Algorithmic Study of Single-Layer Bus Routing for High-Speed Boards

Muhammet Mustafa Ozdal and Martin D. F. Wong, *Senior Member, IEEE*

***Abstract*—As the clock frequencies used in industrial applications increase, the timing requirements on routing problems become tighter, and current routing tools cannot successfully handle these constraints anymore. In this paper, the authors focus on the high-performance single-layer bus routing problem, where the objective is to match the lengths of all nets belonging to each bus. An effective approach to solve this problem is to allocate extra routing resources around short nets during routing, and use those resources for length extension afterwards. First, a provably optimal algorithm for routing nets with minimum-area maximum-length constraints is proposed. Then, this algorithm is extended to the case where minimum constraints are given as exact length bounds, and it is also proven that this algorithm is near-optimal. Both algorithms proposed are shown to be scalable for large circuits, since the respective time complexities are $O(A)$ and $O(A \log A)$, where A is the area of the intermediate region between chips.**

***Index Terms*—Algorithms, design automation, physical design, routing.**

I. Introduction

As THE clock frequencies used in industrial circuits increase, the timing requirements imposed on routing problems become tighter. There have been several algorithms proposed for the objective of minimizing path lengths, or satisfying prespecified maximum-length constraints, especially in the context of timing-driven routing [1]–[6]. However, the problem of routing nets with lower bound constraints has not been studied extensively in the literature. The main reason is that these bounds were loose most of the time, and non-sophisticated strategies (such as greedy length extension in postprocessing) were sufficient for most applications. However, as circuits start to use clock frequencies in the order of gigahertz in the current technology, the timing constraints become extremely tight, and automatic routing becomes more challenging. The current industrial tools cannot successfully route many high-end designs, and the manual routing process takes about a month in a typical design cycle [7]. Therefore, more aggressive routing algorithms that handle tight constraints are needed for current industrial circuits.

In a recent work [8], a general Lagrangian relaxation (LR)-based framework has been proposed for printed circuit board (PCB) routing with minimum- and maximum-length constraints. Although that framework can be applied to general routing problems, it is heuristics based, and it has no optimality guarantees. In this paper, a more restricted yet common high-performance routing problem is studied, and provably optimal algorithms are proposed for it.

The authors have been working on a project to develop a large-scale routing system for high-end IBM PCBs. Here, a typical board contains several bus structures between multichip modules (MCMs), memory, and input/output (I/O) modules, in addition to individual nets. In a typical bus structure, data are clocked into registers, or other circuits; so all signals traveling over different wires of the bus must arrive at their destinations approximately at the same time. To achieve this, all the wires constituting this bus need to have approximately the same length. The precision with which matching must be done is directly related to the clock frequency [9]. As the clock frequency increases, the skew requirements on the propagation delays become more strict, and hence, a higher degree of length matching is required. Furthermore, in some applications, the use of vias in multilayer designs is restricted to reduce the manufacturing costs. For such circuits, a routing algorithm that assigns nets to different layers and then routes each net on a single layer is needed.

In this paper, the authors focus on the single-layer bus routing problem, which is illustrated in Fig. 1 by an example. Here, assume that layer assignment has already been performed, and all nets have been routed from their individual pins to chip boundaries.[1] Now, the problem is to route nets between pairs of components such that all nets belonging to the same bus have approximately the same length. In the example of Fig. 1, two separate bus structures are displayed. However, in reality, there may be more than one bus structure, interleaved with each other, or there may be individual nets not belonging to any bus. For this reason, the general problem, where each net has individual minimum- and maximum-length constraints, will be focused upon in this paper.

Actually, this problem is similar to the river routing [10] problem, in the sense that all terminal points are aligned with

Manuscript received October 7, 2004. This work was supported in part by the National Science Foundation under Grant CCR-0306244 and by an IBM Faculty Partnership Award. The preliminary version of this paper has been presented in IEEE/ACM ICCAD in November 2004. This paper was recommended by Associate Editor T. Yoshimura.

M. M. Ozdal was with the Department of Electrical and Computer Engineering, University of Illinois at Urbana–Champaign, Urbana, IL 61801 USA. He is now with the Intel Corporation, Hillsboro, OR 97124 USA.

M. D. F. Wong is with the Department of Electrical and Computer Engineering, University of Illinois at Urbana–Champaign, Urbana, IL 61801 USA.

Digital Object Identifier 10.1109/TCAD.2005.853685

[1]Since via usage is not allowed, the previous phases of the routing system make sure that net ordering within a single layer is compatible among different components.

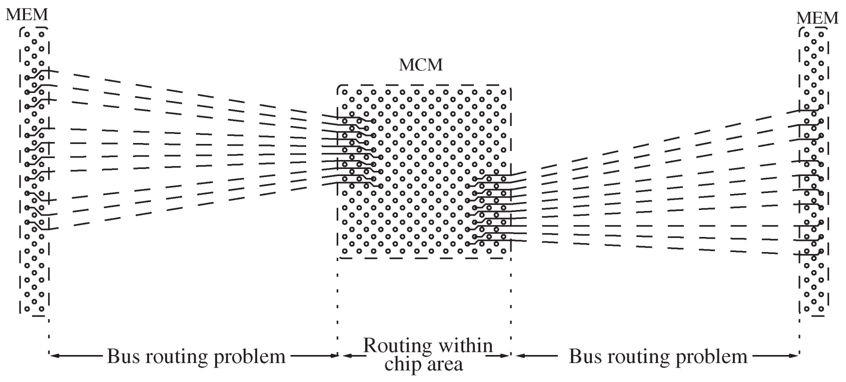


Fig. 1. Two separate bus structures are displayed between MCM and memory modules of a sample board.

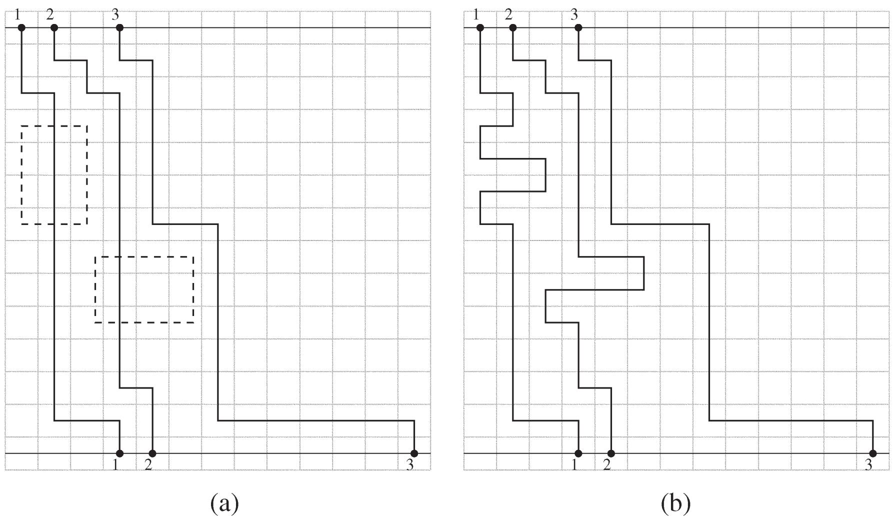


Fig. 2. (a) Sample routing solution with area allocation, where dashed lines represent the allocated areas around routes. (b) Final routing solution, where shorter nets have extended their lengths using the allocated areas.

each other on two opposite sides of the circuit, and a single-layer routing solution is desired. River routing has been studied extensively in the literature [11]–[15]. A common application for river routing has been routing bus structures across a channel [11]. Today, a similar problem for high-performance bus routing is confronted, with the additional constraint that each net must be routed within minimum–maximum-length constraints due to very high clock frequencies.

In this paper, first, the problem of minimum-area maximum-length routing is defined in Section II. The objective here is to route each net within its maximum-length constraint, while allocating at least a prespecified amount of routing area around it. Then, in postprocessing, snaking can be performed within this area to extend the lengths of short nets. Intuitively, shorter nets belonging to a bus need to allocate more area around their routes so that length matching will be possible at the end. A linear-time optimal algorithm for this problem is proposed in Section II. Then, this algorithm is extended in Section III to solve a general river routing problem with minimum–maximum-length constraints and it is proven to be near-optimal. Finally, in Section IV, experiments are performed to compare the proposed algorithms with the recent work [8].

```
find the leftmost boundary $L_i$ for each net $i$
find the rightmost boundary $R_i$ for each net $i$
for each net $i$ from left to right
    while Route($L_i$, $L_{i+1}$) violates minimum-area constraint
        flip an appropriate corner of $L_{i+1}$ rightwards
```

Fig. 3. High-level algorithm for routing problem with minimum-area maximum-length constraints.

II. Minimum-Area Maximum-Length Routing

A. Problem Formulation

The objective here is to find a routing solution, where each net has some extra space allocated around its route. The idea is that if a short net allocates enough routing resources around itself, it is possible to extend its length in postprocessing using those extra resources. Fig. 2(a) gives a sample routing solution, where shorter nets have allocated extra routing areas around their routes. Fig. 2(b) illustrates how those areas can be used for matching the lengths of all three nets. Based on this idea, it is assumed that each minimum-length constraint $(T_i^{\min})$ can

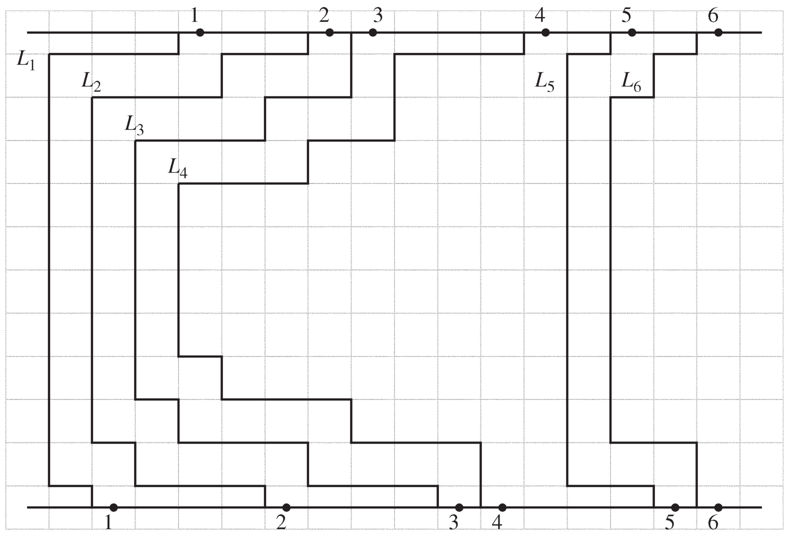


Fig. 4. Left boundaries $L_1 - L_6$ for six nets. Terminal points of the nets are shown as filled circles. A feasible route for net i cannot cross L_i at any point.

