Device-Level Early Floorplanning Algorithms for RF Circuits

Mehmet Aktuna, Rob A. Rutenbar and L. Richard Carley

Department of Electrical and Computer Engineering
Carnegie Mellon University, Pittsburgh, PA 15213
email: aktuna@ece.cmu.edu

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, proximity, planarity, etc.—we suggest a new layout strategy for these circuits: early floorplanning at the device level. This paper develops a floorplanner for RF 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.

1. Introduction

Thegrowingmarketforwirelesstechnologieshasincreasedtheneed fordesigntoolsforhigh-frequencycircuits.Mostworktodateinthis areahasfocusedonthedifficultproblemsofverificationandsimulation for such designs, *e.g.*, [Feldmann 96], [Kundert 90], [Telichevesky 95] [Kundert 97]. 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 significantimpactofevensmallgeometricperturbationsontheoverall 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. The idea is similar to [Cohn 91b]: for maximum control over performance, we need simultaneous placement and routing so that we may evaluate subtle performance issues correctly.

Much of the related CAD work for layout here has focussed on lower-speed CMOS analog designs, e.g. [Cohn 91a], [Cohn 94], [Lampaert 95], [Malavasi 93], [Malavasi 96], [Rutenbar 96] and is not directly applicable at higher frequencies. There is some recent RF circuit synthesis, *e.g.*, [Crols 95] 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 [Jansen 86, 88]. Other work in the area includes semi-automated approaches that rely on knowledge of the relative position of all cells [HP 92], [Zurcher 85]. 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 [Charbon 95]. In their approach, sensitivity analysis is employed to compute upper bounds for critical parasitics in the circuit, which the router then attempts to respect. None of these techniques plan for both device placement and wiring, and none of them can target difficult constraints such as planarity.

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 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 [Charbon 93], [Choudhury 93], [Malavasi 96]. 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. Sec. 2 revisits the general floorplanning strategy outlined here, and describes more carefully our assumptions. Sec. 3 describes our device-level floorplanner. Sec. 4 describes the device-level router used to evaluate each floorplan. Sec 5. offers experimental results to demonstrate the merits of the approach. Sec. 6 offers concluding remarks.

2. Assumptions and Strategy

The critical problems in an RF circuit layout are different from lower-speed analog or digital CMOS circuits:

- **Performance**: Every geometric property of a wire is a performance concern. Signal degradation can be caused by bends and

airbridges (the 3-D 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.

- **Area**: Wiring often dominates the layout area. Wire width plus spacing is 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 layout in Figure 1 from [Lewis 87].
- **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 performance-critical nets take precedence. These wire-specific 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 multi-point 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 trade-off. In fact, very complex floorplans can be realized with slicing trees. Figure 1 also shows a slicing tree overlaid on top of the manual layout as a practical example on the usability of slicing trees.
- **Wire meandering 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 genetic algorithm formulation. The genetic algorithm creates new solution candidates from promising floorplans and invokes the router for their evaluation.

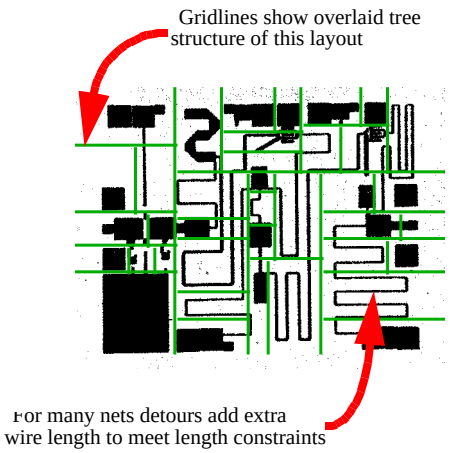


Fig. 1. Slicing overlay of RF limiting amplifier [Lewis 87]

In the following two sections, we discuss our device-level floorplanning and routing algorithms in detail.

3. 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 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 genetic algorithm optimizer around it, genetic optimization issues are also discussed here.

3.1 Floorplanning by Genetic Optimization

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

- **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 [Goldberg 91] with a tournament size of 2.
- **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* [De Jong 75]. 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.

