12-1993

Parallel Algorithms for Single-Layer Channel Routing

Ronald I. Greenberg
Rgreen@luc.edu

Shih-Chuan Hung

Jau-Der Shih

Recommended Citation

Ronald I. Greenberg, Shih-Chuan Hung, and Jau-Der Shih. Parallel algorithms for single-layer channel routing. In Algorithms and Computation: 4th International Symposium, ISAAC '93 Proceedings, volume 762 of Lecture Notes in Computer Science, pages 456--465. Springer-Verlag, 1993.

Proceedings of ISAAC '93, pp. 456-465

Parallel Algorithms for Single-Layer Channel Routing*

Ronald I. Greenberg, Shih-Chuan Hung, and Jau-Der Shih

Electrical Engineering Department
University of Maryland
College Park, MD 20742

Abstract. We provide efficient parallel algorithms for the minimum separation, offset range, and optimal offset problems for single-layer channel routing. We consider all the variations of these problems that have linear-time sequential solutions rather than limiting attention to the "river-routing" context, where single-sided connections are disallowed. For the minimum separation problem, we obtain $O(\lg N)$ time on a CREW PRAM or $O(\frac{\lg N}{\lg \lg N})$ time on a CRCW PRAM, both with optimal work (processor-time product) of $O(N)$, where N is the number of terminals. For the offset range problem, we obtain the same time and processor bounds as long as only one side of the channel contains single-sided nets. For the optimal offset problem with single-sided nets on one side of the channel, we obtain time $O(\lg N \lg \lg N)$ on a CREW PRAM or $O(\lg N)$ time on a CRCW PRAM with $O(N \lg \lg N)$ work. Not only does this improve on previous results for river routing, but we can obtain an even better time of $O((\lg \lg N)^2)$ on the CRCW PRAM in the river routing context.

1 Introduction

Much attention has been given to single-layer routing for VLSI. Most popular has been river routing [6], the connection of two (horizontal) rows of corresponding terminals using the channel region between the rows of terminals; see also [16] and the references therein. More general arrangements of modules and nets have been considered for testing routability of terminals in fixed positions, but it is also desirable to answer more sophisticated questions. For example, the *minimum separation* problem involves finding the minimum vertical separation between two rows of terminals that is required for routability (given that the horizontal positions of the terminals are completely fixed). In other problems, we are allowed to offset the upper row of terminals as a block to the left or the right, though the individual terminals do not shift position relative to one another. In particular, the *optimal offset problem* involves finding the offset that minimizes the amount of separation necessary to route the channel. The *offset range problem* involves finding all offsets that give enough room to route at a given separation.

* Supported in part by NSF grant CCR-9109550.

We consider these problems in all contexts for which linear-time sequential algorithms are known instead of considering only river routing, where each net is restricted to have exactly one terminal on each side of the channel. The input we assume is two arrays of terminals sorted by x-coordinate. The top terminals (and, in arithmetic contexts, their x-coordinates) are denoted $t_0, t_1, \ldots, t_{m-1}$, and the bottom terminals are denoted $b_0, b_1, \ldots, b_{n-1}$. We use N to denote $m+n$. Associated with each terminal is a net number. Terminals belonging to the same net are to be connected together. We also assume that each terminal has a pointer to the next terminal of the same net in a clockwise ordering of the terminals. (It may be possible to eliminate this assumption under some circumstances, e.g., constant number of terminals per net, allowing randomization, and/or allowing a modest increase in time or work, by applying results on sorting of small integers (e.g, [10, 15]).) For simplicity, we use a rectilinear, grid-based model in which terminals lie on gridpoints and wires are disjoint paths through grid edges. Also, for convenience, we allow routing on channel boundaries.

We henceforth assume that each net has two terminals. Multiterminal nets can be handled by a transformation described in [7], that might be considered "folklore". Then a *single-sided net* has its two terminals on the same side of the channel, whereas a *two-sided net* is the type of net allowed in river routing.

We also assume henceforth that the channel is routable in one layer, i.e., no two nets are topologically forced to cross. (This condition can be verified without increasing the running time of our parallel algorithm by doing parentheses matching [2].)