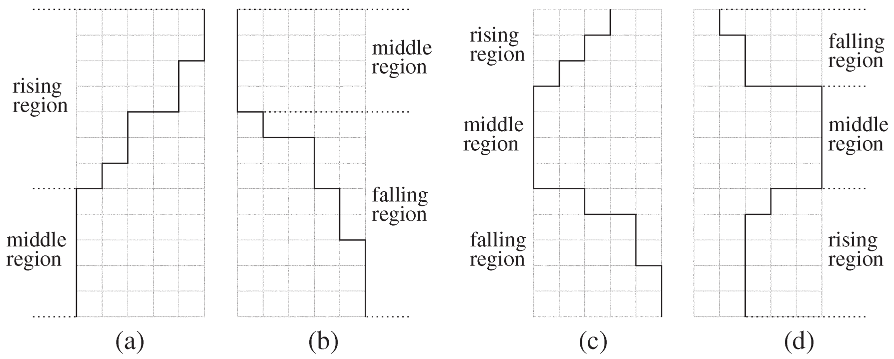


Fig. 5. Types of boundaries. (a) Rising monotonic. (b) Falling monotonic. (c) Concave nonmonotonic. (d) Convex nonmonotonic.

actually be given as a minimum-area constraint $(A_i^{\min})$, where the area of route i (A_i) is defined as the total number of grid cells allocated by route i.

This problem can formally be stated as follows: Given a set of nets $\mathcal{N}$ and minimum-area $(A_i^{\min})$ maximum-length $(T_i^{\max})$ constraints for each net i, find a single-layer routing solution such that $A_i \geq A_i^{\min}$ and $T_i \leq T_i^{\max}$, where A_i and T_i denote the area and the length of route i, respectively.

Here, it is assumed that there is an underlying routing grid where routes go center-to-center of grid cells. As mentioned in the previous section, it is also assumed that all terminal points are aligned on two opposite sides of the circuit. For simplicity of the presentation, the proposed algorithms will be given for the case where terminal points are at the topmost and bottommost rows of the grid.

B. Optimal Algorithm

The high-level description of the proposed algorithm is given in Fig. 3. The algorithm starts with finding the leftmost boundary (L_i), and the rightmost boundary (R_i) for each net i. Here, L_i and R_i define the interval within which net i must be routed. These boundaries depend on: 1) the boundaries to the left and right of net i; and 2) the maximum detour allowed for net i due to its maximum-length constraint. Fig. 4 illustrates the idea for left boundaries. The right boundaries can also be found similarly. Note here that, a boundary L_i follows L_{i-1} to the left as long as maximum-length constraint of net i is not violated. For example, L_5 in Fig. 4 stops short of following L_4 due to the maximum-length constraint of net 5.

After finding the initial positions of all left and right boundaries, the algorithm attempts to find a valid route for each net, starting from the leftmost one. At any point in time, the route of net i is defined based on L_i and L_{i+1} as follows: The main route of net i follows the trail of L_i as close as possible; and all the remaining grid cells between L_i and L_{i+1} are allocated by net i as extra routing area. Since the left and right boundaries are defined based on the maximum-length constraints, it is guaranteed that any route within those boundaries will satisfy maximum constraints. Therefore, the algorithm checks only the minimum-area constraints. The strategy here is to incrementally flip the left boundary of the right neighbor until the area constraint of the current net is satisfied.

Before giving details of the flip operations, some properties need to be defined for boundaries.

Definition 2.1: A boundary is defined to be monotonic if its trail contains no detour, and nonmonotonic otherwise. A

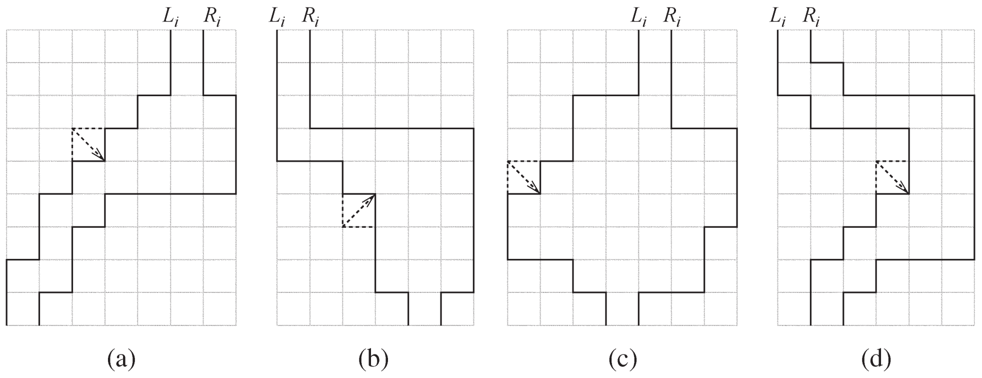


Fig. 6. Next flip on left boundary L_i if L_i is (a) rising monotonic, (b) falling monotonic, (c) concave nonmonotonic, and (d) convex nonmonotonic. Dashed lines illustrate the flip operation. The final L_i is shown in solid lines.

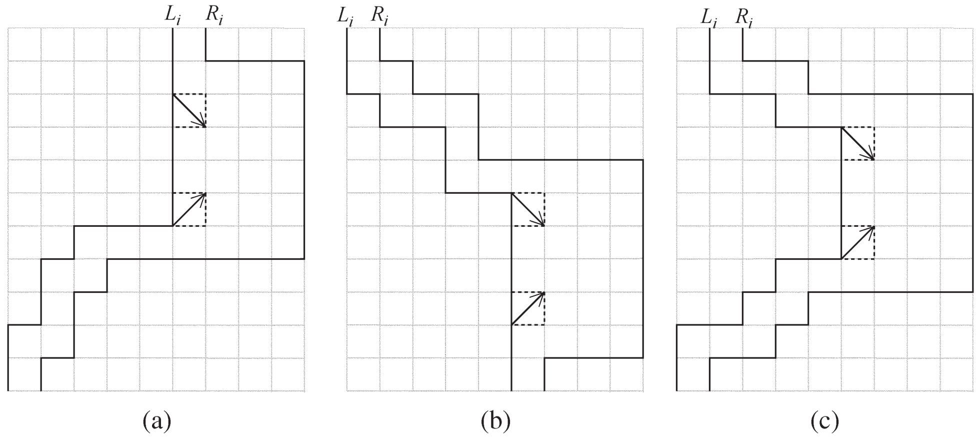


Fig. 7. Special cases corresponding to (a) rising monotonic L_i, (b) falling monotonic L_i, and (c) convex nonmonotonic L_i. Dashed lines illustrate two alternative flips for each case.

monotonic boundary can be either falling or rising, depending on the relative positions of its terminals. A nonmonotonic boundary can be either concave or convex, depending on the direction of the detour. A boundary is defined to have three regions: rising, middle, and falling regions.

These concepts are illustrated in Fig. 5. Note that a rising monotonic boundary has an empty falling region, and vice versa. Due to the algorithm used here, a nonmonotonic boundary cannot have detours in two different directions at any point in time; so this case is not defined. It is also possible to show that a left boundary cannot be convex nonmonotonic in the beginning of the algorithm. Similarly, a right boundary cannot be concave nonmonotonic.

Based on Definition 2.1, the operation flip L_i can be defined as follows.

- If L_i is rising monotonic: Flip the top corner of the leftmost segment of L_i that is not adjacent to R_i. If no such corner exists, see the special case below.
- If L_i is falling monotonic: Flip the bottom corner of the leftmost segment of L_i that is not adjacent to R_i. If no such corner exists, see the special case below.
- If L_i is concave nonmonotonic: Flip the top corner of the middle region. Since R_i is guaranteed not to be concave nonmonotonic, such a flip will always be possible.
- If L_i is convex nonmonotonic: Consider the leftmost segment of L_i that is not adjacent to R_i. If this segment is in the rising region of L_i [as in Fig. 6(d)], then flip its top corner. If it is in the falling region of L_i, then flip its bottom corner. If it is in the middle region, then see the special case below.

Fig. 6 illustrates each case with an example. Recall that the algorithm given in Fig. 3 processes nets from left to right, and L_i is flipped to allocate more area for net $i-1$. It is obvious that L_i cannot be pushed on or beyond any segment of R_i, since that would make it impossible to route the next net.

