Device-Level Early Floorplanning Algorithms for RF Circuits

Mehmet Aktuna, Rob A. Rutenbar, *Fellow, IEEE*, and L. Richard Carley, *Fellow, IEEE*

***Abstract*—High-frequency circuits are notoriously difficult to lay out because of the tight coupling between device-level placement and wiring. Given that successful electrical performance requires careful control of the lowest-level geometric features-wire bends, precise length, planarity, etc., we suggest a new layout strategy for these circuits: early floorplanning at the device level. This paper develops a floorplanner for radio-frequency circuits based on a genetic algorithm (GA) that supports fully simultaneous placement and routing. The GA evolves slicing-style floorplans comprising devices and planned areas for wire meanders. Each floorplan candidate is fully routed with a gridless, detailed maze-router which can dynamically resize the floorplan as necessary. Experimental results demonstrate the ability of this approach to successfully optimize for wire planarity, realize multiple constraints on net lengths or phases, and achieve reasonable area in modest CPU times.**

***Index Terms*—Algorithms, integrated circuit layout, routing.**

I. Introduction

THE GROWING market for wireless technologies has increased the need for design tools for high-frequency circuits. Most work to date in this area has focused on the difficult problems of verification and simulation for such designs, e.g., [13], [19], [20], and [32]. As the number of designs proliferates, however, other phases of the design process are becoming bottlenecks. Layout is a notorious problem for these designs because of the tight coupling between device placement and wiring, and the potentially significant impact of even small geometric perturbations on the overall performance of the circuit.

Radio-frequency (RF) circuits have unique properties which make their automated layout impossible with standard techniques developed for lower frequency analog and digital circuits. Because every geometric property of the layout of an individual wire—its length, bends, proximity to other wires or devices—may play a key role in the electrical performance of the overall circuit, most RF layouts are optimized for performance first and density second. Worse, in some cases the crossing of two wires creates an unacceptable level of signal degradation and parasitic coupling, requiring a completely planar layout for some high-performance circuits.

Given this level of electrical and geometric coupling, we suggest in this paper a new layout strategy: device-level early floorplanning. The central idea, borrowed from chip-level floorplanning, is to resolve as early as possible all problematic device/wiring interactions by correctly planning the placement and the wiring of the full circuit. The scale of these problems admits an aggressive optimization-based attack. In our approach, a genetic algorithm (GA) evolves a population of device-level candidate floorplans; the location of not only the active devices but also the necessary extra space for planned wire meanders (extra detours taken by individual wires to control total length or phase) are managed by this floorplanning process. Each candidate floorplan is evaluated by completely routing it with a fast, gridless, detailed maze router which can dynamically resize the floorplan as necessary. We extend here our earlier treatments of [1] and [2]. The idea is similar to Cohn *et al.* [8]: for maximum control over performance, we need simultaneous placement and routing so that we may evaluate subtle performance issues correctly.

Much of the related computer-aided design (CAD) work for layout here has focused on lower-speed CMOS analog designs, e.g., [7], [9], [21], [23], [24], and [30], and is not directly applicable at higher frequencies. There is some recent RF circuit synthesis, e.g., [10], which focuses on efficient representations of these circuits for use in numerical optimization. Most CAD work targeting RF circuits comprises interactive tools that aid the designer to speed manual design iterations [15], [16]. Other work in the area includes semi-automated approaches that rely on knowledge of the relative position of all cells [14], [35]. However, these template-based approaches with predefined cells strongly limit the design alternatives possible. Recently, Charbon *et al.* introduced a performance-driven router for RF circuits [5]. In their approach, sensitivity analysis is employed to compute upper bounds for critical parasitics in the circuit, which the router then attempts to respect. Nagao *et al.* [25] also recently targeted a subset of the RF layout problem, using a variant of the sequence pair method and introducing a planarity-preserving routing algorithm. None of these techniques plan simultaneously for both device placement and wiring, and none of them can target difficult constraints such as planar wiring with precise length control.

Our goal in this work is to create a basic substrate of geometric algorithms that can manage the complex geometric interactions that determine performance for an RF layout. We assume here that the critical electrical concerns can be

Manuscript received June 10, 1998; revised October 29, 1998. This work was supported by HP-EEsof and the Semiconductor Research Corporation. This paper was recommended by Associate Editor M. Wong.

M. Aktuna is with the Department of Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh, PA 15213 USA (e-mail: aktuna@ece.cmu.edu).

R. A. Rutenbar is with the Department of Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh, PA 15213 USA (e-mail: rutenbar@ece.cmu.edu).

R. Carley is with the Department of Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh, PA 15213 USA (e-mail: carley@ece.cmu.edu).

Publisher Item Identifier S 0278-0070(99)02316-7.

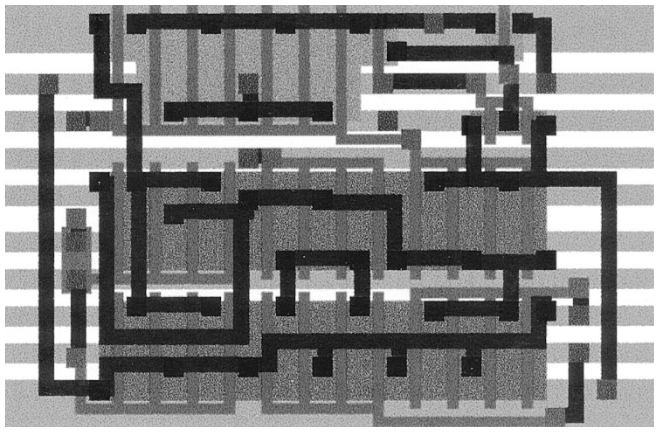
reduced—at least approximately—to a set of purely geometrical constraints that guide the device-level floorplanning task. Automatic, sensitivity-based constraint-mapping techniques have been demonstrated in [4], [6], and [24]. In practice, we expect designers to use a mix of expertise and extraction/simulation to guide this floorplanning process.

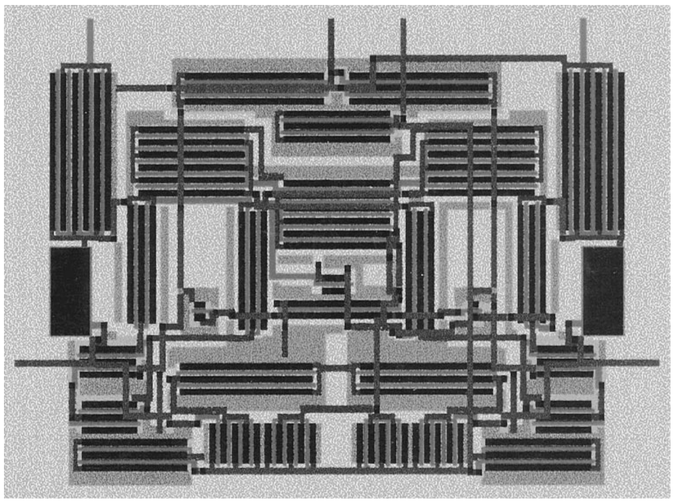
The remainder of the paper is organized as follows. Section II revisits the general floorplanning strategy outlined here, and describes more carefully our assumptions. Section III describes our device-level floorplanner. Section IV describes the device-level router used to evaluate each floorplan. Section V offers experimental results to demonstrate the merits of the approach. Section VI offers concluding remarks.

II. Assumptions and Strategy

RF cells are significantly different from either digital or lower-speed analog cells of similar size. Fig. 1 shows a simple side-by-side comparison of the three cell types. We can observe the following.

- Digital cells have mostly small devices, and for MOS technologies are often laid out in a stylized, row-dominated fashion. MOS devices are aggressively merged for performance and density, and over-the-device wiring is common. This cell, from [3] is a CMOS dynamic logic gate.
- Analog cells feature a wider typical range of device sizes and device shapes, and must manage precision issues not found in digital cells, such as required device/wiring alignment, matching, proximity, or symmetry. These cells are typically optimized first for performance, and then for density. This cell, from [26] is a CMOS operational amplifier.
- RF cells, unlike either low-speed analog or digital cells, are mostly wire dominated. There is no MOS-like device merging. There are no wires routed over devices. There are new precision issues, involving precise control of wire topology and length; indeed these layouts are often required to be planar. These cells are aggressively optimized for performance with density being more of a secondary concern. This cell, from [22] is an RF limiting amplifier. More precisely, we can enumerate the specific layout issues that make RF cell synthesis difficult.
    1) *Performance:* Every geometric property of a wire is a performance concern. Signal degradation can be caused by bends and airbridges (the three-dimensional structures used to allow wires to cross with an insulated air-space in between) on a particular net, and may impact the functionality of the overall design.
    2) *Routing:* Wiring often dominates the layout area. Wire width plus wire spacing is often quite large, causing substantial area consumption for routing. More importantly, wire detours which result from explicit length constraints on nets often take up a large fraction of the layout area. This can be seen in the manual RF layout in Fig. 2.


(a)


(b)

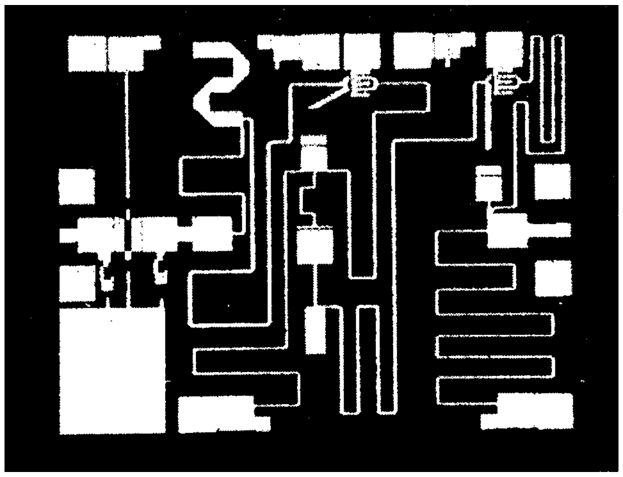

(c)

Fig. 1. Typical digital, analog, and RF cell layouts. (a) Dynamic logic gate [3]. (b) CMOS op amp [26]. (c) Layout of RF limiting amplifier [22].

