PERFORMANCE DRIVEN GLOBAL ROUTING FOR STANDARD CELL DESIGN *

Jason Cong Patrick H. Madden

April 7, 1997

Abstract

Advances in fabrication technology have resulted in a continual shrinkage of device dimensions. This has resulted in smaller device delays, greater resistance along interconnect wires, and a greater impact of interconnect on total system performance. These changes have driven a considerable number of studies on single-net interconnect optimization, but relatively little work has been done to integrate the results on single-net optimization with the problem of global routing and interconnect optimization for the entire circuit.

In this paper, we present the DECIMATE global router for performance driven standard cell design. The router applies both interconnect topology optimization and variable-width wire sizing optimization results to the global routing problem, while maintaining routing areas that are comparable with TimberWolf Systems' well-known commercial global router.

Optimal selection of interconnection structures is shown to be an NP-Hard problem; we provide a simple heuristic for the problem, and show that it is effective with experiments on industry benchmarks. Under the Elmore delay model, our global router produces as much as a 35% reduction in critical path delay over TimberWolf Systems' global router, while path length reductions are as large as 52%. Circuit area optimization is performed taking into account variably-sized wires, fixed routing topologies, and pre-existing obstacles; an improved cost function obtains as much as an 11.6% reduction in channel density over the result in [16].

1 Introduction

With the advent of deep submicron design, a number of process parameters have changed, resulting in an increasing importance for interconnect optimization. Interconnect delay can now consume from 50% to 70% of the clock cycle in many cases [3]; reduction in interconnect delay will have a significant impact on the overall performance of the circuit.

To address these new design parameters, a number of approaches to single net interconnect topology optimization have been proposed, such as bounded-radius bounded-cost trees[10], AHHK trees[1], maximum performance trees[9], A-trees[14], IDW/CFD trees[21], SORT and SERT trees [4], and P-Trees[28]. These methods consider both the traditional concern of low total tree length, and also the path length or Elmore delay between the source node and the timing-critical sink nodes. Many of these algorithms have been surveyed in [24] and [12].

In addition to topology optimization, sizing of interconnect wires has also been shown effective for delay reduction [13, 15, 27, 33, 8, 31]. Traditionally, minimum width wires were used for most connections; for high performance submicron design, however, this may be inappropriate.

*This work is partially supported by DARPA/ITO under Contract J-FBI-93-112 and NSF under Young Investigator Award MIP-9357582.

The global routing problem (with area minimization objectives) is NP-hard in general, and has been studied by a number of researchers.

A hierarchical decomposition of the global routing problem has been used with some success [18, 6], dividing the core area into progressively smaller regions. Simulated annealing has also been applied to the global routing problem [26], where both net topologies and cell positions may be affected. Linear programming methods have been used to select the assignment of segments in [2, 22], in order to minimize the maximum density across a channel. Related to the LP methods are those based on network flow or multicommodity flow models [30, 36, 7, 34]. In [20], the authors use path-based timing constraints and utilize the features specific to bipolar design to optimize both the delay and area of the global routing result.

The global router described in [16, 17] provides the basis for the work in this paper. It involves two phases, a constructive step in which feedthroughs are inserted, and an iterative deletion step which first constructs redundant connection graphs for each net, and then removes redundant edges to minimize overall channel density.

Most of the works on interconnect optimization deal with only single net optimization, and do not address the issues of how to integrate the optimization techniques into global routing. On the other hand, most existing global routing approaches fail to address global delay minimization for deep submicron design. This paper presents the DECIMATE global router, which addresses both of these problems, and offers the following features.

- Through the use of high performance interconnect topologies, the router addresses global path delay concerns for timing driven circuit design. We apply the topology algorithms of [14, 23, 5] and the wiresizing work of [15, 13] in our global router.
- Area optimization takes into account both the timing-critical and non-timing-critical nets, as well as variable width routing and pre-existing congestion. The router also avoids the rip-up and re-route approach common in global routing, obtaining low area solutions directly. Area results are competitive with the well respected TimberWolf Systems' global router on widely available industry benchmarks, showing that interconnect optimization can be considered with little or no sacrifice in circuit area.

2 Problem Formulation

The standard cell model has been well studied, and is widely used for ASIC designs. Standard cell design allows the construction of relatively high performance circuits with moderately low design effort.

In standard cell design, a circuit is composed of a set of logic *cells*; each cell contains a number of connection points, commonly called *pins* or *ports*. There may be *equivalent pins*, which allow connection at multiple locations on a cell (these points are electrically equivalent). A *net* is a set of pins which must be interconnected. Cells are arranged into horizontal rows. Interconnection takes place in horizontal *channels* between the rows, using *channel segments*. In two layer design, one metal layer is used for the horizontal channel segments, while the second metal layer is used for vertical connections between the cells and the channel segments. If three or more layers are available, *over the cell* routing may be used. This paper assumes a two-layer model, and discusses extensions to three or more layers in Section 5. Through the insertion of *feedthroughs* in the row or the utilization of *built in feedthroughs*, inter-channel connectivity can be obtained.

The traditional global routing problem is to determine the connection pattern for each net to minimize the overall routing area. The *density* of a channel corresponds to the maximum number of horizontal segments passing any point within the channel. The density metric has been found to be an accurate estimate of the routing area required by the channel; the extension of this metric to variably sized interconnect wires is straightforward. The area required to route a circuit is a function of the longest cell row and the density across all channels (with consideration of variable wire widths).

In high performance design, we are concerned not only with circuit area, but also with signal delay and operating rates. Signal delay through a complex VLSI circuit involves a series of interconnect nets and logic gates. To maximize device operating rates, delay along the *critical path* (the longest delay path) through the circuit must be minimized. Note that determination of the critical path through a circuit is an NP-hard problem in general (due to the false path problem). For the purposes of this discussion, we consider the *static* critical paths, which are the maximum delay paths from primary inputs or flip-flop outputs to primary outputs or flip-flop inputs, where the delay of a path is the *sum* of gate delays and interconnect delays along the path. If we wish to perform delay optimization, we must select a subset of nets for interconnect optimization, while avoiding large penalties in terms of circuit area.

As an example, a pair of possible interconnect topologies are shown in Figure 1. The first example shows an interconnect solution for a timing critical net minimized for total wire length (a traditional global routing objective). While wire length is low, delay from the driver (at the top of the circuit) to the critical sink (towards the left side of the circuit) is high. The second example shows a routing (in this case, an A-Tree [14]) which provides shortest paths from the driver to the critical sinks, and also provides reduced delay.

