Fast Layout Generation of RF Embedded Passive Circuits Using Mathematical Programming

Mohit Pathak, *Student Member, IEEE*, and Sung Kyu Lim, *Senior Member, IEEE*

***Abstract*— In this paper, we present a methodology for the automatic generation of layout for radio frequency (RF) designs using embedded passives. Our methodology is divided into three steps: 1) pre-layout optimization; 2) placement and routing; and 3) post-layout optimization. We show that our methodology generates layouts with small area and good performance response within a fraction of design time compared with fully manual design effort. We make use of circuit models to represent and optimize the physical layout of the resistance-inductance-capacitance (RLC) components as well as the entire RF circuit that uses them. We perform non-linear mathematical programming-based optimization at various stages of the methodology to achieve high quality layouts. Full wave electromagnetic simulations are kept completely out of the design loop, so our methodology significantly reduces design time. We have used our methodology to successfully generate layouts for large-scale filter circuits.**

***Index Terms*— Embedded passive, filter design, layout generation.**

I. Introduction

PASSIVE elements are an important part of microelectronic devices. The number of passive components in handheld devices and computers is greater than 80% of the total part counts. Moreover, the passive to active ratio continues to grow [1]. Embedded passive is an emerging technology that has a potential for increased reliability, improved electrical performance, size shrinkage, and reduced cost [2]. Using this technology, various passive components used in systems are integrated into packaging substrate via multiple layers. A very popular choice for the embedded passive substrate is liquid crystalline polymer (LCP) (see Fig. 1). LCP is a low-loss material ($tan\delta = 0.002$) with relative permittivity (ϵ_r) of 2.95. The material properties are invariant up to 20 GHz with negligible moisture absorption (0.04%). The process is also known to be low cost and low temperature [2]. Thus, LCP-based embedded passives promise high quality passives implemented in the packaging substrate.

However, the design process for the circuits using embedded passives is non-trivial due to the complex electromagnetic interactions that cause undesired parasitics, leading to non-ideal circuit behavior. The desired response of a given radio frequency (RF) layout is tightly-coupled with the response of the individual components and the parasitics of wires that connect them. The manual design cycle for such layouts can be very time-consuming, typically requiring a few weeks, if not months, of effort. The design time bottleneck in this case is the computationally-intensive electromagnetic simulations that the designers perform to ensure that their layouts achieve desired responses and performance. Depending on the design requirements, many iterations of "layout update+electromagnetic (EM) simulation" cycle are needed, thereby increasing the overall design time and time-to-market.

Our goal in this paper is to reduce this prohibitive design time of manual approach by an automatic layout generation tool. Our method is library-based, where we assume a library of embedded passive resistance-inductance-capacitance (RLC) components with varying values and sizes is given. This library contains the circuit models, layouts, and performance models for the RLC components. Given a circuit topology to be implemented using embedded passives, we first select an initial set of components to be used from the library based on the RLC values. We then combine the circuit models of individual components and build one for the entire design. The initial selection of components is then updated by our novel non-linear optimization method. Once this pre-layout optimization is completed, we perform automatic placement and routing for the components in the final component list. We present a novel integer linear programming (ILP)-based method to ensure the high quality of the final layout while alleviating the computational burden of the ILP solver. Once the layout construction is completed, we perform post-layout optimization that involves component resizing and rerouting. Our novel approach ensures that the final layout is the most area-compact and meets the desired performance requirements. Compared with the fully-manual approach, our automatic method offers the following advantages.

1) *Design Time:* Our method generates a layout with reasonable quality within a fraction of design time. Moreover, our final layout can further be refined manually by the RF designers. Since it is easier to tweak a given good solution to meet the desired goals rather than starting from scratch, our approach reduces the overall design time significantly compared with the fully-manual approach.
2) *Design Space Exploration:* Our method can easily generate *multiple* layouts with *similar* response and *varying*

Manuscript received November 2, 2010; revised June 18, 2011; accepted September 1, 2011. Date of publication December 2, 2011; date of current version January 5, 2012. This work was supported in part by the National Science Foundation under CAREER Grant CCF-0546382 and the Interconnect Focus Center. Recommended for publication by Associate Editor J.-H. Zhao upon evaluation of reviewers' comments.