The results obtained in this paper are summarized in Table 1, where the river routing model is as described above, the general model includes any single-layer channel routing problem, and the intermediate model is one in which all single-sided nets are on one side of the channel.

Table 1. Running time and work (processor-time product) for the algorithms presented in this paper.

		CREW		CRCW	
problem	model	time	work	time	work
min. sep.	general	$O(\lg N)$	$O(N)$	$O(\lg N/\lg\lg N)$	$O(N)$
offset range	intermediate	$O(\lg N)$	$O(N)$	$O(\lg N/\lg\lg N)$	$O(N)$
optimal offset	intermediate	$O(\lg N \lg\lg N)$	$O(N \lg\lg N)$	$O(\lg N)$	$O(N \lg\lg N)$
optimal offset	river	$O(\lg N \lg\lg N)$	$O(N \lg\lg N)$	$O((\lg\lg N)^2)$	$O(N \lg\lg N)$

Most prior work on single-layer routing has been limited to sequential models of computation; linear-time sequential algorithms for the problems considered in this paper can be found in [7], [9], and [16]. For the river routing model only, parallel algorithms for the problems in this paper are given by Aggarwal and

Park [1]. For optimal offset, their results are $O(\lg^2 N)$ time on the CREW and $O(\lg N \lg\lg N)$ time on the CRCW, both with $O(N \lg N)$ work. We obtain superior bounds even for the more general problem labeled as the "intermediate" model in Table 1. We improve these bounds even further for river routing on the CRCW by using the offset range results of Aggarwal and Park quoted in Sect. 3.3. Chang, JáJá, and Ryu [4], independently obtain the same bounds as Aggarwal and Park for minimum separation in the river routing model (matching our CREW bounds for the general model and improving the CRCW bounds of the general model) and also give an optimal (work) algorithm for routability testing for switchboxes.

The remainder of this paper is organized as follows. In Sect. 2, we explain the parallel operations used in this paper. We also indicate how to conveniently express the routability conditions for *single-layer channel routing*. These conditions are then used in Sect. 3 to solve the minimum separation, offset range and optimal offset problems. Section 4 gives some concluding remarks.

2 Preliminaries

2.1 Basic Parallel Operations

Given a sequence of N elements $\{x_1, x_2, \ldots, x_N\}$ with a binary associative operator $*$, the *prefix sums* are all the partial sums defined by:

$$p_i = x_1 * x_2 * \ldots * x_i, 1 \leq i \leq N$$

For a given array $A(i)$ with $1 \leq i \leq N$, the *range maxima problem* is to find the element with maximum value between two given positions i and j. A query can be answered in $O(1)$ time after the preprocessing described in [11].

The range maxima preprocessing can be implemented in $O(\lg N)$ (respectively $O(\frac{\lg N}{\lg\lg N})$) time with $O(N)$ work on a CREW (CRCW) PRAM [3]. The prefix sums computation can be performed with the same time and processor bounds on the CREW [13], and on the CRCW as long as the input elements are integers in the interval $[1, N]$ [5].

2.2 Cut Conditions

We need a few definitions in order to use a general theory of single-layer routing developed by Maley. Define a *critical cut* to be a line segment that connects a top and bottom terminal or runs from a terminal straight across to the opposite side of the channel. Define a *pivotal cut* to be a line segment that connects a top and bottom terminal or runs at 45° from a terminal to the opposite side of the channel. Also let the *flow* across a cut χ be the number of nets that must cross χ, namely those nets having terminals on both sides of χ and those having an endpoint of χ as a terminal. The *capacity* of χ is one greater than the maximum

of the horizontal and vertical separations of its endpoints; if χ is the line segment from (x_1, y_1) to (x_2, y_2), then

$$capacity(\chi) = max\{|x_1 - x_2|, |y_1 - y_2|\} + 1 \ .$$

The cut χ is *safe* if $flow(\chi) \leq capacity(\chi)$, which means that there is enough space along χ for the wires to get through.

