some fixed k of its outputs, but without the ability to distinguish between those outputs. In addition to being used for multiplexing callers into channels, concentrator switches also play a central role in the constructions of efficient superconcentrators [8], permutation and broadcast networks [20], [27], [18], [13].

As with other switching networks, three parameters underpin the performance of a concentrator: its cost, depth, and concentration time. In this paper, the *cost* of a concentrator is the number of constant fanin, bit-level switches it contains, its depth is the maximum number of such switches on a path from an input to an output, and its concentration time is the time it takes to realize any concentration assignment. The unit of time is a delay through a constant fanin, bit-level switch or logic gate.

Concentrators have been studied under two different models in the literature. In the first model, a concentrator is viewed as a graph with two distinguished sets of vertices, called inputs and outputs [4]. A concentration pattern is a set of vertex-disjoint paths between the inputs and outputs through the graph. The cost in this model is defined as the number of edges in the concentrator graph, and the depth is defined as the maximum number of edges on a path from an input to an output.

In a seminal work, Pinsker showed by a combinatorial argument that there exist concentrators with linear cost and logarithmic depth [22]. Later Margulis [17] gave a concentrator which, he showed, had linear cost, but could not determine the exact constant factor in the cost expression. Based on Margulis' construction, Gabber and Galil [6] gave an explicit concentrator construction with linear cost and logarithmic depth. Since this construction was revealed, several other explicit constructions with smaller constants have also been obtained [24], [1], [14].

Despite their linear cost, and logarithmic depth, it is difficult to route these concentrators as they are all based on a certain kind of bipartite graph, called an *expander* [22], [2]. Realizing concentration assignments on these concentrators requires finding matchings on expanders. Unfortunately, the best matching algorithm for bipartite graphs requires $O(n^{2.5})$ serial time [7], [9], [5], and even though there exist parallel matching algorithms that run in polylogarithmic orders of time, these algorithms require parallel computers with complex interconnection topologies [16], [19], [5], [26].

Recently, another concentrator model was introduced in [12] to incorporate the routing problem into the construction of a concentrator. In this model, the concentration problem is broken into two subproblems, namely *ranking* and *routing*. It was shown in [12] that the ranking problem can be solved by using an $O(n \lg n)$ bit-level cost prefix circuit with $O(\lg^2 n)$ bit-level depth. It was also shown there that the routing problem can be solved by using an $O(n \lg^2 n)$ bit-level cost cube network with $O(\lg n)$ bit-level depth. More recently, this construction was modified in [10] to reduce the total bit-level cost to $O(n \lg n)$ and the bit-level depth to $O(\lg n \lg \lg n)$.

While these concentrators provide low cost, small depth and short routing time, their implementations require different types of components such as arithmetic circuits and switching

Paper approved by E. G. Sable, the Editor for Communication Switching of the IEEE Communications Society. Manuscript received May 6, 1992; revised August 19, 1992. This work was 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. This paper was presented in part at the International Conference on Parallel Processing, St. Charles, IL, USA, August 1992.

M. V. Chien is with PRC Inc., McLean, VA, USA

A. Y. Oruç with the Electrical Engineering Department and Institute for Advanced Computer Studies University of Maryland, College Park, MD 20742 USA.

IEEE Log Number 9401948.

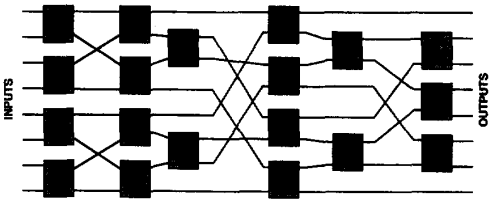


Fig. 1. An 8-input Batcher's odd-even merge concentrator.

devices. In this paper, we introduce a concentrator construction that can be implemented by using only multiplexer circuits. The main concept behind our concentrator is the notion of adaptive binary sorting [28] which combines the ranking and routing steps into a single step. This concentrator has $O(n \lg n)$ bit-level cost, $O(\lg^2 n)$ bit-level depth and routing time, and with pipelining, its cost can be reduced to $O(n \lg \lg n)$ without increasing its depth and routing time. These cost, depth and routing time complexities are competitive with the complexities of the concentrators mentioned above.