The authors are with the School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA 30332 USA (e-mail: mohitp@gatech.edu; limsk@ece.gatech.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCPMT.2011.2170214

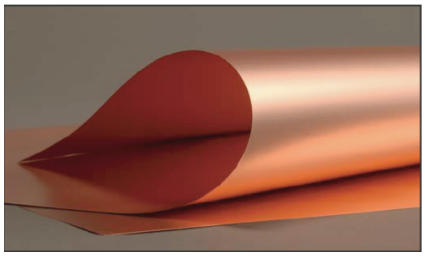

Fig. 1. Copper-cladded flex LCP sheet.

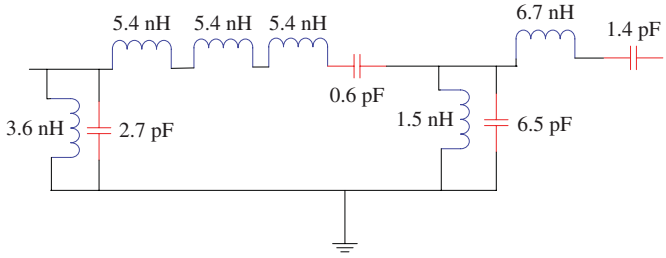

Fig. 2. Circuit schematic of a ten component band pass filter.

dimensions. This allows the designers to explore the design space and choose the ideal shape for the entire design.

Our experiments indicate that this entire process typically takes less than 30 minutes for 10–20 component RF filters. In most cases, the final filter response is very close to ideal circuit response, which can be very hard to achieve even for expert RF circuit designers. Compared with other heuristic approaches, our mathematical programming-based methods generate better quality layouts within a comparable runtime.

The rest of this paper is organized as follows. In Section III we discuss previous works in analog and RF circuit layout generation. In Section IV we discuss our overall design flow. In Section V we discuss our pre-layout optimization method. In Section VI we discuss integer programming-based method to generate place and route solutions. In Section VII we discuss how to optimize a given place and route solution for performance and area. We present our experimental results in Section VIII and conclude in Section IX.

II. Motivation

In this section, we briefly discuss how interconnect parasitics can impact the performance metrics of a given in RF circuit layout. We also discuss the role of mathematical programming in achieving good quality layouts. Let us assume that we are given a circuit as shown in Fig. 2 for which we would like to generate its corresponding layout that meets the required performance objectives. A simple approach to generate the layout of the given circuit would be as follows.

1) Choose components based on the values as shown in the Fig. 2 from a library of components.
2) Manually place the components (based on certain objectives) as seen best.
3) Manually route various interconnects.
4) Analyze the quality of the designs based on EM simulations.

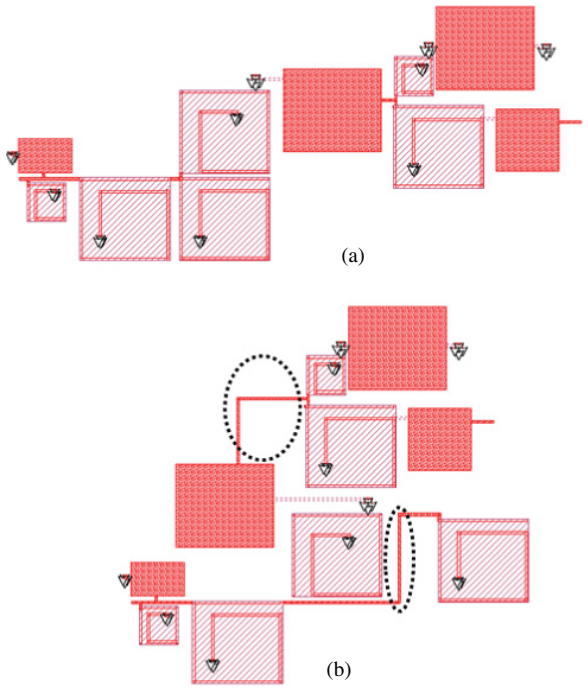

Fig. 3. Layout comparison of a ten-component bandpass filter (BPF). (a) Wirelength optimized. (b) Wirelength not optimized. The circled regions show excessive wirelength being used when wirelength is not optimized.