There are three special cases mentioned above that need to be handled separately. These are illustrated in Fig. 7. Observe that the rightmost segments can be flipped either from the top or from the bottom. This decision has to be made so that the following invariant is maintained throughout the execution: Each boundary is one of the following: 1) rising monotonic; 2) falling monotonic; 3) concave nonmonotonic; and 4) convex nonmonotonic. Fig. 8(b) shows an example where this invariant is violated because of an incorrect decision. On the other hand, if the flip is performed from the bottom as in part (c), then the invariant is successfully maintained. Note that flipping L_i modifies some of the boundaries L_j $(j > i)$ to the right of L_i, as in the example of Fig. 8. Here, the invariant for all

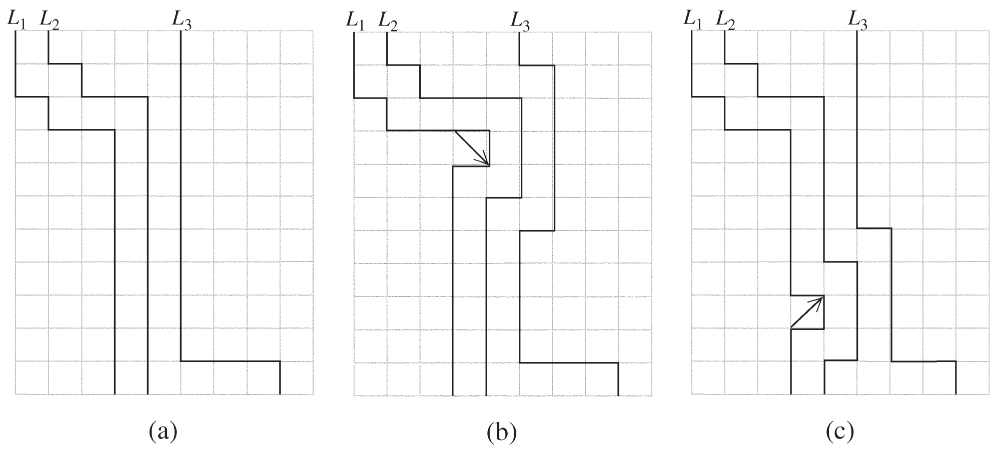


Fig. 8. Illustrating two alternative flip operations for a special case. (a) Flip is to be performed at the rightmost segment of L_1. (b) Flip is performed from the top, and L_3 violates the invariant. (c) Flip is performed from the bottom, and the invariant is maintained.

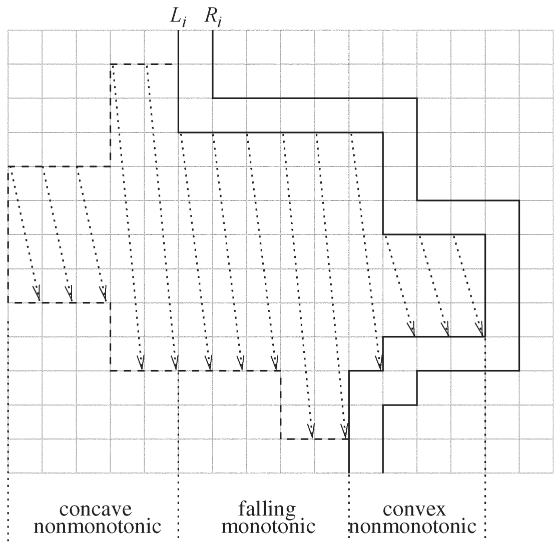


Fig. 9. Sequence of flips on left boundary L_i is illustrated. The initial and final positions of L_i are shown with dashed and solid lines, respectively. Each columnwise flip is shown as a dotted arrow. The intervals within which L_i is monotonic or nonmonotonic are also specified.

boundaries between L_i and L_k needs to be checked, where k is the smallest value such that $k > i$, and L_k will not be convex nonmonotonic after the flip. Note that the boundaries to the right of L_k do not need to be checked, because only convex nonmonotonic boundaries can cause others (to the right) to violate the invariant.

A sequence of flips on an initially concave nonmonotonic boundary L_i is illustrated in Fig. 9 as a general example. As mentioned above, it is always possible to flip the top corner of a concave nonmonotonic boundary. Therefore, a sequence of flips is performed on L_i, until it becomes falling monotonic. Then, it stays monotonic until all its segments except the rightmost one are pushed adjacent to R_i. After that, it becomes convex nonmonotonic, and its middle region is flipped until there remains a trail only with a single width for net i.

The example of Fig. 9 suggests that flips do not need to be performed only one unit at a time, as shown in Fig. 6. Instead, two binary searches can be used to find the minimum number of flips necessary to satisfy the minimum-area requirement of the current net. Here, the first binary search performs flips one column at a time (as in Fig. 9) to find the minimum number of columnwise flips. Then, the second binary search finds the minimum number of single-unit flips necessary on the last columnwise flip.

The time complexity of this algorithm is $O(A)$, where A is the area of the channel. The operations of finding the left and right boundaries for all nets dominate this time complexity. Note that $O(\log A_i)$ iterations in the binary search described above are needed to route net i, where A_i denotes the total area between L_i and L_{i+1}. The area constraint for net i can be checked in constant time in each iteration.[2] Note also that checking the invariants for special cases (as in Fig. 7) does not affect the time complexity. The main reason is that this check needs to be done only once for each net; i.e., for the last partial column.

C. Proof of Correctness

It will be proven that if a feasible solution exists for a given minimum-area maximum-length routing problem, the proposed algorithm is guaranteed to find it.

Lemma 2.1: Consider the initial positions of two adjacent boundaries L_i and L_{i+1}. It is guaranteed that every segment in the rising and falling regions of L_{i+1} is adjacent to either some segment of L_i, or the channel boundary. In other words, extra routing space between L_i and L_{i+1} can only exist to the left of the middle region of L_{i+1}.

Proof: The main intuition is that maximum detour for L_{i+1} is determined by the position of its middle region. The bends that are not in the middle region of L_{i+1} must be due to the blockage of L_i. ■

Definition 2.2: A grid cell in the final routing solution is defined to be critically allocated if and only if its removal causes either a route to be disconnected, or a minimum-area constraint to be violated.

It can be argued that the grid cells that are not critically allocated may cause some minimum-area constraints to be

[2]It is assumed that a boundary L_j $(j > i + 1)$ is updated lazily, i.e., only before net $j - 1$ is to be routed.

violated. As an example, consider the interval between L_4 and L_5 in Fig. 4. If the minimum-area constraint of net 4 is not large enough, there will be several grid cells here that are not critically allocated. On the other hand, the minimum-area constraint for net 6 can be violated, since it cannot be routed further to the left due to the blockage of L_5. In other words, some routing resources are wasted in one part of the circuit, while there are not enough resources in other parts. Our optimality proof will be based on the fact that the proposed algorithm uses routing resources at least as efficiently as any feasible solution.

Remark 2.2: If there is no extra space between the initial positions of L_i and L_{i+1}, then all grid cells between the final positions of L_i and L_{i+1} will be critically allocated.

Lemma 2.3: If there is some extra space between the initial positions of L_i and L_{i+1}, then either 1) all grid cells between the final positions of L_i and L_{i+1} will be critically allocated or 2) the final position of L_{i+1} will be the same as its initial position.

Proof: The left boundary L_{i+1} will be flipped only after all extra routing space between L_i and L_{i+1} is critically allocated by some net to the left of net $i+1$. From Lemma 2.1, it is known that the extra space must be to the left of the middle region of L_{i+1}. Since the algorithm continuously flips L_i from its leftmost available segment, the position of L_{i+1} will stay the same until all extra space to its left has been critically allocated. ■

The following discussion will be based on the comparison of the solution of the proposed algorithm with an arbitrary feasible solution. Let T_i and A_i denote the length of and area allocated for net i, respectively, in the proposed solution. (Note that since it is assumed that the route of net i follows the trail of L_i, the length of L_i is also equal to T_i.) Let T_i^F and A_i^F denote the corresponding quantities in the arbitrary feasible solution.

Lemma 2.4: Consider the final positions of left boundaries in the solution of the proposed algorithm. If all grid cells have been critically allocated between L_1 and L_n, then there exists no convex nonmonotonic left boundary L_i, $1 \leq i \leq n$, such that $T_i > A_i^F$.

Proof: By contradiction, consider the smallest i such that L_i is convex nonmonotic, and $T_i > A_i^F$. It is known that no left boundary can be convex nonmonotonic in the beginning of the proposed algorithm. Hence, before obtaining the final L_i, the proposed algorithm must have tested a configuration L_i' such that $T_i' = A_i^F$, and no more flip is possible on L_i' without increasing its length T_i'. Note that in the feasible solution considered above, all grid cells allocated for nets $[1, i)$ must be within the region between L_1 and L_i', because L_1 is the absolute leftmost boundary for any solution, and L_i' is the rightmost possible boundary if net i has a length of at most A_i^F. Therefore, it can be stated that the set of grid cells allocated for net $[1, i)$ in the feasible solution is a proper subset of the region between L_1 and L_i. Now, consider two cases.

Case 1) There is no L_j $(j < i)$ that is concave nonmonotonic and $T_j > A_j^F$. In other words, for all k, $1 \leq k < i$, it is the case that $T_k \leq A_k^F$; hence, $A_k \leq A_k^F$, since all grid cells between L_1 and L_n are assumed to be critically allocated. However, it has been shown above that the number of grid cells between L_1 and L_i is greater than the number of grid cells allocated for nets $[1, i)$ in the feasible solution if L_i is convex nonmonotonic and $T_i > A_i^F$. This is a contradiction, and the proof is complete for this case.

Case 2) There is at least one L_j $(j < i)$ that is concave nonmonotonic and $T_j > A_j^F$. Now, consider the largest such j value. Since net j already satisfies its minimum-area constraint (i.e., $T_j > A_j^F$), L_{j+1} will not be flipped by net j. Note that this means that L_{j+1} cannot be convex nonmonotonic; hence, $j < i-1$. By the same arguments above, the set of grid cells allocated for nets $[j+1, i)$ in the feasible solution must be a proper subset of the region between L_{j+1} and L_i. Furthermore, it is known that for all k, $j+1 \leq k < i$, it is the case that $T_k \leq A_k^F$; hence, $A_k \leq A_k^F$, due to the assumption of critical allocation. Again, this is a contradiction, and the proof is complete. ■

Theorem 2.5: Assume that a feasible solution exists for a given problem containing nets $[1 \ldots n]$. If the proposed algorithm critically allocates all grid cells between final positions of L_1 and L_n, then it is guaranteed that the solution found will be feasible.