The rest of this paper is organized as follows. Section II presents the multiplexer-based concentrator. Section III describes how time-multiplexing and pipelining can be used to reduce the cost of this concentrator without increasing its depth and routing time. Section IV shows how this concentrator can be used to obtain a superconcentrator with the same cost, depth and routing time complexities. The paper is concluded in Section V.

II. Binary Sorters as Concentrators

Both concentrators to be described are based on adaptive binary sorting.[1] Suppose that all those inputs of a concentrator that need to be concentrated are assigned a 0 bit, and the remaining inputs are assigned a 1 bit. If the inputs are sorted in ascending order with their assigned bits used as the keys for sorting, then all those inputs that are assigned 0 bits (or 1 bits) can be mapped together into a set of consecutive locations which then amounts to a concentration of the inputs. We carry out this sorting on a network of multiplexers. Unlike conventional sorting networks that are constructed out of compare/exchange elements only [11], [3], our binary sorters are adaptive in that we probe various points in a sorter to determine the paths of inputs. For comparison, an 8-input Batcher's odd-even merge sorting network is depicted in Fig. 1, where the shaded boxes represent two-input compare/exchange elements. This network is nonadaptive in that it can be constructed with such elements only. For n inputs its cost is $O(n \lg^2 n)$ and its depth is $O(\lg^2 n)$.

Our adaptive binary sorter is based on the following observation.

Definition 1: A binary sequence is called bi-sorted if each of its two halves is sorted.

Theorem 1 [28]: If a bi-sorted binary sequence is cut into four equal-size subsequences, then at least two of the subsequences contain only 0's or only 1's and the other two, when concatenated form a bi-sorted sequence.

[1]For ease of discussion, but with no loss of generality, all networks described in this paper are assumed to have a power of two inputs.

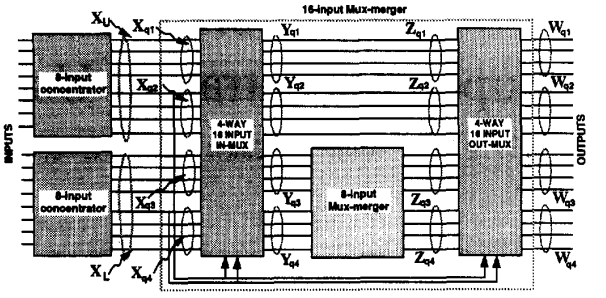


Fig. 2. A 16-input concentrator.

TABLE I
Behavior of Mux-Merger

Select inputs	Input pattern	FOUR-WAY IN-MUX select	FOUR-WAY OUT-MUX select
00	X_{q1} and X_{q3} are all 0's, $X_{q2} * X_{q4}$ is bi-sorted	X_{q1} to Y_{q1}, X_{q2} to Y_{q3}, X_{q3} to Y_{q2}, and X_{q4} to Y_{q4}	Z_{q1} to W_{q1}, Z_{q2} to W_{q2}, Z_{q3} to W_{q3}, and Z_{q4} to W_{q4}
01	X_{q1} is all 0's, X_{q4} is all 1's and $X_{q2} * X_{q3}$ is bi-sorted	X_{q1} to Y_{q1}, X_{q2} to Y_{q3}, X_{q3} to Y_{q4}, and X_{q4} to Y_{q2}	Z_{q1} to W_{q1}, Z_{q2} to W_{q4}, Z_{q3} to W_{q2}, and Z_{q4} to W_{q3}
10	$X_{q1} * X_{q4}$ is bi-sorted, X_{q2} is all 1's, and X_{q3} is all 0's	X_{q1} to Y_{q3}, X_{q2} to Y_{q2}, X_{q3} to Y_{q1}, and X_{q4} to Y_{q4}	Z_{q1} to W_{q1}, Z_{q2} to W_{q4}, Z_{q3} to W_{q2}, and Z_{q4} to W_{q3}
11	$X_{q1} * X_{q3}$ is bi-sorted, X_{q2} and X_{q4} are all 1's	X_{q1} to Y_{q3}, X_{q2} to Y_{q2}, X_{q3} to Y_{q4}, and X_{q4} to Y_{q1}	Z_{q1} to W_{q3}, Z_{q2} to W_{q4}, Z_{q3} to W_{q1}, and Z_{q4} to W_{q2}