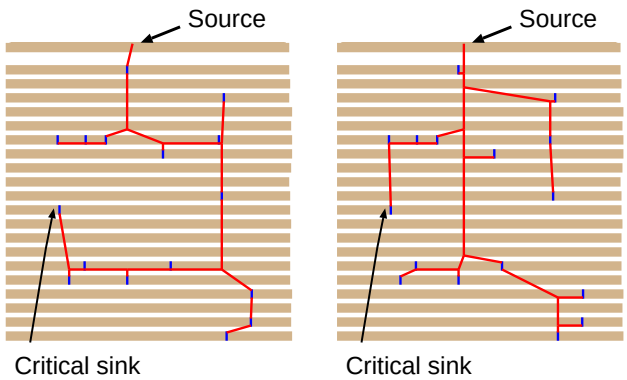


Figure 1: An area-minimizing interconnect structure for the net B3 in the MCNC example STRUCT, and a delay minimizing structure. While a minimum-length Steiner tree is appropriate for area minimization, it may result in high delay for deep submicron design.

3 The DECIMATE Global Router

The goal of a global router is to specify interconnect structures (topologies *and* wire sizes) and feedthrough locations for signal nets. For the solution to be acceptable, it must allow for low core area, and also low critical path delay.

With this in mind, we partition the global routing problem into three subproblems, and then solve these subproblems efficiently. The global router first generates interconnect structures appropriate for each net, then inserts and assigns feedthroughs to each net, and finally selects channel segments for net connections.

3.1 Interconnect Structure Generation and Selection

For submicron design, area-minimum interconnect does not necessarily result in low delay, and low delay interconnect may not be area-minimum. For high performance routing, we are faced with the problem of selecting a subset of nets to route with low-delay structures.

We define the *Interconnect Selection Problem* as follows: Each net $n_i \in N$ has a set of possible interconnect structures $I_{i,a}, I_{i,b}, ...I_{i,k}$, with costs $C_{i,a}, C_{i,b}, ..., C_{i,k}$. The source-to-sink delays for each net are computed by an appropriate delay model. The Interconnect Selection Problem is to select an interconnect structure for each net which obtains minimum circuit delay subject to a cost constraint.

For example, we might consider minimum-length spanning trees (MSTs), minimum-length Steiner trees, or any of the high-performance interconnect structures mentioned in the introduction, with or without driver or wire sizing, for each net. The cost of an interconnect structure might be measured as the total wire length, the impact of the structure on circuit area, or by some other value. While there may be an infinite number of possible interconnect structures, and evaluation of structure cost may be quite complex, we will restrict our consideration to a tractable set of candidate structures, and will use wire length, linear delay, or Elmore delay as our objective (more accurate delay models may be used as well). While the number of structures we consider for each net may be small, and the cost functions are well defined, the problem is still quite difficult.

Fact 1 *The Interconnect Selection Problem is NP-Hard.*

Proof: Consider any instance of the well known NP-Hard Knapsack problem [29], and assume that the costs and delays of different interconnect structures in an instance of an Interconnect Selection Problem can be

assigned independently. We can construct a chain of inverters, and assign each interconnection a cost and gain for an "optimized" structure that matches the weight and gain of an item from the Knapsack problem. Solution of the Interconnect Selection problem also solves the Knapsack problem. □

The problem as defined above is considerably simpler than the practical problem. We assume here that costs and delays of interconnects are independent, although this is not the case in practice. Under complex delay models, the choice of an interconnect structure may affect the delays to different pins in the same net in different ways. Thus, even if we do not constrain the overall cost, the problem of delay minimization can still be quite difficult.

In our global router, we support minimum spanning trees, low area topologies (using the algorithms of [23] and [5]), high-performance topologies (using the algorithm from [14]), and also variable-width interconnect sizing (using the optimal wire-sizing algorithms from [13, 15]). Our heuristic for the Interconnect Selection Problem initially implements minimum spanning trees for all nets (the motivation for this is shown in Sections 3.2 and 4), and then evaluates the delay of the circuit. If delay constraints are not met, nets are iteratively selected for delay improvement. We implement a simple greedy algorithm for net selection. For a net n, with driver s and critical sink t, with path length between the driver and sink in routing tree T as $d_T(s,t)$, and Manhattan distance as $d(s,t)$, the *gain* of the net is defined as $d_T(s,t) - d(s,t)$. The highest gain net (with total tree length used to break ties) is selected and then converted to a high performance topology, and may also be sized. The process continues until delay objectives are met, no gain can be found, or improvement falls below a user-determined threshold. Specific interconnect structures for nets may also be selected manually, allowing experienced designers to perform sophisticated delay analysis and optimization.

A large gain indicates that the path length between s and t can be improved significantly through topology optimization. This may result in a reduction in path interconnect resistance, and also a reduction in delay. Larger trees have a greater chance of improvement through wire sizing, so tree size is used to break ties.

In brief, our approach to interconnect generation and selection is as follows.

- Construct interconnect topologies for all nets.
- Select the highest *gain* net on a critical path.
- Apply topology optimization and wire sizing optimization to compute a high-performance interconnect structure.
- Recalculate critical paths.
- Repeat selection and optimization until improvement falls below a specified threshold, or delay bounds are met.

While this heuristic is exceedingly simple, it is shown to be effective in Section 4, particularly in circuits with large nets (where simple interconnect topologies may differ greatly from high performance topologies). For other circuits, a hill-climbing heuristic with a greater variety of interconnect structures, or manual selection of interconnect by an experienced designer, may be more appropriate.

3.2 MSTs vs. Steiner Trees

As mentioned above, we generally implement minimum spanning trees for net interconnection, even though we support a number of Steiner heuristics as part of the global router. While Steiner heuristics can obtain tree length improvements of 10% or more on random examples, we found improvements to be less than 2.1% with standard cell designs (see Table 1 in Section 4).

First, note that a large percentage of nets contain only two pins, eliminating any possibility for Steiner tree improvement. For nets with three or more pins, we find that the pin positioning shown in Figure 2A is representative of standard cell interconnect problems, and that Figure 2B (which allows tree length reduction when Steiner points are considered) is relatively uncommon.

