Computer Science: Faculty Publications and Other Works

Faculty Publications

9-1992

Minimum Separation for Single-Layer Channel Routing

Ronald I. Greenberg
Loyola University Chicago, Rgreen@luc.edu

F. Miller Maley

Follow this and additional works at: https://ecommons.luc.edu/cs_facpubs

Part of the Computer Sciences Commons, and the VLSI and Circuits, Embedded and Hardware Systems Commons

Author Manuscript

This is a pre-publication author manuscript of the final, published article.

Finding Single-Layer Channel Routing Width in Linear Time

Ronald I. Greenberg[*]
Department of Electrical Engineering and
Institute for Advanced Computer Studies
University of Maryland
College Park, MD 20742
rig@eng.umd.edu

F. Miller Maley[†]
Department of Mathematics
Princeton University
Princeton, NJ 08544
maley@math.princeton.edu

Abstract

We provide a linear-time algorithm for determining the minimum separation required to route a channel when the connections can be realized in one layer. Differing from the usual "river-routing" context, we allow single-sided connections. The algorithm also works directly for problems with multiterminal nets and does not require wires to lie on an underlying grid, though it does use the rectilinear wiring model. The approach can also be used to obtain a routability test for single-layer switchboxes that corrects errors in the literature and is simpler than other correct approaches. These problems are of interest because they may arise as subtasks in routing problems involving more than one layer.

Keywords: design of algorithms, VLSI layout, single-layer wire routing, channel routing

1   Introduction

Much attention has been given to planar or single-layer wire routing for VLSI chips. Most popular has been river routing in the restricted sense of the term, the connection of two parallel rows of corresponding points[1], e.g., [4, 7, 10, 12, 13]. Other works have considered routing within a rectangle [2], placement and routing within a ring of pads [1], or routing between very general arrangements of modules [3, 6, 9].

This paper is primarily concerned with the *minimum separation problem* for single-layer channel routing, i.e., finding the minimum separation of two rows of terminals (in fixed horizontal positions) that allows the routing to be performed (subject to the design rules). We show that time linear in the number of terminals suffices to solve this problem even if the input deviates from the restricted river routing framework by including single-sided and multiterminal nets. Solution to the minimum separation problem implies, of course, a solution to the *routability problem*, which simply asks whether a routing is possible given a fixed vertical separation as well as fixed horizontal positions of the terminals. We also briefly discuss generalization from channels to switchboxes, demonstrating, in particular, a simple routability test for switchboxes. We draw much inspiration from the routability testing techniques of Cole and Siegel [3], but this paper is both simpler by virtue of concentrating on the special cases of channels and switchboxes, and more comprehensive in its analysis of those cases. The simplicity of our approach is evidenced by the inclusion of detailed but concise pseudocode for finding minimum channel width. The extension from river routing to general single-layer channel routing is not as easy as it may appear; we explain in Section 5 the erroneous nature of some published solutions to the switchbox routability problem.

A motivation for studying single-layer channel routing problems is that they may arise as subproblems within a multilayer context. For example, the heuristic multilayer channel router MulCh [5] finds good solutions to general channel routing problems by dividing them into essentially independent subproblems of one, two, or three layers. The ability to efficiently determine the minimum width of single-layer subproblems provides MulCh with a greater ability to find a good way of partitioning the original problem. Many of the test problems for MulCh included single-sided and multiterminal nets.