Proof: Let X_n denote a bi-sorted binary sequence of size n. Let X_{q1}, X_{q2}, X_{q3} and X_{q4} denote the four quarters of X_n from top to bottom, respectively, and X_U and X_L denote its upper and lower halves (refer to Fig. 2). If there are more 0's than 1's in X_U, then X_{q1} must contain all 0's and X_{q2} must be a sorted sequence of size $n/4$. On the other hand, if there are either an equal or more number of 1's than 0's in X_U, then X_{q2} must contain all 1's, and X_{q1} must be a sorted sequence of size $n/4$. Similar statements hold for X_{q3} and X_{q4}. Therefore, at least two of X_{q1}, X_{q2}, X_{q3} and X_{q4} must be all 0's or all 1's and the other two must form a bi-sorted sequence when concatenated. ||

Thus, if a binary sequence of n inputs is bi-sorted using two $n/2$-input sorters then the proof of the theorem suggests how to identify the two quarters of the bi-sorted sequence that form a bi-sorted sequence of size $n/2$ when concatenated, as well as its all zero or all one quarters. The network in Fig. 2 is constructed based on this fact, where the network enclosed by the rectangle in dash lines, and called a *Mux-Merger*, merges the bi-sorted sequence at the outputs of the two sorters into a sorted sequence.

Table I lists the possible patterns of bi-sorted sequences and shows how the Mux-Merger selects the quarter-size subsequences for each pattern.[2] The entries in the table denote

[2]The symbol * used in the table denotes a concatenation operator.

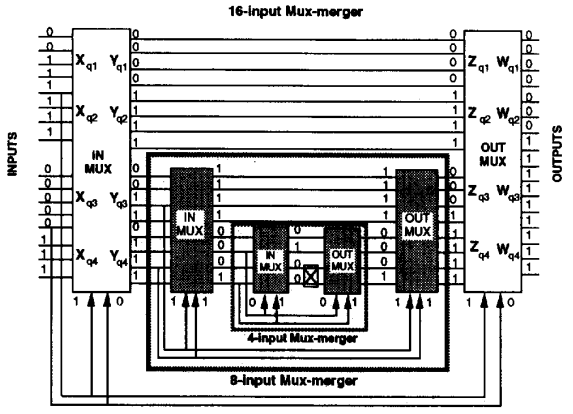


Fig. 3. An example showing the operations of a 16-input Mux-Merger.

various points in the network as marked in Fig. 2. As suggested by the proof of Theorem 1, the middle two bits of the two sorted halves of outputs can be one of four binary patterns, $00, 01, 10, 11$. For each of these, the patterns of X_{q1}, X_{q2}, X_{q3} and X_{q4} are uniquely determined, and the selections of the four quarters can be made accordingly as shown in the table. The n-input, four-way IN-MUX and OUT-MUX circuits shown in the figure for $n = 16$ map their four quarters of inputs to their four quarters of outputs without altering their relative ordering according to the binary code they receive at their select inputs as shown in the table. All these maps can be realized by connecting each input in both IN-MUX and OUT-MUX circuits to four outputs, one in each group of outputs. Thus, an n-input four-way IN-MUX or OUT-MUX circuit can be implemented by using n 4×1 multiplexers.