3) *Optimization:* Area minimization is not the primary concern. Optimizing the planarity of the routing—necessary when crossing introduces unacceptable coupling and desirable to reduce expensive airbridges—and meeting length constraints on

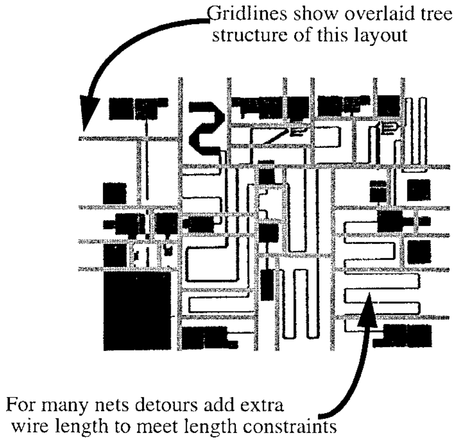

Gridlines show overlaid tree structure of this layout
For many nets detours add extra wire length to meet length constraints

Fig. 2. Layout of RF limiting amplifier [22] with slicing tree floorplan overlaid.

performance-critical nets take precedence. These wirespecific constraints directly determine the functionality of the circuit. Given these RF-specific layout issues, we propose a new approach to layout for RF circuits: early floorplanning. Before going into specific algorithmic and implementation details in the following sections, we summarize here our basic strategy and the critical engineering decisions on which it depends.

- *Target Technology:* We assume a one-layer signal routing technology with airbridges for net crossings. Even when multiple metal layers are available, despite their area cost, airbridges can be more desirable for performance-critical nets since they have better signal degradation properties compared to vias that are needed to switch layers. We assume multipoint nets are allowed.
- *Device Level Floorplanning:* We evolve floorplans that specify the locations of both active and passive individual devices of an RF circuit.
- *Representation:* We use slicing trees to represent the floorplans. This restricts the floorplans to some extent, but since slicing trees can efficiently be manipulated for optimization, we find this a good tradeoff. In fact, very complex floorplans can be realized with slicing trees. Fig. 2 shows a slicing tree overlaid on top of the manual RF layout from Fig. 1, and illustrates the practical example on the usability of slicing trees.
- *Wire Meanders as Placeable Objects:* Length constraints on wires are specified as part of the input. In a particular floorplan, wire detours or meanders may be needed to meet this exact length. In our approach, wire meanders are located in the same floorplan room as one of the devices to which the net connects. The size of the room in the floorplan is adjusted by the router to accommodate extra meandering if needed.
- *Geometric Problem Abstraction:* Rather than evaluating the exact performance characteristics of our layouts by using expensive circuit simulation or electromagnetic field analysis, we focus on optimizing the geometric properties of the layout. This is primarily due to the high computational cost of field analysis which we cannot afford for the thousands of layouts we evaluate. Circuit simulation needs the results of field analysis to determine element values and is also computationally intensive. Our strategy strives to provide the designer with fully routed layout alternatives that meet essential geometric concerns such as net length constraints and planarity. The designer, using sensitivity-based analysis tools, can then further adjust the layout to resolve subtle performance issues.
- *Stochastic Optimization of Layouts:* We evolve floorplans using a GA formulation. The GA creates new solution candidates from promising floorplans and invokes the router for their evaluation. In the following two sections, we discuss our device-level floorplanning and routing algorithms in detail.

III. Device-Level Floorplanning

In this section we describe the details of our device-level floorplanning strategy for RF cells. Despite the fact that there is no explicit placement stage after floorplanning, the floorplanner does not produce fixed device locations for the router to work on. The result of the floorplanning is a starting point for the router to work on, laying out the relative placements of devices and the wire detours that are adjacent to them. The router will further expand this "seed" placement and dynamically create a "sized" floorplan that can accommodate the optimized routing. Because of this, the device-level floorplanning strategy introduced here should be regarded as a preliminary stage before the floorplan is finalized. Due to the tight links between the floorplanner and the GA optimizer around it, genetic optimization issues are also discussed here.

A. Floorplanning by Genetic Optimization

We recast the floorplan optimization problem as a stochastic optimization using a GA. The critical components of any genetic algorithm are as follows.

- *A Representation for Individual Solutions:* In our case this is a slicing tree representation of floorplans.
- *A Population of Solutions:* We evolve a population of device-level RF cell floorplans, with typical population size of a few hundred.
- *A Selection Scheme:* We use tournament selection as described in [17] with a tournament size of two.
- *Evolution Operators:* We use a new subtree-driven crossover scheme and mutation operations adapted from simulated annealing of slicing trees.
- *An Evaluation Method:* We use our router for evaluation of the floorplans. Our specific genetic algorithm implementation uses a continuous population model that replaces a user-controllable fraction of the population in every generation. The default is replacing the 30% of the population with the worst scores. Keeping the best individuals of the population after a generation is called elitism [12]. It should be noted that the continuous population model implements elitism implicitly, since the individuals with better scores will always be preserved.

The score of the best individual in the population is tracked during the course of evolution. This is used to determine the stopping criteria, which is to stop if the score of the best answer found has not changed in a user-defined number of generations more than a tolerance percentage. The default is to stop if the best individual has not changed more than 1% in the last 100 generations.

B. Floorplan Representation

In our strategy, we represent the floorplan using canonical polish expressions of slicing trees [28], [34], and the genetic algorithm evolves the polish expressions directly. Slicing trees capture the relative placement of objects in a compact way. More importantly, the optimizer can produce a new floorplan from a given one with little computational effort, allowing efficient search of the design space. The choice of slicing trees for representation will not allow the realization of some nonslicing floorplans. With the target application of at most 50 devices—where for us a "device" is any active or passive component that must be placed and wired in an RF cell—this does not impact area significantly. We believe efficient traversal of the slicing tree design space can more than make up for this restriction.

Each of the objects in the polish expression is either a device or a device with planned space for wire meandering next to it. The floorplanner chooses an aspect ratio for every module using the Stockmeyer algorithm [34]. When there is no user defined constraint on the overall aspect ratio of the layout, the choice of aspect ratio for each module is optimal for area with respect to the given slicing tree. This optimization allows the router to start with the best packing possible for each slicing tree.

When there is a constraint on the aspect ratio for the overall layout, the floorplanner will look for aspect ratios that are within given upper and lower bounds. If there are multiple aspect ratios within the allowed range the one with minimum area is selected. If none of the aspect ratios are within the allowed range, the one closest is selected. Note that after routing, the final aspect ratio of the layout may be different than what the Stockmeyer algorithm determined after placement. After routing, a penalty is imposed if the final aspect ratio is not within specified bounds. This guides the evolution of floorplans toward solutions with more desirable aspect ratios.

C. Evolution Operators

In their floorplanner that uses simulated annealing to evolve canonical polish expressions of slicing trees, Wong *et al.* [34] used three moves to perturb the current floorplan. These were as follows.

M1) Swap two adjacent objects.

M2) Flip every cut in a chain in the polish expression, where a chain is a maximal series of operators not delimited by objects.

M3) Swap an adjacent operator operand pair.

Any new slicing floorplan of n rectangles can be reached from another with some sequence comprised solely of these three moves. Our genetic algorithm uses the same three moves as mutation operators to introduce diversity into the population. However, for efficiency we also need to mate pairs of floorplans, with the goal of propagating the best components of each. For this purpose, we introduce a new crossover operator based on subtrees. Subtrees are a good choice of building blocks for slicing trees since they encapsulate the adjacency relations among subsets of nearby devices. The crossover operator preserves the subtrees in parents as much as possible with the hope of preserving the adjacency relations that allowed the parents to have a good score. We call this strategy subtree-driven evolution. Two parents chosen for mating by the genetic algorithm then go through mutation with a fixed mutation probability.

We can describe this mating process qualitatively as follows. There are two basic cases. In the simplest, the crossover operator picks a random location in the first parent tree. If the crossover location holds a device, the child is obtained by swapping two devices in the second parent tree to enforce the same location for the crossover module in the second parent. This resembles an M1-type mutation operator, but it is influenced by the other parent.

The more interesting, second type of crossover occurs when the crossover location in the first parent, P1, contains an operator. The subtree is implanted into a suitable place in the other parent tree, P2. The primary goal while doing this is to introduce a subtree from P1 into P2 with minimal disruption. This is done in two main ways.

- The crossover algorithm rigorously searches for a subtree of comparable size in P2 for implantation, in order not to destroy a significant portion of the organization of P2.
- Further conservation is sought even within this tree of comparable size, by looking for subtrees that can be preserved.

Fig. 3 shows a simple crossover example where nodes labeled "|" mark operators whose children are horizontally adjacent, and nodes labeled "-" delimit vertically adjacent children. Note how the subtree with f-k-g-h is implanted in the final offspring, and how the a-c subtree is preserved. Fig. 4 has simple psuedocode for the operation. Together, these evolution operators can efficiently find dense, low-area slicing floorplans that can meet the geometric constraints imposed as input. But to evaluate these layouts accurately in their context as RF circuit designs, we need to route them.

IV. Detailed Routing of Floorplans

We use a novel router as the evaluation tool for evolving floorplans generated by the genetic optimizer. However, in our overall layout strategy this router has responsibilities beyond those of a traditional router. Our router completes the placement process by determining exact device locations and placing airbridges, which may occupy substantial area in RF circuits.