---

[1]This is the only usage of the term "river routing" in this paper; we refer to more complicated variations of the problem as "single-layer" or "planar" routing.

mate b_1 b_2 b_7 t_7 t_6 t_5 t_4 b_8 t_{10} t_9 b_9 t_{13} t_{12}

terminal t_1 t_2 t_3 t_4 t_5 t_6 t_7 t_8 t_9 t_{10} t_{11} t_{12} t_{13}

terminal b_1 b_2 b_3 b_4 b_5 b_6 b_7 b_8 b_9 b_{10} b_{11} b_{12} b_{13} b_{14} b_{15}

mate t_1 t_2 b_6 b_5 b_4 b_3 t_3 t_8 t_9 b_{13} b_{12} b_{11} b_{10} b_{15} b_{14}
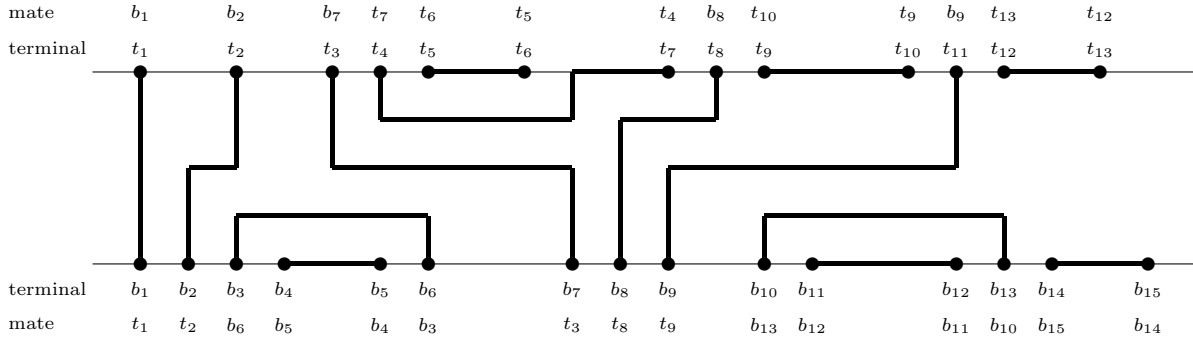
Figure 1: A representative single-layer channel routing problem. Each terminal T among the top terminals $\{t_i\}$ or the bottom terminals $\{b_i\}$ must connect to exactly one terminal $mate(T)$.

The remainder of this paper is organized as follows. Section 2 gives a more detailed definition of the single-layer channel routing problem, and Section 3 provides the algorithm for finding minimum channel separation. Sections 2 and 3 restrict attention to 2-point nets, while Section 4 explains why multiterminal nets can be handled with essentially the same algorithm. Section 5 discusses the generalization to switchboxes, and Section 6 provides concluding remarks.

2   Problem Definition and Notation

Figure 1 illustrates a single-layer channel routing problem and some of the notation to be used. For simplicity, we consider only 2-point nets and explain how to handle multiterminal nets in Section 4. We label the terminals along the top and bottom $t_1, t_2, \ldots, t_m$ and $b_1, b_2, \ldots, b_n$, respectively. In river routing, the problem would be to connect t_i to b_i for every i; in our more general channel routing problem we allow single-sided connections as between b_3 and b_6 in Figure 1.[2] For each terminal T, the notation mate(T) represents the terminal that is to be connected to T. The pair of terminals and/or the connecting wiring are referred to as a *net*. We also make the notation for terminals do double duty; where a terminal appears in an arithmetic context (comparison or subtraction), it represents the x-coordinate or horizontal position of the terminal.

A few details remain to fully define the problem. First, we assume rectilinear wiring, i.e., all wire segments are horizontal or vertical. Wires (viewed as 0 width) must stay unit distance apart to incorporate the width and spacing requirements for VLSI chips. If this condition can be met while wiring together the requisite terminals in a channel (with a specified separation), we say that the channel is *routable*. It is not necessary that wires and terminals be restricted to an underlying grid, as long as no two terminals are less than unit distance apart; it is merely convenient in this paper to draw the illustrations on an underlying grid.

As a final modeling decision, it is convenient to allow wires to be routed immediately alongside the channel boundaries. If there are practical objections to this allowance, it is still a simple matter to get the correct result from the algorithm in this paper. Specifically, if boundary routing is not permitted, it is merely necessary to add 2 to the separation computed by our algorithm; this procedure yields a suboptimal result only if all nets are 2-point nets running straight across the channel, an easy special case to check for. Another possible model is to allow boundary routing only for the connection of terminals less than two units apart (on adjacent gridpoints in the case of a gridded model). This problem can be easily reduced to the problem with boundary routing completely disallowed by simply stripping out certain terminals.

