Efficient Multichip Partial Concentrator Switches

Thomas H. Cormen

Laboratory for Computer Science
Massachusetts Institute of Technology
Cambridge, Massachusetts 02139

Abstract

Due to chip area and pin count constraints, large concentrator switches sometimes must be partitioned among several chips. This paper presents designs for two multichip partial concentrator switches, both of which follow from a lemma showing that an ε-nearsorter is also an $(n, m, 1 - \varepsilon/m)$ partial concentrator.

The first switch, based on the Revsort algorithm, is an $(n, m, 1 - O(n^{3/4}/m))$ partial concentrator switch with at most $2\sqrt{n} + \lceil (\lg n)/2 \rceil$ data pins per chip, $\Theta(\sqrt{n})$ chips, and volume $\Theta(n^{3/2})$. A message incurs $3 \lg n + O(1)$ gate delays in passing through the switch.

The second switch, based on Columnsort, is an $(n, m, 1 - O(n^{2-2\beta}/m))$ partial concentrator switch with $\Theta(n^\beta)$ data pins per chip, $\Theta(n^{1-\beta})$ chips, and volume $\Theta(n^{1+\beta})$, for any $1/2 \leq \beta \leq 1$. A message incurs $4\beta \lg n + O(1)$ gate delays.

1 Introduction

The problem of concentrating relatively few signals on many input lines onto a lesser number of output lines must be solved in many kinds of communication networks. In many parallel computing systems, information is packaged into messages which are routed among the processors. The switches that route these messages sometimes require more chip area or input and output wires than a single chip can supply. This paper presents two designs for fast multichip partial concentrator switches suitable for routing bit-serial messages in a parallel supercomputer. The key lemma of this paper may be used to justify other partial concentrator designs.

An *n-by-m perfect concentrator switch* has n input wires $X_1, X_2, \ldots, X_n$ and $m \leq n$ output wires $Y_1, Y_2, \ldots, Y_m$. The switch can establish m disjoint electrical paths from any set of m input wires to the m output wires. A perfect concentrator switch always routes as many messages as possible. Specifically, whenever k out of the n input wires of an n-by-m perfect concentrator switch carry messages, one of the following is true:

- If $k \leq m$, then an electrical path is established from each input wire that contains a message to an output wire.
- If $k > m$, then each output wire has an electrical path established from an input wire that contains a message.

When $k > m$, some messages cannot be successfully routed, in which case we say the switch is *congested*. Typical ways of handling unsuccessfully routed messages in a routing network are to buffer them, to misroute them, or to simply drop them and rely on a higher-level acknowledgment protocol to detect this situation and resend them. The switch designs in this paper are compatible with any of these congestion control methods.

One way to create a perfect concentrator switch is with a hyperconcentrator switch. An *n-by-n hyperconcentrator switch* has n input wires $X_1, X_2, \ldots, X_n$ and n output wires $Y_1, Y_2, \ldots, Y_n$. The switch can establish disjoint electrical paths from any set of k input wires, for any $1 \leq k \leq n$, to the first k output wires $Y_1, Y_2, \ldots, Y_k$. In other words, we route the k messages to the first k output wires. We can make any n-by-m perfect concentrator switch from an n-by-n hyperconcentrator switch by simply choosing the first m output wires of the hyperconcentrator switch, $Y_1, Y_2, \ldots, Y_m$, as the m output wires of the perfect concentrator switch.

An efficient n-by-n hyperconcentrator switch design is given in [1] and [2]. This switch has a highly regular layout in both ratioed nMOS and domino CMOS technologies, and a signal incurs exactly $2 \lg n$ gate delays through the switch.[1] This switch uses $\Theta(n^2)$ components and has area $\Theta(n^2)$.

Partitioning this hyperconcentrator switch among multiple chips with p pins each requires $\Omega((n/p)^2)$ chips, since each p-pin chip has area $O(p^2)$ and there are $\Theta(n^2)$ components to partition. We may need to partition the switch for two reasons:

1. The $\Theta(n^2)$ area may exceed the available chip area.
2. If the switch is to be packaged by itself on a chip, it may require more input and output pins than are provided by the packaging technology.

A different hyperconcentrator switch, comprised of a parallel prefix circuit and a butterfly network [1], can be built in volume $\Theta(n^{3/2})$ with $O(n \lg n)$ chips and as few as four data pins per chip, but this switch is not combinational. Although its sequential control is not very complex, it is not as simple as that of a combinational circuit.

Partial concentrator switches, as we shall see in Sections 4 and 5, can be combinational with relatively low

[0] This research was supported in part by the Defense Advanced Research Projects Agency under Contract N00014–80–C–0622 and in part by a National Science Foundation Fellowship.

[1] We use the notation $\lg n$ to denote $\log_2 n$.

gate delays. Yet, given chips with p pins, we can partition n-input partial concentrator switches using only $\Theta(n/p)$ chips. An (n, m, α) *partial concentrator switch* has n input wires $X_1, X_2, \ldots, X_n$, $m \leq n$ output wires $Y_1, Y_2, \ldots, Y_m$, and a fraction $0 < \alpha \leq 1$ such that disjoint electrical paths may be established from any set of k input wires, for any $1 \leq k \leq \alpha m$, to k output wires.