Since the concatenated two quarters form a bi-sorted sequence by Theorem 1, the same Mux-Merge process can be applied recursively. As an example, Fig. 3 shows the operations of a 16-input Mux-Merger, for a 16-element bi-sorted binary sequence, $00111111/00000111$. Since the two middle elements are 1 and 0, by Table I, IN-MUX circuit moves X_{q1} to Y_{q3}, X_{q2} to Y_{q2}, X_{q3} to Y_{q1}, and X_{q4} to Y_{q4}. The resulting 8-element bi-sorted sequence $Y_{q1} * Y_{q4}$, i.e., 00110111, then becomes the input for the next level of Mux-Merger. The process is repeated through the next level of Mux-Merger, and finally, the multiplexer, OUT-MUX reorders the four sorted quarters to form a sorted output.

The entire binary sorting network can be constructed using IN-MUX and OUT-MUX circuits if the half-size sorters in Fig. 2 are recursively replaced by the same sorter construction. An example is depicted in Fig. 4 for $n = 16$.

The cost and depth complexities of this binary sorting network for n inputs can be recursively expressed as

$$C(n) = 2C(n/2) + C_m(n) \quad (1)$$

$$D(n) = D(n/2) + D_m(n) \quad (2)$$

where $C(n)$ and $D(n)$ denote the cost and depth of the entire network respectively, and $C_m(n)$ and $D_m(n)$ denote the cost and depth of the Mux-Merger. $C_m(n)$ and $D_m(n)$ can also be

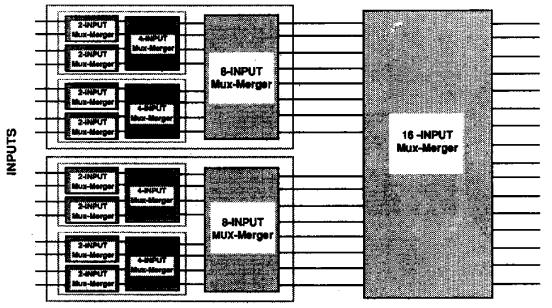


Fig. 4. A 16-input concentrator constructed with Mux-Mergers.

expressed recursively as

$$C_m(n) = C_m(n/2) + C_{IN-MUX}(n) + C_{OUT-MUX}(n) \quad (3)$$

$$D_m(n) = D_m(n/2) + D_{IN-MUX}(n) + D_{OUT-MUX}(n) \quad (4)$$

where $C_{IN-MUX}(n)$ and $C_{OUT-MUX}(n)$ denote the costs of IN-MUX and OUT-MUX circuits, and $D_{IN-MUX}(n)$ and $D_{OUT-MUX}(n)$ denote their depths. It is easy to see that IN-MUX and OUT-MUX circuits can be constructed with $O(n)$ constant fanin, bit-level logic gates, and in $O(1)$ bit-level depth. Substituting these in the above equations and solving the recurrences with $C_m(2) = O(1)$ and $D_m(2) = O(1)$, gives $C_m(n) = O(n)$ and $D_m(n) = O(\lg n)$. These, combined with (1) and (2), yield $C(n) = O(n \lg n)$ and $D(n) = O(\lg^2 n)$. It should be obvious that the routing time complexity of this binary sorting network is the same as its depth complexity. Thus, these results establish an n-input concentrator with $O(n \lg n)$ cost, $O(\lg^2 n)$ depth, and $O(\lg^2 n)$ concentration time, all in bit-level. We note that the constants in these complexity expressions are small and are bounded from above by the number of gates that are needed to implement a 4×1 multiplexer in terms of elementary logic gates. A naive implementation would use at most four three-input AND gates and one four-input OR gate, and would have a depth of 2. These numbers are comparable to the number of gates that would be needed to implement a two-input compare/exchange element which is used in the construction of Batcher's odd-even merge sorters.

III. Time-Multiplexed Concentration

The cost of the concentrator described in the previous section can be reduced to $O(n \lg \lg n)$ without increasing its depth or routing time. For this, the network shown in Fig. 5 can be used. In this network, we use a multiplexing scheme proposed in [15]. The idea is to multiplex the binary inputs through a binary sorting network with a smaller number of inputs. The input sequence is first arbitrarily divided into k groups of n/k elements, where $2 \leq k \leq n$ and k divides n. Each group of inputs is run through an $(n, n/k)$-MUX circuit[3] sequentially, and sorted by an n/k-input binary sorter.