3   Finding Minimum Channel Separation

This section shows how to determine minimum separation for the channel routing problem just described. The first step in the demonstration is the invocation of a general theory of single-layer routing [3, 9] specialized to channel routing to obtain a set of necessary and sufficient conditions for routability. Then it is shown

---

[2]Though we use the term "single-sided" for these connections, the term "sides" will be used in Section 5 to refer to the left and right of the channel as opposed to the top and bottom.

that we can reduce the number of conditions that must be checked, and, finally, that the minimum channel width satisfying the conditions can be determined in linear time.

It is most convenient for us to use the single-layer routing theory of [9] in a slightly altered form consistent with the approach of Cole and Siegel [3]. The basic idea of the theory is that routability can be decided by checking whether a limited collection of *cuts* are *safe*. Roughly, a cut is a line segment from top to bottom of the channel, and it is safe if there is enough room for all the nets that must cross it to do so. To make the set of important cuts be a particularly convenient set, we slightly alter a few definitions from [9] by considering cuts as closed rather than open line segments. The key definitions that place the theory in the form we desire are summarized below.

> **Definition:** A *critical cut* is a line segment connecting a terminal on the top of the channel to a terminal on the bottom or a line segment running from a terminal straight across to the other channel boundary.

> **Definition:** The *flow* of a cut c, denoted flow(c), is the number of wires that *must* cross c, including the wires incident at an endpoint of c. More precisely, the flow is the number of nets that have terminals lying on opposite sides of c or that have a terminal coincident with an endpoint of c. Note that this definition depends only on the cut and the terminal positions, not on any particular way of drawing the nets.

> **Definition:** The *capacity* of a cut c, denoted cap(c), is one plus the maximum of the horizontal and vertical separations of the endpoints of the cut. That is, if c connects the points with coordinates (x_t, y_t) and (x_b, y_b), then cap(c) = $\max\{|x_t - x_b|, |y_t - y_b|\} + 1$.

> **Definition:** A cut c is *safe* if an only if flow(c) $\leq$ cap(c).

We can now state formally the result we need from the general theory of single-layer routing.

> **Lemma 1** *A channel is routable if and only if all the critical cuts are safe.*

Proof sketch. The lemma follows immediately from the corresponding result in [9, Section 2.1] using the slightly different definitions of *flow*, *capacity*, and *critical cut*. ■

The restriction to checking critical cuts gives us $O(N^2)$ conditions that must be checked, where $N = m+n$ is the number of terminals. In the river routing problem, it suffices to check a smaller set of $O(N)$ cuts, but when single-sided connections are allowed it is not evident how to reduce the number of cuts below $\Omega(N^2)$. Nonetheless, it is possible to eliminate some of the critical cuts in a manner dependent on the flows and capacities of the cuts. We still consider $\Omega(N^2)$ cuts in the worst case, but the flows and capacities are sufficiently related that linear time suffices to determine the minimum channel width making all the cuts safe. Henceforth, since it suffices to restrict attention to critical cuts, we use the term "cut" to refer specifically to a critical cut.

The necessary limitation on the critical cuts to be checked is provided by the simple distinction between *dense* and *sparse* cuts. A sparse cut is one that is safe regardless of the channel separation, i.e., the horizontal distance between the cut endpoints is large enough relative to the flow that the cut is guaranteed to be safe. A cut that is not sparse is referred to as dense. For convenience, we also classify as dense any cut running from a terminal straight across the channel. (Such a cut is both sparse and dense if its flow is 1.) With the distinction between dense and sparse cuts, we can provide a simple expression for the channel separation as given in the following corollary to Lemma 1.