A lightly loaded partial concentrator switch is similar to a perfect concentrator switch. If there are k messages entering an (n, m, α) partial concentrator switch, one of the following is true:

- If $k \leq \alpha m$, then an electrical path is established from each input wire that contains a message to an output wire.
- If $k > \alpha m$, then at least αm electrical paths are established from input wires containing messages to output wires.

We call the fraction α the *load ratio*. If a partial concentrator switch is lightly loaded, i.e., the number of messages entering is at most αm, then *all* the messages are routed to output wires.

An $(n/\alpha, m/\alpha, \alpha)$ partial concentrator switch can be used anywhere an n-by-m perfect concentrator switch is required. Consider a set of $k \leq m$ messages to be routed through an n-by-m perfect concentrator switch. For the $(n/\alpha, m/\alpha, \alpha)$ partial concentrator switch, we have that $k \leq m = \alpha \cdot (m/\alpha)$, and thus all k messages are routed to output wires. If there are instead $k > m$ messages to be routed through the perfect concentrator switch, we have that $k > m = \alpha \cdot (m/\alpha)$ for the $(n/\alpha, m/\alpha, \alpha)$ partial concentrator switch, and thus m output wires carry messages. In either case, the partial concentrator switch performs the same function as the perfect concentrator switch, at the cost of a $1/\alpha$-factor increase in the number of input and output wires.

In this paper, we show a connection between nearsorting and partial concentration. We then use this relationship to design two efficient multichip partial concentrator switches, both of which use the hyperconcentrator switch of [1] and [2] as a subcircuit on a single chip.

The remainder of this paper is organized as follows. Section 2 covers some basic terminology and describes the message format upon which the switches are based. Section 3 defines nearsorting and shows the relationship between nearsorting and partial concentration. Section 4 presents a design for a partial concentrator switch based on the Revsort algorithm for sorting on a mesh; Section 5 does the same, but based on the Columnsort algorithm for sorting on a mesh. Finally, Section 6 contains further remarks about multichip concentrator switches.

2 Preliminaries

In this section, we define some basic terminology and mathematical conventions and present the message format assumed by the switch designs.

We assume that the switches route *bit-serial messages.* Each message is formed by a stream of bits arriving at a wire at the rate of one bit per clock cycle. The first bit of each message that arrives at an input wire is the *valid bit*, indicating whether subsequent bits arriving on that wire form a valid message or an invalid message. The bit sequence following a valid bit of 1 forms a *valid message*, which we would like to be routed from an input wire to an output wire of the switch. From there it may pass through the remainder of the routing network. A valid bit of 0 indicates an *invalid message*, which does not need to be routed to an output wire.

The valid bits all arrive at the input wires of a switch during the same clock cycle, which we call *setup*. An external control line signals setup. Message bits entering through input wires at cycles after setup follow the electrical paths in the switch that are established during setup.

We shall adopt some notational conventions to ease the exposition in the remainder of this paper. Uppercase symbols denote wire names and lowercase symbols denote integer values. We shall also use uppercase symbols to denote bit values on the wires they name when the usage is unambiguous. Wire names will usually be subscripted.

A sequence of values is *sorted* if it is in nonincreasing order. The valid bits output by an n-by-n hyperconcentrator switch are thus sorted, since if there are k valid messages, we have

$$\begin{aligned} Y_1, Y_2, \ldots, Y_k &= 1 \\ Y_{k+1}, Y_{k+2}, \ldots, Y_n &= 0 \end{aligned}$$

during setup.

Concentrators were originally presented as graphs in, for example, [4, 5, 8]. The term "hyperconcentrator" is due to Valiant. Vertex-disjoint paths from designated input nodes to designated output nodes are the concentrator graph counterpart of the combinational routing paths established during setup in the concentrator switches of this paper.

3 Nearsorting and Partial Concentration

In this section, we define ε-nearsorting and show its relationship to partial concentration. The key lemma proven in this section is used in the next two sections to justify partial concentrator switch constructions.

A sequence of values is *ε-nearsorted* if each element in the sequence is within ε positions of where it belongs in the fully sorted sequence. For example, the sequence $5, 3, 6, 1, 4, 2$ is 2-nearsorted since each element is at most two places away from its correct position in the fully sorted sequence $6, 5, 4, 3, 2, 1$. The value ε need not be a constant; we will usually let ε be a function of the size of the sequence. A fully sorted sequence is also 0-nearsorted.

Since we are only interested in nearsorting valid bits, for the remainder of this paper we shall be concerned only with inputs whose value is either 0 or 1. We say that a sequence of values is *clean* if they all have the same value; otherwise the sequence is *dirty*. The following lemma describes an ε-nearsorted sequence of 0's and 1's.