Lemma 1. *A channel is routable if and only if every critical cut or every pivotal cut is safe.* □

This lemma follows from the corresponding results in [14, §2.1,2.3,2.6.5]. (Our slightly different definitions of flow and capacity allow Maley's formulation in terms of cuts emanating from "feature" endpoints to correspond to cuts emanating from terminals. Since we allow routing on the channel boundaries, the only "features" are terminals and two routing obstacles (horizontal lines) located one unit outside of what we have been referring to as the channel boundaries.)

We can further strengthen the result for critical cuts as follows. Define the *span* of a cut χ to be the horizontal distance between its endpoints. Call χ *sparse* if χ is not vertical and $flow(\chi) \leq span(\chi) + 1$, and *dense* otherwise. A sparse cut is safe regardless of the separation, but a dense cut χ is safe if and only if the separation is at least $flow(\chi) - 1$.

Lemma 2. *The minimum channel separation is the maximum of* $flow(\chi) - 1$ *over dense critical cuts* χ. □

When all single-sided nets are on the bottom, we can strengthen the result for pivotal cuts, but first we must review results regarding *contours* of single-sided nets. Define the *contour* of single-sided nets on the bottom to be the upper boundary of the routing region consumed in the routing of these nets that minimizes total wire length. That is, when the nets are routed as tightly as possible against the boundary of the channel, the contour is formed by the uppermost nets and portions of the channel boundary.

The following Lemma from [4] shows that a contour of single-sided nets can be found efficiently.

Lemma 3. *The bendpoints in the contour of a set of* N *single-sided nets can be found in* $O(\lg N)$ *(respectively* $O(\frac{\lg N}{\lg \lg N})$*) time using* $O(\frac{N}{\lg N})$ *(respectively* $O(\frac{N \lg \lg N}{\lg N})$*) processors on a CREW (CRCW) PRAM.* □

Now, we are ready to state a result of [9] relating to pivotal cuts:

Lemma 4. *A channel with all single-sided nets on the bottom is routable if and only if all* 45° *cuts from bottom terminals of two-sided nets and all* 45° *cuts crossing the contour of single-sided nets at a convex corner are safe.* □

3 The Algorithms

3.1 The Minimum Separation Problem

Our algorithm for this problem is based on Lemma 2. To ensure that vertical cuts are captured, first add a dummy terminal across from each real terminal. Then we find the minimum separation that makes all dense (critical) cuts emanating from bottom terminals safe. To find all dense cuts emanating from b_j, we search for the two farthest dense cuts, one going to the right and one going to the left from b_j; these two cuts form a "cone" such that cuts emanating from b_j are dense if and only if they lie inside the cone.

We now provide some further definitions and notations used in this subsection. First, we say that a terminal is covered by a single-sided net on its side of the channel if it lies in the closed interval defined by the endpoints of the net. Two-sided nets are said to lie to the left or right of a terminal on the top (or bottom) according to the location of the net's top (respectively bottom) terminal. Also, a two-sided net is a right net if its top terminal is to the right of its bottom terminal; it is a left net if its top terminal is to the left of its bottom terminal. Define $R(\tau)$ to be the number of right nets to the left of terminal τ, and $L(\tau)$ to be the number of left nets to the left of τ. Also define $S(\tau)$ to be the number of single-sided nets covering τ. Define $\mathrm{IL}(\tau)$ (and $\mathrm{IR}(\tau)$) to be 1 if τ is a terminal of a left (respectively right) net, and zero otherwise.

The heart of this algorithm is to form the cone for each terminal b_j. According to the definition, a nonvertical cut $\overline{t_i b_j}$ is dense if $|t_i - b_j| + 1 < flow(\overline{t_i b_j})$. We now show how to find the farthest cuts emanating from each terminal b_j on the bottom that are dense. Note that for any dense cut $\overline{t_i b_j}$, $R(t_i) \leq R(b_j)$, and $L(t_i) \geq L(b_j)$. Thus, for dense cuts, $flow(\overline{t_i b_j}) = L(t_i) - L(b_j) + \mathrm{IL}(t_i) + R(b_j) - R(t_i) + \mathrm{IR}(b_j) + S(t_i) + S(b_j)$. Defining, $\mathrm{bl}(j)$ to be $b_j + L(b_j) - R(b_j) - S(b_j) - \mathrm{IR}(b_j) + 1$ and $\mathrm{tl}(i)$ to be $t_i + L(t_i) - R(t_i) + S(t_i) + \mathrm{IL}(t_i)$, we need to find the smallest t_i such that $\mathrm{bl}(j) < \mathrm{tl}(i)$; for similar definitions of $\mathrm{tr}(i)$ and $\mathrm{br}(j)$, we also find the largest t_i such that $\mathrm{tr}(i) < \mathrm{br}(j)$. It can be shown that the four functions bl, tl, br, and tr are non-decreasing. Now, we can give an algorithm for forming the cone for each bottom terminal.