Proof: The proof is based on induction on the number of concave nonmonotonic left boundaries L_i for which $T_i > A_i^F$:

Base case) There exists no L_k $(1 \leq k \leq n)$ for which $T_k > A_k^F$. For all k, $1 \leq k \leq n$, it will be the case that $T_k \leq A_k^F$; and hence, $A_k \leq A_k^F$, due to critical allocation. By contradiction, consider the leftmost net j of which minimum-area constraint has been violated. This means that L_{j+1} has been maximally flipped to the right in the proposed algorithm. On the other hand, any feasible solution for nets $[1, j]$ must be within the region between L_1 and L_{j+1}, by definition. Since it has already been shown that $A_k \leq A_k^F$ for all nets, it must be the case that $A_k = A_k^F$ for all k, $1 \leq k \leq j$. This contradicts with the assumption that minimum-area constraint for net j is not satisfied.

General case) Consider the smallest j $(1 \leq j \leq n)$ for which $T_j \geq A_j^F$. The same proof above applies to the nets $[1, j)$; hence, their minimum-length constraints must have been satisfied. Now, consider the subproblem containing nets $(j, n]$. Since $T_j > A_i^F$ and L_j is concave nonmonotonic, the area between L_j and R_n is a superset of the set of grid cells allocated by nets $(j, n]$ in the feasible solution. Hence, the inductive hypothesis applies for it. ■

Theorem 2.6: If there exists a feasible solution for a given minimum-area maximum-length routing problem, then it is guaranteed that the given algorithm is going to find it.

Proof: The proof is based on (reverse) induction, where the base case contains only the rightmost net n. It is obvious that the theorem is correct for the base case. Now, it will be proven that the theorem holds for an input problem containing nets $[1 \ldots n]$. Consider the smallest j value $(1 \leq j < n)$ such that there are some grid cells not critically allocated between L_j and L_{j+1}. If no such j exists, then the proof is complete due to Theorem 2.5. Otherwise, the final position of L_{j+1} will be the same as its initial position due to Lemma 2.3. This implies that the subproblem containing nets $[j+1 \ldots n]$ remains unmodified; hence, the induction hypothesis applies for it. On the other hand, Theorem 2.5 can be used to show that the solution found for nets $[1 \ldots j]$ is feasible since all grid cells are critically allocated between L_1 and L_j. As a result, a feasible solution will be found (if one exists) for all nets in the given input problem. ■

III. Bus Routing With Minimum–Maximum-Length Constraints

A. Problem Formulation

In this section, the algorithm given in Section II is extended to the problem of river routing with minimum–maximum-length constraints. The main difference here is that the minimum constraints are also given as exact length bounds, instead of minimum-area requirements.

In the original river routing problem [13], the input is a single-layer rectangular routing channel, and a set of two-terminal nets, where all terminals are aligned at the top and the bottom of the channel. In this section, this problem is extended for the case where all nets have to be routed within prespecified minimum–maximum-length bounds.

Most of the existing work on river routing assume monotonic routes both in horizontal and vertical directions, since it does not hurt routability [10]. However, in the present case, explicit detours will be needed to satisfy minimum-length constraints. Hence, monotonicity is assumed only in the vertical direction. In other words, routes are allowed to proceed in three directions: left, right, and down.

B. Routing Algorithm

For this problem, almost the same algorithm given in Fig. 3 is used. The main difference here is that the minimum-length constraint of $\mathrm{Route}(L_i, L_{i+1})$ needs to be checked in each iteration, instead of the minimum-area constraint. Recall that it was trivial to calculate the total area between L_i and L_{i+1}, after each flip on L_{i+1}. However, the maximum length achievable by route i now needs to be calculated, so that whether minimum-length constraint for net i can be satisfied within the interval defined by L_i and L_{i+1} can be determined.

For the purpose of calculating the maximum-length route efficiently, a graph $\mathcal{G}$ is defined, corresponding to the interval between L_i and L_{i+1}. For each row, two vertices are defined on the leftmost and rightmost horizontal grid edges within the interval.[3] Then, an edge is defined from each vertex in row k to each vertex in row $k+1$, corresponding to the paths between the respective cells. The weight of an edge is defined to be the number of grid cells wasted (i.e., not used by the route), if this edge is selected. Fig. 10(a) shows a maximum-length route between two terminal points, and Fig. 10(b) shows the corresponding path in the graph model $\mathcal{G}$.

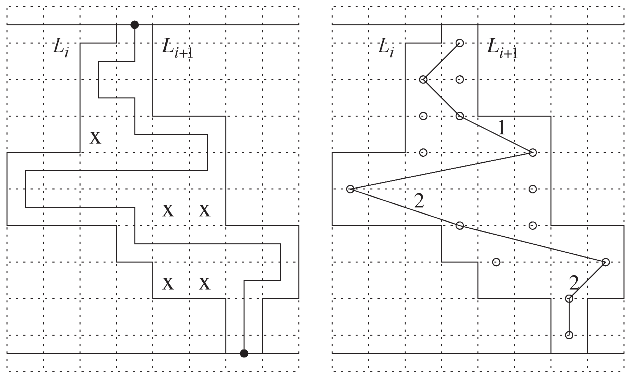


Fig. 10. (a) Sample maximum-length route, where wasted cells are marked with "X." (b) Corresponding shortest path in $\mathcal{G}$. For clarity, only the edges on the shortest path, and only the nonzero edge weights are displayed.

It is straightforward to show that the shortest path between the top and bottom vertices in $\mathcal{G}$ corresponds to the maximum-length route in the original problem; and the total path length in $\mathcal{G}$ is actually equal to the number of grid cells wasted by this route. Observe in the example of Fig. 10(a) that there are five grid cells wasted within the given interval, and the length of the corresponding shortest path in Fig. 10(b) is also five.

Based on this graph model, the time complexity of calculating the maximum length achievable by $\mathrm{Route}(L_i, L_{i+1})$ is $O(W)$, where W denotes the channel width, i.e., the number of rows between the top and bottom terminal points. As discussed in Section II-B, the optimum position of L_{i+1} can be found using two binary searches on the flip sequence. Therefore, $O(\log A_i)$ iterations are needed to route net i, where A_i denotes the total area between L_i and L_{i+1}. As a result, the overall time complexity of routing all nets within minimum–maximum-length bounds becomes $O(nW \log A)$, where n is the number of nets and A is the total area of the channel. Note that since $nW \leq A$, the time bound can also be stated as $O(A \log A)$.

C. Analysis of the Algorithm

It will be shown that the algorithm described in the previous subsection is optimal within a constant factor.

Definition 3.1: Let $\mathrm{Route}(L_i, L_{i+1})$ denote the route with the maximum length within the interval of L_i and L_{i+1}. The waste due to net i [denoted as $\mathrm{waste}(i)$] is defined as the

[3]If there is a single grid edge in a particular row, there will be a single vertex defined. However, for consistency of presentation, assume that the two vertices overlap with each other in this case.

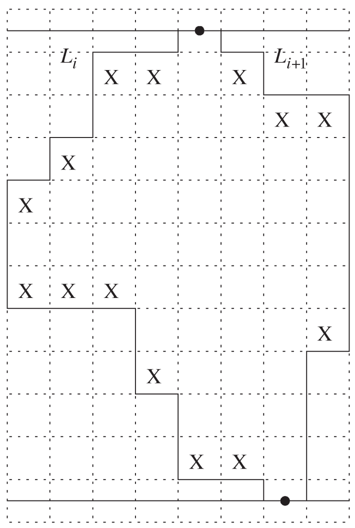


Fig. 11. Sample interval illustrating the upper bound on $\mathrm{waste}(i)$. It is possible to construct a route that wastes at most the grid cells marked with "X."

number of grid cells that are within the respective interval, but not used by $\mathrm{Route}(L_i, L_{i+1})$.

Observe in the example of Fig. 10(a) that $\mathrm{waste}(i)$ is 5, since the maximum-length route of net i cannot use 5 grid cells.

Lemma 3.1: For each net i, $\mathrm{waste}(i)$ is guaranteed to be less than the sum of horizontal spans of the trails corresponding to L_i and L_{i+1}.

Proof: Fig. 11 illustrates the idea with an example. Assume that a path in the corresponding graph $\mathcal{G}$ is constructed by always choosing edges going from a vertex adjacent to L_i to another vertex adjacent to L_{i+1}, or vice versa. It can be shown that such a path will only waste the grid cells marked with "X" in Fig. 11 in the worst case. Since the shortest path in $\mathcal{G}$ (corresponding to the longest route in the channel) is guaranteed to waste at most as many grid cells as this path, the lemma follows. ■

For the following analysis, let C_i denote the leftmost column of the left boundary L_i that has been flipped. If L_i has never been flipped, let it denote the leftmost column of L_i.

Remark 3.2: For each i, $1 \le i < n$, it is the case that $C_i < C_{i+1}$.

Lemma 3.3: The extra grid cells allocated for net i can only be at columns within the interval $[C_i, C_{i+1}]$. In other words, there exists only a single trail between L_i and L_{i+1} outside the interval $[C_i, C_{i+1}]$ (see Fig. 12). For consistency of presentation, C_{n+1} is defined to be the rightmost channel boundary, for a set of n nets.