Lemma 1 *A sequence of n bits, containing k 1's and $n-k$ 0's, is ε-nearsorted if and only if it consists of a clean sequence of at least $k-\varepsilon$ 1's followed by a dirty sequence of at most 2ε bits followed by a clean sequence of at least $n-k-\varepsilon$ 0's.*

Figure 1: A fully sorted sequence of k 1's and $n-k$ 0's and an ε-nearsorted sequence of the same values. The ε-nearsorted sequence consists of a clean sequence of at least $k-\varepsilon$ 1's followed by a dirty sequence of at most 2ε bits followed by a clean sequence of at least $n-k-\varepsilon$ 0's.

Proof ($\Rightarrow$) As shown in Figure 1, a fully sorted sequence of k 1's and $n-k$ 0's is simply k 1's followed by $n-k$ 0's. In an ε-nearsorted sequence of the same values, each 1 appears within the first $k+\varepsilon$ positions, and each 0 appears within the last $n-k+\varepsilon$ positions. The only dirty sequence within the ε-nearsorted sequence is therefore centered at the kth position and extends ε positions to either side. The lemma then follows.

($\Leftarrow$) Again referring to Figure 1, each 1 is within the first $k+\varepsilon$ positions, and each 0 is within the last $n-k+\varepsilon$ positions. The sequence is thus ε-nearsorted. $\square$

The following lemma is the key lemma that relates ε-nearsorting to partial concentration.

Lemma 2 *Let P be a switch with n inputs $X_1, X_2, \ldots, X_n$ and n outputs $Y_1, Y_2, \ldots, Y_n$, and suppose that P ε-nearsorts valid bits. Then by restricting the outputs of P to $Y_1, Y_2, \ldots, Y_m$, for any $m \le n$, P is an (n, m, α) partial concentrator switch, where $\alpha = 1 - \varepsilon/m$.*

Proof Consider any input to switch P containing k 1's and $n-k$ 0's. We have $\alpha m = (1-\varepsilon/m)m = m - \varepsilon$, and there are two cases.

Case 1: $k \le \alpha m = m - \varepsilon$. We have $m \ge k + \varepsilon$. Since P is an ε-nearsorter, each 1 appears within the outputs $\{Y_1, Y_2, \ldots, Y_{k+\varepsilon}\} \subseteq \{Y_1, Y_2, \ldots, Y_m\}$. Thus, each 1 is routed to an output of the partial concentrator switch.

Case 2: $k > \alpha m = m - \varepsilon$. We have $m < k + \varepsilon$. Again, each 1 appears within the outputs $\{Y_1, Y_2, \ldots, Y_{k+\varepsilon}\}$. From Lemma 1, we know that at most ε of the outputs $\{Y_1, Y_2, \ldots, Y_{k+\varepsilon}\}$ carry 0's, so at most ε of the outputs $\{Y_1, Y_2, \ldots, Y_m\}$ carry 0's. Thus, at least $m - \varepsilon = \alpha m$ of the outputs $\{Y_1, Y_2, \ldots, Y_m\}$ carry 1's.

We conclude that by restricting the outputs of P to $Y_1, Y_2, \ldots, Y_m$, P is an $(n, m, 1-\varepsilon/m)$ partial concentrator switch. $\square$

The converse of Lemma 2 is not necessarily true. As shown in Figure 2, if an $(n, m, 1-\varepsilon/m)$ partial concentrator switch routes $m-\varepsilon$ out of $k > \alpha m = m - \varepsilon$ 1's to the first m outputs, the remaining $k - m + \varepsilon$ 1's may be routed to the last $k - m + \varepsilon$ out of the n outputs. In this case, if there are more than ε outputs between Y_k and $Y_{n-(k-m+\varepsilon)}$, then the output sequence is not ε-nearsorted.

Figure 2: The output of an $(n, m, 1-\varepsilon/m)$ partial concentrator switch that is not ε-nearsorted. This switch routes $m-\varepsilon$ out of $k > m-\varepsilon$ 1's to the first m outputs, but the remaining $k-m+\varepsilon$ 1's are routed to the last $k-m+\varepsilon$ out of the n outputs. If we have $k+\varepsilon < n-(k-m+\varepsilon)$, or equivalently, $k+\varepsilon < (n+m)/2$, then the last $k-m+\varepsilon$ 1's are not within ε positions of output Y_k, and thus the output sequence is not ε-nearsorted.

4 A Revsort-Based Partial Concentrator Switch

In this section, we present a design for an (n, m, α) partial concentrator switch that uses $\Theta(\sqrt{n})$ chips with only $\Theta(\sqrt{n})$ data pins each. The basic building block is the hyperconcentrator switch of [1] and [2] placed on a chip. Each message incurs $3\lg n + O(1)$ gate delays in passing through the switch. The load ratio is $\alpha = 1 - O(n^{3/4}/m)$. Most of the results of this section originally appeared in [1].

This partial concentrator switch can be implemented in

- two dimensions with $\Theta(n^2)$ area and one chip type with $2\sqrt{n}$ data pins, or
- three dimensions with $\Theta(n^{3/2})$ volume, two chip types with at most $2\sqrt{n} + \lceil(\lg n)/2\rceil$ pins, and two board types.