We consider the lack of improvement for a Steiner tree heuristics on actual placements to be a side-effect of a common placement cost metric. A typical objective function is to minimize the sum of the perimeters (or half the perimiters) of the bounding boxes for all nets in a circuit. Consider the case shown in Figure 3; for a cell which contains two pins, one for net A, and one for net B, there are certain locations within the bounding box for net A that minimize cost with respect to net B. In particular, if net B has one other pin, it is likely to be in one of the four shaded regions shown (as the area of the shaded region is substantially larger than the non-shaded region). We can expect that an optimization process will attempt to move the cell towards one of the four shaded regions; if expansion of the bounding box for A is to be avoided, the desired position will be at the *corner* of the bounding box. Therefore, if bounding box perimeter is a placement objective, we can

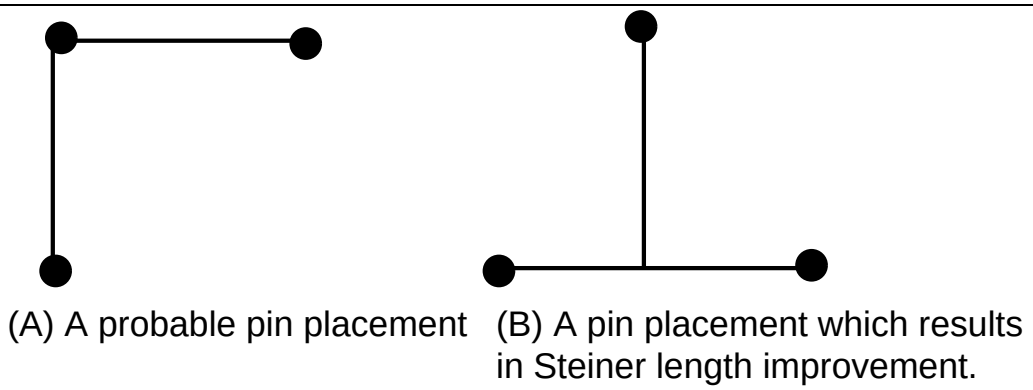


Figure 2: A likely positioning of pins for a three pin net (A), and a positioning of three points which leads to an improvement with a Steiner tree heuristic (B).

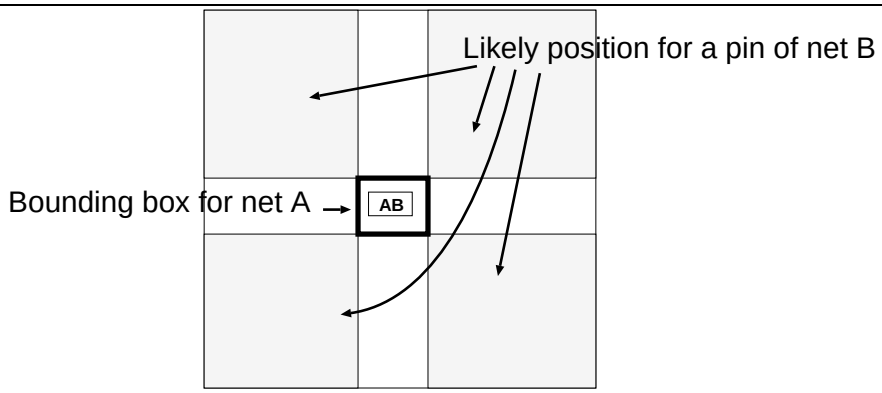


Figure 3: The bounding box for a net A, and the likely position for another pin for some net B.

expect pins to be placed towards the corners of their net bounding boxes, and tree length reductions through the insertion of Steiner points will be small.

3.3 Support for Wire Sizing Optimization

As mentioned in the introduction, we support variably sized interconnect in our global router. If wire sizing is to be performed, we utilize the optimal sizing algorithms of [15, 13] to determine sizes for the edges of nets that are selected for topology optimization. When the net is embedded for the Iterative Deletion process (described in Section 3.5), the channel segment widths are obtained from the width of the topology edges.

To accurately model the delay effects of sized interconnect wires, we model wire capacitance as the sum of area capacitance (which changes with wire width), and the fringing capacitance (which is fixed). Wire resistance is inversely related to the wire width.

The effects of wire width on circuit area are as follows. Each interconnect segment is modeled as a wire with additional spacing (to meet design rule constraints). When a wire is sized, the spacing does not increase. Therefore, a wire that is twice the minimum width takes less routing area than two parallel minimum width segments.

In our current implementation, wire width is uniform within each segment; it was shown in [11] that when the length of segments were small, solutions had near optimal performance. We can easily use the bundled refinement algorithm of [11] to vary wire width within a segment if necessary.

```
Generate feedthrough requests for each row.
Repeat
        Assign a the lowest cost request
        to a built-in feedthrough
Until no more assignments can be made
If requests are unassigned
  Insert sufficient feedthroughs
        to support all remaining requests.
  Release all feedthrough assignments
  Repeat
          Assign a the lowest cost request to a
          feedthrough
  Until no more assignments can be made
Perform greedy swap of assignments between requests.
```

Figure 4: The feedthrough assignment routine.

3.4 Feedthrough Assignment

If an edge of a net interconnect topology spans one or more standard cell rows in two-layer design, feedthroughs in those rows must be inserted or assigned to the net.

While a number of approaches to single-row feedthrough assignment have been considered, we find that a lack of a *global* perspective on the problem can seriously degrade solutions. To capture the global nature of the problem, the DECIMATE global router constructs lists of *all* feedthrough requirements and resources across *all* rows. Assignment is performed in a greedy manner, assigning a single feedthrough to an edge at each step by increasing order of cost (even though the edge may require more than one feedthrough). The assignment is improved by a greedy pairwise swapping phase.

During our research, we have explored two alternative feedthrough assignment approaches. The first, similar to that of [16], assigns feedthroughs to each interconnect edge based on an edge ordering. In any ordering, edges which receive assignments early in the process obtain high quality assignments, while edges considered later receive lower quality assignments (resulting in significant horizontal jogs). This problem persists under a variety of edge orderings. Horizontal jogs of the feedthrough assignments resulted in a substantial increase in total channel density. On average, our global feed-at-a-time approach resulted in an average 7.1% decrease in total channel density over the edge-by-edge approach.

A second approach based on a bipartite matching between a single row of feedthrough resources, and the requirements of all edges crossing the row, was also explored. Results from this approach were also inferior. The quality of results from this approach suffered from a lack of consideration of *multiple row* feedthrough requirements. In our experiments, we observed that roughly half of the edges which required feedthroughs needed *more than one*. Any single-row based assignment must be performed without fixed locations for many of the feedthroughs required; an assignment which minimizes an objective function for one row may result in a very difficult problem for another row, and global solution quality suffers. Bipartite matching is also computationally expensive and scales poorly to large examples.

While our current method is relatively simple, it produces significantly lower total channel density compared to the alternatives. Additionally, we found that the bipartite matching approach was impractical for large examples.