procedure FIND-CONES

1. Compute $\mathrm{tl}(i), \mathrm{bl}(j), \mathrm{tr}(i)$ and $\mathrm{br}(j)$ for $0 \leq i \leq m-1$ and $0 \leq j \leq n-1$.
2. Merge $\mathrm{tl}(i)$ with $\mathrm{bl}(j)$ and $\mathrm{tr}(i)$ with $\mathrm{br}(j)$ in order of nondecreasing values. If a tie occurs, put $\mathrm{br}(j)$ before $\mathrm{tr}(i)$ and put $\mathrm{bl}(j)$ after $\mathrm{tl}(i)$.
3. For each $\mathrm{bl}(j)$, find the nearest $\mathrm{tl}(i)$ to the right in the merged sequence. If we do not find such a $\mathrm{tl}(i)$ corresponding to a t_i with lesser x-coordinate than b_j, then the farthest dense cut to the left from b_j is vertical. Similarly, for each $\mathrm{br}(j)$, find the nearest $\mathrm{tr}(i)$ to the left in the merged sequence, and select a vertical cut if necessary.

The merging can be done using the approach of Kruskal [12] in $O(\lg \lg N)$ time with $O(N)$ work on a CREW PRAM. Also steps 1 and 3 can be implemented by using prefix-sums. Therefore, algorithm FIND-CONES can be implemented with optimal work in $O(\lg N)$ (respectively $O(\frac{\lg N}{\lg \lg N})$) time on a CREW (CRCW) PRAM.

Once we find the cone for each bottom terminal, we can use the information to find the minimum separation for the single-layer channel routing problem:

procedure MINIMUM-SEPARATION

1. Apply algorithm FIND-CONES to find the farthest dense cuts to form a cone for every terminal on the bottom.
2. Find the maximum flow $F(b_j)$ among the cuts inside the cone for every terminal b_j.
3. The minimum separation is $-1 + \max\{F(b_1), F(b_2), \ldots, F(b_n)\}$.

Theorem 5. *Algorithm* MINIMUM-SEPARATION *finds minimum separation for single-layer channel routing with $O(N)$ work in time $O(\lg N)$ on a CREW PRAM and in time $O(\frac{\lg N}{\lg \lg N})$ on a CRCW PRAM.*

Proof. We have already explained how step 1 can be performed within the specified time and processor bounds, and step 3 simply involves a minimum that can be computed in the same bounds. To find the maximum flow in each cone in step 2, we use the range maxima technique. Since, the flow for a dense cut $\overline{t_i b_j}$ is $L(t_i) - L(b_j) + \mathrm{IL}(t_i) + R(b_j) - R(t_i) + \mathrm{IR}(b_j) + S(t_i) + S(b_j)$, and the terms dependent on j are fixed for any given cone, the task is to find the maximum of $\mathrm{tl}(i) - t_i$ over each cone. The preprocessing for range maxima and the single query per b_j can also be implemented within the stated bounds. □

3.2 The Offset Range Problem

In this subsection, we consider the offset range problem for single-layer channel routing with single-sided nets on one side. Without loss of generality, assume that all single-sided nets are on the bottom. Additional notation used in this subsection is as follows. Let s be the separation and d the offset (the positive or negative distance by which the upper block of terminals is moved right from its original position) . Define $T(\tau)$ to be the number of two-sided nets to the left of the terminal τ. Also, define $\mathrm{IT}(\tau)$ to be one if τ belongs to a two-sided net, and zero otherwise.