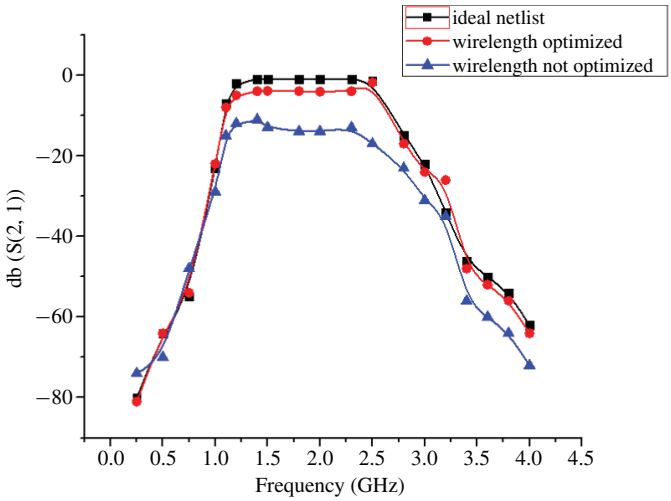

Fig. 4. Response comparison of the layouts shown in Fig. 3.

5) Repeat the above steps until the desired goals are achieved.

In the above "manual" flow, two steps are critical. First, how to place the components (= placement). Second, how to connect various components together (= routing). The way we arrange the components can impact the total area of the layout. Typically, we would like to minimize the area because larger area may result in larger costs. In addition, how we arrange the components also determines how much wirelength is needed to connect different components together. Moreover, to meet the performance objectives, we desire to minimize the wirelength needed to connect different components. In Fig. 3, we show two layouts of the ten component band pass

filter shown in Fig. 2, where the layouts differ in terms of wirelength. Based on the responses shown in Fig. 4, we see that reducing wirelength helps us to meet the performance objectives easier.

Based on our analysis, the wires (including vias) needed to connect different components largely behave as inductors. However, these wires (or essentially parasitic inductance) can make it difficult to achieve the performance requirements. From an RF designers point of view, adding these extra wires may cause a change in various design objectives such as insertion loss, return loss, or center frequency of a given filter. While designers can perform sensitivity analysis and see the impact of parasitic wires on the circuit performance, such an analysis is dependant on: 1) amount of parasitic wire added (essentially parasitic inductance); 2) where the parasitic wire is added; and 3) the particular topology of the circuit. The goal of our automatic layout generator is to generate layouts for a wide variety of circuit topologies and to do it quickly with as little feedback from the designer as possible. Thus, during our layout generation stage, we try to minimize wirelength needed to connect different components to avoid problems caused by the parasitic wires.

Based on our discussions above, our two main objectives during automatic layout synthesis is to minimize area (due to cost related factors) and reduce wirelength (to avoid parasitic inductance.) We effectively use ILP methods to meet these objectives. While ILP is an effective technique to minimize area and reduce wirelength, it still cannot completely eliminate all the wire parasitics. Thus, the initial layout obtained may not meet the required performance objectives. To tackle this issue, we further apply non-linear programming during our post-layout optimization stage. The details of the non-linear optimization is discussed in Section VII. We use a commercial non-linear optimization tool in our flows. This ILP +non-linear hybrid approach is shown to be highly effective in synthesizing RF layouts that meet the desired performance goals.