3.5 Iterative Deletion

Selection of high-performance interconnect structures will result in many critical nets having fixed routing topologies, with variable width routing being possible. The pins connected to these nets, and the cells to which they belong, also have fixed positions: shifting of these cells may change the net delay, requiring new interconnect structures to be determined. As high-performance interconnect is not necessarily area minimizing,

$d_{above} = abs(f_x - p_{above_x})$
$d_{below} = abs(f_x - p_{below_x})$
If $d_{above} = 0$ and d_{above} is in an adjacent
  row to r, $cost = 0$
Otherwise
If $d_{below} = 0$ and d_{below} is in an adjacent
  row to r, $cost = 0$
Otherwise
$cost = d_{above} + d_{below}$
If r is not in an adjacent row to either
  d_{above} or d_{below},
  $cost = K + cost$

Figure 5: Cost calculation for feedthroughs. A preference is given to assignments which connect directly across a channel (avoiding a horizontal jog).

we are interested in finding interconnect structures for the non-critical nets which are compatible with the fixed high-performance structures.

We approach this problem through the application of the *iterative deletion* method. This method was first introduced in [16], and is a generalization of the solution to the switchable segment problem [37, 35].

3.5.1 The Switchable Segment Problem

The switchable segment problem involves determination of the orientation of edges connecting pairs of pins in the same row (referred to as 2-pin linear nets). Generally, a pair of pins in a row can be connected with a segment in either the channel above the row, or the channel below. An example of a 3-net switchable segment problem is shown in Figure 7.

Solving the problem entails finding the orientation (either above the row or below) of each segment for each pair of pins which minimizes the total channel density. This problem was shown to be NP-hard, and a heuristic no more than 1.5 times optimal was presented in [37].

Many global routing approaches attempt to minimize the total density across all channels by switching segments up and down iteratively. In [35], experiments indicated that 38% of segments were switchable.

3.5.2 Iterative Deletion

Instead of optimizing only simple 2-pin linear nets, we attempt to find an optimal subset of edges from the "simplified net connection graph," introduced in [16].

For each net, the simplified net connection graph is constructed by creating edges between the adjacent pins of a net in each channel. This graph may contain redundant edges, which will be iteratively removed based on their relative costs. An example of such a graph is shown in Figure 8. A simple biconnectivity algorithm can be used to determine which edges are redundant (can be removed), and which are required for connectivity of the net. It was shown in [16] that the number of edges in the simplified net connection graph, and the number of edges that must be removed, are linear with the number of pins in the circuit, and that this formulation contains a minimum-density solution.

We only consider edges between adjacent pins in the same channel for the following reason. Given three pins ordered A, B, C, an edge AC can be replaced by the pair of edges AB and BC without any increase in channel density. Thus, we can be sure that any solution that would contain the AC edge is no better than the solution containing the pair, and edge AC can be left out during iterative deletion for area minimization.

The DECIMATE global router applies the iterative deletion method in the following way. First, it constructs the simplified net connection graph for each non-critical net. We use S to denote the set of all edges from all simplified net connection graphs. Edges are then removed one-by-one from S by selecting the highest cost redundant edge, and then recalculating costs for the remaining edges. Deletion continues until all redundant edges are removed. Details of the cost functions for edge selection are given in Section 3.5.3.

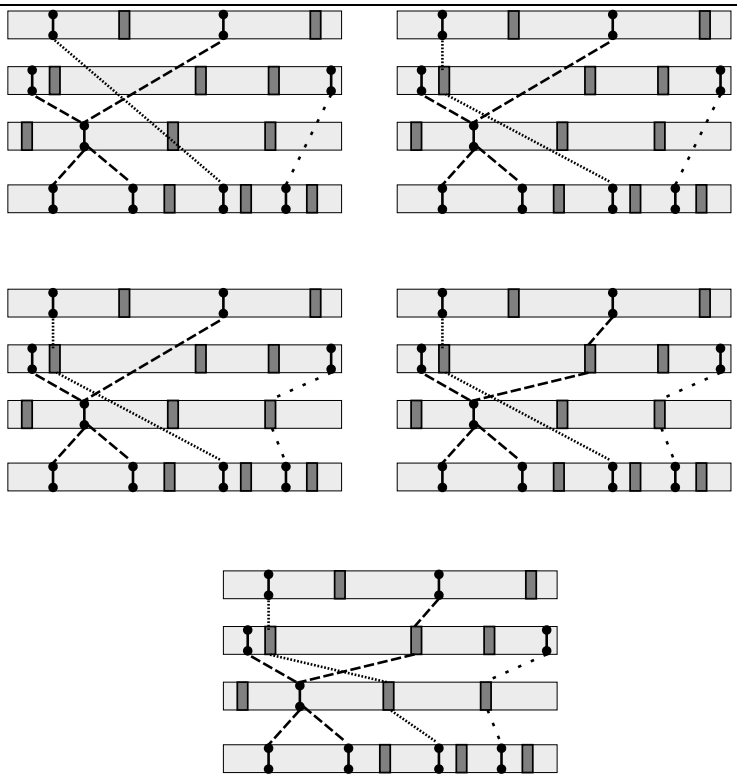


Figure 6: Assignment of feedthroughs to multiple nets. At each step, a single feedthrough is assigned to an edge, even though the edge may eventually require several feedthroughs.

There are a number of important properties of this approach. First, the number of edges to be removed is linear with the number of pins; it was shown in [16] that there are at most $1.5N$ edges in the connection graph (where N is the number of pins). To maintain connectivity, the final graph must have $O(N)$ edges, so $O(0.5N)$ edges must be deleted.

Since we start with all possible connections, the algorithm has a *global* view of the congestion distribution. This allows for effective area and congestion minimization without going through a lengthy, usually *ad hoc*, process of rip-up and re-route.

Note that the iterative deletion method is applied only to nets which are not timing critical; as the iterative deletion process may modify a the topology of a timing critical net significantly, this could prevent timing goals from being met.

A non-timing critical net may become critical after iterative deletion. If this occurs, the segment assignment of the net is replaced with a segment assignment compatible with the nets initial topology. This process continues until the delay bounds determined for the initial structures are met by the structures after iterative deletion. Therefore, we guarantee that the area optimization performed by iterative deletion will not worsen the circuit performance.

We begin with a set of edges S from all nets; this set will in general contain a number of redundant edges. During iterative deletion, edges are removed from S until all remaining edges are required for connectivity. There exists an optimal set $O \subseteq S$ which provides connectivity, and also provides minimum density.

3.5.3 Edge Selection

In [16], selection of the maximum cost edge was done by considering the length of an edge, the maximum density of the channel, and the density across the edge. Deletion was done in the most dense regions first, with edge length used to break ties.