We use a detailed, one-wire-at-a-time area router. The router does not have separate global and detailed routing stages, instead it takes wiring details into account while selecting paths. There are two major reasons for this. The first concern

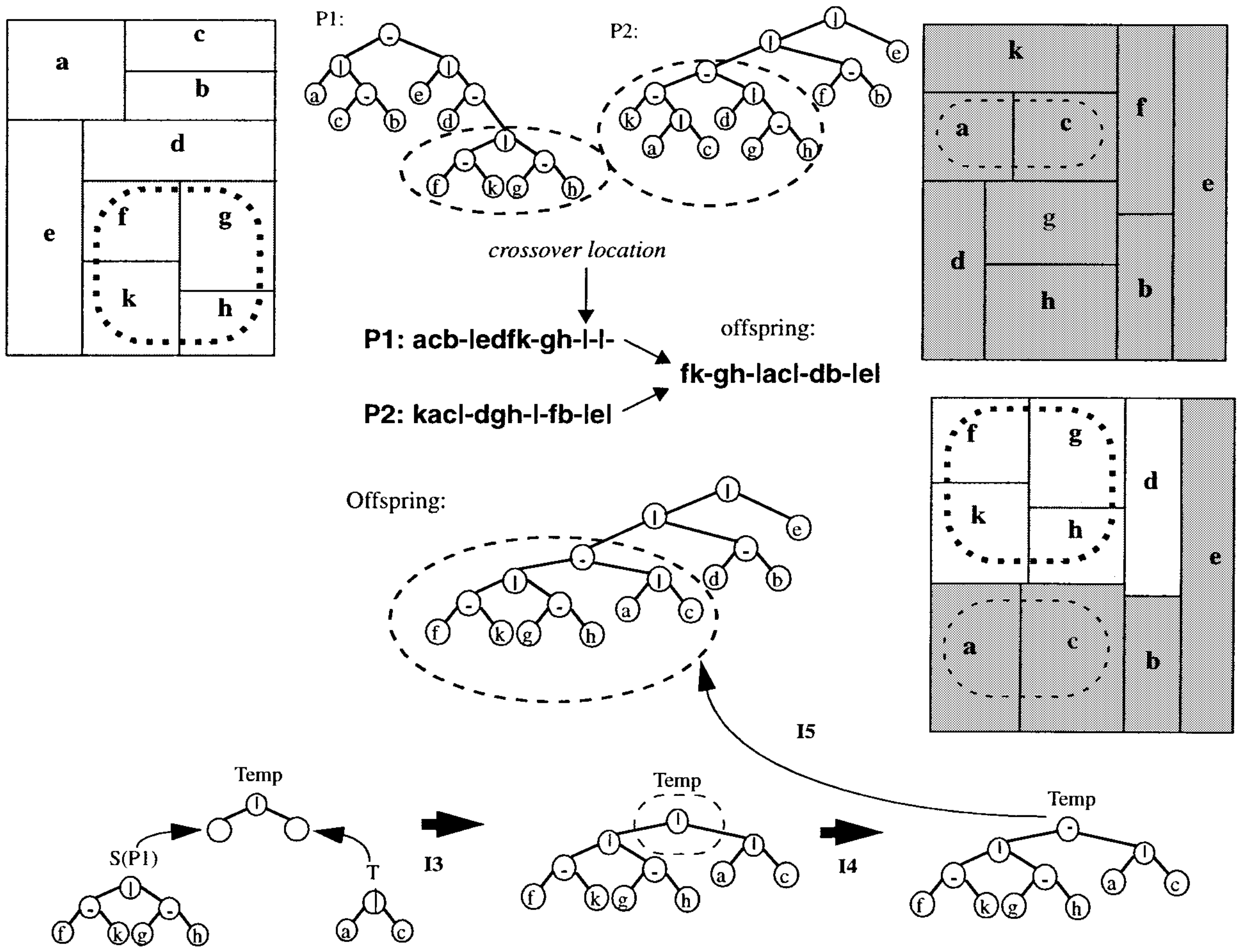
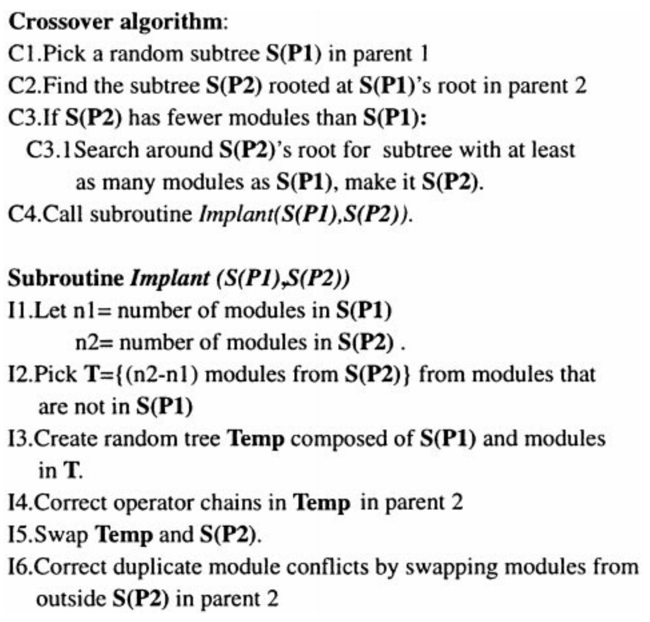


Fig. 3. Crossover example for combining two floorplans with implanting. Parent and resulting offspring floorplans appear at the top; steps of the implant process appear at the bottom.



```
Crossover algorithm:
C1.Pick a random subtree S(P1) in parent 1
C2.Find the subtree S(P2) rooted at S(P1)'s root in parent 2
C3.If S(P2) has fewer modules than S(P1):
   C3.1Search around S(P2)'s root for subtree with at least
        as many modules as S(P1), make it S(P2).
C4.Call subroutine Implant(S(P1),S(P2)).

Subroutine Implant (S(P1),S(P2))
I1.Let n1= number of modules in S(P1)
        n2= number of modules in S(P2) .
I2.Pick T={(n2-n1) modules from S(P2)} from modules that
    are not in S(P1)
I3.Create random tree Temp composed of S(P1) and modules
    in T.
I4.Correct operator chains in Temp in parent 2
I5.Swap Temp and S(P2).
I6.Correct duplicate module conflicts by swapping modules from
    outside S(P2) in parent 2
```

Fig. 4. Algorithm for crossover with subtree implantation.

is planarity. Without taking routing details into account, the number of net crossings will not be optimized. This is very important since the number of airbridges has a major impact on the quality of a global path. Maximizing planarity is important not only because it decreases area due to fewer airbridges, but also because it reduces signal degradation at airbridges. Similarly, bends require attention to detail for minimization. The second concern is the need to update channel dimensions (i.e., the wiring areas between our placed devices, where all routing occurs) as routing progresses, which can only be done with detailed routing information. In our dynamic sizing formulation, channel dimensions change dramatically as wires and airbridges are embedded, and global routing decisions have to be made taking this into account.

A. Routing Strategy

We use a graph-based router with the cost function described in pseudocode form as

Σ (length of floorplan edges in path) $\cdot$ (net width)
 $+\,\Sigma$ (length of routing in floorplan nodes for path)
 $\cdot$ (net width)
 $+\,\Sigma$ airbridge penalty for crossings in expanded path
 $+\,\Sigma$ bend penalty for bends in expanded path
 $+\,\Sigma$ (number of other nets in edge)
 $\cdot$ (length of floorplan edge) $\cdot$ (net width).

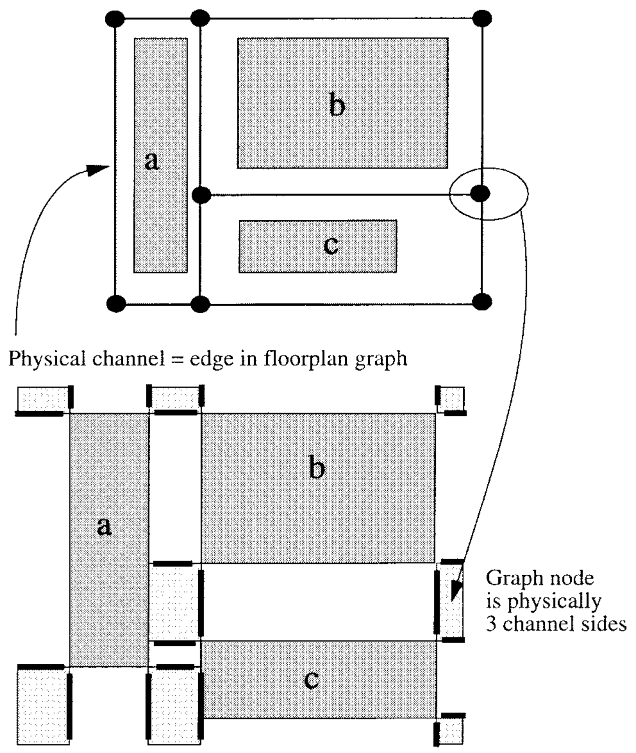


Fig. 5. Floorplan graph of simple layout.

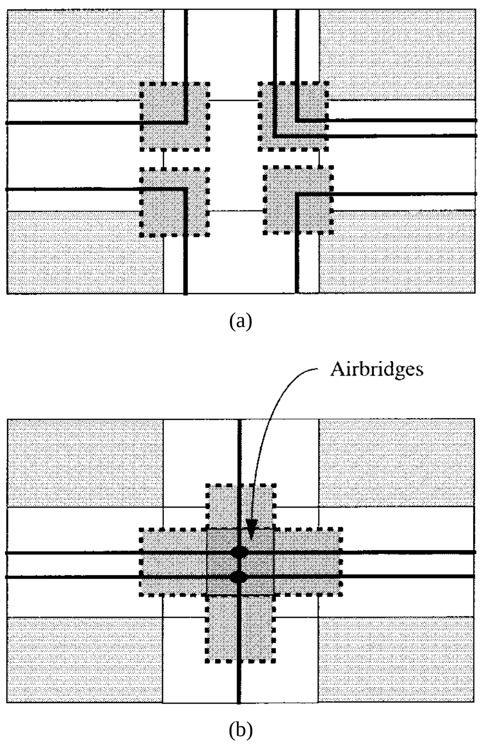


Fig. 6. (a) The four net-lists for nets turning the corners and (b) the two net-lists for nets crossing a node vertically.