The design is based on Schnorr and Shamir's Revsort algorithm for sorting on a mesh [7], which, although not optimal for sorting on a mesh, is simple. The idea behind the partial concentrator switch is to nearsort a $\sqrt{n}$-by-$\sqrt{n}$ matrix of valid bits. The m output wires of the switch correspond to the first m nearsorted matrix entries.

We need some basic definitions. We assume that the rows and columns of the $\sqrt{n} \times \sqrt{n}$ matrix are numbered $0, 1, \ldots, \sqrt{n}-1$ and that $\sqrt{n} = 2^q$ for some integer q. We also define, for any integer i, $0 \le i < \sqrt{n}$, $rev(i)$ to be the binary number obtained by reversing the q bits in the binary representation of i, including the leading zeros. For example, when $\sqrt{n} = 16$, $rev(3)$ is 12.

The partial concentrator switch is built from three stages, each stage containing $\sqrt{n}$ hyperconcentrator chips. Each $\sqrt{n}$-by-$\sqrt{n}$ hyperconcentrator chip serves to fully sort a row or column of valid bits in the underlying matrix. We shall denote by $H_{l,i}$ the ith hyperconcentrator chip in stage l, for $1 \le l \le 3$ and $0 \le i < \sqrt{n}$, with input wires $X_{l,i,0}, X_{l,i,1}, \ldots, X_{l,i,\sqrt{n}-1}$ and output wires $Y_{l,i,0}, Y_{l,i,1}, \ldots, Y_{l,i,\sqrt{n}-1}$.

The general idea of the construction of the partial concentrator switch is as follows. Each stage 1 chip corresponds to a column of the matrix, so the stage 1 chips fully sort the valid bits in each column. The input and output wires $X_{1,j,i}$ and $Y_{1,j,i}$ represent the value of the matrix element at row i and column j before and after sorting.

The wiring between stages 1 and 2 is effectively a matrix transposition, accomplished by connecting the output wire $Y_{1,j,i}$ to the input wire $X_{2,i,j}$ for $0 \leq i, j < \sqrt{n}$. Each stage 2 chip then corresponds to a row of the matrix, so the stage 2 chips fully sort the valid bits in each row. The input and output wires $X_{2,i,j}$ and $Y_{2,i,j}$ represent the value of the matrix element at row i and column j before and after sorting.

The wiring between stages 2 and 3 is the composition of two matrix permutations. We first cyclically rotate row i by $rev(i)$ places to the right, for $0 \leq i < \sqrt{n}$. That is, the matrix element in row i and column j, for $0 \leq i, j < \sqrt{n}$, is moved to row i and column $(rev(i) + j) \bmod \sqrt{n}$. The matrix is then transposed. Each stage 3 chip then corresponds to a column of the matrix, so the stage 3 chips fully sort the valid bits in each column. The two permutations are accomplished in one wiring step by connecting the output wire $Y_{2,i,j}$ to the input wire $X_{3,(rev(i)+j) \bmod \sqrt{n},i}$, for $0 \leq i, j < \sqrt{n}$.

The output wires of the partial concentrator switch are the first m output wires of the matrix in row-major order, or $Y_{3,j,i}$ for $0 \leq i < \lfloor m/\sqrt{n} \rfloor$ and $0 \leq j < \sqrt{n}$ or $i = \lfloor m/\sqrt{n} \rfloor$ and $0 \leq j < m \bmod \sqrt{n}$.

Like the hyperconcentrator chips from which it is built, the partial concentrator switch is a combinational circuit. The routing paths are established by the valid bits during setup, and subsequent bits follow along these paths.

To see that this construction does indeed yield an $(n, m, 1 - O(n^{3/4}/m))$ partial concentrator switch, we first observe that its operation is equivalent to the following algorithm, which corresponds to the first $1\frac{1}{2}$ iterations of Revsort:

Algorithm 1 Given a $\sqrt{n} \times \sqrt{n}$ matrix with $\sqrt{n} = 2^q$ and matrix element values of 0 or 1, perform the following four steps:

1. Fully sort the columns.
2. Fully sort the rows.
3. For $0 \leq i < \sqrt{n}$, cyclically rotate row i by $rev(i)$ places to the right, i.e., move the element in column j to column $(rev(i) + j) \bmod \sqrt{n}$.
4. Fully sort the columns.

The three sorting steps correspond to the three stages of hyperconcentrator chips in the partial concentrator switch construction. The wiring between stages 1 and 2 corresponds to changing from sorting columns to sorting rows. The wiring between stages 2 and 3 corresponds to the cyclic rotations within rows and changing from sorting rows to sorting columns. We are now ready to prove that this construction works.

Theorem 3 *The Revsort-based construction yields an $(n, m, 1 - O(n^{3/4}/m))$ partial concentrator switch.*

Proof Both [1] and [7] show that after running Algorithm 1 on a $\sqrt{n} \times \sqrt{n}$ matrix with elements valued 0 or 1, the matrix consists of only clean rows of 1's at the top, clean rows of 0's at the bottom, and at most $2\lceil n^{1/4} \rceil - 1$ dirty rows in the middle. Since each row contains $\sqrt{n}$ elements, there are at most $O(n^{3/4})$ dirty bits. By Lemma 1, the sequence is $O(n^{3/4})$-nearsorted, and by Lemma 2, the circuit is an $(n, m, 1 - O(n^{3/4}/m))$ partial concentrator switch. □

Figure 3: A two-dimensional layout of the Revsort-based partial concentrator switch with $n = 64$ inputs and $m = 28$ outputs. The electrical paths established by 24 valid messages are shown with heavy lines. The output wires are the top four output wires of hyperconcentrator chips $H_{3,0}, H_{3,1}, H_{3,2}, H_{3,3}$ and the top three output wires of hyperconcentrator chips $H_{3,4}, H_{3,5}, H_{3,6}, H_{3,7}$.