We extend the approach through more sophisticated selection of edges for removal. To motivate our new cost function, we first state a number of observations.

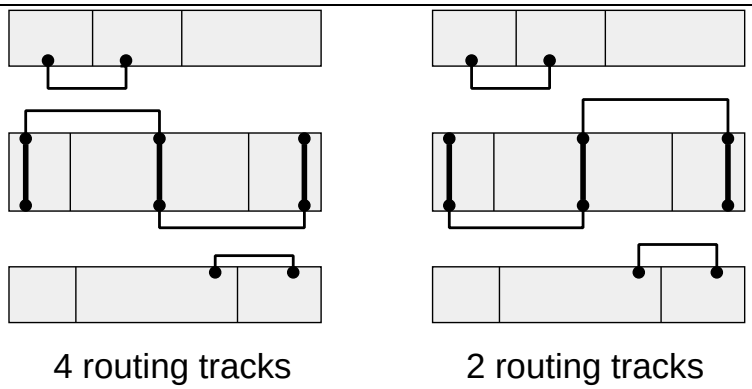


Figure 7: Each 2-pin net in the same cell row can be routed using one of the two switchable segments. The selection of switchable segments can have a major impact on the total density, and thus affect the circuit area.

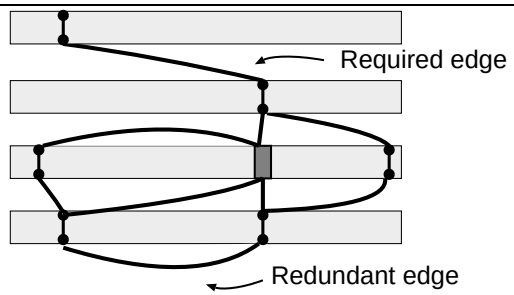


Figure 8: The simplified net connection graph contains edges between adjacent pins in a channel. The final net topology is a subset of these edges.

For a single channel, we refer to the density for all edges in S as the *current* density. The channel density for all required edges in S is the *required* density. Edges than are redundant must be part of some cycle of length N (we select the smallest cycle containing the edge to obtain N); we define the *probable width* of such an edge to be $\frac{N-1}{N}$. The *probable* density is then obtained from the density of all edges considering their probable width. The extension of each of these measures to the variable-width routing model is straightforward.

The smallest cycle lengths are determined as follows. We first mark each redundant as having an unknown cycle length. For each unmarked edge, we execute a shortest path algorithm between the endpoints of the edge, without allowing the use the edge itself; this provides length of the smallest cycle for the edge. Note that we can use this value to label *other* edges on the cycle as well, and selecting the smallest label for each edge if more than one label exists.

Our motivation for the use of probable density is as follows. For a long cycle, removal of a single edge can make all other edges on the cycle required, and thus these edges are more likely to exist in the solution obtained after iterative deletion. When the probable density of an area is high, it indicates that density reduction in this area may be difficult; we therefore have a preference for removing edges from this area first.

We use $e_{required}$, $e_{current}$, and $e_{probable}$ to denote the various density measures over an edge e. We use $c_{required}$, $c_{current}$, and $c_{probable}$ for the density measures across an entire channel c. In general, we will consider an edge e with respect to the channel c that contains it. Given these definitions, we have the following results.

Lemma 1 *Required density provides a lower bound on channel density in any connected subset of S.*

Note that during iterative deletion, required density increases monotonically, while the current density decreases monotonically.

Lemma 2 *Edges with current density less than or equal to the channel required density can be part of any optimal-density subset of S.*

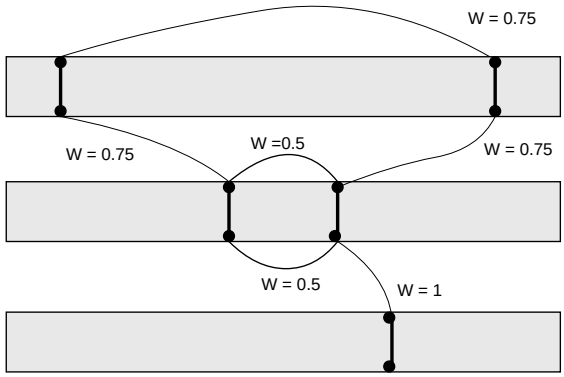
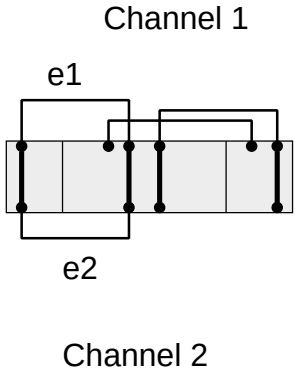


Figure 9: The *probable* width of an edge is determined by the size of the smallest cycle it belongs to.



Various density measures

	Current	Probable	Required
e1	2	1.5	1
channel 1	2	2	2
e2	1	0.5	0
channel 2	1	0.5	0

Figure 10: *current*, *required*, and *probable* density measures for edges and channels.

Proof: Assume we have the optimal solution O, which is a subset of the current solution S. If we insert all the edges as constrained above, density will not rise above the current density of S, which has the required density as a lower bound. □

Using these two lemmas, we have the following theorems, which influence our new cost function.

Theorem 1 *Given an edge set S, and an optimal (non-redundant) edge set O, where $O \subseteq S$, and the total density of O is less than that of S. There exists edge $e \in S$ in channel c such that $e_{current} = c_{current}$, $e_{current} > c_{required}$, and $e \notin O$.*

Proof: By contradiction, assume that no such edge exists. Then for each channel, the optimal solution contains channels which either did not have their density reduced ($c_{current} = c_{required}$), or all edges that passed through the most dense region of a channel are required in the optimal solution. If this is the case, the total density of O is equal to that of S, contradicting the assumption that improvement was possible. □

Theorem 2 *Given a net n, with redundant edge e, $e_{current} > c_{required}$, and no other redundant edge e' of the same net, where $e'_{current} > c'_{required}$. Then removal of e from the current solution is compatible with some optimal subset O.*

Proof: From Lemma 2, we know that even if all remaining edges in n become required, they cannot increase the density of any channel beyond its current required value; thus, they cannot affect the density of some optimal solution O. □

Construct the simplified net connection graph
Repeat
    Delete the highest cost redundant edge
    Recalculate edge costs
Until no edge can be removed

Figure 11: Iterative Deletion algorithm

Based on these results, the $weight$ of edge e is

$$\begin{cases} length(e) & \text{if } e_{current} \leq c_{required} \\ 3K + length(e) \times e_{probable} & \text{if } e_{current} = c_{current} \\ 2K + length(e) \times e_{probable} & \text{otherwise} \end{cases}$$

Edge cost (with e_{max} as the highest weighted edge other than e in the same cycle) is defined as $cost(e) = 2 \times weight(e) - weight(e_{max})$. We consider the difference between a pair of edges in the same cycle for the following reason: when one edge is removed, the other may become required for net connectivity. Thus, in addition to receiving the benefit of removing the high cost edge, we may also incur a penalty in the other edge becoming required. In our current implementation, we consider the two highest cost edges anywhere in a net, as consideration of specific cycles is computationally expensive.