3.2 Floorplan Representation

In our strategy, we represent the floorplan using canonical polish expressions of slicing trees [Otten 83] [Wong 86], 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 [Wong 86]. The choice of the aspect ratio for each module is optimal with respect to the given slicing tree. This optimization allows the router to start with the best packing possible for each slicing tree.

3.3 Evolution Operators

In their floorplanner that uses simulated annealing to evolve canonical polish expressions of slicing trees, [Wong 89] used three moves to perturb the current floorplan. These were:

- 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 floorplans 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.

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.

A more interesting type of crossover will occur 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.

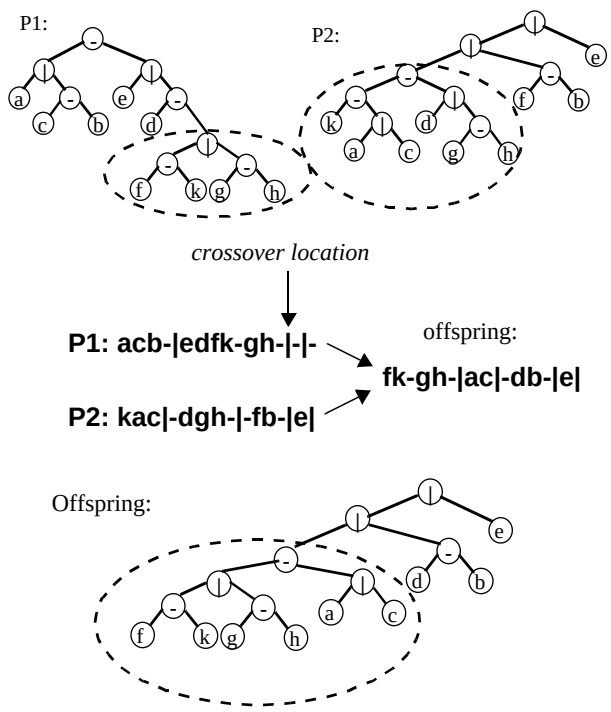


Fig. 2. Crossover example

Figure 2 shows a simple crossover example. Figure 3 has simple pseudocode 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.

4. 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. We choose to do the global and detailed routing simultaneously. There are two major reasons for this. The first concern 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 as routing progresses, which can only be done with detailed routing information. In our dynamic sizing formulation, channel dimen-

Crossover algorithm:

C1.Pick a random subtree S(P1) in parent 1 (P1)

C2.Find the subtree S(P2) rooted at S(P1)'s root in parent 2 (P2)

C3.If S(P2) is not large enough:

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),P2)*.

Subroutine *Implant (S(P1)*:,S(P2),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.Swap Temp and S(P2).

I5. Correct operator chains in Temp in P2

I6.Correct module conflicts by swapping modules from outside S(P2) in P2

Fig. 3. Algorithm for crossover with subtree implantation

sions change dramatically as wires and airbridges are embedded, and global routing decisions have to be made taking this into account.

4.1 Routing Strategy

We use a cost-based maze router with the cost function described in pseudo-code form as:

$$\Sigma \text{ (length of floorplan edges in path)} \bullet \text{(net width)}$$
$$+ \Sigma \text{ (length of routing in floorplan nodes for path)} \bullet \text{(net width)}$$
$$+ \Sigma \text{ airbridge penalty for crossings in expanded path}$$
$$+ \Sigma \text{ bend penalty for bends in expanded path}$$
$$+ \Sigma \text{ (number of other nets in edge)} \bullet \text{(length of floorplan edge)} \bullet \text{(net width)}$$

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 routing regions 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.