Proof: Consider a left boundary L_i and the corresponding column C_i. By definition, no segment of L_i to the right of C_i has been flipped. Therefore, any bend on L_i that is to the right of C_i must be either adjacent to the channel boundary, or be due to the blockage of L_{i-1}. Hence, all nets to the left of net i can have only a single trail at columns to the right of C_i. On the other hand, remember that the proposed algorithm always flips the leftmost segment of L_i that is not adjacent to R_i. Therefore, all segments of L_i to the left of C_i (if any) must have been flipped until the corresponding right boundaries. Hence, for each net j, $j \ge i$, there can be at most a single trail to the left of C_i. These concepts are illustrated by an example in Fig. 12. ■

Theorem 3.4: For a given routing problem with minimum–maximum-length constraints, the number of grid cells wasted by all nets will be less than $4H$, where H is the number of grid cells in one row of the channel.

Proof: Due to Lemma 3.3, the grid cells wasted by net i can only be at columns within the interval $[C_i, C_{i+1}]$. In addition, the total size of the horizontal trails of L_i and L_{i+1} in the interval $[C_i, C_{i+1}]$ will be at most $2(C_{i+1} - C_i + 1)$. Therefore, due to Lemma 3.1, it can be stated that $\mathrm{waste}(i) \le 2(C_{i+1} - C_i + 1)$. As a result, the total number of grid cells wasted by n nets will be at most $2(H + n - 1)$, which is less than $4H$. ■

Definition 3.2: Let $\mathcal{C}^W$ denote a river routing problem with channel width W and with length constraints $T_i^{\min}$, $T_i^{\max}$ for each net i. The projection of $\mathcal{C}^W$ onto a channel width of $W - k$ (denoted as $\mathcal{C}^{W-k}$) is defined to be the same routing problem as $\mathcal{C}^W$, except that the channel width in $\mathcal{C}^{W-k}$ is $W - k$, and length constraints are $T_i^{\min} - k$ and $T_i^{\max} - k$, respectively, for each net.

For the rest of the analysis, the following notations will be used:

$\mathcal{C}_L^W$	given minimum-length maximum-length routing problem with length constraints $T_i^{\min}$ and $T_i^{\max}$, for each net i;
$\mathcal{C}_L^{W-3}$	projection of $\mathcal{C}_L^W$ onto channel width $W-3$, with length constraints $T_i^{\min} - 3$ and $T_i^{\max} - 3$, for each net i;
$\mathcal{C}_A^{W-3}$	minimum-area maximum-length routing problem obtained by replacing minimum-length constraints of $\mathcal{C}_L^{W-3}$ with minimum-area constraints $A_i^{\min}$ for each net i, where $A_i^{\min}$ is defined as follows: if the minimum detour required to satisfy the minimum-length constraint of net i in $\mathcal{C}_L^{W-3}$ is even, then $A_i^{\min} = T_i^{\min} - 3$; otherwise,[4] $A_i^{\min} = T_i^{\min} - 2$;
$\mathcal{S}_A^{W-3}$	solution to $\mathcal{C}_A^{W-3}$ produced by the minimum-area maximum-length routing algorithm given in Section II;
$\mathcal{S}_A^W$	projection of $\mathcal{S}_A^{W-3}$ to channel width of W (see Appendix for the definition of solution projection);
$\mathcal{S}_L^W$	solution obtained after routing each net within the area allocated for it in $\mathcal{S}_A^W$ (as in the example of Fig. 10).

Our objective in the following analysis is to show that if a feasible solution to $\mathcal{C}_L^{W-3}$ exists, then the proposed minimum-length maximum-length routing algorithm is guaranteed to find a feasible solution to $\mathcal{C}_L^W$.

Remark 3.5: A feasible solution to $\mathcal{C}_L^{W-3}$ is also a feasible solution to $\mathcal{C}_A^{W-3}$.

Remark 3.6: If there exists a feasible solution to $\mathcal{C}_A^{W-3}$, then $\mathcal{S}_A^{W-3}$ is guaranteed to be feasible (due to Theorem 2.6).

[4]Let MD_i denote the Manhattan distance between the terminals of net i. It is obvious that any valid route for net i will have a length of $\mathrm{MD}_i + d_i$, where d_i is an even number. If the minimum detour required to satisfy the min-length constraint $T_i^{\min} - 3$ is an odd number, a feasible solution will have a length of at least $T_i^{\min} - 2$, since odd detour is not possible.

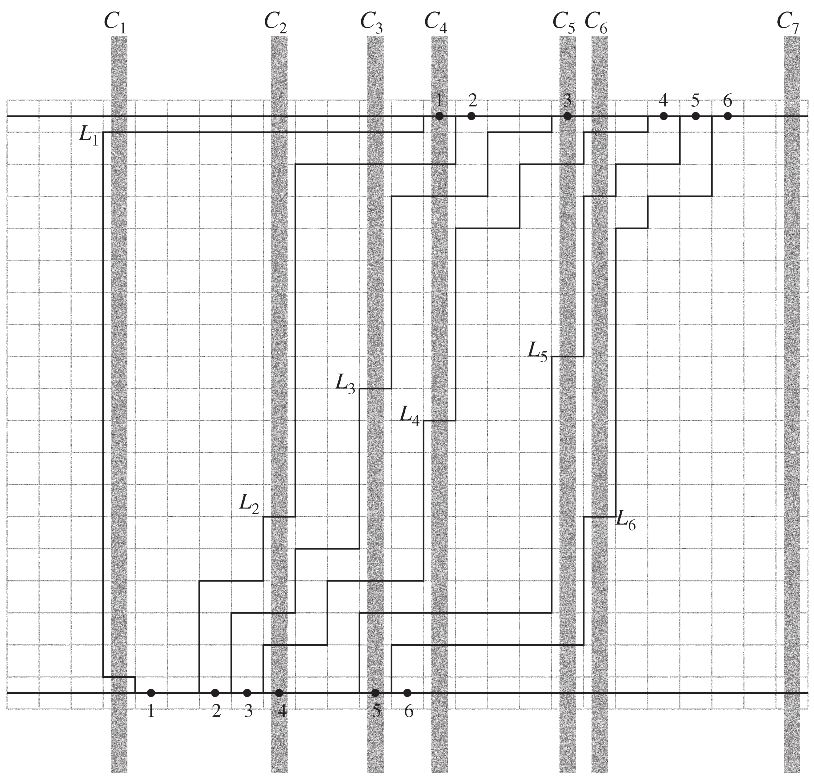


Fig. 12. Final positions of left boundaries are illustrated for six nets. For each left boundary L_i, the leftmost flipped column (if any) C_i is also highlighted. The extra grid cells between L_i and L_{i+1} exist only at columns within the interval $[C_i, C_{i+1}]$. For consistency, C_7 is defined to be the rightmost channel boundary.

Lemma 3.7: It is possible to obtain a feasible S_L^W, which satisfies all length constraints of $\mathcal{C}_L^W$, by projecting S_A^{W-3} to S_A^W, and then routing each net within the area allocated for it in S_A^W.

Proof: The detailed proof is given in the Appendix. ■

Remark 3.8: S_L^W is within the solution space explored by the proposed minimum-length maximum-length routing algorithm.

Theorem 3.9: Assume that $\mathcal{C}_L^W$ is the given minimum-length maximum-length routing problem, and $\mathcal{C}_L^{W-3}$ is the projected problem onto channel width $W-3$. If a feasible solution exists for $\mathcal{C}_L^{W-3}$, then the proposed algorithm is guaranteed to find a feasible solution for $\mathcal{C}_L^W$.

Proof: From Remarks 3.5 and 3.6, and Lemma 3.7, it is known that S_L^W will be feasible if there exists a feasible solution to $\mathcal{C}_L^{W-3}$. From Remark 3.8, the proposed algorithm will find either S_L^W, or another feasible solution. ■

D. Discussions and Practical Considerations

Recall that the given algorithm in Section II finds the optimal solution if the minimum constraints are given as area constraints. Theorem 3.4 suggests that if a prespecified amount of routing area is allocated for each net using this optimal algorithm, then all routing resources within the allocated areas will be successfully used for length extension, except for at most a number of $4H$ grid cells. Since the total area of the channel is WH, where W is the channel width, and typically is much larger than 4, the number of grid cells wasted will be negligible.

On the other hand, Theorem 3.9 makes a stronger statement, giving an approximation factor for the actual routing problem with minimum–maximum-length constraints. One implication of this theorem is that if a bus routing problem that has a feasible solution is given, the lengths of all nets can be guaranteed to match by extending the channel width by 3 units. The reason is that all length constraints $T_i^{\min}$ and $T_i^{\max}$ are increased by exactly the same amount (i.e., 3 units); and the proposed algorithm is guaranteed to find a feasible solution for the extended channel.

Furthermore, typical industrial circuits have channel widths containing hundreds, or even thousands of grid cells. One can argue that if a feasible solution exists for the original channel width W, most probably, a feasible solution will exist for the channel width $W-3$, since the difference will be almost negligible. Therefore, in practice, the proposed algorithm will find the feasible solution for the original problem without the need for extending the channel length.

Note here that the proposed algorithm in this section uses a certain type of snaking, and it does not explore the whole solution space—e.g., the route in Fig. 10 cannot go up-and-down, since it must be monotonic in the channel direction. Yet, Theorem 3.9 states that the solution found by the proposed algorithm is guaranteed to be close to the (most general) optimal solution. In other words, the proposed solution will be sufficiently close to optimum even though only a particular type of snaking is considered. However, it is possible to use different types of length-extension methods to reduce the number of bends (see discussion below). Note here that the general

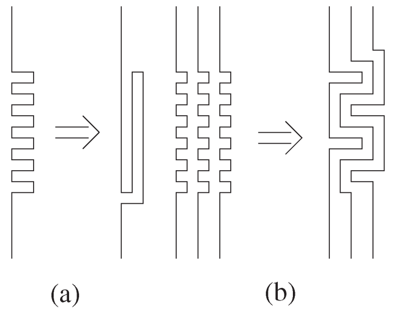


Fig. 13. Sample postprocessing techniques to reduce the number of bends. (a) Adjacent jogs are replaced with a longer segment. (b) Jogs for adjacent nets are merged together.

framework, given in Fig. 3, allows using such alternative approaches, as long as it is possible to check the maximum length achievable for net i within the area between L_i and L_{i+1}. Yet, another practical approach can be to use the minimum-area maximum-length routing algorithm given in Section II-B first, and then to perform any type of length extension in postprocessing using the allocated areas.