The large constant K, and the ratio between the weights, is used to ensure removal of edges from the most dense regions first (using Theorem 1), and to select edges from nets where the remaining edges cannot increase the required density (using Theorem 2).

3.6 Summary of the DECIMATE Global Router

In summary, our global router performs the following steps.

- Interconnect structure selection and optimization for *global* delay objectives.
- Feedthrough assignment based on consideration of resources and requirements across all rows and nets.
- Iterative deletion to minimize channel density and congestion across the circuit.
- Restoration of net topologies for nets where the structure of a non-critical net has been modified during iterative deletion, and prevents delay objectives from being met.

4 Experimental Results

Results of our global router were compared with those of the well known TimberWolf place and route package. This package was previously under development as a university research project [35, 38], and is now supported as a commercial product by TimberWolf Systems. We routed a number of MCNC benchmark circuits [25], with placements for these benchmarks being produced by TimberWolfSC. Note that the placements used by the DECIMATE router were obtained after the TimberWolf global router finished: thus, cell mirroring and swapping performed by the TimberWolf global router was included in the placement used by the DECIMATE router. However, all feedthroughs inserted by the TimberWolf global router were removed, and all built-in feedthrough assignments were ignored.

To motivate our use of minimum spanning trees for interconnect topologies, we first present Table 1, showing the percentage improvement of an algorithm based on the ERT Steiner tree heuristic of [5] over minimum spanning tree length. We consider the existence of *equivalent pins*, and take advantage of them to reduce tree length. On random examples, we found that the ERT heuristic produced tree lengths approaching that of the 1-Steiner heuristic[23]; the 1-Steiner heuristic obtains improvements of roughly 11% on large problems. Given pin locations from an actual placement, however, improvements were significantly lower. In this table, we consider wirelength of nets with more than two pins; total wirelength improvements are thus even lower.

Benchmark	Improvement
fract	1.4%
struct	0.7%
primary 1	0.2%
primary 2	1.9%
biomed	1.4%
industry 1	2.1%
industry 2	2.0%
industry 3	1.3%
avqsmall	0.8%
avqlarge	1.1%

Table 1: Percentage reduction in total tree length of the Steiner heuristic of [6] over minimum spanning tree length for a variety of industry benchmarks.

Benchmark	TW	ID[17]	New ID
fract	39	40	39
struct	162	185	164
primary 1	166	184	167
primary 2	377	412	364
biomed	728	729	668
industry 1	449	472	447
industry 2	1006	1107	1057
industry 3	1380	1482	1422
avqsmall	1012	1061	973
avqlarge	1009	1076	993

Table 2: Comparison of cost functions for Iterative Deletion. We show total channel density for an initial TimberWolf placement routing (TW), iterative deletion using the placement and feedthrough assignment of TimberWolf and a simple cost function (ID), and iterative deletion with the improved cost function (New ID).

Not surprisingly, the slight difference in total tree length results in a slight difference in circuit area. In global routing experiments where we compare minimum spanning tree routing topologies to topologies determined by the ERT heuristic of [5], circuit area results were identical. When we apply the 1-Steiner heuristic of [23], circuit area results are slightly worse, as this heuristic does not take advantage of equivalent pins to reduce tree lengths or minimize feedthroughs.

To demonstrate the effectiveness of the new iterative deletion cost function, we compare density results using our new cost function, iterative deletion with the cost function of [17], and the result of the TimberWolf global router in Table 2. Using feedthrough assignments produced by the TimberWolf global router, the new cost function for iterative deletion resulted in density reductions ranging from 2.5% to 11.6%. Note that while the iterative deletion process does not employ any form of rip-up and re-route, the density results obtained by the new cost function were only 1.1% higher on average than the TimberWolf result.

In some instances, we may wish to have interconnect for an entire net be larger than the minimum width. As an example, consider a high frequency clock net: if this net has a large driver, a high current can result, causing metal migration problems if the interconnect is undersized. To address this problem, a clock net might have it's interconnect sized to some factor of the minimum width. In Table 3, we show the track density results of adjusting the width of clock net interconnections to twice the normal width for a TimberWolf global routing, and the density when iterative deletion is applied to the same pin positions, feedthrough assignment, and width constraint.

To examine the impact of topology optimization on global routing, we next compare maximum row length, total channel density, and maximum path length performance of the DECIMATE global router, with and without topology optimization, to that of the TimberWolf global router in Table 4. The INDUSTRY benchmarks

Benchmark	TimberWolf	Iterative Deletion
biomed	810	732
avqsmall	1207	1129
avqlarge	1307	1124

Table 3: Comparison of track density when clock nets are resized to twice the minimum width. When iterative deletion is applied to the new constraints, significantly lower track density is obtained.

		TimberWolf		DECIMATE			DECIMATE + optimization		
Benchmark	Rows	L	D	L	D	PL	L	D	PL
fract	6	1368	39	1374	41	1	1374	42	1
struct	21	4874	162	4894	164	0.87	4894	166	0.87
primary 1	17	4760	166	4760	174	1.21	4760	186	0.97
primary 2	22	10400	377	10400	386	0.96	10400	388	0.95
biomed	44	10548	728	10548	536	0.58	10548	535	0.48
industry 1	17	5148	449	5214	433	–	–	–	–
industry 2	69	14536	1006	14536	1062	–	–	–	–
industry 3	52	27152	1380	27152	1385	–	–	–	–
avqsmall	81	10272	1012	10272	981	0.83	10304	994	0.67
avqlarge	83	10688	1009	10688	1076	1.04	10608	1084	0.68
v1i3	23	3400	238	3370	202	–	–	–	–
v1i69	23	3448	234	3392	200	–	–	–	–

Table 4: Row length (L), total channel density (D), and path length (PL) comparisons with TimberWolf, with and without topology optimization. Path lengths are scaled so that the TimberWolf result is unity.

do not contain signal direction information, so no path length results are reported for these benchmarks. The benchmarks BIOMED, AVQSMALL, and AVQLARGE provided significant improvement in maximum path lengths, with topology optimization providing 52%, 32%, and 31% reductions over the TimberWolf routing result. For deep submicron design, interconnect path lengths can impact signal delay significantly. The two designs v1i3 and v1i69 were developed at UCLA as part of a low power wireless modem; the remaining designs are from the MCNC suite of standard cell benchmarks.