> **Corollary 2** *Minimum channel separation is given by* $-1 + \max\limits_{c \text{ dense}} \text{flow}(c)$. ■

We now make two important observations about the structure of dense cuts in a channel. First, the set of dense cuts emanating from any given terminal forms a "cone". In Figure 2, for example, the dotted and dashed lines show all the critical cuts emanating from terminal a on the bottom of the channel. The cuts to points between terminals l and r inclusive are all dense, while the cuts outside this cone are all sparse. To see that this situation prevails in general, just compare any sparse cut to a cut "immediately below" it. For
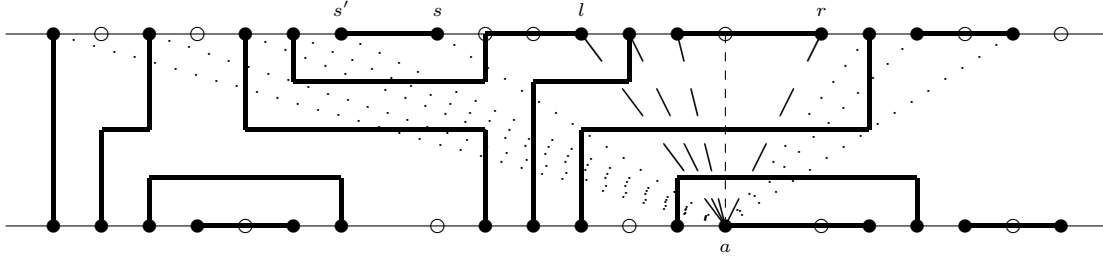
Figure 2: The dotted and dashed lines show the sparse and dense cuts, respectively, emanating from terminal a. In general, the dense cuts emanating from any terminal form a "cone". This figure also shows the dummy terminals (hollow circles) that are added for convenience in detailing the algorithm.

example, in Figure 2, the sparsity of cut $[a, s]$ implies sparsity of $[a, s']$, since the horizontal extent of $[a, s']$ is at least one greater than that of $[a, s]$, but the flow of $[a, s']$ can be at most one greater than that of $[a, s]$. (In this example, the horizontal extent is two greater, and the flows are the same.) The second observation is that if we consider a bottom terminal a' to the right of a, the terminals l' and r' corresponding to the leftmost and rightmost dense cuts from a' are at least as far to the right as l and r, respectively. (If, for example, l' was to the left of l, the sparsity of $[a, l']$ would contradict the fact that the set of dense cuts emanating from l' must form a cone.)

Since we have observed that l and r in Figure 2 move only to the right as a does, we can move a across the channel bottom and update l and r at each stage with only a linear number of movements of a, l, and r all together. Thus, we need only show that we can move each of a, l, and r one terminal to the right in constant time while maintaining the maximum of the flows of the cuts in the cone. The maximum flow over all positions of a is the maximum flow of all dense cuts in the channel, except that we may miss straight cuts emanating from top terminals. We remedy this omission by adding dummy bottom terminals at x-positions that have a top terminal but no bottom terminal. (For any dummy terminal T, mate(T) will have a special null value.) In fact, for simplicity of coding the algorithm, it is convenient to also add dummy top terminals at x-positions that have a bottom terminal but no top terminal. The positions of the dummy terminals are illustrated in Figure 2.

Moving l or r one terminal position to the right is relatively easy. Moving r a step to the right, just requires computing the flow of a new cut whose flow differs by -1, 0, or 1 from the flow of the "preceding" cut. Let us define a function flowdiff to represent this difference:

$$\text{flowdiff}(u, v, w) = \text{flow}([u, v]) - \text{flow}([u, w]) .$$

When v and w are adjacent terminals on one of the channel boundaries (and u is a terminal anywhere on the other boundary), the value of flowdiff(u, v, w) can be computed by checking on which sides of the cuts $[u, v]$ and $[u, w]$ the mates of v and w lie. As r is moved right, each new flow is tallied in a table of flows in the current collection of cuts; that is, for each flow value we keep a count of how many cuts have that flow. When l is moved a step to the right we simply decrement the appropriate tally in the table of flows. (Repetition of the flowdiff computation can be avoided here by remembering the flow associated with each top terminal.) The maximum of the flows of the cuts in the current collection is updated by comparison of the old maximum to the flow of the new cut when r is moved right. When l is moved right and we decrement the appropriate flow tally, we simply check whether that flow was the maximum and the tally for that flow has gone to zero; if so, we decrement the maximum, since the flow values that have nonzero tallies must always be a continuous range of integers.