The type of snaking used in this algorithm (as illustrated in Fig. 2) is also frequently used in current industrial circuits [7]. However, it is possible to reduce the number of bends by using some postprocessing methods, if necessary. Two possible techniques are illustrated in Fig. 13. Here, Fig. 13(a) gives a straightforward replacement of adjacent jogs with a single longer detour. On the other hand, in Fig. 13(b), the jogs for three nets are merged to obtain a solution with less number of bends.

IV. Experimental Results

Experiments have been performed to compare the proposed algorithm with a recently proposed LR-based approach [8]. All algorithms in this section have been implemented in C++ and executed on an Intel Pentium 4 2.4 GHz system with 512 MB memory and on a Linux operating system.

For the purpose of illustration, the proposed algorithm has been first applied on a test circuit, shown in Fig. 14. Each net in this circuit has prespecified minimum–maximum-length constraints, and the solution displayed satisfies all those constraints.

Then, experiments have been performed on test circuits, as demonstrated in Table I. Here, "avg. spacing" is measured in terms of the number of grid cells between terminal points of adjacent nets, and it indicates how dense the problem is. On the other hand, the column "length stdev" gives the standard deviation in net target lengths. Each problem in this table contains around 200–300 nets, with individual length constraints for each net. The grid size for the smallest circuit in this table is 390 × 200, and the largest one is 776 × 290. The last three problems here have been extracted from an IBM design, corresponding to the bus routing problems between MCM, memory, and STI modules. Here, layer assignment and routing inside chips have been performed by the previous phases of the routing system, as described in Section I, and the input for the bus routing problem is a set of noncrossing nets on each layer. While the first seven circuits in this table have single layers, the IBM circuits have multiple layers.

As seen in this table, the proposed algorithm performs significantly better than the LR-based approach on most circuits, in terms of both quality and run time. As the average spacing between nets decrease, the solution quality for LR-based approach degrades, and it takes more time to find a routing solution. The main reason for this is that the LR-based approach uses a variant of Pathfinder [16], [17] algorithm in the low level, where routing conflicts are resolved through negotiations. As the problems get denser, these negotiations take more and more time, and they do not always successfully lead to a good result. On the other hand, the algorithm proposed in this paper has optimality guarantees, and it finds a good solution as long as it exists. Observe that, the proposed algorithm has also very good runtime characteristics, since the underlying flip operations can be done in a fast and efficient way.

V. Conclusion

Two algorithms have been proposed for high-performance bus routing problem. The first algorithm is for the problem of routing with minimum-area maximum-length constraints, where length matching is assumed to be performed in postprocessing using the allocated areas. The second algorithm extends these ideas to the problem where minimum constraints are given as exact length bounds. The first algorithm is proven to be optimal, while the second one is provably near-optimal. The respective time complexities of these algorithms are $O(A)$ and $O(A \log A)$, where A is the area of the intermediate region between chips. Our experiments demonstrate the effectiveness of these algorithms compared to a recently proposed approach. These algorithms can be implemented as a part of large-scale routing system for high-performance PCBs.

Appendix
Solution Projection to a Wider Channel

It has been indicated in Lemma 3.7 that it is possible to project the solution of $\mathcal{C}_A^{W-3}$ to channel width W for the purpose of obtaining a feasible solution to $\mathcal{C}_L^W$. In this Appendix, the details of this proof will be given, by using the same notation in Section III-C for $\mathcal{C}_L^W$, $\mathcal{C}_L^{W-3}$, $\mathcal{C}_A^{W-3}$, $\mathcal{S}_A^{W-3}$, $\mathcal{S}_A^W$, and $\mathcal{S}_L^W$. It is important to note here that solution projection is not an actual part of the algorithm that has been proposed, but it is used only to prove Lemma 3.7.

Definition A.1: Assume that the proposed minimum-area maximum-length routing algorithm is applied on $\mathcal{C}_A^{W-3}$. As described in Section II, a number of flips are performed on each left boundary L_i during the execution of the algorithm. Let $\#\mathrm{fc}_i^{W-3}$ denote the number of columnwise flips and $\#\mathrm{fs}_i^{W-3}$ the number of single flips on the last flipped column of left boundary L_i [see Fig. 15(a)]. The projected solution $\mathcal{S}_A^W$ is defined to be constructed as follows.

- Find the leftmost and rightmost boundaries L_i and R_i for each net i for channel width W.

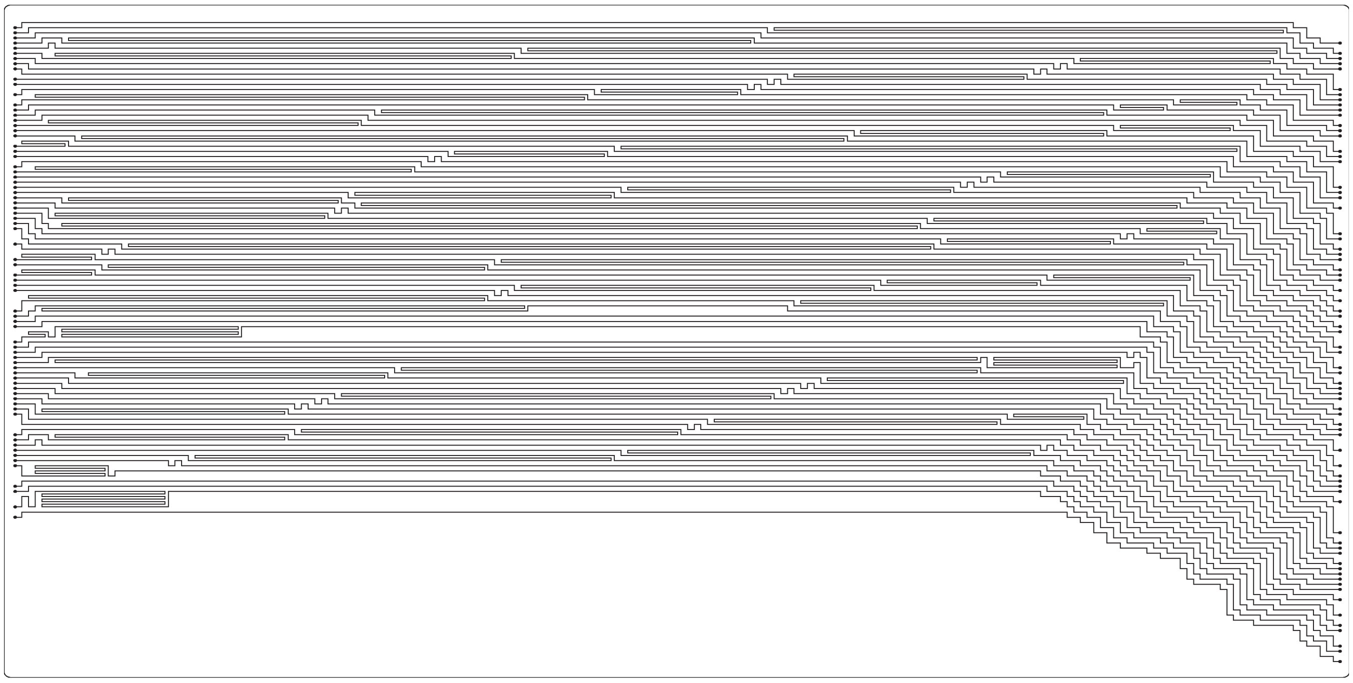


Fig. 14. Output of our algorithm on a bus routing problem. Only part of the circuit is displayed here due to space limitations. The postprocessing technique illustrated in Fig. 13(a) has been applied on the output of the algorithm given in Section III to obtain the final routing solution. Note that although a small problem is chosen here for illustration purposes, typical industrial problems have channel widths on the order of hundreds, or even thousands, and the proposed algorithm is scalable for such large problems.

TABLE I
COMPARISON OF OUR ALGORITHM WITH THE LR-BASED APPROACH

Circuit	Average Spacing	Length stdev	Our Algorithm: Number of Nets Failed	Our Algorithm: Time (minute:second)	LR-Based: Number of Nets Failed	LR-Based: Time (minute:second)
C1	2.59	31.96	0	0:02	1	12:54
C2	2.80	47.80	0	0:02	2	7:45
C3	2.18	66.12	1	0:02	7	20:41
C4	1.81	44.62	0	0:03	18	21:30
C5	1.53	11.13	0	0:02	21	8:31
C6	1 64	41 26	0	0:02	29	17·58
C7	1.52	53.46	1	0:02	86	45:34
IBM_1	7.75	35.78	3	0:02	3	6:32
IBM_2	6.41	45.98	1	0:02	1	4:31
IBM_3	9.16	73.76	0	0:03	0	4:09

- For each net i:
  - Perform $\#\mathrm{fc}_i^W$ columnwise flips on L_i, where $\#\mathrm{fc}_i^W = \#\mathrm{fc}_i^{W-3}$.
  - Perform $\#\mathrm{fs}_i^W$ single flips on the next column of L_i, where $\#\mathrm{fs}_i^W$ is defined as follows: If $\#\mathrm{fs}_i^{W-3}$ is equal to 0, then $\#\mathrm{fs}_i^W = 0$; else if L_i is convex nonmonotonic, then $\#\mathrm{fs}_i^W = \#\mathrm{fs}_i^{W-3} + 2$; otherwise, $\#\mathrm{fs}_i^W = \#\mathrm{fs}_i^{W-3} + 1$.
- The route of net i in $\mathcal{S}_A^W$ is defined to follow the trail of L_i as close as possible; and all the grid cells between L_i and L_{i+1} are defined to be allocated by net i.