[3] An $(n, n/k)$-MUX circuit is just a collection of n/k $k \times 1$ multiplexers.

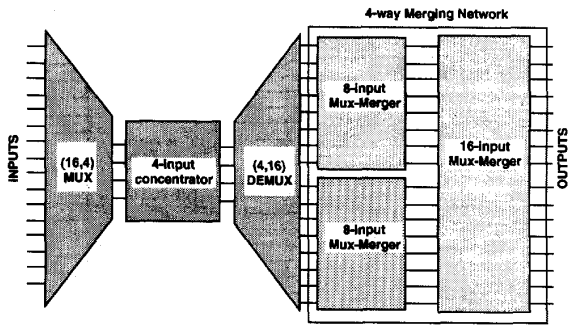


Fig. 5. A 16-input concentrator using a multiplexing scheme, where $k = 4$.

The sorted n/k-element sequences are then moved through an $(n/k, n)$-DEMUX circuit[4] to the inputs of a k-way merger, where the first n/k inputs of the k-way merger are occupied by the first sorted n/k-element sequence, the second n/k inputs are occupied by the second sorted n/k-element sequence, and so on.

The k-way merger is implemented in terms of Mux-Mergers by using an iterative construction. The first stage of the merger uses $k/2$ $2n/k$-input Mux-Mergers to merge pairs of n/k-element sorted sequences, the second stage uses $k/4$ $4n/k$-input Mux-Mergers to merge pairs of $2n/k$-element sorted sequences, and in general, the ith stage uses $k/2^i$ $2^i n/k$-input Mux-Mergers to merge pairs of $2^{i-1}n/k$-element sorted sequences, $1 \leq i \leq \lg k$. For the network given in Fig. 5, $n = 16$, and $k = 4$.

Let $C(n,k)$ and $D(n,k)$ denote the cost and depth of this network, respectively. It follows from the preceding discussion that

$$C(n,k) = C_d(n,n/k) + C_s(n/k) + C_w(n,k) \quad (5)$$

$$D(n,k) = D_d(n,n/k) + D_s(n/k) + D_w(n,k), \quad (6)$$

where $C_d(n,n/k)$ and $D_d(n,n/k)$ denote the cost and depth of the $(n,n/k)$-MUX and $(n/k,n)$-DEMUX circuits combined together, $C_s(n/k)$ and $D_s(n/k)$ denote the cost and depth of the n/k-input binary sorter, and $C_w(n,k)$ and $D_w(n,k)$ denote the cost and depth of the k-way merger. The $(n,n/k)-$MUX (DEMUX) circuit can be implemented by using $O(n/k)$ $O(\lg k)$-level binary trees whose nodes are 2×1 multiplexers (1×2 demultiplexers) [10], [15]. The bit-level cost and depth of these implementations can be shown to be $O(n)$ and $O(\lg k)$, respectively. If we use the binary sorting network construction of the preceding section then the cost and depth of the n/k-input sorter are $O((n/k)\lg(n/k))$ and $O(\lg^2(n/k))$, respectively. As for the k-way merger, summing up the costs and depths of the Mux-Mergers in its construction,

$$C_w(n,k) = \sum_{i=1}^{\lg k} O(\frac{k}{2^i}\frac{2^i n}{k}) = O(n \lg k) \quad (7)$$

$$D_w(n,k) = \sum_{i=1}^{\lg k} O(\lg \frac{2^i n}{k}) = O(\lg n \lg k) \quad (8)$$

[4]An $(n/k,n)$-DEMUX circuit is a collection of n/k $1 \times k$ demultiplexers.