This cost-based method is a practical method for capturing the effects of routing details during path selection. It also gives the user the flexibility to choose the criticality of airbridges and bends on a net-by-net basis, since the corresponding penalties are proportional to user-set coefficients. The last term in the cost function penalizes parallel runs of nets in crowded regions, proportional to the length of the parallel run. This is introduced to control congestion in channels and nodes with the aim of reducing long parallel runs of wires. This—at least qualitatively—reduces some crosstalk problems.

One key mechanism of the router is dynamic floorplan resizing. The router starts off on a floorplan with fixed-aspect-ratio devices placed with no extra routing space between them. To ensure sufficient space, the layout is dynamically enlarged while each net is embedded. By avoiding fixed, predetermined channel widths, the layout quality is substantially improved in RF circuits since the variation of channel widths is higher than for lower-frequency analog or digital circuits. Having a dynamic resizing mechanism also means that a channel is never blocked due to congestion. The dynamic resizing ability is also key to meeting length constraints since we can resize as needed to create extra wire detour space.

B. Routing Data Structures and Basic Routing Engine

The routing engine is a graph based maze router that minimizes channel congestion, number of air bridges and bends. A floorplan graph is a commonly used tool for describing the topological relation of routing regions and placed blocks [11]. Our gridless router works on an extended version of the floorplan graph. Fig. 5 shows a simple layout and its associated floorplan graph. Each edge of the graph corresponds to a routing channel with device pins on at most two opposing sides. Each node in the graph corresponds to the intersection of channels. Our maze expands nets along one edge and through one node as its atomic expansion step. In our extended floorplan graph, nodes and edges are rectangular regions and are not necessarily perfectly aligned. The router keeps track of the sizes and the alignment offsets for each edge or node.

Further extending the floorplan graph, the router relies on topological enumerations of nets in each floorplan graph edge and node. Each enumeration tracks how the net moves through the routing region associated with the edge or node: from which side it entered, its pin order on this entering side, whether it goes straight through or chooses to turn in the region, its exiting side, its pin order on this side. Given that we currently route in just one layer with a focus on maintaining planarity, this topological representation of each routing region is critical. Each node in the graph can have several of these topological entry/order/exit lists. At every graph node, nets are grouped into lists using as the unique key the entry–exit channels for the net. Therefore there is a net list for every combination of two edges incident on a node. Since a node may have two, three, or four edges incident on it, it may have up to $C(4,2) = 6$ net-lists. Two typical cases are illustrated in Fig. 6. Fig. 6(a) shows five nets changing direction as they traverse a node; the shaded boxes highlight the four topological entry/order/exit lists that track nets that change direction through this node. Similarly, in Fig. 6(b) we highlight the two other lists that track nets traversing the node vertically or horizontally. If a node has nets in both of these latter two lists, they are nonplanar in the node and will require airbridges to cross each other in the node as shown in Fig. 6(b).

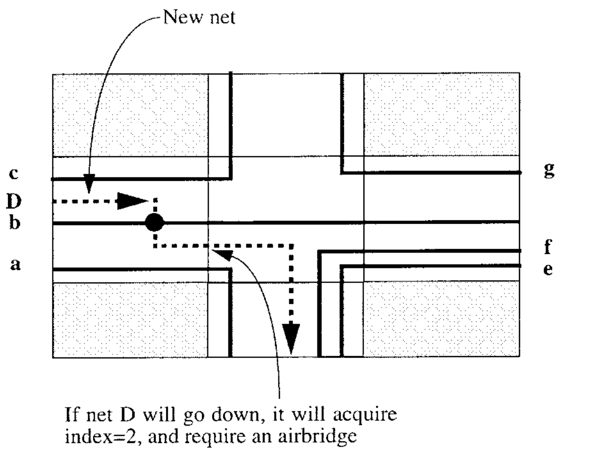

New net
c
D
b
a
g
f
e
If net D will go down, it will acquire index=2, and require an airbridge

Fig. 7. Wavefront expansion for a new net among previously routed nets.

Nets in these topological lists are ordered via their pin-order on both their entering and exiting sides. For example, counting bottom-up, net f in Fig. 7 will be the second net in the bottom-right-corner topological list for the node. Given any edge or node in the graph, we can combine the appropriate topological lists from the adjacent node/edges in the graph and create an ordered topological list of all nets on any side of any routable region in the layout. We will call this list the pin ordering for brevity. We also assign an index to each net in pin orderings. For the channel to the left of the node in Fig. 7 the bottom-up pin ordering is $(a\, b\, c)$ and the indexes for a, b, and c are one, two, and three, respectively. Pin orderings allow the router to deal straightforwardly with planarity issues in a channel. While expanding potential paths, the router considers the net as a pin to be inserted into the existing pin orders on the entering and exiting sides of the routing region being traversed. The new pin's index entering a channel is known since the order in the previous routing region has already been determined and nets do not change order in a node. Referring again to Fig. 7, the index for the new net D is three on the left side of the channel. Given the entering index, the router computes the index leaving the channel to minimize airbridges required. For net D in our example, the index will be two on the right side of the channel. Once this is determined, the index entering the next channel on the bottom side of the node is also determined to be two since the pin order $(a\, D\, f\, e)$ is known. This index depends on the direction the net will proceed in the succeeding node, after traversing this channel; this is illustrated in Fig. 8. Therefore, every channel is expanded in all possible directions in which the net can next proceed. More details of this process will be discussed in the next section on wavefront expansion.

C. Wavefront Expansion

Routing of signals is done in a single layer of metal with air bridges inserted to resolve nonplanarities. The path search algorithm is an extended form of general maze routing [29]. However, the cost of a net location on the next node-side is rather more complex than costs in general cell expansion. The cost of adding a net location to the evolving wavefront is the sum of the cost to cross a node and an edge. Both costs

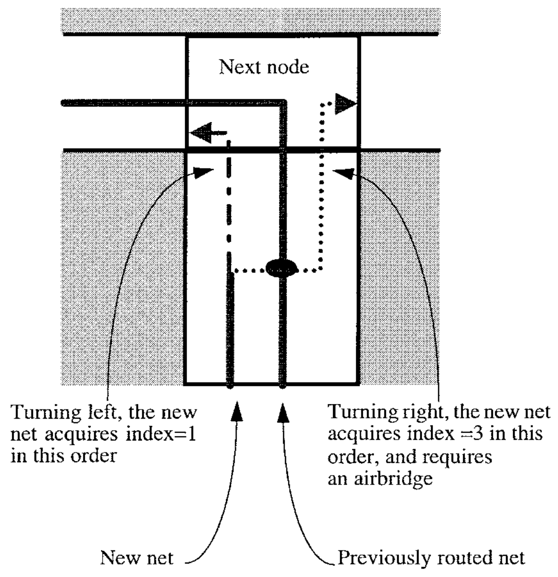

Next node
Turning left, the new net acquires index=1 in this order
Turning right, the new net acquires index =3 in this order, and requires an airbridge
New net
Previously routed net

Fig. 8. Wavefront expansion for a net in two directions.

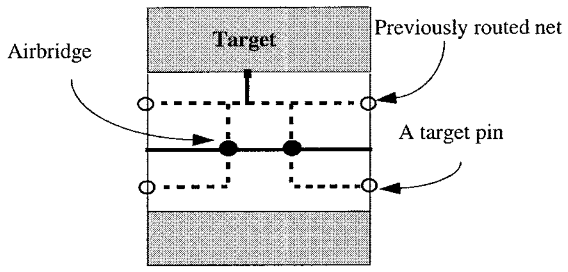


Fig. 9. Four target pins for a new net, two of which require airbridges.

include bend and airbridge costs. After a path is selected by the maze router, it is embedded into the current layout by inserting the required air bridges and resizing the channels as needed. We discuss details of the expansion process in this section.

In our formulation, device terminals generally have fixed positions as part of the circuit description. That is, for every terminal, the designer specifies a device, one of the device's four edges, and the distance from the left or bottom corner. Another option we provide the designer is specifying multiple edges without a coordinate. This is intended for devices like capacitors and inductors where the connections can be made anywhere on the perimeter of the device. For such nets the router will choose the center of the cheapest edge among the ones available.

Maze routing requires a source and a set of targets before expansion can start. Given the floorplan graph, we propagate device terminals as pins onto the escaping sides of the routing region we are expanding through. While doing this, all distinct possible pin locations have to be represented so as not to restrict the router. For example, as illustrated in Fig. 9, for the target one of the two device terminals being connected by a net is chosen. Then all possible net indexes on the two ends of the channel are initially recorded as target pins. It should be noted that the target pins have different costs to the terminal since they may have different distance or different number of airbridges required. The goal of path search is to reach one of these target pins with minimum cost. Sources are generated with a similar procedure.

The cost of expansion depends on the following:

1) the length and congestion of the channel involved;
2) the number of airbridges and bends required to cross it;
3) the relevant physical dimensions: the width of the channel, the dimensions of the node at its end, and the width of the wire being routed.

Since the number of airbridges required to cross an edge depends on the direction in which the net will proceed, each new routing region entry/exit being added to the wavefront may have a different set of airbridges required, and hence a different cost. Therefore, the router expands each of these topological path alternatives separately. To be precise, what is pushed to the wavefront by expansion during path search for net n is a record that has: 1) one side of one node in the floorplan graph and 2) the index for net n in the pin order for that side the router proposes as an entry point for net n. If different paths lead to different net indexes (i.e., pin orders) on the same side of one region for the net being routed, they will coexist on the wavefront.

The following three causes of wire bends are tracked and penalized during expansion:

1) bends caused by connecting a vertical edge to a horizontal edge;
2) bends caused by airbridges;
3) bends caused by connecting two edges that are both horizontal or both vertical, but not aligned.