Figure 3 shows a two-dimensional layout of the switch using $3\sqrt{n}$ hyperconcentrator chips, with $2\sqrt{n}$ data pins each. We simply use crossbar wiring to permute the wires between hyperconcentrator chips of consecutive stages. The area of this layout is $\Theta(n^2)$ since the crossbar wiring area is $\Theta(n^2)$, which dominates the total chip area of $\Theta(n^{3/2})$. (Each stage of $\sqrt{n}$-by-$\sqrt{n}$ hyperconcentrator chips consists of $\sqrt{n}$ chips, each with area $\Theta(n)$, for a total chip area of $\Theta(n^{3/2})$.)

A signal incurs $2\lceil \lg \sqrt{n} \rceil + O(1)$ gate delays in passing through each chip. The $2\lceil \lg \sqrt{n} \rceil$ gate delays are from the hyperconcentrator switch within the chip. The I/O pad circuitry accounts for the additional $O(1)$ delay. The total number of gate delays incurred by a signal passing through the entire partial concentrator switch is thus

$$\begin{aligned} 6\lceil \lg \sqrt{n} \rceil + O(1) &\leq 6 \lg \sqrt{n} + O(1) \\ &= 3 \lg n + O(1) . \end{aligned}$$

As shown in Figure 4, we can package the partial concentrator switch in three dimensions using volume

Figure 4: The three-dimensional packaging of the Revsort-based partial concentrator switch for $n = 64$. Each stack contains $\sqrt{n}$ circuit boards and corresponds to one stage. Each board contains one $\sqrt{n}$-by-$\sqrt{n}$ hyperconcentrator chip, and boards in stack 2 follow the hyperconcentrator chip by a $\sqrt{n}$-bit barrel shifter chip to perform the cyclic rotation of each row. The $\lg \sqrt{n}$ control bits that determine the shift amount for each barrel shifter are hardwired.

$\Theta(n^{3/2})$. Each circuit board contains one $\sqrt{n}$-by-$\sqrt{n}$ hyperconcentrator chip, corresponding to one row or column of the matrix. Each of the three stacks contains $\sqrt{n}$ boards and represents one stage. The wires cross stack junctions in a $\sqrt{n} \times \sqrt{n}$ array, with the valid bit value of the wire in row i and column j equal to the value of the matrix element in the same position at the corresponding step of Algorithm 1.

The matrix transpose between stages 1 and 2 is performed in the natural way, with the ith output wire from board j in stage 1 going straight across the junction to be the jth input wire of board i in stage 2. The wiring permutation between the hyperconcentrator chips of stages 2 and 3 includes the cyclic rotations of the rows, followed by the transpose. The transpose is performed in the natural way once again. We perform the cyclic rotation by following each stage 2 hyperconcentrator chip by a $\sqrt{n}$-bit barrel shifter on the same board. The barrel shifter has $\sqrt{n}$ input wires, $\sqrt{n}$ output wires, and $\lceil \lg \sqrt{n} \rceil$ control bits which, interpreted as a binary integer, determine the rotation amount. We hardwire the control bits in the ith board to have the value $rev(i)$.

We use only two board types, $3\sqrt{n}$ hyperconcentrator chips, and $\sqrt{n}$ barrel shifters in building the switch. All $2\sqrt{n}$ boards in stages 1 and 3 are identical, as are all $\sqrt{n}$ stage 2 boards. The barrel shifters require $2\sqrt{n} + \lceil \lg \sqrt{n} \rceil = 2\sqrt{n} + \lceil (\lg n)/2 \rceil$ data pins. The hardwiring of the barrel shifter control bit values can be performed after the boards have been fabricated.

To see that the volume is $\Theta(n^{3/2})$, we need only consider the stage 2 stack, which has the most components. Each board contains a $\sqrt{n}$-by-$\sqrt{n}$ hyperconcentrator chip and a $\sqrt{n}$-bit barrel shifter, both having area $\Theta(n)$. The whole stack of $\sqrt{n}$ boards, and therefore the entire switch, has volume $\Theta(n^{3/2})$.

Since the barrel shift amounts are hardwired and never change, the barrel shifters introduce only a constant number of gate delays. A signal therefore incurs $3 \lg n + O(1)$ gate delays in passing through the three-dimensional switch.

Letting p, the number of pins per chip, be $\Theta(\sqrt{n})$, both the two-dimensional and three-dimensional layouts use only $\Theta(n/p)$ chips.