Finally, constant time also suffices to move a one terminal to the right. This is true because when a moves one terminal to the right, the flows of cuts to relevant top terminals all change by the same amount $(0, 1, \text{or } -1)$. That is, for all top terminals in the cone of dense cuts for the old position of a, the change in flow is the same. This follows from a straightforward case analysis showing that either all or none of these top terminals are strictly "between" the nets corresponding to the adjacent bottom terminals under consideration. (A key observation is that a cut of flow at most 2 can only be dense if it runs straight across

4

procedure CHANNEL-SEPARATION

1 Add dummy terminals (with null mates) so that every x-position with
   a terminal has both a top and bottom terminal, yielding top and
   bottom terminal sets $\{t_1, t_2, \ldots, t_{last}\}$ and $\{b_1, b_2, \ldots, b_{last}\}$.

2 $separation \leftarrow 0$; $a \leftarrow 1$; $l \leftarrow 1$; $r \leftarrow 1$

3 $offset \leftarrow 0$; $flow(1) \leftarrow \text{flow}([b_1, t_1])$; $maxflow \leftarrow flow(1)$
   $\langle offset + flow(r) = \text{flow}([b_a, t_r]) \ always \rangle$

4 $tally(i) \leftarrow 0$ for $-n \leq i \leq \frac{1}{2}(m+n)$; $tally(flow(1)) \leftarrow 1$

5 **while** $a \leq last$

6   **while** $r < last$

7          $flow(r+1) \leftarrow flow(r) + \text{flowdiff}(b_a, t_{r+1}, t_r)$

8   **if** $t_{r+1} - b_a \geq \max\{1, flow(r+1) + offset - 1\}$ **then**

9   **break** $\quad \langle sparse \ cut \rangle$

10   **endif**

11         $r \leftarrow r + 1$

12         $tally(flow(r)) \leftarrow tally(flow(r)) + 1$

13         $maxflow \leftarrow \max\{maxflow, flow(r)\}$

14   **endwhile**

15   **while** $b_a - t_l \geq \max\{1, flow(l) + offset - 1\}$ $\quad \langle sparse \ cut \rangle$

16         $tally(flow(l)) \leftarrow tally(flow(l)) - 1$

17   **if** $maxflow = flow(l)$ and $tally(flow(l)) = 0$ **then**

18             $maxflow \leftarrow maxflow - 1$

19   **endif**

20         $l \leftarrow l + 1$

21   **endwhile**

22     $separation \leftarrow \max\{separation, maxflow + offset - 1\}$

23   **if** $a < last$ **then**

24         $a \leftarrow a + 1$; $offset \leftarrow offset + \text{flowdiff}(t_r, b_a, b_{a-1})$

25   **endif**

26 **endwhile**

Figure 3: This algorithm computes the maximum of the flows of the dense cuts, thereby determining the minimum channel separation for which all cuts are safe. Comments appear inside angle brackets. Variable names are set in italics; function names appear in roman type.

the channel.) Thus, when a moves, it is only necessary to adjust a flow offset that will be applied to all the flows in the collection. That is, after the updates to l and r have been completed for a given position of a, we have determined only a nominal maximum of the flows in the current cone. To this nominal maximum, we add the flow offset; then we compare to the maximum flow found at previous positions of a.