Bend costs are user-specified on a per-net basis, giving the user finer control of bends on critical nets.

While expanding the wavefront, it is not sufficient to account for just the edge lengths, i.e., the simple distance of traversal across each physical channel, to compute the exact routing length of a net. Edge widths as well as detours in nodes have to be taken into account. These are accounted for by a simple algorithm that keeps track of turns at node corners during expansion.

D. Resolving Planarity During Expansion

The maze router computes the number of air bridges required to insert an expanding net into each channel using locally stored pin order information; this is a critical component of the routing cost. In our model, evolving nets may need to take nonplanar paths through channels to make the turns necessary to reach their targets. However, we do not allow nets to make nonplanar turns through floorplan graph nodes. During expansion through a channel, we use a simple heuristic to determine if a planar embedding is feasible. This is also made tractable by the fact that we restrict ourselves to a single wiring layer. The number of net crossings will also change with the order nets are routed. We do not explore different orders in this version and use the net ordering the designer provided.

If two nets are nonplanar in a channel, an air bridge has to be inserted at the point they become adjacent along the channel. The number of airbridges a new net requires to cross a channel depends on two pieces of information which are known for the expanding net. We call the following the planarity data set:

1) the entrance index of the net among previously routed nets in the edge;
2) the direction in which the net will proceed after it exits this edge.

The simple idea is that a net n will fail to find a planar embedding if its entry pin order to the channel requires it to insert between other previously embedded wires, and net n needs to make a turn that crosses some of these wires. For example, net D in Fig. 7 has to cross net b in order to make a turn toward the bottom of the node because it entered the channel above net b. Comparison of the planarity data set for the new net n and each other net in the channel suffices to determine if an airbridge is required. By doing this for all nets in the channel, a set of nets which must each be crossed, and the index in which the new net leaves the edge, are computed. This determines the index of entry into the successor routing region, along with the exact cost of airbridges in the edge just traversed. We use a simple linear scan down the spine of the channel, and compare the evolving net to each previously embedded net. This operation is linear in the number of nets embedded in the channel. The approach is similar to the pin-order net-combing technique introduced by Groeneveld in [18] for selecting an ordering for all pins at junction edges, given the global routing. However unlike their problem, we already have the ordering for previously routed nets and need to make a decision only for the new net being expanded. Once a path is decided for a new net, it will be embedded to the graph edges and nodes on the path. The topological lists in each edge or node are updated at embedding time.

E. Interaction Between the Router and the Floorplanner

It is important to note in our strategy how precise net lengths are achieved. We do not require the maze router to embed each controlled net at a precise length; rather, we rely on evolution to create floorplans which can be routed with nets of the correct length. This is less random than it might initially appear: the floorplanner plans space for meanders on individual nets, and the router then negotiates with the floorplan to ensure that the combined length of the embedded wires and the flexible wiring in the meandered spaces meets the length constraints. This is illustrated in Fig. 10. This is critical to the success of the overall approach since, on a net-by-net basis, the router is constantly resizing the floorplan. It is not possible to embed a net once, early, at a specified length and then maintain this as an invariant as subsequent routes embed.

After embedding all nets, length constraints are checked. If a net is shorter than it has to be, it is detoured in the space adjacent to its source. When that device does not have enough unused space in its floorplan room, appropriate floorplan resizers are called to create enough space in the floorplan for meandering this wire.

When the net is longer than its constraint, this shows that the current slicing tree is not suitable to meet the constraint, and we impose a penalty to the score of the slicing tree, reducing its chance of survival into the next generation of layout solutions evolved by the genetic algorithm. RF signal phase constraints

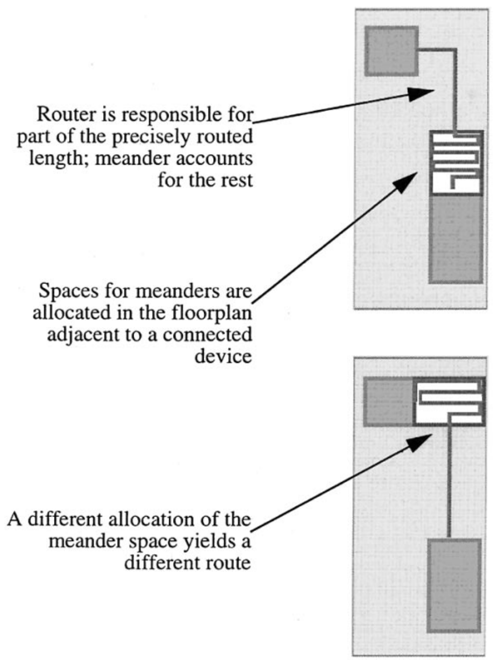

Router is responsible for part of the precisely routed length; meander accounts for the rest
Spaces for meanders are allocated in the floorplan adjacent to a connected device
A different allocation of the meander space yields a different route

Fig. 10. Interaction between floorplanner and router for precise-length control.

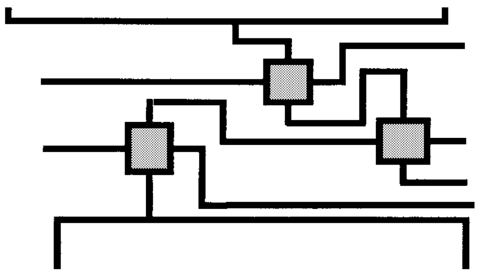


Fig. 11. A set of airbridges placed in a channel.

can be mapped to length constraints and satisfied with the same mechanism. The only difference is that a penalty is not needed since a net can always be made longer until it reaches the required phase.

The router makes fine adjustments on the floorplan by resizing it. Embedding of nets is done after the maze router finds a path. Channel heights and lengths are adjusted starting from the source while embedding the net with its air bridges. While embedding, the ordered list of airbridges in each channel is updated with the new airbridges. The airbridge list defines the placement order of airbridges as well; in our model, all airbridges will be placed along the length of the channel, as illustrated in Fig. 11. The height and length of the channel are also adjusted to accommodate the new net with resizing operations. Qualitatively, there are five major causes for the resizing operations the router invokes.

1) *Insufficient Channel Height to Place Wires and Airbridges:* The router makes sure that all wires can fit side by side for the current channel height. The required channel height is also a function of the airbridges in the channel. A net is not necessarily present in the whole channel—it may exit to a terminal. Furthermore, airbridges may have different sizes, therefore the maximum height will depend on the placement of airbridges and the current set of nets around them.
2) *Insufficient Channel Length to Place Airbridges and the Bends They Require:* While placing the airbridges the router keeps track of the space they take side by side. As illustrated in Fig. 11, this depends not only on the width of the airbridges but the width of the nets that have to bend around to get in and out of the airbridges.
3) *Improper Airbridge-Device Pin Alignment in the Channel:* The router makes sure that there is enough room in the channel for required airbridges before a net exits. For example in Fig. 11, there has to be enough space to the right of the net exiting to a terminal above to fit the two airbridges to the right of the terminal.
4) *Insufficient Wire Meandering Space Next to a Device:* When a net needs a large meander, new space may have to be created next to a device to fit the meander properly.
5) *Insufficient Space for Airbridges at a Channel Intersection (a Floorplan Graph Node):* When there are airbridges in a node as in Fig. 6, the router makes sure the node is large enough to contain them.

These resize operations are handled by tracking at all times the available space or slacks around devices and wires. Managing slacks around modules for floorplanning is common; unusual here is that we also track slacks for wire segments in each routing region. Recall that our goal is fully simultaneous placement and routing, to explore a layout space with tight coupling between device placement and device routing. Each proposed floorplan is fully routed, and each new wire can embed so as to move all other devices and previously embedded nets. We track slacks to allow fast, incremental resizing across all layout geometry as each wire embeds.

To accomplish this, we keep a detailed record of all geometry and slacks around them. For every device the slacks on its four sides are tracked. For example, in Fig. 12 device A has slacks $s1$, $s2$, $s3$, and $s4$ in its slicing floorplan room. Besides devices, channels and nodes are first-class objects whose sizes are tracked. Channels have both horizontal and vertical slacks. Channel 1 in Fig. 12 has horizontal slacks $s5$ and $s7$ and the vertical slack $s6$. Slacks originate from unused space created by slicing during the Stockmeyer detailed floorplan sizing algorithm [34], or previous resizing operations. Fig. 13(a) shows slacks above device A and to the left of device B created by the slicing tree algorithm. The first of these two slacks increases when a net is routed where the second one decreases as shown in Fig. 13(b). Fig. 13(c) shows a new slack created to the right of device B by resizing operations when another net gets routed. When a space request originates, devices, channels and nodes will try to satisfy it from their own slacks as much as possible and then propagate the remainder across the layout. Eventually all objects will have enough space. To allow the communication for this propagation, every device has pointers to nodes on its sides which in turn have pointers to all channels incident on them. The channels in turn have pointers to devices and nodes around them. For example, in Fig. 12 device A keeps track of the first and last nodes on its right, nodes 1 and 3. Since nodes and channels are stitched

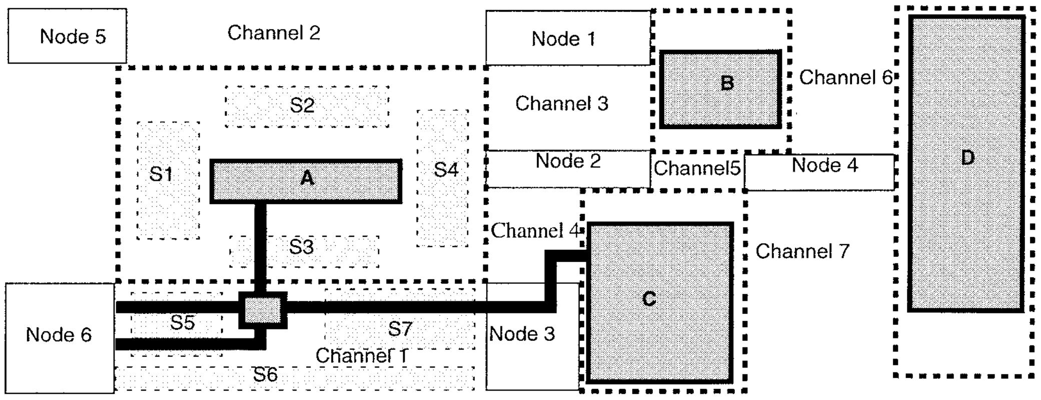


Fig. 12. Slacks around devices and channels in a floorplan.