III. Previous Work

Automatic layout generation methods for analog and RF circuits can be broadly divided into two major categories: template-driven method, and full-custom method. In template-driven method, a geometric template for the layout of the circuit is provided for a given design. The final layout is completed by correctly generating the components and the wires that fit into the selected template. Such an approach works the best when changes in circuit parameters may result in little change in the layout structure. Such an approach works when we already have a good layout template for a given circuit. A template-driven approach also tends to have shorter runtime because the solution space of the layout has been reduced significantly. The works in [2]–[5] belong to this group. The works in [4], and [5] look at performance optimization of the layout while considering interconnect parasitics. They ensure that the routing topology of the given layout is maintained during resizing. The authors in [2] discuss techniques for automating the design of embedded passive components. Note that these works focus only on resizing a given layout topology instead of constructing layouts.

In full-custom method, the entire layout for a given circuit is built by the tool. The main goal is to generate a layout that satisfies the performance specifications while minimizing the overall layout area. In this approach, no known template of the layout is provided as an input and the tool has to generate its own layout. The layout is also optimized for area, wirelength, and aspect ratio under various geometric constraints to meet the design requirements. Tools such as ILAC [6] and KOAN/ANAGRAM [7] are well known. KOAN relied on a very small library of device generators and incorporated important layout optimizations into the placer. The placer is based on simulated annealing algorithm and can dynamically fold, merge, and abut metal-oxide-semiconductor devices. ANAGRAM is a maze-style detailed router, capable of supporting several forms of routing considering parasitic avoidance and symmetry. The authors in [8] preformed floor planning for RF circuits, where they evaluate RF layouts in terms of geometric performances such as wirelength and planarity constraints.

However, these tools primarily focus on geometric objectives. These objectives in general improve analog response of the final designs, but the approaches that tackle the analog metrics more directly are desired. A major challenge in this case is on how to manage undesired parasitics created in the overall analog/RF layouts. Tools that use this approach include ROAD [9] and PUPPY-A [10]. In these tools, sensitivity analysis is used to quantify the impact of low-level layout decisions on final analog performance. This paper in [11] quantifies the impact of layout decisions on circuit performance and showed how to achieve maximum bounds on various parasitics. This paper in [12] targets primarily complementary-metal-oxide-semiconductor technology, not packaging substrate such as LCP or LTCC. They do not describe a way of estimating parasitics for more than two-pin nets. Such an approach would not work for RF embedded passive circuit since knowing the exact nature of interconnect parasitics is critical. Sommer *et al*. [13] presented a layout synthesis algorithm for embedded passive components such as capacitor, resistor, and inductor. But, they do not discuss how to use them to construct an entire circuit.

IV. Design Flow

An overview of our automatic layout generation tool flow is shown in Fig. 5. The basic approach is to perform pre-layout optimization, followed by the generation of a placement and routing solution, and post-layout optimization for the entire layout.

1) Step 1: Component shapes are chosen from the library based on initial values in the given circuit.
2) Step 2: Pre-layout optimization using approximate wirelength and via connectivity information is performed. The component selection and resizing are performed in this step.
3) Step 3: A place-and-route solution optimizing various geometrical objectives such as area and wirelength is generated. Circuit responses are carefully monitored and optimized based on the circuit model of the entire layout.

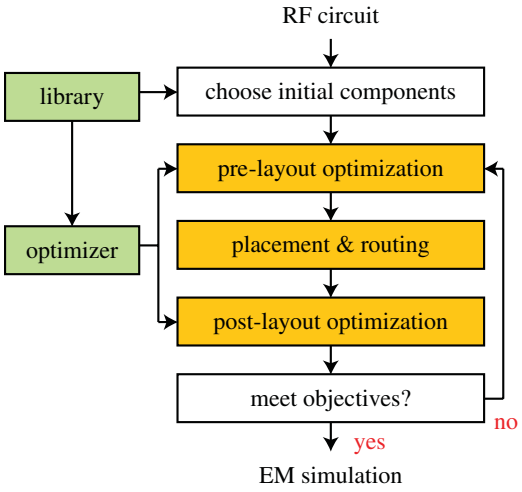

Fig. 5. Design flow of our automatic layout generation of embedded passive RF circuits.