According to Lemma 4, we only need to ensure that all 45° cuts from bottom terminals of two-sided nets and all 45° cuts crossing the contour of single-sided nets at a convex corner are safe. In this subsection, we achieve that task by checking all 45° cuts from bottom terminals $b_0, b_1, \ldots, b_{n-1}$. For a cut χ emanating from b_j, the flow contributed by single-sided nets is $S(b_j)$; to be safe,

χ can accommodate at most $s + 1 - S(b_j)$ more flow. For simplicity, we denote $s + 1 - S(b_j)$ as e_j. Then, for the left-going 45° cut from b_j to be safe, we need that $b_j - s > t_i + d$ when the number of two-sided nets crossing cut $\overline{t_i b_j}$ exceeds e_j. Finding the i that gives the tightest constraint, we have $b_j - s > t_{T(b_j)-e_j+\mathrm{IT}(b_j)} + d$. A similar argument for the right-going cut yields $b_j + s < t_{T(b_j)+e_j+1} + d$. So the feasible offsets, if any, are given by

$$\max_{0 \le j \le n-1} \{b_j + s - t_{T(b_j)+e_j+1}\} < d < \min_{0 \le j \le n-1} \{b_j - s - t_{T(b_j)-e_j+\mathrm{IT}(b_j)}\} .$$

The most difficult operation involved in the computation just specified is finding the $S(b_j)$ and $T(b_j)$ values, for which performing prefix sums suffices, so we have the following theorem:

Theorem 6. *The offset range for single-layer channel routing with single-sided nets on one side can be found with $O(N)$ work in $O(\lg N)$ (respectively $O(\frac{\lg N}{\lg \lg N})$) time on a CREW (CRCW) PRAM.* □

3.3 The Optimal Offset Problem

This subsection considers the optimal offset problem for river routing and channels with single-sided nets on one side. In both cases, the optimal offset problem is solved by using an algorithm for offset range as a subroutine. For channels with single-sided nets on one side, we use the results of Sect. 3.2 to obtain optimal offset results better than those of Aggarwal and Park, even though their results apply only to river routing. For river routing on the CRCW, we further improve their optimal offset bounds by using their results for offset range, $O(\lg \lg N)$ time and $O(N)$ work. (On the CREW, their offset range results offer no improvement over the results of Sect. 3.2.)

We first explain how to solve the problem for river routing, and then we extend the algorithm to channels with single-sided nets on one side. For ease of presentation, we assume the number of nets N is 2^p for some integer p.

Our algorithm adapts Mirzaian's halving technique [16] for relating optimal offset to offset range. Let T^0 be the original set of two-sided nets, and define T^k to be the set of even-numbered nets of T^{k-1}, for $1 \le k \le p$. Also define optsep(A) to be the minimum separation attainable with an optimal offset for a channel with the set A of nets. (Once optsep(A) is determined, the solution of the offset range problem can be used to determine the optimal offsets.) The following lemma adapted from [16][2] states the relationship between optsep(T^k) and optsep(T^{k+1}):

Lemma 7. *Let $s^k = \mathrm{optsep}(T^k)$ and $s^{k+1} = \mathrm{optsep}(T^{k+1})$, then $0 \le s^k - 2s^{k+1} \le 2$.* □

[2] This lemma differs slightly from [16] because we are allowing routing on both boundaries of the channel.

From the above lemma, once we know s^{k+1}, then s^k can be solved by checking only three possible separations to achieve the optimal offset. Now, s^{k-1} can also be solved by checking all the possible separations derived from the three separations. (Again each possible separation of s^k induces 3 possible separations for s^{k-1}.) By considering the union of these separations, we only have to check 7 possible separations to solve s^{k-1}. Continuing this line of reasoning, we have the following corollary:

Corollary 8. *If s^k is known, then s^{k-l} can be solved by checking only $2^{l+1}-1$ possible separations.* □

Our algorithm finds the optimal separations $s^{p/2^i}$, for $1 \leq i \leq \lg p$. Each separation is determined from previously computed separations by using Corollary 8. Figure 1 shows the algorithm to solve the optimal offset problem.

```
      procedure OPTIMAL-SEPARATION
1     for i ← lg p to 1 do
2         find T^{p/2^i}
3     endfor
4     for i ← 1 to lg p do
5         find s^{p/2^i}
6     endfor
7     find s^0
```

Fig. 1. This algorithm finds the minimum separation for the optimal offset problem.