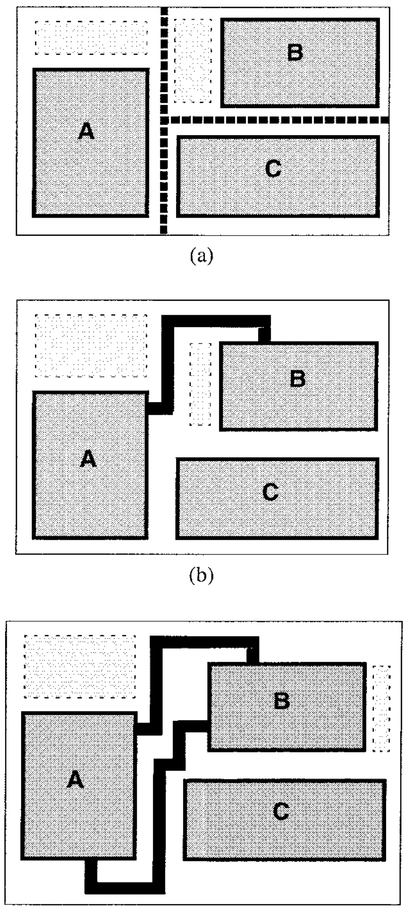


Fig. 13. Evolution of slacks in a floorplan as routing progresses.

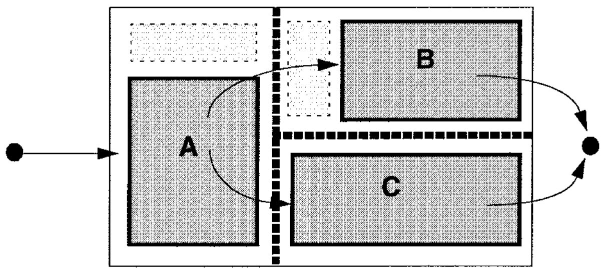


Fig. 14. Horizontal constraint graph overlaid on a floorplan with shown slicing tree.

with pointers, when device A has to move right it can probe nodes 1, 2, and 3 to move objects to its right further by passing requests to all these nodes.

Resizing of channels is done by moving the devices defining the channel. Once a vertical channel requests more width, the device to the right is passed a request for extra space, where a horizontal channel would probe the device on top of it. Resizing moves are always to the "right" or "up," in the plane of the device-level floorplan. For example, if channel 1 in Fig. 12 needs more vertical space it will pass a request to device A for more space. Device A will first try to satisfy the request from $s3$ then $s2$. If devices can not satisfy the request from their own slacks, extra space requests are passed to all channels on the opposite side through nodes on the opposite side. In this case, device A would probe nodes 1 and 5 to move channel 2 as needed. After a device moves, the required slack is added to its neighboring channels on the side from which the request originated. For example, if channel 6 makes device D move right, after the move device D adds slack to channel 7. Channel 6, being the originator for the move, does not get extra slack here.

This resizing method is equivalent to finding and maintaining critical paths on the floorplan graph and computing slacks in the noncritical paths. Fig. 14 shows the horizontal constraint graph for a simplified layout where the path that goes through devices A and C is critical, therefore the path that goes through devices A and B has a horizontal slack at B. However our approach does not need to identify explicit critical paths across the floorplan for every resizing; in practice, the incremental updates do not propagate across the whole floorplan for most resizing operations.

After devices move, they also check that the neighboring channels and nodes can still be routed properly. The new location of the device may leave insufficient room for airbridges and wires around them or may make aligning them with device terminals impossible. To fix this, extra resizing requests may be generated.

The router is also responsible for the placement of pads and other fixed objects. We allow the designer to specify how far a device should be from one or multiple edges with given tolerances. For a pad this is usually a specification to place the

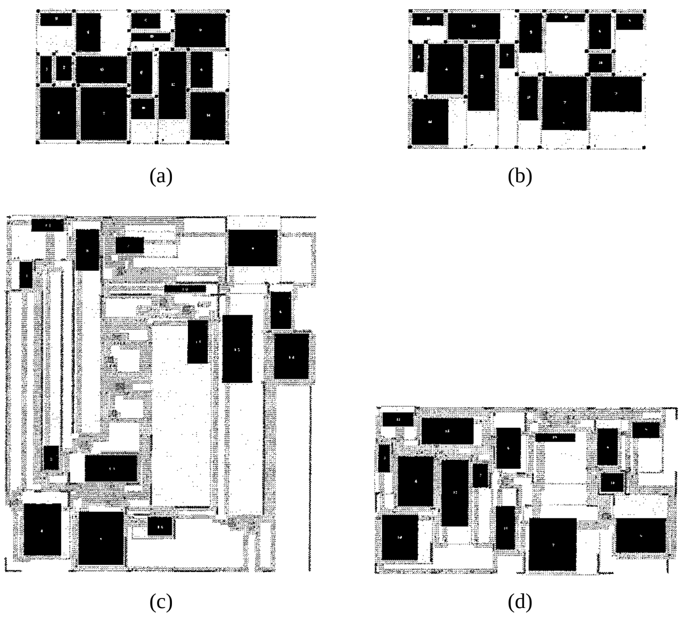


Fig. 15. Validating the need to evaluate device-level floorplans with complete, detailed routing. (a) Floorplan from optimization with sizeless wires; area: 1612. (b) Floorplan from optimization with real wires. area: 2119. (c) Routing the floorplan from (a); area: 7236; airbridges: 12. (d) Routing the floorplan from (b); area: 3311; airbridges: four.

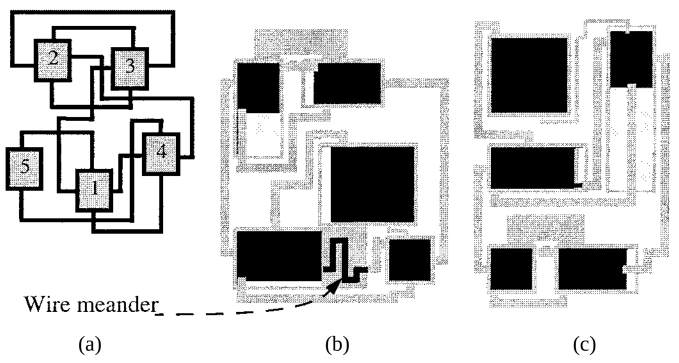


Fig. 16. Impact of bend costs on optimization.

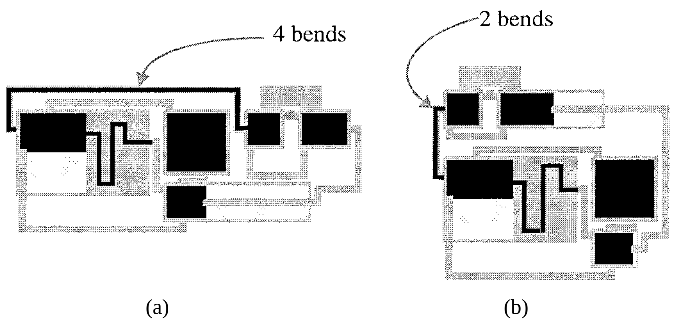


Fig. 17. Manual and automatic layouts for limiting amplifier of Fig. 2.

pad at zero distance from a specific edge of the layout with zero tolerance. We also allow a specification to place pads on any one of the four edges without the designer choosing a particular one. The router moves pads to the specified locations if there is no obstruction and imposes a cost penalty if the specification is not met. This simple approach is reasonably functional for pad support at the cost of increased runtime since we need larger populations to accommodate the pad constraints.

V. Results

The algorithms we presented in the preceding sections have been implemented in roughly 18 000 lines of C++ code. We employed a modified version of a genetic algorithm library, GAlib, available from MIT [33].

To begin, let us validate one of the core assumptions of the overall strategy: the need to have detailed routing geometry to evaluate the quality of each proposed device-level floorplan. We use a synthetic netlist with 15 devices, 20 nets, and no length constraints. The netlist is evolved twice. First, we use a "sizeless" wire routing without airbridges; the idea is that each route is of zero-width and so does not require any negotiation with the floorplan for resizing. In effect, this minimizes a simplified wire length for each net. This floorplan is optimized for idealized estimates of area and wirelength. We then evolve another floorplan with the tools full real-geometry routing capabilities.

It is possible to find very dense—indeed superior—floorplans if we ignore the details of routing. Unfortunately, when we then actually route these floorplans, the results can be dramatically inferior, as illustrated in Fig. 15. Without real wires, the floorplan at the top left offers a better packing of the devices. But when routed, it is clear this is a poor solution candidate: it has three times the number of airbridges and more than twice the area compared to the layout resulting from full optimization. This effect is especially pronounced in our RF circuit layouts because of the need to route in a single wiring layer under length constraints, and the significant area penalty (much larger than a conventional via) of each air bridge to resolve nonplanar connections. We believe that this simple result demonstrates to the need to capture fine details of the routing simultaneous to the device placement. Next, we shall highlight two specific capabilities of our floorplanning strategy: the ability to control length precisely, and the ability to optimize wire bends.

First, we show the impact of optimizing for precise wire lengths. We use the simple netlist shown in Fig. 16. The floorplan at left is optimized first without taking the length constraint into account, and then (at the right) with the length constraint. When we ignore precise net length requirements during device-level floorplan evolution, we cannot guarantee that subsequent routing can meet the constraints. In this case we can—by adding a meander as shown at the left in the figure—but at increased area. The layout at the right is simply better planned to meet the constraint, and thus saves area.