The arguments provided above suffice to show that the procedure in Figure 3 computes the minimum channel separation in linear time. In the pseudocode, a, l, and r are essentially as above, but now they are indices in the appropriate list of terminals. The variables $maxflow$ and $offset$ represent the nominal maximum flow in the current collection and the flow offset as just discussed. We use the array $flow$ to keep track of the flows of cuts terminating at given top terminals, and the array $tally$ for the count of how many cuts have a given flow value. Lines 1–4 perform initialization. (Note that a range of $-n$ to $n + \frac{1}{2}(m+n)$ suffices for the indices of $tally$, since the flow of any cut is in the range $[1, \frac{1}{2}(m+n)]$, the offset is in the range $[-n, n]$, and flow minus offset is upper bounded by $\frac{1}{2}(m+n)$ for dense cuts.) Lines 6–14 update r for a given position of a, and then lines 15–21 update l. Lines 5 and 22–26 move a across the channel, update $offset$, and update the separation, which we have seen is one less than the running max of the flows in the cones of dense cuts.

4   Handling Multiterminal Nets

This section gives a proof sketch that the only change required to handle multiterminal nets is maintenance of a small amount of extra data for computation of the flowdiff function. Instead of working only with mates

of terminals, we precompute for each net the leftmost and rightmost terminal on each side (top and bottom) of the channel. It is easy to see that this information suffices to do the required flowdiff computations; notice that flow remains well-defined under the definition near the beginning of Section 3.

To see that our algorithm works for multiterminal problems, consider transformation to a 2-terminal problem by replacing each t-terminal net (including $t{=}2$) with a "ring" of t 2-terminal nets. (Note that our algorithm does not actually perform this transformation; it is considered only for analysis. Also, though the transformation is a standard reduction for routing problems, we know of no formal analysis of the effect of multiterminal nets on routability conditions and minimum width determinations as discussed in this paper.) In the transformation, we split each of the old terminals into two terminals and displace them $1/4$ unit horizontally away (in opposite directions) from the old terminal position. We also consider the minimum separation for terminals and wires to be $1/2$ instead of 1. Now it is straightforward to verify that the transformed problem is routable in separation $s + \frac{1}{2}$ if and only if the original problem is routable in separation s. (To go from a solution to the original to a solution to the transformed problem, think of each old wire as generating two parallel wires $1/4$ unit away and similarly for the boundaries, and remember that routing on the boundaries is allowed. The reverse transformation of solutions can be accomplished by coalescing wires corresponding to a given original net into a Steiner tree, since there is no room for conflicting wires to have gotten into the "middle of the ring".)

Now if we adjust the capacity and flow definitions in the transformed problem to account for wires being of only $1/2$ weight, we can compare the safety conditions for separation $s + \frac{1}{2}$ in the transformed problem to those for separation s in the original problem. (In the transformed problem, flow is $1/2$ times the old definition, and capacity is $1/2$ plus the maximum of the horizontal and vertical separations of the cut endpoints.) We compare the condition for each cut in the original problem to the conditions for the two or four cuts it generates in the transformed problem when the terminals are split. At this point, it is simply a case analysis to verify that the transformed problem, with separation $s + \frac{1}{2}$, has an unsafe cut if and only if the original problem, with separation s, has an unsafe cut. We leave the case analysis to the reader, but a helpful shortcut is as follows. In the original problem, call a terminal "outer" if it is the leftmost or rightmost terminal of its net along its side of the channel; call the other terminals "inner". There is no need to check cuts ending at inner terminals; if such a cut is unsafe, there is another unsafe cut that does not have an inner terminal as an endpoint. Similarly, in the transformed problem, we can ignore cuts with an endpoint derived from an inner terminal.

5   Extension to Switchboxes

This section considers extension of the above results from channels to switchboxes. That is, terminals are allowed at the (vertical) sides of the channel instead of just at the top and bottom. With floating side-terminals, the problem is easily reduced to the problem without side-terminals; when side-terminals are fixed, we obtain a linear-time routability test which is simpler than that of Chang, JáJá, and Ryu [2]. It should be noted that the literature (e.g., [8, 11]) also contains erroneous solutions to this problem.[3]

The first variant of switch box routing, in which side-terminals are allowed to float to arbitrary y-positions may be thought of as merely an extended channel (figuratively and literally). (Included in this notion of floatability is that there are no single-sided side connections.) It suffices to extend the channel, place the side terminals along the top or bottom of the channel, and solve the problem as before. As long as there are no single-sided side connections, the wires attached to the former side terminals must cross the former channel sides, so any routing for the new problem can be converted to a routing for the old problem by merely truncating some wires.