4.2 Basic Routing Engine

The routing engine is a modified maze router that minimizes channel congestion, number of air bridges and bends. The router is

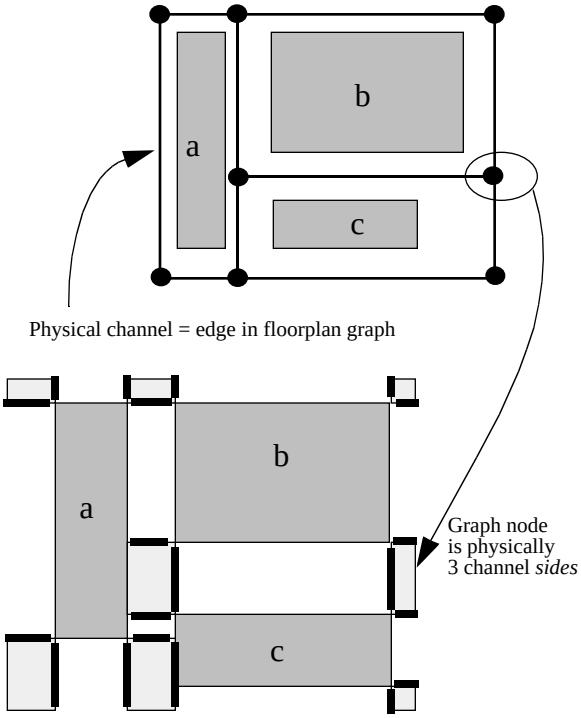


Fig. 4. Floorplan graph of simple layout

gridless and works on the floorplan graph. This substantially reduces runtime since a channel is completely traversed in only one expansion step of the maze router. The airbridges required to cross a channel depend on what direction the net will proceed after traversing that channel. Therefore, every channel is expanded in all possible directions in which the net can next proceed. Each side of the adjacent node defines one of these directions. For each of these sides, expansion produces one or more net orderings. Thus the units of expansion are net locations or orders on *sides* of a node in the floorplan graph, as illustrated in Figure 4.

Routing of signals is done in a single layer of metal with air bridges inserted to resolve non-planarities. The path search algorithm is an extended form of general maze routing [Sherwani 93]. However, the cost of a net location on the next 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 include bend and airbridge costs. Details of expansion will be introduced in the next section. 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.

4.3 Wavefront Expansion

Maze routing requires a *source* and a set of *targets* before expansion can start. One of the two device terminals being connected by a net is chosen as the target. All possible net orderings on *sides* at the ends of the channel that holds the target are initially recorded as *target pins.* The goal of path search will be to reach one of these target pins with minimum cost.

The cost of expanding a side depends on:

- The length and congestion of the channel involved.
- The number of airbridges and bends required to cross it.
- The width of the channel and the nodes at its ends.

The maze router computes the number of air bridges required to insert a net into a channel and considers this in the routing cost. This is done by keeping track of previously embedded nets in lists at every node. There is a net list for every combination of two edges incident on a node. Since a node may have 2, 3 or 4 channels incident on it, it has up to C(4,2) = 6 net lists. All nets in a given list cross the node to connect the same pair of edges. The order of nets in a given list corresponds to the physical routing order while crossing the node.

If two nets are non-planar 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 new net. We call these the *planarity data set*:

- The entrance order of the net among previously routed nets in the edge.
- The direction in which the net will proceed after it exits this edge.

The net lists at nodes are updated at embedding time. Every list entry has a net identification along with its order in the opposite side, the first element of its planarity data set. Since node net lists are formed according to the edge pairs they connect, the direction in which they proceed after a given edge is the same for members of the same list, which is the second entry of the planarity data set. Therefore, for every list entry the planarity data set is available.

Comparison of the planarity data set for the new net and a list member is enough to determine if an airbridge is required. By doing this for all nets in the channel, a *list* of nets that has to be crossed, and the *order* in which the new net leaves the edge is computed. This determines the order in the next side, along with the exact cost of airbridges in the edge just traversed.

Since the number of airbridges required to cross an edge depends on the direction in which the net will proceed, each new side being added to the wavefront may have a different set of airbridges required, and hence a different cost. This is the primary reason for expanding for each side separately.

Expansion pushes a net with its order among existing nets on a given side to the wavefront. Multiple net orders on the same side may co-exist on the wavefront if different paths lead to different net orders.

Three causes of wire bends are tracked and penalized during expansion:

- Bends caused by connecting a vertical edge to a horizontal edge.
- Bends caused by airbridges.
- 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.