Projection of a concave nonmonotonic left boundary from channel width $W-3$ to channel width W is illustrated in Fig. 15 as an example.

For the rest of the analysis, the following notation will be used:

C_i	leftmost column of left boundary L_i that has been flipped (if L_i has never been flipped, let it denote the leftmost column of L_i);
A_i^{W-3}, A_i^W	number of grid cells (i.e., area) allocated for net i in $\mathcal{S}_A^{W-3}$ and $\mathcal{S}_A^W$, respectively;
T_i^{W-3}, T_i^W	length of net i in $\mathcal{S}_A^{W-3}$ and $\mathcal{S}_A^W$, respectively;
$T_i^{\min}$	minimum-length constraint for net i in $\mathcal{C}_L^W$.

Remark A.1: The C_i values of $\mathcal{S}_A^W$ are the same as those of $\mathcal{S}_A^{W-3}$.

Lemma A.2: For any i, $1 \le i \le n$, if $C_{i+1} = C_i + 1$ in $\mathcal{S}_A^W$, and if $\mathcal{C}_L^{W-3}$ has a feasible solution, then it is guaranteed that the minimum-length constraint of net i in $\mathcal{S}_L^W$ is satisfied.

Proof: It is known that $A_i^W \ge A_i^{W-3} + 3$, due to the increase in the channel width. Since $C_{i+1} = C_i + 1$, and due to Lemma 3.3, the extra grid cells allocated for net i can only be at columns C_i and C_{i+1}. Fig. 16 illustrates examples for different left boundary types. Observe that if the number of extra grid

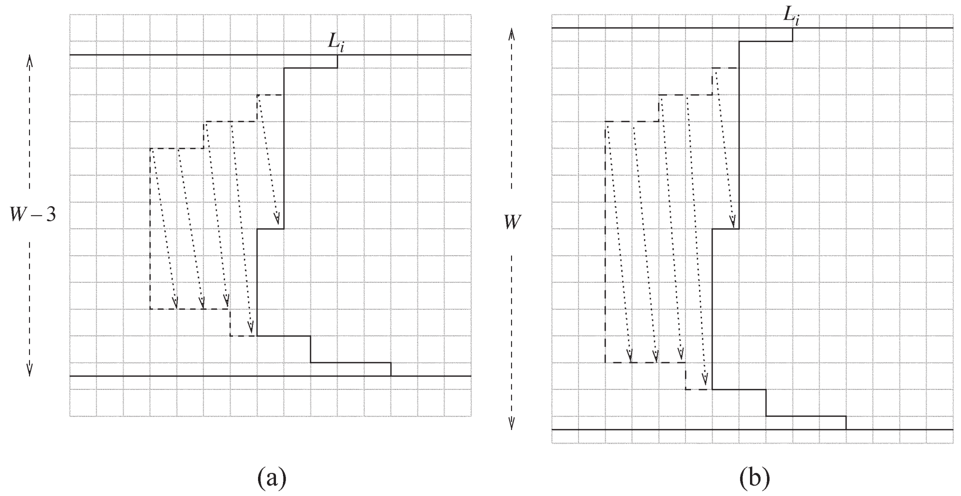


Fig. 15. Illustration of solution projection from channel width $W-3$ to W. The flips performed are shown by dotted arrows, and the final positions of L_i are shown by solid lines. (a) Original output of the minimum-area maximum-length routing algorithm, where $\#\text{fc}_i^{W-3} = 4$ and $\#\text{fs}_i^{W-3} = 5$. (b) Projected solution, where $\#\text{fc}_i^{W} = 4$ and $\#\text{fs}_i^{W} = 6$.

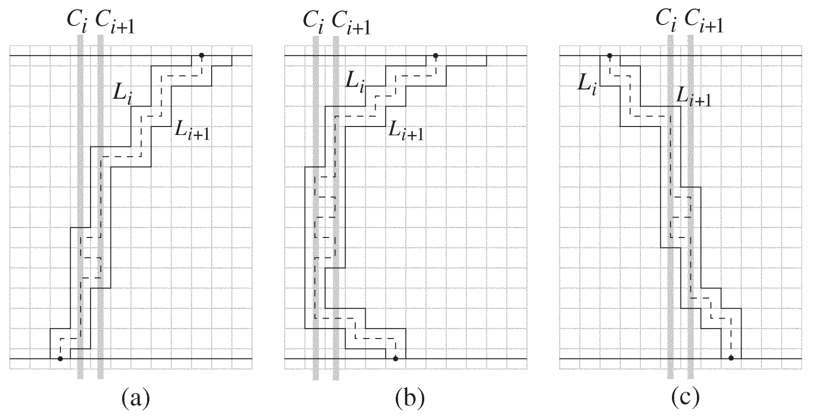


Fig. 16. Examples illustrating the final positions of different left boundary types, where $C_{i+1} = C_i + 1$. Dashed lines indicate the route for net i between left boundaries L_i and L_{i+1} for each case. If the number of extra grid cells between L_i and L_{i+1} is even, then no grid cell is wasted by the corresponding route.

cells allocated for net i is even, then net i can be routed between L_i and L_{i+1} without wasting any grid cell. In this case, $T_i^W = A_i^W \geq A_i^{W-3} + 3 \geq T_i^{\min}$ holds. Remember from definition of $\mathcal{C}_A^{W-3}$ in Section III-C that the number of extra grid cells required to satisfy the minimum-area constraint of net i in $\mathcal{C}_A^{W-3}$ is always even. Since the number of extra grid cells in $\mathcal{S}_A^W$ will be at least as large as the number of extra grid cells in $\mathcal{S}_A^{W-3}$, the lemma follows. ∎

Lemma A.3: For any i, $1 \leq i \leq n$, if $C_{i+1} > C_i + 1$ in $\mathcal{S}_A^W$, and if C_L^{W-3} has a feasible solution, then it is guaranteed that the minimum-length constraint of net i in $\mathcal{S}_L^W$ is satisfied.

Proof: The proof is based on case-by-case analysis of different types of left boundaries L_i and L_{i+1}. Note that since a left boundary can be one of the four types described in Definition 2.1, there are 16 different cases for L_i and L_{i+1}. Fig. 17 illustrates four of these cases, and it is straightforward to extend the ideas here for the remaining 12 cases. The following notations are used in Fig. 17:

s_i	number of columns between (but excluding) C_i and C_{i+1}, i.e., $s_i = C_{i+1} - C_i - 1$;
$\text{waste}(i)$	number of grid cells wasted by the longest route within the region between L_i and L_{i+1} (see Definition 3.1);
$\Delta\text{area}(i)$	area increase between L_i and L_{i+1} after solution projection, i.e., $\Delta\text{area}(i) = A_i^W - A_i^{W-3}$.

Due to Lemma 3.3, the grid cells wasted between L_i and L_{i+1} can only be at columns within the interval $[C_i, C_{i+1}]$. To find an upper bound for $\text{waste}(i)$, a route that snakes between boundaries L_i and L_{i+1} can be constructed, as described in the proof of Lemma 3.1. In each case illustrated in Fig. 17, the grid cells that would be wasted in the worst case by such a route are marked with an "X." Since the route with the maximum length is guaranteed to waste at most this many grid cells, the number of grid cells marked with "X" in each case gives an upper bound for $\text{waste}(i)$.

Due to the increase of channel width from $W-3$ to W, it is known that the number of grid cells at each column will increase by 3. To calculate $\Delta\text{area}(i)$, the extra grid cells that are in the region between L_i and L_{i+1} should be considered. From Definition A.1, it is known that no segment of L_i in the interval

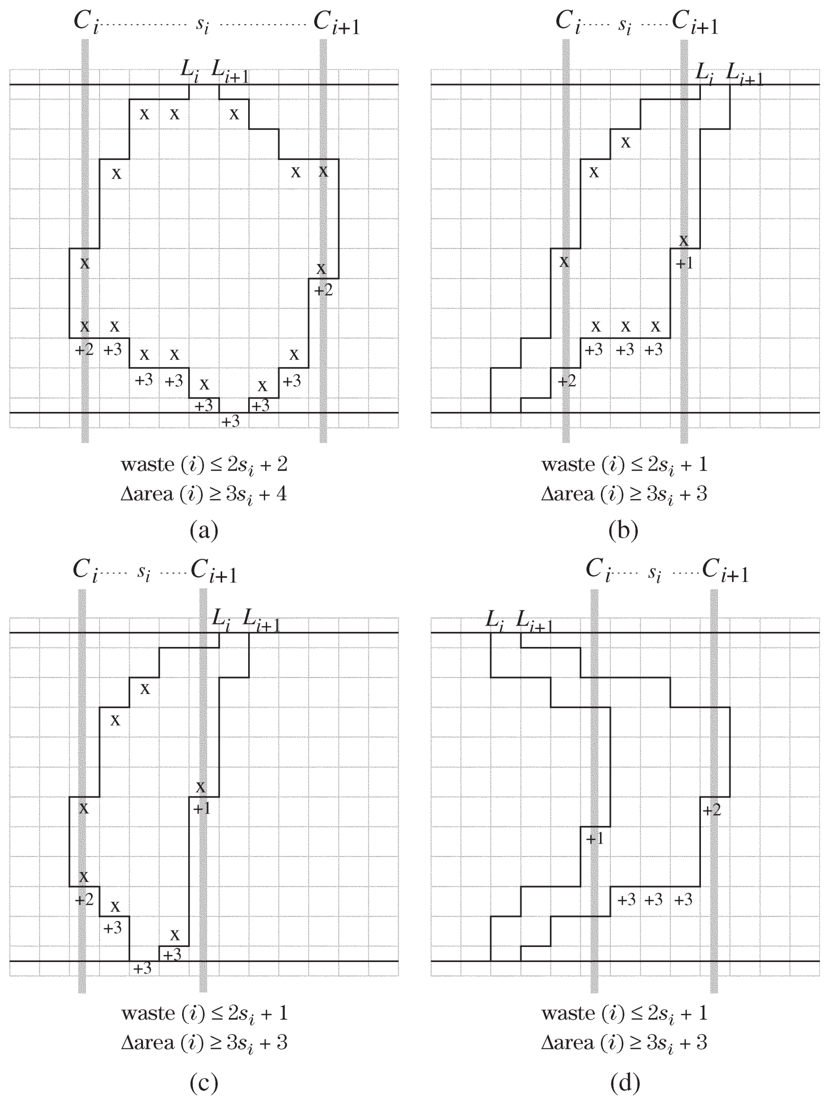

C_i ······ s_i ······ C_{i+1}
L_i L_{i+1}
waste $(i) \leq 2s_i + 2$
Δarea $(i) \geq 3s_i + 4$
(a)
waste $(i) \leq 2s_i + 1$
Δarea $(i) \geq 3s_i + 3$
(b)
waste $(i) \leq 2s_i + 1$
Δarea $(i) \geq 3s_i + 3$
(c)
waste $(i) \leq 2s_i + 1$
Δarea $(i) \geq 3s_i + 3$
(d)