Substituting these into (5) and (6) we find

$$C(n,k) = O(n) + O(\frac{n}{k}\lg\frac{n}{k}) + O(n \lg k), \quad (9)$$

$$D(n,k) = O(\lg k) + O(\lg^2 \frac{n}{k}) + O(\lg n \lg k), \quad (10)$$

and noting that these expressions are minimized when $k = O(\lg n)$, we obtain

$$C(n, \lg n) = O(n \lg \lg n) \quad (11)$$

$$D(n, \lg n) = O(\lg^2 n). \quad (12)$$

We note that the inputs to be sorted in this binary sorter construction are multiplexed, and hence the depth of the network does not give the time it takes to complete the sorting. But given that the depths of all the components in this network are known, its sorting time, $T(n,k)$, can be determined by noting that the k groups of n/k bits must be multiplexed through the n/k-input sorter. Thus,

$$T(n,k) = k(D_d(n,n/k) + D_s(n/k)) + D_w(n,k) \quad (13)$$

$$= O(k \lg k) + O(k \lg^2 \frac{n}{k}) + O(\lg n \lg k) \quad (14)$$

and letting $k = \lg n$ gives

$$T(n, \lg n) = O(\lg^3 n). \quad (15)$$

The sorting time can be reduced to $O(\lg^2 n)$ by noting that the k groups of n/k inputs can be pipelined through the n/k-input binary sorting network. The sorting time with pipelining, $T_{pip}(n,k)$, is given by

$$T_{pip}(n,k) = O(\lg^2 \frac{n}{k}) + O(k) + O(\lg k) + O(\lg n \lg k) \quad (16)$$

where the $O(\lg^2(n/k))$ and $O(\lg k)$ terms account for the time for the first group of n/k elements to exit the pipeline, and the $O(k)$ term accounts for the time that the remaining $k-1$ groups of n/k elements need to get through the pipeline. The $O(\lg n \lg k)$ term is the merging time which takes place after the pipelining is completed. Letting $k = \lg n$ gives $T_{pip}(n, \lg n) = O(\lg^2 n)$ as desired.

IV. Superconcentrators

An (n,m)-superconcentrator is a switching network with n inputs and m outputs that can connect any k of its n inputs, where $k \leq min\{n,m\}$, to any k of its m outputs, but without the ability to distinguish between those outputs. The main distinction between a concentrator and superconcentrator is that, in the latter, the inputs can be mapped to any set of outputs on a one-to-one basis. In contrast, the inputs in a concentrator can be mapped to some fixed but not any set of outputs. Superconcentrators can thus be used to circumvent faults in the output terminals, and thus form a viable alternative to concentrators.

A superconcentrator can be constructed by cascading two concentrators back-to-back [8]. More precisely, an (n,m)-superconcentrator can be constructed by cascading an (n,k)-concentrator with an (m,k)-concentrator as shown in Fig. 6. It must be noted that the concentrators in this construction must

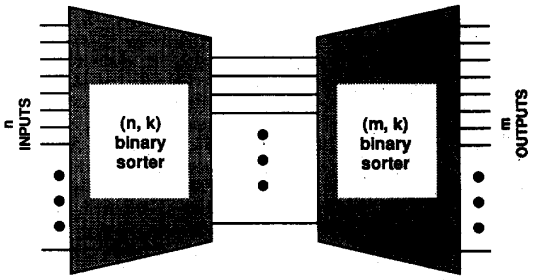


Fig. 6. A superconcentrator using binary sorters.

TABLE II
COMPLEXITIES OF VARIOUS CONCENTRATOR DESIGNS IN BIT LEVEL

Construction	Cost	Depth	Concentration Time
Odd-even Merge Concentrator [3]	$O(n \lg^2 n)$	$O(\lg^2 n)$	$O(\lg^2 n)$
Ranking Concentrator [12]	$O(n \lg^2 n)$	$O(\lg^2 n)$	$O(\lg^2 n)$
Reduced Ranking Concentrator[10]	$O(n \lg n)$	$O(\lg n \lg \lg n)$	$O(\lg n \lg \lg n)$
Mux-Merger Concentrator	$O(n \lg n)$	$O(\lg^2 n)$	$O(\lg^2 n)$
Multiplexed/Pipelined Mux-Merger Concentrator	$O(n \lg \lg n)$	$O(\lg^2 n)$	$O(\lg^2 n)$

be capable of concentrating their inputs to a common set of outputs. The binary sorters described in the previous sections obviously meet this criterion. The only alteration that is needed before they can be used in this superconcentrator construction is to remove all but their first k outputs. With this alteration the routing on the superconcentrator proceeds in two directions. The inputs to be concentrated are sorted to the outputs of the left binary sorter, and the outputs onto which the inputs are to be concentrated are sorted by the right binary sorter to meet the concentrated inputs at the outputs of the left binary sorter.