4.4 Interaction Between the Router and the Floorplan

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 Figure 5. 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 the 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. Signal phase constraints 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 air bridges. While embedding, the ordered list of airbridges in each channel is updated with the new airbridges. The height and length of the channel are also adjusted to accommodate the new net with resizing operations. There are five major causes for the resizing operations the router invokes:

- Insufficient channel width to place wires and airbridges.
- Insufficient channel length to place airbridges and the bends they require.
- Improper airbridge-device pin alignment in the channel.

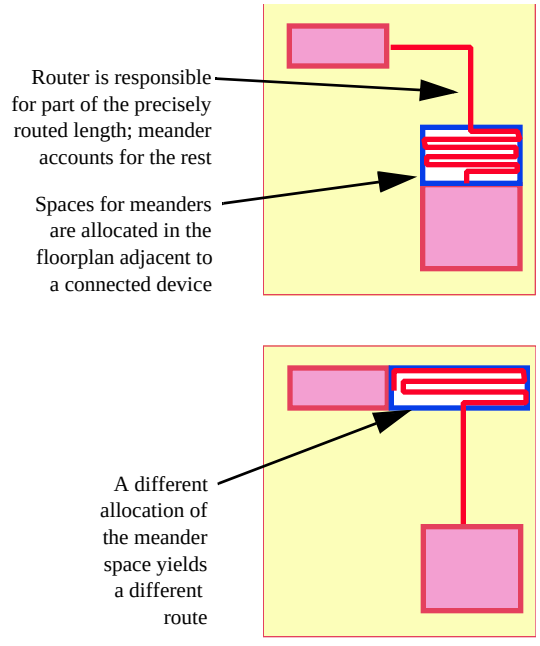
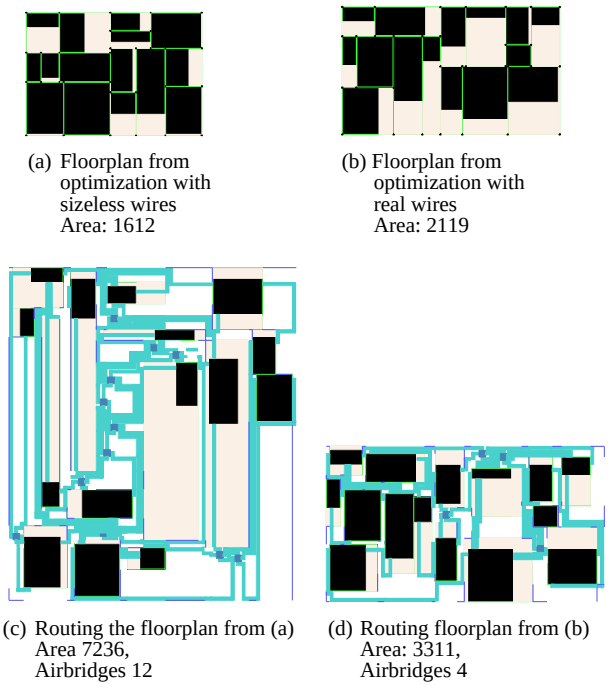


Fig. 5. Interaction between floorplanner and router for precise-length control.



(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 floorplan from (b) Area: 3311, Airbridges 4

Fig. 6. Validating the need to evaluate device-level floorplans with complete, detailed routing.

- Insufficient wire meandering space next to a device.
- Insufficient space for airbridges at a channel intersection (*i.e.*, a floorplan graph node).

Resizing of channels is done by moving the devices defining the channel. These moves are always to the "right" or "up", in the plane of the device-level floorplan.