When the side-terminals are fixed, a linear-time routability test is obtained with little more than two applications of the procedure CHANNEL-SEPARATION. After applying that procedure to the terminals at top and bottom of the switchbox and then applying it with the appropriate shift in orientation to the terminals at left and right of the switchbox, it is only necessary to check for safety of the cuts that connect perpendicular

---

[3]These sources propose finding the contours of the single-sided nets first and then looking only at certain horizontal, vertical, or 45° cuts in the resultant channel with ragged boundaries. In the example shown here, however, the only unsafe cuts connect the two boundary protrusions. Note that for consistency with [8, 11], the example is in the model that does not allow routing on the channel boundaries. Also, we have labeled terminals with net numbers.
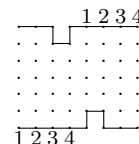
sides of the switchbox. Fortunately, checking these corner cuts is as easy as for river routing without side connections [3]; we need only check the diagonal cuts, which is a simple, linear-time procedure.

6   Conclusion

We have demonstrated a simple, linear-time algorithm for several single-layer routing problems. Specifically, we can determine minimum width for channels or channels with floating side-terminals, and we can test routability in a switchbox with all terminal positions fixed. Further directions for research involve minimizing channel width or switchbox area with differing types of constraints on terminal positioning.

Acknowledgements

References

[1] B. S. Baker and R. Y. Pinter. An algorithm for the optimal placement and routing of a circuit within a ring of pads. In *24th Symp. Found. Comp. Sci.*, pages 360–370. IEEE, 1983.

[2] S.-C. Chang, J. JáJá, and K. W. Ryu. Optimal parallel algorithms for one-layer routing. Technical Report UMIACS TR 89-46, U. of Maryland Inst. for Advanced Computer Studies, Apr. 1989.

[3] R. Cole and A. Siegel. River routing every which way, but loose. In *25th Symp. Found. Comp. Sci.*, pages 65–73. IEEE, 1984.

[4] D. Dolev, K. Karplus, A. Siegel, A. Strong, and J. D. Ullman. Optimal wiring between rectangles. In *13th Symp. Theory Comput.*, pages 312–317. ACM, 1981.

[5] R. I. Greenberg, A. T. Ishii, and A. L. Sangiovanni-Vincentelli. MulCh: A multi-layer channel router using one, two, and three layer partitions. In *ICCAD-88*, pages 88–91. IEEE, 1988.

[6] C. E. Leiserson and F. M. Maley. Algorithms for routing and testing routability of planar VLSI layouts. In *17th Symp. Theory Comput.*, pages 69–78. ACM, 1985.

[7] C. E. Leiserson and R. Y. Pinter. Optimal placement for river routing. *SIAM J. Comput.*, 12(3):447–462, Aug. 1983.

[8] T. Lengauer. *Combinatorial Algorithms for Integrated Circuit Layout*. Wiley, 1990.

[9] F. M. Maley. *Single-Layer Wire Routing and Compaction*. MIT Press, 1990.

[10] A. Mirzaian. River routing in VLSI. *J. Comp. Sys. Sci.*, 34:43–54, 1987.

[11] R. Y. Pinter. River routing: Methodology and analysis. In R. Bryant, editor, *Third Caltech Conf. on VLSI*, pages 141–163. Comp. Sci. Press, 1983.

[12] A. Siegel and D. Dolev. Some geometry for general river routing. *SIAM J. Comput.*, 17(3):583–605, June 1988.

[13] M. Tompa. An optimal solution to a wire-routing problem. *J. Comp. Sys. Sci.*, 23:127–150, 1981.