It is easy to see that the cost, depth and routing complexities of this superconcentrator are of the same order as the cost, depth and routing time complexities of the binary sorters used in its construction.

V. CONCLUDING REMARKS

This paper explored adaptive binary sorting and multiplexing schemes to construct fast and low cost concentrators and superconcentrators. The complexities of the two concentrators described in the paper are listed in Table II along with those given in [3], [12], [10]. As can be seen in the table, the cost of the Mux-Merger concentrator is smaller than that of the odd-even merge concentrator, and matches the cost of the concentrator given in [10]. The cost of the time-multiplexed Mux-Merger concentrator is smaller than all three concentrators. The depth and concentration time of both Mux-Merger and time-multiplexed Mux-Merger concentrator match those of the concentrators given in [3], [12] and are slightly higher than those of the concentrator given in [10].

Unlike the expander based concentrators [22], [17], [23], [6], [1], or those that use ranking trees [12], [10], the Mux-Merger sorter can be implemented using simple 4×1 multiplexers, and does not require any comparators or adders. Furthermore, with time-multiplexing and pipelining, the Mux-Merger binary sorter comes very close to being an optimal concentrator in terms of its cost, with its depth and concentration time being quite competitive. In fact, in a recent paper [28], we were able to optimize this construction, i.e., reduce its cost to $O(n)$ without increasing its concentration time.

The paper also described how to use Mux-Merger sorters to obtain an n-input superconcentrator with $O(n \lg \lg n)$ bit-level cost and $O(\lg^2 n)$ bit-level routing time.

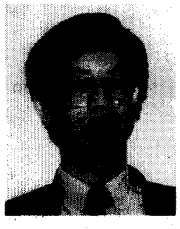
ACKNOWLEDGMENT

The authors thank the anonymous reviewers for their constructive comments. They also thank Dr. E. G. Sable for his expeditious and careful handling of the paper.