Next, we show the impact of controlling bends. We use the same synthetic netlist as from Fig. 16. Fig. 17 shows the results. The layout at left optimizes with uniform bend penalties on all nets. Increasing the bend cost of the highlighted net results in a different floorplan which allows the net to embed with two rather than with four bends, which is minimum for this design. To give a better view of the capabilities of the approach, we turn finally to a larger and more realistic layout with several interacting constraints. We compare an automatic floorplan with a manual layout we

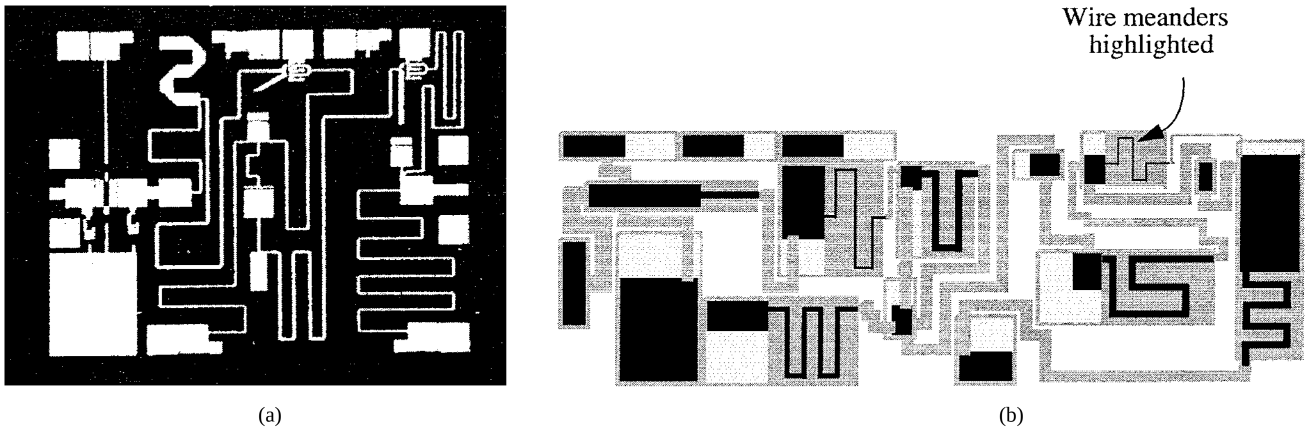


Fig. 18. Manual and automatic layouts for limiting amplifier of Fig. 2. (a) Manual layout from [22]; area: 3.8 × 3.0 mm; norm. area: 1.0; 15 devices, 15 nets, eight length constraints; zero airbridges. (b) Automatic layout; area: 6.75 × 2.33 mm; norm. area: 1.38; meets all eight length constraints; zero airbridges; runtime: 25.7 min.

TABLE I
EXECUTION TIMES TO FLOORPLAN OUR EXAMPLE CIRCUITS

Example Circuit (Figure ref.)	Number Devices /Nets	GA Population Size	Number of GA Generations	CPU Time (sec)	CPU Time per generation (sec)
Fig. 16b)	5/9	100	65	28.8	0.44
Fig. 15c) (with sizeless nets)	15/20	200	102	336	3.29
Fig. 15d)	15/20	200	172	637	3.67
Fig. 18b)	15/15	1000	286	1542[a]	5.39

a.This result was generated on a 233MHz IBM 604 CPU; others in the table used a 100MHz IBM 604 CPU.

extracted from [22]. The manual layout comprises 15 devices and 15 nets, eight of which have precise net length constraints. Since the manual layout is planar, it has no airbridges. We ran our tool on a simplified version with approximately sized rectangles for each device. We imposed the same net length constraints. As in the manual layout, we restricted three pads to the upper edge, two pads to the lower edge, the input to the left edge and the output to the right edge. Running the tool produced the layout at the right of Fig. 18. Our tool was able to evolve a planar layout. More importantly, all length constraints were met. Total runtime for this layout was roughly 26 min on an IBM 233-MHz PowerPC604 workstation.

The automatic layout is roughly 38% larger than the manual layout. This is primarily because wire meanders of the same net are spread across multiple floorplan "rooms" in the manual layout, many of which are dedicated solely to meandering. Currently, our meandering space model does not support this, resulting in inferior density. This suggests the need for a more sophisticated model of how meanders can distribute themselves across a layout. Nevertheless, this is the first time to our knowledge that any RF circuit with these sorts of tightly interacting placement and routing constraints has been automatically floorplanned.

Table I gives the runtimes for the tool with a termination criteria of 1% change tolerance at 50 generations.

In general, the runtime goes up rapidly as the number of devices and nets is increased. This is due to the larger population size required for larger problems, the larger number of generations necessary for convergence, and the longer evaluation times (routing time) for each circuit. However, the tool is capable of optimizing typical designs in 1–15 min. The second row of Table I gives the statistics for one layout optimization with sizeless wires and airbridges to show an example of the incremental cost of floorplan resizing operations. A rough analysis using the decrease in the time spent per generation on examples run with sizeless optimization shows that sizing operations take about 10%–35% of the total runtime, depending on the particular circuit.

One final issue to examine is the robustness of the basic genetic algorithm. We are solving a difficult, constrained problem using a stochastic optimization attack that is essentially an unconstrained minimization (i.e., our GA seeks to find a routed floorplan with a minimal cost score). A reasonable question is that of the robustness, by which mean both the likelihood that the GA can find a feasible solution, and that upon repeated runs of the algorithm feasible solutions dominate. As a test here, we ran 22 separate trials of the layout experiment from Fig. 18; the GA starts from a different random seed in each. Results appear in Fig. 19. The plot on the left shows the distributions of the best layout cost and mean layout cost for each of the 22 final populations at the end of genetic optimization. Despite the wide variation in the mean cost (which is influenced by the existence of some relatively poor layouts in the final populations) there is surprising uniformity of results for the best layout in each population. The scatter plot at the right then sorts these 22 best final layouts with respect to the number of

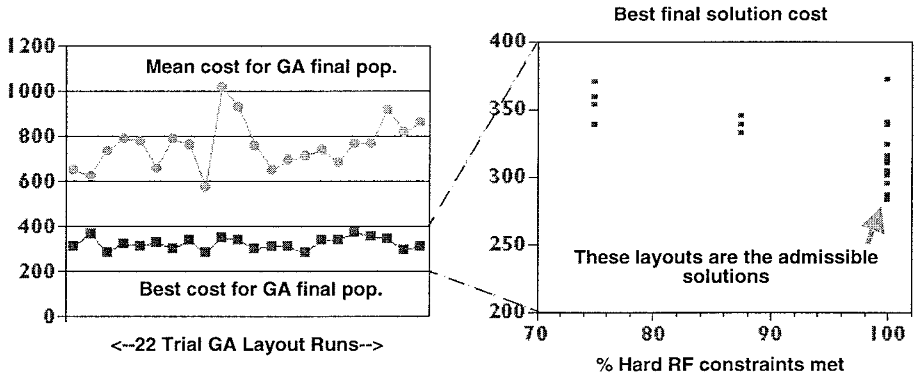


Fig. 19. Analysis of GA layout algorithm repeatability and robustness.

hard RF constraints met. Unsurprisingly, not all final solutions meet all constraints. However, roughly two-thirds of these solutions do meet all constraints. The practical conclusion here is that a few separate runs of our GA RF layout tool will suffice to find a good quality, feasible solution.

Overall, we regard this as a very satisfactory set of results for a first attempt at this difficult, tightly constrained, geometrically complex layout task.

VI. Conclusions

In this paper we suggested that the tight interaction between performance and layout for RF circuits could be addressed by device-level early floorplanning. We developed new algorithms for device-level floorplanning which integrate simultaneous detailed routing. The key idea is to use a complete—though rough—circuit layout to evaluate the low-level geometric interactions that must be carefully controlled in high-frequency designs. One of the more novel features of the approach is the integration of the placement and routing algorithms: the floorplanner plans space for large wire meanders, and the router negotiates fine-grain space for individual nets one segment at a time. This ensures that all layouts can be routed, and that both placement and wiring can be adjusted to optimize for constraints.

A preliminary implementation of these ideas works well on small designs. Our prototype can handle multiple constraints on precise net length, wire bends (and congestion; see [1]), and optimize for overall area, wirelength and—especially critical for RF cells—planarity. For these circuits, the floorplanning process is computationally reasonable.

Preliminary comparison to manual layout suggests the need for a more sophisticated model for embedding wire meanders to achieve density comparable to manual designs. The other obvious extension is to incorporate more direct evaluation of electrical interactions (e.g., local parasitics) on top of the geometric abstractions we introduced in this paper. This will allow us to take into account subtle electromagnetic interactions and make more accurate quantitative tradeoffs to optimize performance of the designed circuits. This should lead the way to a more complete layout optimization strategy for RF circuits, allowing us to move from device-level floorplans to more complete device-level layouts.

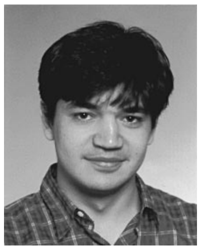
Acknowledgment

The authors are grateful to M. Mlinar of HP-EEsof for several insightful discussions of the potential role of genetic optimization in device-level RF layout.