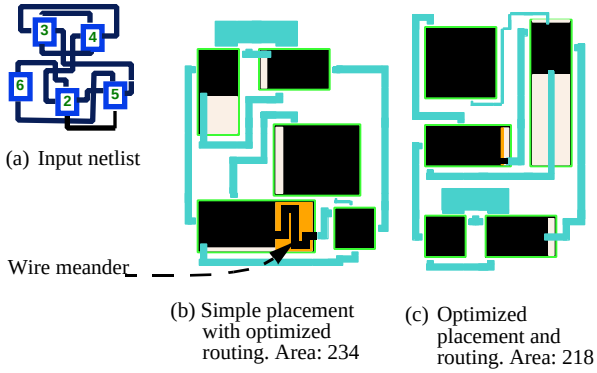
Once a channel requests more width, depending on the orientation of the channel, the device to the right or the device above is passed a request for extra space. Devices first try to satisfy the space requests from the *slacks* created by slicing during the Stockmeyer detailed floorplan sizing algorithm, or previous resizings. If this is not enough, extra space requests are passed to all channels on the opposite side. When a device moves, the required slack is added to its neighboring channels on the side from which the request originated

This resizing method is equivalent to finding and maintaining critical paths on the floorplan graph and computing slacks in the non-critical paths. Devices pass the sizing requests to nodes in the floorplan graph—channel intersections—rather than requesting space directly from the channels.

5. Results

The algorithms we presented in the preceding sections have been implemented in roughly 18000 lines of C++ code. We employed a modified version of a genetic algorithm library, GAlib, available from MIT [Wall 96].

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



(a) Input netlist

(b) Simple placement with optimized routing. Area: 234

(c) Optimized placement and routing. Area: 218

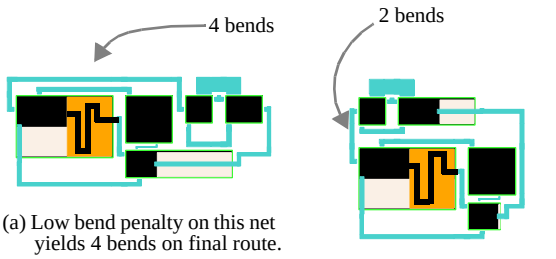
Fig. 7. Impact of length constraints on optimization

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 tool's 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 Figure 6. 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 3 times the 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 single wiring layer under length constraints, and the significant area penalty (much larger than a conventional via) of each air bridge to resolve non-planar 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 Figure 7. 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 *plan ned* to meet the constraint, and thus saves area.



(a) Low bend penalty on this net yields 4 bends on final route.

(b) High bend penalty reduces this to 2 bends.

Fig. 8. Impact of bend costs on optimization

Table 1. Execution times to floorplan our example circuits

Example Circuit (Figure ref.)	Num. Devices /Nets	GA Popul. Size	Number of Gens	CPU (sec)	CPU / gen (sec)
Figure 7b)	5/9	100	65	28.8	0.44
Figure 6c) (sizeless nets)	15/20	200	102	336	3.29
Figure 6d)	15/20	200	172	637	3.67
Figure 9b)	15/15	250	164	738	4.50

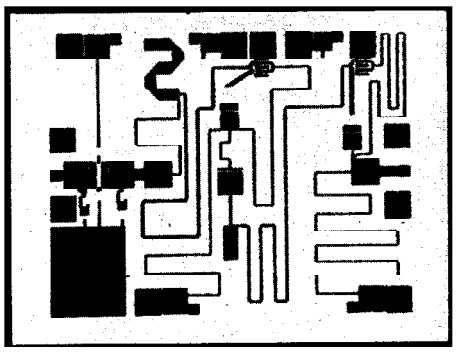
Next, we show the impact of controlling bends. We use the same synthetic netlist as from Figure 7. Figure 8 shows the results. The layout at left optimizes with uniform bend penalties on all nets. Increasing the bend cost of the highlighted net results in a different floorplan which allows the net to embed with 2 rather than with 4 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 extracted from reference [Lewis 87]. The manual layout comprises 15 devices and 15 nets, 8 of which have precise net length constraints. Since the manual layout is planar, it has no airbridges. Reverse engineering this netlist by hand, we ran our tool on a simplified version with approximately sized rectangles for each device. We imposed the same net length constraints. However, fixed pads in the manual design are approximated by movable devices in our netlist. Running the tool produced the layout at the bottom of Figure 9. Our tool was able to evolve a planar layout. More importantly *all* length constraints were met. Total runtime for this layout was roughly 12 minutes on an IBM 100MHz PowerPC604 workstation.