Theorem 9. *The optimal offset for river routing can be found in $O(\lg N \lg\lg N)$ (respectively $O((\lg\lg N)^2)$) time using $O(\frac{N}{\lg N})$ (respectively $O(\frac{N}{\lg\lg N})$) processors on a CREW (CRCW) PRAM.*

Proof. First we explain the CREW result. Lines 1–3 can be executed in time $O(\lg N)$ time using $O(\frac{N}{\lg N})$ processors. (To see this, it suffices to consider repeated "halving" of the net set, with the time at each stage decreasing geometrically to a constant as the number of nets decreases.) In lines 4–6, each $s^{p/2^i}$ can be found in $O(p) = O(\lg N)$ time as follows. First, $s^{p/2}$ has at most $2^{p/2}$ possible values because there are only $2^{p/2}$ nets in $T^{p/2}$. The feasibility of each separation can be checked in $O(p)$ time using $O(\frac{2^{p/2}}{p})$ processors by the offset range algorithm of Sect. 3.2 or [1]. There is a total of $O(\frac{2^p}{p})$ processors, so all possible separations can be checked at the same time. The minimum separation among the feasible separations is the optimal separation. Now, suppose $s^{p/2^i}$ is known; then at most $O(2^{p/2^{i+1}})$ possible values need to be checked for $s^{p/2^{i+1}}$ by Corollary 8. By a similar argument as before, $s^{p/2^{i+1}}$ can be found in $O(p)$ time. Finally, Line 7 is again an $O(p)$ time computation. The total time, including all the passes through the loop in lines 4–6, is $O(p \lg p)$.

On the CRCW, we can do each pass through the loop in lines 1–3 in time $O(\lg\lg N)$ with $O(N/\lg\lg N)$ processors, and there are $O(\lg\lg N)$ passes. The remaining analysis is the same as before except that we use the CRCW result of Aggarwal and Park [1] for offset range, yielding $O(\lg p)$ time to compute each $s^{p/2^i}$ (and s^0). □

When there are single-sided nets, we first use Lemma 3 to find the *contour* of the single-sided nets. At each column, we define the *extension* of the contour to be the distance that the contour extends into the channel at that column. Let T^0 be the original set of two-sided nets and let L^0 be the original contour of single-sided nets. We define T^k as before, and recursively define L^k to be a contour with the extensions of L^{k-1} divided by two (and rounded down to integral values). (The new problem $T^k \cup L^k$ can be viewed as an ordinary problem instance with at most half as many nets as $T^{k-1} \cup L^{k-1}$ by using Lemma 4. The bottom endpoints of the cuts going through convex corners of the new contour can be viewed as the new bottom terminals of single-sided nets that are of interest.) The following lemma from [9] states the relationship between $\text{optsep}(T^k \cup L^k)$ and $\text{optsep}(T^{k+1} \cup L^{k+1})$:

Lemma 10. *Let* $s^k = \text{optsep}(T^k \cup L^k)$ *and* $s^{k+1} = \text{optsep}(T^{k+1} \cup L^{k+1})$. *Then* $s^k \geq 2s^{k+1} - 1$ *and* $s^k \leq 2s^{k+1} + 2$. □

Now we can use the halving technique as before; the only difference is that we have to check 4 possible separations for s^k once s^{k+1} is known. We follow a similar analysis as in Theorem 9, but the CRCW details must be altered to avoid using the result of Aggarwal and Park that only applies to river routing. Instead, we plug in the offset range results from Sect. 3.2.