Fig. 17. Calculation of $\mathrm{waste}(i)$ and $\Delta\mathrm{area}(i)$ values for different types of $L_i - L_{i+1}$ pairs. The grid cells marked with "X" are the cells that will be wasted in the worst case by the longest route for net i. The number of such grid cells gives an upper bound for $\mathrm{waste}(i)$. The increase in the number of grid cells (due to solution projection) at each column between L_i and L_{i+1} is indicated below each column. Note that the lower bound for $\Delta\mathrm{area}(i)$ is calculated as the sum of these values between C_i and C_{i+1}.

(C_i, C_{i+1}) has been flipped. On the other hand, the segments of L_{i+1} (if any) in the interval (C_i, C_{i+1}) have been flipped maximally, i.e., until the right boundary R_{i+1}. Therefore, it can be stated that the three extra grid cells at each column in the interval (C_i, C_{i+1}) are all in the region between L_i and L_{i+1}. To find the corresponding area increase at columns C_i and C_{i+1}, the last flip performed on L_i and L_{i+1} should be considered, respectively. These flips are illustrated in Fig. 17 for each case, and the corresponding increase at these columns are marked below. For instance, if L_i is concave nonmonotonic as in Fig. 17(a), then the corresponding area increase on column C_i will be 2. The reason is that the last flip on L_i will have $\#\mathrm{fs}_i^W = \#\mathrm{fs}_i^{W-3} + 1$ (see Definition A.1), and one out of three extra grid cells on column C_i will be outside the region between L_i and L_{i+1}.

Based on these considerations, the values of $\mathrm{waste}(i)$ and $\Delta\mathrm{area}(i)$ are given for each case in Fig. 17. Note that since $C_{i+1} > C_i + 1$, s_i will have a value of at least 1. Therefore, for each case, $\Delta\mathrm{area}(i) - \mathrm{waste}(i) \geq 3$ holds. Based on this, it can be stated that $A_i^W \geq A_i^{W-3} + \mathrm{waste}(i) + 3 \geq T_i^{\min} + \mathrm{waste}(i)$. This means that a routing solution for each net i that satisfies the minimum-length constraint $T_i^{\min}$ within the region between L_i and L_{i+1} in $\mathcal{S}_L^W$ can be found. ■

Remark A.4: The proof of Lemma 3.7 is complete due to Lemmas A.2 and A.3.

References

[1] K. D. Boese, J. Cong, A. B. Kahng, K. S. Leung, and D. Zhou, "On high-speed VLSI interconnects: Analysis and design," in *Proc. Asia-Pacific Conf. Circuits Systems*, Sydney, Australia, 1992, pp. 35–40.

[2] J. Cong, A. B. Kahng, G. Robins, M. Sarrafzadeh, and C. K. Wong, "Provably good performance-driven global routing," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 11, no. 6, pp. 739–752, Jun. 1992.

[3] A. E. Dunlop, V. D. Agrawal, D. N. Deutsch, M. F. Jukl, P. Kozak, and M. Wiesel, "Chip layout optimization using critical path weighting," in *Proc. Design Automation Conf.*, Albuquerque, NM, 1984, pp. 133–136.

[4] E. Kuh, M. A. B. Jackson, and M. Marek-Sadowska, "Timing-driven routing for building block layout," in *Proc. IEEE Int. Symposium Circuits and Systems*, Philadelphia, PA, 1987, pp. 518–519.

[5] S. Lee and M. D. F. Wong, "Timing-driven routing for FPGAs based on Lagrangian relaxation," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 22, no. 4, pp. 506–510, Apr. 2003.

[6] S. Prastjutrakul and W. J. Kubitz, "A timing-driven global router for custom chip design," in *Proc. IEEE Int. Conf. Computer-Aided Design*, Santa Clara, CA, 1990, pp. 48–51.

[7] J. Ludwig, IBM Systems Group, private communication, 2004.

[8] M. M. Ozdal and M. D. F. Wong, "Length matching routing for high-speed printed circuit boards," in *Proc. IEEE Int. Conf. Computer-Aided Design*, San Jose, CA, Nov. 2003, pp. 394–400.

[9] L. W. Ritchey, "Busses: What are they and how do they work?" *Printed Circuit Design Mag.*, 2000.

[10] T. Lengauer, *Combinatorial Algorithms for Integrated Circuit Layout*. New York: Wiley, 1990.

[11] C. Hsu, "General river routing algorithm," in *Proc. 20th Design Automation Conf.*, Miami, FL, 1983, pp. 578–582.

[12] C. E. Leiserson and R. Y. Pinter, "Optimal placement for river routing," in *Proc. CMU Conf. VLSI Systems and Computations*, 1981.

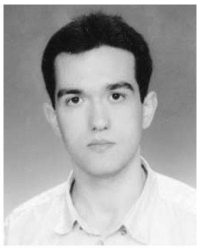
[13] R. Y. Pinter, "On routing two-point nets across a channel," in *Proc. 19th Design Automation Conf.*, Las Vegas, NV, 1982, pp. 894–902.

[14] ——, "River routing: Methodology and analysis," in *Proc. 3rd Caltech Conf. VLSI*, Pasadena, CA, 1983, pp. 141–163.

[15] H. Zhou and M. D. F. Wong, "Optimal river routing with crosstalk constraints," *ACM Transact. Des. Automat. Electron. Syst.*, vol. 3, no. 3, pp. 496–514, 1998.

[16] V. Betz and J. Rose, "Vpr: A new packing, placement and routing tool for FPGA research," in *Proc. 7th Int. Workshop Field-Programmable Logic*, London, U.K., 1997, pp. 213–222.

[17] C. Ebeling, L. McMurchie, S. A. Hauck, and S. Burns, "Placement and routing tools for the triptych FPGA," *IEEE Trans. Very Large Scale (VLSI) Integr. Syst.*, vol. 3, no. 4, pp. 473–482, Dec. 1995.



Muhammet Mustafa Ozdal received the B.S. degree in electrical engineering and the M.S. degree from Bilkent University, Ankara, Turkey, in 1999 and 2001, respectively, and the Ph.D. degree in computer science from the University of Illinois at Urbana-Champaign in 2005.

He is currently with Intel Corporation, Hillsboro, OR. His research interests include computer-aided design algorithms for very large scale integration (VLSI) with primary focus on physical design.

Martin D. F. Wong (M'88–SM'04) received the B.Sc. degree in mathematics from the University of Toronto, Toronto, ON, Canada, in 1979, the M.S. degree in mathematics from the University of Illinois at Urbana–Champaign in 1981, and the Ph.D. degree in computer science from the University of Illinois at Urbana–Champaign in 1987.

He is currently Professor of Electrical and Computer Engineering at the University of Illinois at Urbana–Champaign (UIUC). Before he joined UIUC, he was a Bruton Centennial Professor of Computer Sciences at the University of Texas at Austin (UT-Austin). His research interests are in computer-aided design (CAD) of very large scale integrated (VLSI) circuits, design and analysis of algorithms, and combinatorial optimization. He has published over 250 technical papers and has graduated 31 Ph.D. students. He is the coauthor of *Simulated Annealing for VLSI Design* (Kluwer Academic, 1988) and two invited articles in the *Wiley Encyclopedia of Electrical and Electronics Engineering* (1999).

Dr. Wong received the 2000 IEEE CAD Transactions Best Paper Award for his work on interconnect optimization. He also received best paper awards at DAC-86 and ICCD-95 for his work on floorplan design and routing, respectively. His ICCAD-94 paper on circuit partitioning has been included in the book, *The Best of ICCAD—20 Years of Excellence in Computer Aided Design*, published in 2002. He was the General Chair of the 1999 ACM International Symposium on Physical Design (ISPD-99) and was the Technical Program Chair of the same conference in 1998 (ISPD-98). He is on the Steering Committee of ISPD (ISPD-01, ISPD-02, and ISPD-05). He also regularly serves on the technical program committees of many other VLSI conferences (e.g., DAC, ICCAD, ISPD, DATE, ASPDAC, ISCAS, FPGA, SASIMI, GLS-VLSI, SSMSD). He has served as an Associate Editor for IEEE TRANSACTIONS ON COMPUTERS (1985–2000) and Guest Editor of four special issues for IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN OF INTEGRATED CIRCUITS AND SYSTEMS. He is currently on the Editorial Boards of *ACM Transactions on Design Automation of Electronic Systems* and the IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN OF INTEGRATED CIRCUITS AND SYSTEMS.