5 A Columnsort-Based Partial Concentrator Switch

In this section, we present a design for an (n, m, α) partial concentrator switch that uses $\Theta(n^{1-\beta})$ chips with $\Theta(n^{\beta})$ pins each, where $1/2 \le \beta \le 1$. The basic building block is a $\Theta(n^{\beta})$-by-$\Theta(n^{\beta})$ hyperconcentrator chip. Each message incurs $4\beta \lg n + O(1)$ gate delays in passing through the switch. The load ratio is $\alpha = 1 - O(n^{2-2\beta}/m)$. This switch can be implemented in two dimensions with area $O(n^2)$ or in three dimensions with volume $\Theta(n^{1+\beta})$. Table 1 shows resource measures for the Revsort-based switch and the values of β at which the switch of this section matches them asymptotically.

The design is based on Leighton's Columnsort algorithm [3] for sorting n elements on an $r \times s$ mesh, where $n = rs$ and s evenly divides r. The idea behind this partial concentrator switch is to $(s-1)^2$-nearsort an $r \times s$ matrix of valid bits. As with the switch of the previous section, the m output wires of the switch correspond to the first m matrix entries.

We may identify a matrix entry by either its row and column position or by its position in row-major or column-major order. All numbering starts at 0. Thus, the rows are numbered $0, 1, \ldots, r-1$ and the columns are numbered $0, 1, \ldots, s-1$. The row-major position of the matrix entry in row i and column j is $RM(i,j) = si + j$, and its column-major position is $CM(i,j) = rj + i$. For example, Figure 5 shows the row-major and column-major positions of a 6×3 matrix. We have that $0 \le RM(i,j), CM(i,j) < n$. The row and column position corresponding to the entry in row-major position x is $RM^{-1}(x) = (\lfloor x/s \rfloor, x \bmod s)$.

The partial concentrator switch is built from two stages, each stage containing s hyperconcentrator chips.

	Revsort	Columnsort, $\beta = 1/2$	Columnsort, $\beta = 5/8$	Columnsort, $\beta = 3/4$
pins per chip	$\Theta(n^{1/2})$	$\Theta(n^{1/2})$	$\Theta(n^{5/8})$	$\Theta(n^{3/4})$
chip count	$\Theta(n^{1/2})$	$\Theta(n^{1/2})$	$\Theta(n^{3/8})$	$\Theta(n^{1/4})$
load ratio	$1 - O(n^{3/4}/m)$	$1 - O(n/m)$	$1 - O(n^{3/4}/m)$	$1 - O(n^{1/4}/m)$
gate delays	$3 \lg n + O(1)$	$2 \lg n + O(1)$	$\frac{5}{2} \lg n + O(1)$	$3 \lg n + O(1)$
volume	$\Theta(n^{3/2})$	$\Theta(n^{3/2})$	$\Theta(n^{13/8})$	$\Theta(n^{7/4})$

Table 1: Resource measures for the Revsort-based partial concentrator switch and the values of β at which the Columnsort-based switch matches them asymptotically.

$$\underset{\text{row-major}}{\begin{bmatrix} 0 & 1 & 2 \\ 3 & 4 & 5 \\ 6 & 7 & 8 \\ 9 & 10 & 11 \\ 12 & 13 & 14 \\ 15 & 16 & 17 \end{bmatrix}} \qquad \underset{\text{column-major}}{\begin{bmatrix} 0 & 6 & 12 \\ 1 & 7 & 13 \\ 2 & 8 & 14 \\ 3 & 9 & 15 \\ 4 & 10 & 16 \\ 5 & 11 & 17 \end{bmatrix}}$$

Figure 5: Row-major and column-major positions of elements in a 6×3 matrix.

Since the hyperconcentrator chips are combinational, so is the partial concentrator switch. Each r-by-r hyperconcentrator chip corresponds to a column of the underlying matrix, fully sorting the column. We shall denote by $H_{l,j}$ the jth hyperconcentrator chip in stage l, for $l = 1, 2$ and $0 \le j < s$, with input wires $X_{l,j,0}, X_{l,j,1}, \ldots, X_{l,j,r-1}$ and output wires $Y_{l,j,0}, Y_{l,j,1}, \ldots, Y_{l,j,r-1}$. Wires $X_{l,j,i}$ and $Y_{l,j,i}$ correspond to the matrix element in row i and column j.

The wiring between stages 1 and 2 corresponds to converting the matrix from column-major to row-major ordering, using the composition of functions $RM^{-1} \circ CM$. We connect the output wire $Y_{1,j,i}$ to the input wire $X_{2,(rj+i) \bmod s, \lfloor (rj+i)/s \rfloor}$, for $0 \le i < r$ and $0 \le j < s$.

Once again, the output wires of the partial concentrator switch are the first m output wires of the matrix in row-major order. We use wires $Y_{2,j,i}$ for $0 \le i < \lfloor m/s \rfloor$ and $0 \le j < s$ or $i = \lfloor m/s \rfloor$ and $0 \le j < m \bmod s$.