Corollary 11. *The optimal offset problem for channels with single-sided nets on one side can be solved in* $O(\lg N \lg\lg N)$ *(respectively* $O(\lg N)$*) time using* $O(\frac{N}{\lg N})$ *(respectively* $O(\frac{N \lg\lg N}{\lg N})$*) processors on a CREW (CRCW) PRAM.* □

4 Conclusion

This paper has provided efficient parallel algorithms for (1) the minimum separation problem for general single-layer channels and (2) offset problems for single-layer channels in which only one side of the channel has multiple connections to a single net. In addition, we have improved previous results for optimal offset in river routing problems, where each net has exactly one terminal on each side of the channel. An obvious open question is whether any of the bounds on time or work can be improved. In particular, the algorithms for optimal offset use $N \lg\lg N$ work rather than the $O(N)$ work that can be achived sequentially. An additional open question is whether offset problems can be efficiently solved in parallel when both sides of the channel contain single-sided nets; for sequential computation, this problem is considered in [8].

5 Acknowledgements

Thanks to Uzi Vishkin and Joseph JáJá of the University of Maryland, Omer Berkman of King's College, and Yossi Matias of AT&T Bell Labs for helpful discussions.

References

1. Alok Aggarwal and James Park. Notes on searching in multidimensional monotone arrays. In *29th Annual Symposium on Foundations of Computer Science*, pages 497–512. IEEE Computer Society Press, 1988.
2. O. Berkman, Baruch Schieber, and U. Vishkin. Some doubly logarithmic optimal parallel algorithms based on finding all nearest smaller values. Technical Report UMIACS-TR-88-79, University of Maryland Institute for Advanced Computer Studies, October 1988. To appear in *J. Algorithms.*
3. O. Berkman and U. Vishkin. Recursive star-tree parallel data-structure. Technical Report UMIACS-TR-90-40, University of Maryland Institute for Advanced Computer Studies, March 1990.
4. Shing-Chong Chang, Joseph JáJá, and Kwan Woo Ryu. Optimal parallel algorithms for one-layer routing. Technical Report UMIACS-TR-89-46, University of Maryland Institute for Advanced Computer Studies, April 1989.
5. R. Cole and U. Vishkin. Faster optimal prefix sums and list ranking. *Information and Control*, 81:334–352, 1989.
6. Danny Dolev, Kevin Karplus, Alan Siegel, Alex Strong, and Jeffrey D. Ullman. Optimal algorithms for structural assembly. *VLSI Design*, pages 38–43, 1982. Earlier version in *Proceedings of the 13th ACM Symposium on Theory of Computing.*
7. Ronald I. Greenberg and F. Miller Maley. Minimum separation for single-layer channel routing. *Information Processing Letters*, 43(4):201–205, September 1992.
8. Ronald I. Greenberg and Jau-Der Shih. Feasible offset and optimal offset for single-layer channel routing. In *Proceedings of 2nd Annual Israel Symposium on Theory of Computing and Systems*, pages 193–201. IEEE Computer Society Press, June 1993. Revised version submitted to *SIAM Journal on Discrete Mathematics.*
9. Ronald I. Greenberg and Jau-Der Shih. Single-layer channel routing and placement with single-sided nets. Submitted to *Discrete Applied Mathematics*, 1993.
10. Torben Hagerup. Constant-time parallel integer sorting. In *Proceedings of the 23rd ACM Symposium on Theory of Computing*, pages 299–306. ACM Press, 1991.
11. Joseph JáJá. *An Introduction to Parallel Algorithms.* Addison-Wesley, 1992.
12. D. Kruskal. Searching, merging and sorting in parallel computation. *IEEE Trans. Computers*, C-32(10):942–946, October 1983.
13. R. E. Ladner and M. J. Fischer. Parallel prefix computation. *Journal of the ACM*, 27(4):831–838, October 1980.
14. F. Miller Maley. *Single-Layer Wire Routing and Compaction.* MIT Press, 1990.
15. Yossi Matias and Uzi Vishkin. Converting high probability into nearly-constant time — with applications to parallel hashing. In *Proceedings of the 23rd ACM Symposium on Theory of Computing*, pages 307–316. ACM Press, 1991.
16. Andranik Mirzaian. River routing in VLSI. *Journal of Computer and System Sciences*, 34:43–54, 1987.

This article was processed using the LaTeX macro package with LLNCS style