The automatic layout is roughly 39% 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.

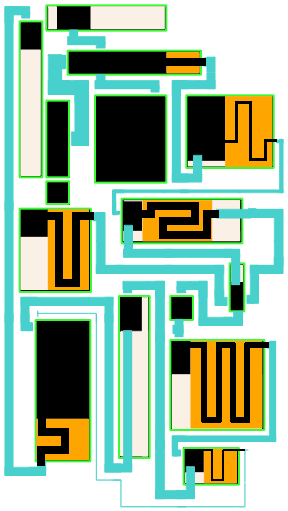
Table 1 gives the runtimes for the tool with a termination criteria of 1% change tolerance at 50 generations, running on a 100MHz PPC 604-based workstation.

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 minutes. The second row of Table 1 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.

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



a) Manual layout from [Lewis 87]:
Area: 3.8X3.0 mm
Norm. area: 1.0
15 Devices 15 Nets
8 Length constraints
0 Airbridges



b) Automatic layout:
Area: 2.98 X 5.32 mm
Norm Area: 1.39
Meets all 8 length constraints
0 Airbridges
Runtime: 12.5 minutes

Fig. 9. Manual and automatic layouts for limiting amplifier of Figure 1

6. 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 [Aktuna 96]), and optimize for overall area, wirelength and—especially critical for RF cells—planarity. For these circuits, the floorplanning process requires only a few minutes of CPU time.

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 trade-offs to optimize performance of the designed circuits. This should lead the way to a more complete layout optimization strategy for RF circuits.

Acknowledgments

This research was funded by HP-EEsof and the Semiconductor Research Corporation. We are grateful to Mitch Mlinar of HP-EEsof for several insightful discussions of the potential role of genetic optimization in device-level RF layout.

References

[Aktuna 96] Mehmet Aktuna, *A Framework for Simultaneous Placement and Routing of Radio Frequency Circuits*, Masters Thesis, Electrical and Computer Engineering Department, Carnegie Mellon University, Dec. 1996.

[Charbon 93] E. Charbon, E. Malavasi, A. Sangiovanni-Vincentelli, "Generalized constraint generation for analog circuit design", *Proc. IEEE/ ACM ICCAD*, pp. 408-414, Nov. 1993.

[Charbon 95] Edoardo Charbon, Gary Holmlund, Bruce Donecker and Alberto Sangiovanni-Vincetelli, "A Performance-Driven Router for RF and Microwave Analog Circuit Design," *Proc. IEEE Custom Integrated Circuits Conference* (1995), pp. 383-386.

[Choudhury 93] U. Choudhury, A. Sangiovanni-Vincentelli, "Constraint-based channel routing for analog and mixed analog/digital circuits," *IEEE Trans. CAD*, Vol.12, No.4, pp.497-510, Apr.1993.

[Cohn 91a] J. M. Cohn, D. J. Garrod, R. A. Rutenbar, L. R. Carley, "KOAN/ ANAGRAMII: Newtoolsfordevice-levelanalogplacementandrouting," *IEEEJSSC*, Vol.26, No.3, March, 1991.

[Cohn91b] J.Cohn, D.Garrod, R.Rutenbar, L.R.Carley, "Techniquesforsimultaneous placement and routing of custom analog cells in KOAN/ANAGRAMII," *Proc.ACM/IEEEICCAD*, pp.394-397, Nov.1991.

[Cohn94] J.MCohn, D.J.Garrod, R.A.Rutenbar, L.R.Carley, *AnalogDevice-LevelLayoutAutomation*, KluwerAcad.Publ., 1994.

[Crols95] J.Crols, S.Donnay, M.Steyaert, G.Gielen, "AHigh-LevelDesignand Optimization Tool for Analog RF Receiver Front-Ends", *Proc. ACM/IEEE ICCAD*, Nov.1995.