Elmore delay performance optimization analysis of the global routing results is approached as follows. First, as interconnect optimization is most appropriate for deep submicron design, we apply 0.5μ CMOS IC process parameters of [14], with driver resistance of 2000Ω, and scale the designs accordingly. We assume all interconnect uses the first metal layer, and do not model vias. We assume a cell delay of zero, as this information is not provided with benchmarks for deep submicron design rules. We note again that we consider the *static* critical path, and only calculate the delay of the interconnect.

We performed Elmore delay measurements on the benchmark BIOMED. We show the row width, channel density, maximum path length, and the maximum Elmore delay path (considering only the interconnect) for the TimberWolf routing, a DECIMATE routing without interconnect optimization, a DECIMATE routing with topology optimization, and a DECIMATE routing with both topology optimization and wire sizing in Table 5. We obtain substantial reductions in path lengths and delay, with the bulk of the delay in the TimberWolf solution occurring in two large nets which have circuitous paths. Due to the small size of the circuit and nets, the optimal wiresizing resulted in minimum width interconnections for many segments, and only limited Elmore delay benefit. Path length differences between the two optimized solutions are a result of different segment assignments (due to the wire sizing) resulting in different path constraints during net restoration. Wire sizing provided limited benefit; for current fabrication technologies, wire sizing may be most appropriate for large nets which can have large drivers, such as clock nets.

In our experiments, we found that interconnect optimization did not necessarily result in a major penalty on circuit area. Relatively few nets required high performance topologies, with the total length of optimized

	L	D	PL	Int. ED
TimberWolf	10548	728	81932	2.925ns
DECIMATE	10548	536	47230	2.045ns
DECIMATE + topology	10548	535	39422	1.907ns
DECIMATE + topology + wire sizing	10548	543	38566	1.899ns

Table 5: Comparison of Elmore delay results on the BIOMED benchmark; we assume 0.5μ CMOS IC parameters. We show maximum row length (L), total channel density (D), maximum path length (PL), and Elmore delay of the interconnect (Int. ED).

Benchmark	TimberWolfGR	DECIMATE
fract	1	0
struct	17	5
primary 1	8	2
primary 2	41	39
biomed	(265) 441	131
industry 1	(106) 110	20
industry 2	424	461
industry 3	428	907
avqsmall	(1402) 1572	1045
avqlarge	(1566) 1520	1266

Table 6: Run times (in seconds) for TimberWolfGR and DECIMATE (without interconnect optimization) on a number of MCNC benchmarks.

interconnect topologies being quite close to that of the minimum-Steiner tree alternative.

Run times for the TimberWolf and DECIMATE global routers are shown in Table 6. Reported results are in terms of CPU seconds on a Sun Ultra-2. For the TimberWolf results, some benchmarks received two passes of global routing: the TimberWolf package completed placement, completed global routing, and then performed another entire iteration of place and route. In these cases, the run time in parentheses is the time for the first routing, while the second value is for the second routing.

Run times for the DECIMATE global router were lower than those of TimberWolfGR for most benchmarks. In our current implementation, the feedthrough routines are not highly optimized, and contribute significantly to the run time of the larger examples.

An earlier version of the global router was organized to allow easy evaluation of a number of iterative deletion cost functions. Optimization of the router to support only the cost function detailed here resulted in a factor of 12 improvement in run times. We expect optimization of the feedthrough assignment code to also provide run time benefits, and anticipate an additional factor of 2 to 4 reduction in run times.

When interconnect optimization is performed, we observe a substantial increase in run times. Contributing to this increase are the repeated calculations of signal delays for each net (using an extremely general an unoptimized set of routines), the construction of high-performance interconnect structures for critical nets, and the calculation of wiresizing solutions for critical nets. Our present implementation utilizes a network interface to obtain interconnect structures and wiresizing solutions; we plan to extend this interface to support parallel construction of interconnect structures, and possibly also parallel interconnect delay calculation.

5 Conclusions and Future Work

In this paper, we have presented a timing driven global router for standard cell design. It produces area results comparable to that of a current commercial global router, obtains reduced path lengths, lower delay, and does

so with relatively low run times and memory consumption.

We have presented a new problem for performance driven interconnect selection, shown that it is NP-Hard, and have provided a simple heuristic algorithm. We have also introduced the concepts of required and probable density, and have presented a pair of theorems which lead to an improved cost function for the iterative deletion method.

More complex circuit structures may require a more sophisticated approach to interconnect selection; we are currently investigating a variety of methods and delay models. Hill-climbing approaches are an obvious next step, as well as the application of slack-based approaches [32, 19].

The following extensions are also under way.

- Direct support for multi-layer standard cell global routing, with explicit consideration of channel capacity available with each routing layer, and use of the over-the-cell routing area to replace feedthrough cells.
- Extension to a multi-layer general cell routing model.
- Support for further interconnect optimization through buffer insertion and driver sizing[33, 28].
- Support for additional optimized interconnect topologies such as those described in [24, 12].
- Support for more accurate deslay modeling, and possibly an interface to more accurate critical path analysis tools.

We are also working with an industry research group to obtain benchmarks representative of current high-performance IC designs.

References

[1] C. J. Alpert, T. C. Hu, J. H. Huang, and A. B. Kahng, "A Direct Combination of the Prim and Dijkstra Constructions for Improved Performance-Driven Routing," *Proc. ISCAS*, pp. 1869-1872, 1993.

[2] K. Aoshima and E. S. Kuh, "Multi-Channel Optimization in Gate-Array LSI Layout," *Proc. ISCAS*, 1983, pp. 1005-1008.

[3] H. B. Bakoglu, *Circuits, Interconnections, and Packaging for VLSI*, Addison-Wesley, 1990.

[4] K. D. Boese, A. B. Kahng, B. A. McCoy, and G. Robins, "Near-Optimal Critical Sink Routing Tree Constructions," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 14(12), pp. 1417–1436, Dec. 1995.

[5] M. Borah, R. M. Owens, and M. J. Irwin, "An Edge-Based Heuristic for Steiner Routing," *IEEE Trans. CAD*, no. 13, 1994, pp. 1563-1568.

[6] R. J. Brouwer and P. Banerjee, "PHIGURE: A Parallel Heirarchical Global Router," *Proc. 27th DAC*, 1990, pp. 650-653.