REFERENCES

[1] N. Alon, "Eigenvalues and expanders," *Combinatorics*, vol. 6, pp. 83–96, 1986.
[2] L. Bassalygo, "Asymptotically optimal switching circuits," *Probl. Inform. Trans.*, vol. 17, pp. 206–211, July-Sept. 1981.
[3] K. E. Batcher. "Sorting networks and their applications," in *Proceedings of AFIPS Spring Joint Conference*, pp. 307–314, 1968.
[4] F. R. K. Chung, "On concentrators, superconcentrators, generalizers, and nonblocking networks," *Bell Syst. Tech. J.*, vol. 58, pp. 1765–1777, Oct. 1978.
[5] J. Carpinelli and A. Y. Oruç, "Applications of edge-coloring algorithms to routing in parallel computers," in *Proc. 3rd Int. Conf. Supercomput.*, Santa Clara, CA, May 1988.
[6] O. Gabber and Z. Galil, "Explicit constructions of linear sized superconcentrators," *J. Comput. Syst. Sci.*, vol. 22, pp. 407–420, 1981.
[7] J. Hopcroft and R. Karp, "An $n^{5/2}$ algorithm for maximum matching in bipartite graphs," *SIAM J. Comput.*, vol. 2, pp. 225–231, Dec. 1973.
[8] J. Y. Hui, *Switching and Traffic Theory for Integrated Broadband Network*. Boston, MA: Kluwer, 1990.
[9] F. K. Hwang, "Control algorithms for rearrangeable Clos networks," *IEEE Trans. Commun.*, vol COM-31, pp. 952–954, Aug. 1983.
[10] C. Y. Jan and A. Y. Oruç, "Fast self-routing permutation switching on an asymptotically minimum cost network," *IEEE Trans. Comput.*, vol. 42, pp. 1369–1379, Dec. 1993.
[11] D. Knuth. *The Art of Computer Programming: Sorting and Searching*. Reading, MA: Addison and Wesley, 1973.
[12] D. Koppelman and A. Y. Oruç, "A self-routing permutation network," *J. Parallel Distrib. Comput.*, vol. 6, pp. 140–151, Aug. 1990.
[13] C. Lea, "A new broadcast switch," *IEEE Trans. Commun.*, vol. 36, pp. 1128–1137, Oct. 1988.
[14] C.-Y. Lee and A. Y. Oruç, "Explicit designs of linear-size expanders, concentrators and superconcentrators," Manuscript, Oct. 1991.
[15] M. Lu and A. Y. Oruç, "Linear-cost networks for realizing unicast assignments in polylogarithmic time," Manuscript, Jan. 1992.
[16] G. F. Lev, N. Pippenger, and L. Valiant, "A fast parallel routing algorithm for routing in permutation networks," *IEEE Trans. Comput.*, vol. C-30, pp. 93–100, Feb. 1981.
[17] G. A. Margulis, "Explicit constructions of concentrators," *Probl. Inform. Transmission*, vol. 9, no. 4, pp. 325–332, 1973.
[18] G. M. Masson, G. C. Gingher, and S. Nakamura, "A sampler of circuit switching networks," *IEEE Comput. Mag.*, vol. 12, pp. 32–48, June 1979.
[19] D. Nassimi and S. Sahni, "Parallel algorithms to set up the Benes network," *IEEE Trans. Comput.*, vol. C-31, pp. 148–154, Feb. 1982.
[20] J. P. Ofman, "A universal automaton," *Trans. Moscow Mathemat. Soc.*, vol. 14, pp. 200–215, 1965.
[21] D. L. Pehrson, "Interfacing and data concentration," in *Computer Communication Networks*, N. Abramson and F. Kuo, Eds. Englewood Cliffs, NJ: Prentice Hall, 1973, ch. 6, pp. 197–236.
[22] M. S. Pinsker, "On the complexity of a concentrator," in *Proc. 7th Int. Teletraffic Congr.*, Stockholm, Sweden, 1973, pp. 318/1–318/4.

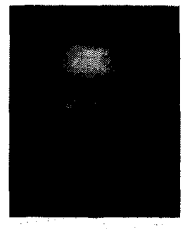
[23] N. Pippenger, "Superconcentrators," *SIAM J. Comput.*, vol. 6, pp. 298–304, 1977.
[24] ——, "Telephone switching networks," in *Symp. Appl. Mathemat.*, vol. 26, American Mathematical Society, pp. 101–133, 1982.
[25] S. F. Smith, *Telephony and Telegraphy*. Oxford, England: Oxford University Press, 1978, 2nd ed.
[26] T. H. Spencer, "Parallel matching on expanders," Dep. Comput. Sci., Rensselaer Polytech. Inst., Tech. Rep. 91-14, Troy, NY, 1991.
[27] C. D. Thompson, "Generalized connection networks for parallel processor intercommunication," *IEEE Trans. Comput.*, vol. C-27, pp. 1119–1125, Dec. 1978.
[28] M. V. Chien and A. Yavuz Oruç, "Adaptive binary sorting schemes and associated interconnection networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 5, pp. 561–572, June 1994.



Minze V. Chien (S'86–M'87–M'92) received the B.Sc. degree in physics from Fu Jen Catholic University, Taipei, Taiwan, in 1980, and the M.Sc. and Ph.D. degrees in electrical engineering from the University of Maryland, College Park, MD, in 1988 and 1992, respectively.

His current research interests include interconnection networks, parallel computer architectures, VLSI design and architectures, distibuted computing, and computer networking, and neural networks.

Dr. Chien is a member of the IEEE Computer Society.



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 Associate Professor in the Department of Electrical Engineering at the University of Maryland, College Park. 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 computer and communication systems.

Dr. Oruç is a member of the IEEE Computer, Communication, and Information Theory Societies.