References

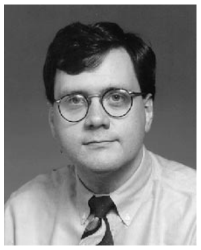
[1] M. Aktuna, “A framework for simultaneous placement and routing of radio-frequency circuits,” Masters thesis, Elect., Comput. Eng. Dept., Carnegie Mellon Univ., Pittsburgh, PA, , Dec. 1996.
[2] M. Aktuna, R. A. Rutenbar, and L. R. Carley, “Device-level early floorplanning algorithms for RF circuits,” in *Proc. ACM Int. Symp. Physical Design,* 1998.
[3] B. Basaran, “Optimal diffusion sharing in digital and analog CMOS layout,” Ph.D. thesis, Dept. of Elect., Comput. Eng., Carnegie Mellon Univ., Pittsburgh, PA, Mar. 1997.
[4] E. Charbon, E. Malavasi, and A. Sangiovanni-Vincentelli, “Generalized constraint generation for analog circuit design,” in *Proc. IEEE/ACM ICCAD,* Nov. 1993, pp. 408–414.
[5] E. Charbon, G. Holmlund, B. Donecker, and A. Sangiovanni-Vincetelli, “A performance-driven router for RF and microwave analog circuit design,” in *Proc. IEEE Custom Integrated Circuits Conf.* 1995, pp. 383–386.
[6] U. Choudhury and A. Sangiovanni-Vincentelli, “Constraint-based channel routing for analog and mixed analog/digital circuits,” *IEEE Trans. Computer-Aided Design,* vol. 12, no. 4, pp. 497–510, Apr. 1993.
[7] J. M. Cohn, D. J. Garrod, R. A. Rutenbar, and L. R. Carley, “KOAN/ANAGRAM II: New tools for device-level analog placement and routing,” *IEEE JSSC,* vol. 26, Mar. 1991.
[8] ——, “Techniques for simultaneous placement and routing of custom analog cells in KOAN/ANAGRAM II,” in *Proc. ACM/IEEE ICCAD,* Nov. 1991, pp. 394–397.
[9] ——, *Analog Device-Level Layout Automation.* Norwell, MA; Kluwer Academic, 1994.
[10] J. Crols, S. Donnay, M. Steyaert, and G. Gielen, “A high-level design and optimization tool for analog RF receiver front-ends,” presented at *ACM/IEEE ICCAD,* Nov. 1995.
[11] W. Dai and E. S. Kuh, “Simultaneous floor planning and global routing for hierarchical building-block layout,” *IEEE Trans. Computer-Aided Design,* vol. CAD-6, no. 5, Sept. 1987.
[12] K. A. De Jong, “An analysis of the behavior of a class of genetic adaptive systems,” Doctoral dissertation, Univ. Michigan, Ann Arbor, 1975.
[13] P. Feldmann, “Computation of circuit waveform envelopes using an efficient, matrix decomposed harmonic balance algorithm,” presented at *ACM/IEEE ICCAD,* Nov. 1991.
[14] *Hewlett-Packard Microwave and RF Design System Manuals,* Santa Rosa Systems Division, Santa Rosa, CA, Dec. 1992.
[15] R. H. Jansen, “LINMIC: A CAD package for the layout-oriented design of single- and multi-layer MIC’s/MMIC’s up to mm-wave frequencies,” *Microwave J.,* pp. 151–161, Feb. 1986.
[16] R. H. Jansen, R. G. Arnold, and I. G. Eddison, “A comprehensive CAD approach to design of MMIC’s up to MM-wave frequencies,” *IEEE J. Microwave Theory Tech.,* vol. 36, pp. 208–219, Feb. 1988.
[17] D. E. Goldberg, K. Deb, and B. Korb, “Do not worry, be messy,” in *Proc. 4th Int. Conf. Genetic Algorithms,* 1991, pp. 24–30.

[18] P. Groeneveld, “Wire ordering for detailed routing,” *IEEE DTC*, pp. 6–17, Dec. 1989.
[19] K. S. Kundert, J. K. White, and A. Sangiovanni-Vincentelli, *Steady-State Methods for Simulating Analog and Microwave Circuits.* Norwell, MA: Kluwer Academic, 1990.
[20] K. S. Kundert and D. Sharrit, “Simulation methods for RF circuits,” in *Proc. ACM/IEEE ICCAD*, Nov. 1997.
[21] K. Lampaert, G. Gielen, and W. M. Sansen, “A performance-driven placement tool for analog integrated circuits,” *IEEE JSSC*, vol. 30, pp. 773–780, July 1995.
[22] G. K. Lewis, I. J. Bahl, E. L. Griffin, and E. R. Schineller, “GaAs MMIC’s for digital radio-frequency memory (DRFM) subsystems,” *IEEE Trans. Microwave Theory Tech.*, vol. MTT-35, no. 12, p. 1478, Dec. 1987.
[23] E. Malavasi and A. Sangiovanni-Vincentelli, “Area routing for analog layout,” *IEEE Trans. Computer-Aided Design*, vol. 12, no. 8, pp. 1186–1197, Aug. 1993.
[24] E. Malavasi, E. Felt, E. Charbon, and A. Sangiovanni-Vincentelli, “Automation of IC layout with analog constraints,” *IEEE Trans. Computer-Aided Design*, 1996.
[25] A. Nagao, I. Shirakawa, and T. Kambe, “A layout approach to monolithic microwave IC,” presented at *ACM Int. Symp. Physical Design*, 1998.
[26] K. Nakamura and L. R. Carley, “An enhanced fully differential folded-cascode operational amplifier,” *IEEE JSSC*, vol. 27, Apr. 1992.
[27] R. H. J. M. Otten, “Automatic floor-plan design,” in *Proc. 19th ACM/IEEE Design Automation Conf.*, 1982, pp. 261–267.
[28] ——, “Efficient floorplan optimization,” in *Proc. IEEE Int. Conf. Computer Design*, 1983, pp. 499–502.
[29] F. Rubin, “The Lee path connection algorithm,” *IEEE Trans. Computer-Aided Design*, vol. 3, no. 4, pp. 308–318, Oct. 1974.
[30] R. A. Rutenbar, L. R. Carley, P. C. Maulik, E. S. Ochotta, T. Mukherjee, B. Basaran, S. Mitra, S. K. Nag, and B. R. Stanisic, “Synthesis and layout for analog and mixed signal IC’s in the ACACIA system,” in *Advances in Analog Circuit Design.* Norwell, MA: Kluwer Academic, 1996.
[31] N. Sherwani, *Algorithms for Physical Design Automation.* Norwell, MA: Kluwer Academic, 1993, pp. 215–223.
[32] R. Telichevesky, K. S. Kundert, and J. K. White, “Efficient steady-state-analysis based on matrix-free Krylov-subspace methods,” presented at *ACM/IEEE DAC*, June 1995.
[33] M. Wall, Massachusetts Inst. Technol, available: http://lancet.mit.edu/ga.
[34] D. F. Wong and C. L. Liu, “A new algorithm for floorplan design,” in *Proc. 23rd ACM/IEEE Design Automation Conf.*, 1986, pp. 101–107.
[35] J. F. Zurcher, “MICROS—A CAD/CAM program for fast realization of microstrip masks,” *IEEE Trans. Microwave Theory Tech.*, vol. MTT-S, pp. 481–484, 1985.



Mehmet Aktuna received the S.B. degree in electrical computer and systems engineering from Harvard University, Cambridge, MA, in 1994 and the M.S. degree in electrical and computer engineering from Carnegie Mellon University, Pittsburgh, PA, in 1996, where he is currently working towards the Ph.D. degree.

During the summer of 1996, he worked on microprocessor routing at Intel Corporation, Hillsboro, OR. He worked with the Physical Design Group of HP-EEsof, Santa Rosa, CA, during the summer of 1997. His current research interests include routing and placement for analog and digital circuits, and high-frequency interconnect modeling.



Rob A. Rutenbar (S’77–M’84–SM’90–F’98) received the Ph.D. degree from the University of Michigan in 1984 and subsequently joined the faculty of Carnegie Mellon University, Pittsburgh, PA.

He is currently Professor of Electrical and Computer Engineering, and (by courtesy) of Computer Science. His research interests focus on circuit and layout synthesis algorithms for mixed-signal ASIC’s, for high-speed digital systems and FPGA’s.

In 1987, Dr. Rutenbar received a Presidential Young Investigator Award from the National Science Foundation (NSF). He has been on the program committees for the IEEE International Conference on CAD, the ACM/IEEE Design Automation Conference, the European Design and Test Conference, and the ACM International Symposium on FPGA’s. From 1992–1995, he was on the Editorial Board of IEEE SPECTRUM. He was General Chair of the 1996 ICCAD and is currently on the Program Committee of the ACM International Symposium on Physical Design. He chaired the Analog Technical Advisory Board for Cadence Design Systems from 1992–1996. He was Director of Carnegie Mellon’s Center for Electronic Design Automation from 1993–1998. He is a member of the ACM.

L. Richard Carley (S’77–M’81–SM’90–F’97) received the S.B. degree in 1976, the M.S. degree in 1978, and the Ph.D. degree in 1984 from the Massachusetts Institute of Technology (MIT), Cambridge.

He has worked for MIT’s Lincoln Laboratories and has acted as a Consultant in the areas of analog and mixed analog/digital circuit design, analog circuit design automation, and signal processing for data storage. In 1984, he joined Carnegie Mellon University, and in 1992 he was promoted to the rank of Full Professor of Electrical and Computer Engineering. His research interests include the development of computer-aided design (CAD) tools to support analog circuit design, the design of high-performance analog/digital signal processing IC’s, and the design of integrated microelectromechanical systems (MEMS). Since joining Carnegie Mellon he has been granted eight patents and authored or coauthored more than 120 technical papers.

Dr. Carley is a member of program committees of the Custom IC Conference (CICC), the Magnetic Recording Conference (TMRC), and the International Symposium on Low-Power Electronics Design (ISLPED). He has served as an Associate Editor for the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS and on the Editorial Board of the *Analog Signal Processing Journal*. He received a National Science Foundation (NSF) Presidential Young Investigator Award in 1985, a Best Technical Paper Award at the 1987 Design Automation Conference, a Distinguished Paper Mention at the 1991 International Conference on Computer-Aided Design, and a Best Panel Award at the 1993 International Solid-State Circuits Conference. He was awarded the Guillemin Prize for best undergraduate thesis in the electrical engineering department at MIT.