[De Jong 75] K. A. De Jong, *An Analysis of the Behavior of a Class of Genetic AdaptiveSystems*, (DoctoralDissertation, UniversityofMichigan) DissertationAbstractsInternational36(10)5140B.pp.102

[Feldmann 96] P. Feldmann," Computation of Circuit Waveform Envelopes Using an Efficient, Matrix Decomposed Harmonic Balance Algorithm", *Proc.ACM/IEEEICCAD*, Nov.1991.

[HP 92] *Hewlett-Packard Microwave and RF Design System Manuals*, Santa RosaSystemsDivision, SantaRosa, CA, Dec.1992.

[Jansen 86] R. H. Jansen, "LINMIC: A CAD Package for the Layout-Oriented DesignofSingle-andMulti-LayerMICs/MMICsuptomm-WaveFrequencies," *MicrowaveJournal* (Feb1986), pp.151-161

[Jansen 88] R. H. Jansen, R. G. Arnonld and I. G. Eddison, "A Comprehensive CADApproachtoDesignofMMICsuptoMM-WaveFrequencies," *IEEE JournalMTT-T* (Feb.1988), vol.36, n.2, pp.208-219

[Goldberg91] D.E.Goldberg, K.Deb, andB.Korb, "DonotWorry, BeMessy," *Proceedings of the fourth International Conference on Genetic Algorithms* (1991), pp.24-30

[Kundert90] K.S.Kundert, J.K.White, A.Sangiovanni-Vincentelli, *Steady-State MethodsforSimulatingAnalogandMicrowaveCircuits*, KluwerAcademic Publishers(1990).

[Kundert97] K.S.KundertandD.Sharrit, "SimulationMethodsforRFCircuits," *Proc.ACM/IEEEICCAD*, Nov.1997.

[Lampaert 95] K. Lampaert, G. Gielen, W.M. Sansen, "A Performance-Driven PlacementToolforAnalogIntegratedCircuits," *IEEEJSSC*, Vol.30, No.7, pp.773-780, July1995.

[Lewis87] G.K.Lewis, I.J.Bahl, E.L.Griffin, E.R.Schineller, "GaAsMMIC's for Digital Radio Frequency Memory (DRFM) Subsystems," *MTT* (Dec 1987), vol.35, n.12pp.1478

[Malavasi93] E.Malavasi, A.Sangiovanni-Vincentelli, "Arearoutingforanalog layout," *IEEETrans.CAD*, Vol.12, No.8, pp.1186-1197, Aug, 1993.

[Malavasi 96] E. Malavasi, E. Felt, E. Charbon and A. Sangiovanni-Vincentelli, "Automation of IC Layout with Analog Constraints," *IEEE Trans. CAD*, 1996.

[Otten 82] R. H.J. M. Otten, "Automatic Floor-plan Design," *Proc. 19th ACM/ IEEEDesignAutomationConf.* (1982), pp.261-267

[Otten83] R.H.J.M.Otten, "EfficientFloorplanOptimization," *Proc.IEEEIntl. Conf.onComputerDesign* (1983), pp.499-502

[Rutenbar96] R.A.Rutenbar, L.R.Carley, *etal*., "SynthesisandLayoutforAnalogandMixedSignalICsintheACACIASystem," chapterin *Advancesin AnalogCircuitDesign*, KluwerAcademicPublishers, Norwell:MA, 1996.

[Sherwani 93] Naveed Sherwani, *Algorithms for Physical Design Automation*, KluwerAcademicPublishers(1993), pp.215-223

[Telichevesky95] R.Telichevesky, K.S.Kundert, J.K.White, "EfficientSteady-State-Analysis based on Matrix-Free Krylov-Subspace Methods", *Proc. ACM/IEEEDAC*, June1995.

[Wall96] MatthewWall, MIT, http://lancet.mit.edu/ga

[Wong86] D.F.WongandC.L.Liu, "ANewAlgorithmforFloorplanDesign," *Proc.23rdACM/IEEEDesignAutomationConf.* (1986), pp.101-107

[Zurcher 85], J.F. Zurcher, "MICROS- A CAD/CAM Program for Fast Realization of Microstrip Masks," *MTT-S*, pp. 481-484, 1985.