[7] R. C. Carden IV and C.-K. Cheng, "A Global Router Using an Efficient Approximate Multicommodity Multiterminal Flow Algorithm," *Proc. 28th DAC*, 1991, pp. 316-321.

[8] C. P. Chen, Y. P. Chen, and D. F. Wong, "Optimal Wire-Sizing Formula Under the Elmore Delay Model," *Proc. ACM/IEEE Design Automation Conf.*, 1996, pp. 487–490.

[9] J. P. Cohoon and L. J. Randall, "Critical Net Routing," *Proc. IEEE Int'l. Conf. on Computer Design*, pp. 174-177, 1991.

[10] J. Cong, A. B. Kahng, G. Robins, and M. Sarrafzadeh, "Provably Good Performance-Driven Global Routing," *IEEE Trans. Computer Aided Design*, Vol. 11, No. 6, pp. 739-752, June 1992.

[11] J. Cong and L. He, "Optimal Wiresizing for Interconnects with Multiple Sources," *ACM Trans. on Design Automation of Electronic Systems*, 1(4), pp. 478–511, Oct. 1996.

[12] J. Cong, L. He, C.-K. Koh, and P. H. Madden, "Interconnect Optimization for High Performance VLSI Design," *Integration*, j. 21, 1996, pp. 1-94.

[13] J. Cong and K.-S. Leung, "Optimal Wiresizing Under the Distributed Elmore Delay Model," *IEEE Trans. on Computer-Aided Design*, 14(3), March 1995, pp. 321-336.

[14] J. Cong, K.-S. Leung, and D. Zhou, "Performance-Driven Interconnect Design Based on Distributed RC Delay Model," *Proc. 30th ACM/IEEE DAC*, pp. 606-611, 1993.

[15] J. Cong and C.-K. Koh, "Simultaneous Driver and Wire Sizing for Performance and Power Optimization," *IEEE Trans. on Very Large Scale Integration (VLSI) Systems*, 2(4), December 1994, pp. 408-423.

[16] J. Cong and B. Preas, "A New Algorithm for Standard Cell Global Routing," *Proc. of Int. Conf. on Computer-Aided Design*, 1988, pp. 176-179.

[17] J. Cong and B. Preas, "A New Algorithm for Standard Cell Global Routing," *Integration*, j. 14, 1992, pp. 49-65.

[18] W.-M. Dai and E. S. Kuh, "Simultaneous Floor Planning and Global Routing for Hierarchical Building-Block Layout," *IEEE Transactions on Computer-Aided Design*, vol. CAD-6, no. 5, 1987, pp. 828-837.

[19] J. Frankle, "Iterative and Adaptive Slack Allocation for Performance-driven Layout and FPGA Routing," *Proc. ACM/IEEE DAC*, pp. 536-542, 1992.

[20] I. Harada and H. Kitazawa, "A Global Router Optimizing Timing and Area for High-Speed Bipolar LSI's," *Proc. IEEE DAC*, pp. 177-181, 1994.

[21] X. Hong, T. Xue, E. S. Kuh, C. K. Cheng, and J. Huang, "Performance-Driven Steiner Tree Algorithms for Global Routing," *Proc. ACM/IEEE DAC*, pp. 177-181, 1993.

[22] J. Huang, X.-L. Hong, C.-K. Cheng and E. S. Kuh, "An Efficient Timing-Driven Global Routing Algorithm," *Proceedings of the 30th Design Automation Conference*, 1993, pp. 596-600.

[23] A. B. Kahng and G. Robins, "A New Class of Iterative Steiner Tree Heuristics with Good Performance," *IEEE Transactions on CAD* 11(7), July 1992, pp. 893-902.

[24] A. B. Kahng and G. Robins, *On Optimal Interconnections for VLSI*, Kluwer Academic Publishers, 1994.

[25] K. Koźmiński, "Benchmarks for Layout Synthesis – Evolution and Current Status," *Proc. Of 28th Design Automation Conference*, 1991, pp. 265-270.

[26] K.-W. Lee and C. Sechen, "A New Global Router for Row-Based Layout," *Proc. of ICCAD*, 1988, pp. 180-183.

[27] J. Lillis, C. K. Cheng and T. T. Y. Lin, "Optimal Wire Sizing and Buffer Insertion for Low Power and a Generalized Delay Model," *Proc. IEEE Int'l. Conf. on Computer-Aided Design*, Nov. 1995, pp. 138-143.

[28] J. Lillis, C. K. Cheng, T. T. Y. Lin, and C. Y. Ho, "New Performance Driven Routing Techniques With Explicit Area/Delay Tradeoff and Simultaneous Wire Sizing," *Proc. ACM/IEEE Design Automation Conf.*, June 1996, pp. 395-400.

[29] M. R. Garey and D. S. Johnson, "Computers and Intractability," W. H. Freeman and Co., 1979.

[30] G. Meixner and U. Lauther, "A New Global Router Based on a Flow Model and Linear Assignment," *Proc. of Int. Conf. on Computer-Aided Design*, 1990, pp. 44-47.

[31] N. Menezes, R. Baldick, and L. T. Pileggi, "A Sequential Quadratic Programming Approach to Concurrent Gate and Wire Sizing," *Proc. Int'l Conf. on Computer-Aided Design*, 1995, pp. 144-151.

[32] R. Nair, C. L. Berman, P. S. Hauge, and E. J. Yoffa, "Generation of Performance Constraints for Layout," *IEEE T. CAD*, Vol. 8, No. 8, 1989, pp. 860-874.

[33] T. Okamoto and J. Cong, "Buffered Steiner Tree Construction with Wire Sizing for Interconnect Layout Optimization," *Proc. Int'l Conf. on Computer-Aided Design*, Nov. 1996, pp. 44–49.

[34] T. Okamoto, M. Ishikawa, and T. Fujita, "A New Feed-Through Assignment Algorithm Based on a Flow Model," *Proc. ICCAD*, Nov. 1993, pp. 775–778.

[35] C. Sechen and A. Sangiovanni-Vincentelli, "TimberWolf3.2: A New Standard Cell Placement and Global Routing Package," *Proceedings of the 23rd Design Automation Conference*, 1986, pp. 432-439.

[36] E. Shragowitz and S. Keel, "A Global Router Based on a Multicommodity Flow Model," *Integration*, j. 5, 1987, pp. 3-16.

[37] K. J. Supowit, "Reducing Channel Density in Standard Cell Layout," *Proceedings of the 20th Design Automation Conference*, 1983, pp. 263-269.

[38] W. Swartz and C. Sechen, "Timing Driven Placement for Large Standard Cell Circuits," *Proc. ACM/IEEE DAC*, pp. 211-215, 1995.