To show that this circuit $(s-1)^2$-nearsorts the valid bits, we first observe that its operation is equivalent to the following algorithm, which corresponds to the first three steps of Columnsort:

Algorithm 2 Given an $r \times s$ matrix of n elements, where $n = rs$, and matrix values of 0 or 1, perform the following three steps:

1. Fully sort the columns.
2. Convert the matrix from column-major to row-major order, i.e., move the element in row i and column j to row $\lfloor (rj+i)/s \rfloor$ and column $(rj+i) \bmod s$.
3. Fully sort the columns.

The two stages of hyperconcentrator chips correspond to steps 1 and 3, and the wiring between the stages corresponds to step 2. This correspondence between the circuit and Columnsort allows us to prove the following theorem.

Theorem 4 *The Columnsort-based construction yields an $(n, m, 1-(s-1)^2/m)$ partial concentrator switch.*

Proof Leighton shows in [3] that Algorithm 2 is an $(s-1)^2$-nearsorter when the matrix elements are taken in row-major order. By Lemma 2, the circuit is an $(n, m, 1-(s-1)^2/m)$ partial concentrator switch when the outputs are taken in row-major order. □

To achieve the results stated at the beginning of this section, we let $r = \Theta(n^{\beta})$ and $s = \Theta(n^{1-\beta})$. To ensure that $n = rs$ and that s divides r as n increases, we require that we have $1/2 \le \beta \le 1$. The load ratio is then

$$\begin{aligned} \alpha &= 1 - \frac{(s-1)^2}{m} \\ &= 1 - \Theta\left(\frac{n^{2-2\beta}}{m}\right) . \end{aligned}$$

The number of chips is $2s = \Theta(n^{1-\beta})$, and each chip requires $2r = \Theta(n^{\beta})$ data pins.

The delay through the switch is $2 \cdot 2 \lg r + O(1) = 4 \lg r + O(1)$. Letting $r \le cn^{\beta} + o(n^{\beta})$ for some constant c, we have that the delay is

$$\begin{aligned} 4 \lg r + O(1) &\le 4 \lg(cn^{\beta} + o(n^{\beta})) + O(1) \\ &\le 4 \lg((c+1)n^{\beta}) \quad \text{(for suff. large } n\text{)} \\ &= 4\beta \lg n + 4\beta \lg(c+1) \\ &= 4\beta \lg n + O(1) . \end{aligned}$$

A two-dimensional layout using $O(n^2)$ area is shown in Figure 6. As in the Revsort-based switch, we use $n \times n$ crossbar wiring to connect the stages.

Figure 7 shows a three-dimensional packaging of the switch using volume $\Theta(r^2 s) = \Theta(n^{1+\beta})$. As in Figure 6, we have $r = 8$ and $s = 4$. There are two stacks of boards, with each stack containing s boards and corresponding to one stage of hyperconcentrator chips, and each board containing one r-by-r hyperconcentrator chip.

The tricky part of this construction is the wiring between stages, which must perform the permutation $RM^{-1} \circ CM$. On the first stack, we group together output wires whose column-major numberings are congruent modulo s, or equivalently, those whose row numbers are congruent modulo s. Each such group contains r/s wires. In Figure 7, for example, since we have $s = 4$, we group together wires $H_{1,0,0}$ and $H_{1,0,4}$, $H_{1,0,1}$ and $H_{1,0,5}$, $H_{1,0,2}$ and $H_{1,0,6}$, $H_{1,0,3}$ and $H_{1,0,7}$, etc. In order to allow them to enter the stage 2 chips, these wires are then "transposed" in small interstack connectors to align them horizontally

Figure 6: A two-dimensional layout of the Columnsort-based partial concentrator switch with $n = 32$ inputs and $m = 18$ outputs. The underlying matrix is 8×4. The electrical paths established by 14 valid messages are shown with heavy lines. The output wires are the first five output wires of hyperconcentrator chips $H_{2,0}$ and $H_{2,1}$ and the first four output wires of hyperconcentrator chips $H_{2,2}$ and $H_{2,3}$.

Figure 7: The three-dimensional packaging of the Columnsort-based partial concentrator switch for $r = 8$ and $s = 4$. Each stack contains s chips, each of which is an r-by-r hyperconcentrator. The wiring between the stages of chips performs the $RM^{-1} \circ CM$ permutation. The interstack connectors transpose the wires from vertical to horizontal alignment.

Figure 8: The transposition of w wires from vertical to horizontal alignment, shown for $w = 4$, using volume $\Theta(w^2)$.

instead of vertically. Figure 8 shows one way to transpose a group of r/s wires in volume $\Theta((r/s)^2)$.

The first stack dominates the volume of this construction. We have s boards, and each board contains a $\Theta(r^2)$-area hyperconcentrator chip and an $O(r^2)$-area wiring permutation. The total volume of each stack is thus $\Theta(r^2 s) = \Theta(n^{1+\beta})$. There are s^2 interstack connectors, each with volume $O((r/s)^2)$, for a total interstack volume of $O(r^2) = O(n^{2\beta})$. Since we have $\beta \leq 1$, the total interstack volume is $O(n^{1+\beta})$. The total volume of the partial concentrator switch is thus $\Theta(n^{1+\beta})$.

For both the two-dimensional and three-dimensional layouts, letting p, the number of pins per chip, be $\Theta(r)$, we use only $\Theta(s) = \Theta(n/p)$ chips. The three-dimensional layout, however, uses $s^2 = \Theta((n/p)^2)$ interstack connectors, but these connectors contain only wiring and no active components.

6 Concluding Remarks

In this section, we briefly discuss the characteristics of the partial concentrator switches we have seen and then discuss multichip hyperconcentrator switches. Finally, we pose some open questions.

Both of the partial concentrator switches we have examined are efficient in that they are relatively fast and can be packaged with a relatively low volume. They also allow air to flow through in all three dimensions and may thus be air-cooled.

The β parameter of the Columnsort-based switch defines a tradeoff continuum for the characteristics of the switch. As evidenced by Table 1, as the value of β increases, so do the number of pins per chip, delay, and volume, but the load ratio improves and the number of chips decreases.

Rather than simulating just the first steps of Revsort and Columnsort, one could simulate the full algorithms to fully sort the valid bits and thus build multichip hyperconcentrator switches. Compared to the partial concentrator switches presented above, such hyperconcentrator switches have increased delay, and a Revsort-based hyperconcentrator switch has a greater chip count and asymptotic volume than its partial concentrator counterpart.

Schnorr and Shamir show in [7] that if steps 1–3 of Algorithm 1 are repeated $\lceil \lg\lg\sqrt{n} \rceil$ times, the resulting matrix contains at most eight dirty rows. We can then complete the full sorting by running three iterations of the Shearsort algorithm [6]. An n-by-n hyperconcentrator switch based on the full Revsort algorithm consists of $\lceil \lg\lg\sqrt{n} \rceil$ repetitions of stacks 1 and 2 of Figure 4 followed by three pairs of different stacks that simulate Shearsort. (Each Shearsort stack consists of $\sqrt{n}$ boards, each of which contains a $\sqrt{n}$-by-$\sqrt{n}$ hyperconcentrator chip and fixed permutation wiring.) A signal passes through $2\lg\lg n + 4$ hyperconcentrator chips in such an n-by-n hyperconcentrator switch, incurring $4\lg n \lg\lg n + 8\lg n + O(\lg\lg n)$ gate delays. The switch uses a total of $\Theta(\sqrt{n}\lg\lg n)$ chips in volume $\Theta(n^{3/2}\lg\lg n)$.

Similarly, by simulating all eight steps of Columnsort, we can build a hyperconcentrator switch with the same asymptotic volume and chip count as the partial concentrator switch of Section 5. A signal passes through four chips and incurs $8\beta \lg n + O(1)$ gate delays through such an n-by-n hyperconcentrator switch.

Rather than wondering how fast a multichip hyperconcentrator switch we can build, we might ask for what functions $f(p)$ can we build an $(\Omega(f(p)), m, 1 - o(p/m))$ partial concentrator switch, given chips with p pins and using only two stages of chips. The Columnsort-based construction, for example, gives us $f(p) = p^{2-\varepsilon}$ for any $0 < \varepsilon \leq 1$. Can we achieve $f(p) = \Omega(p^2)$? In general, how large a function $f(p)$ can we achieve with k stages?

There may be ε-nearsorters based on networks other than the two-dimensional mesh to which we can apply Lemma 2. What types of partial concentrator switches can we build by applying Lemma 2 to other ε-nearsorters?

Acknowledgements

Thanks to Charles Leiserson for suggesting this line of research and for his frequent help and guidance. Thanks also to James Park for his helpful comments.

References

[1] T. H. Cormen, "Concentrator switches for routing messages in parallel computers," Masters thesis, Dept. of Electrical Engineering and Computer Science, M.I.T., Cambridge, Mass., (1986), 66pp.

[2] T. H. Cormen and C. E. Leiserson, "A hyperconcentrator switch for routing bit-serial messages," *Proceedings of the 15th Annual International Conference on Parallel Processing*, (Aug. 1986), pp. 721–728.

[3] F. T. Leighton, "Tight bounds on the complexity of parallel sorting," *IEEE Transactions on Computers*, Vol. C-34, No. 4, (Apr. 1985), pp. 344–354.

[4] M. S. Pinsker, "On the complexity of a concentrator," *Proceedings of the 7th International Teletraffic Conference*, Stockholm, (1973), pp. 318/1–318/4.

[5] N. Pippenger, "Superconcentrators," *SIAM Journal on Computing*, Vol. 6, No. 2, (June 1977), pp. 298–304.

[6] I. D. Scherson, S. Sen, and A. Shamir, "Shear sort: a true two-dimensional sorting technique for VLSI networks," *Proceedings of the 15th Annual International Conference on Parallel Processing*, (Aug. 1986), pp. 903–908.

[7] C. P. Schnorr and A. Shamir, "An optimal sorting algorithm for mesh connected computers," *Proceedings of the 18th Annual ACM Symposium on Theory of Computing*, (May 1986), pp. 255–263.

[8] L. G. Valiant, "Graph-theoretic properties in computational complexity," *JCSS*, Vol. 13, No. 3, (Dec. 1976), pp. 278–285.