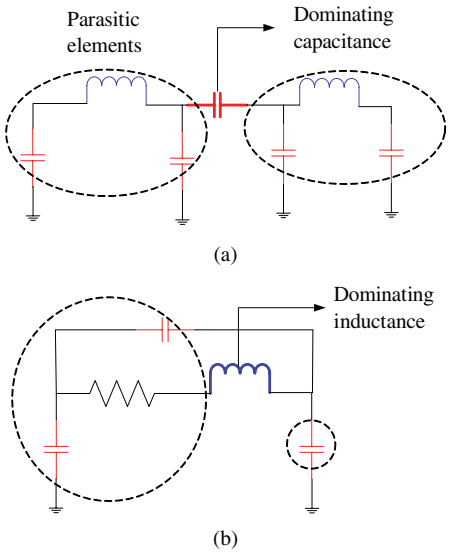

Fig. 6. Circuit models of (a) single capacitor and (b) single inductor, showing the dominating and parasitic elements. The parasitic elements are circled.

4) Step 4: Post-layout optimization is performed considering the entire layout. In this step, we try to minimize the layout area while meeting the performance objectives.

We use commercial non-linear optimization tools in our optimization engine. Finally, we perform a full-wave EM simulation using SONNET for final verification of the layout generated. Compared to the traditional design flow where "manual placement and routing" and "EM simulation" are iterated, our new design flow constructs a high-quality layout in a fraction of the time since the time-consuming EM simulation is kept out of the loop. Our layout optimization is mainly based on the circuit representation of the components and the overall layout. Our final solution can be used for further manual touch-up if necessary.

V. Pre-Layout Optimization

In this section we describe the details of pre-layout optimization. Given a library of embedded passives, a circuit netlist to be implemented, and its initial selection of the components from the library, the objective of our pre-layout optimization is to find new component dimensions which better meet the desired circuit objectives while considering intra-component parasitics. In addition to approximate the effect of nets connecting different components, each net is assumed to have a non-zero wirelength. The net is considered to have a single Steiner point. A single Steiner point results in a star-shaped net topology. Steiner tree problem involves connecting a designated set of vertices (pins), using a minimum weight tree. In addition to the set of designated vertices, additional points in space may be used to reduce the length of the Steiner tree. These additional points are known as Steiner points. At this stage of the design we assume that the net has a single Steiner point. This topology consists of all pins of the net plus one additional point (Steiner point). We then connect every pin in the net to this Steiner point. The idea is to obtain approximate shapes of components considering the effect of both intra-component and inter-component parasitics. This ensures that the layout solutions obtained during place-and-route step contain geometric shapes that are closer to the final layout obtained.

A. Component Modeling

Fig. 6 shows the circuit model of a capacitor and an inductor. The circuit elements in each component are grouped into two categories, namely, dominating element and parasitic elements. The dominating element value reflects the main inductance or the capacitance value of a given component. During initial component selection, components were selected based on their dominating element values. Due to the effect of parasitics in each component, however, the initial selection does not always meet the desired circuit response. The goal of our pre-layout optimization is to find new components (possibly with new dimensions) such that they meet the desired circuit responses while considering both the dominating and parasitic elements. In addition to intra-component parasitics, additional parasitics such as vias connecting to ground and interconnects are also modeled. Due to these changes in the circuit model, the overall shape of the component after the optimization may change, i.e., components are resized. Note that our optimization is not based on the response of individual components but on the circuit model of the *entire* design.

B. Non-Linear Optimizer

To optimize (= resize) the components in a given unplaced/unrouted circuit, we use a commercial non-linear circuit optimization tool. The input to this problem is the circuit representation of the design (= netlist) and the embedded passive library. The objective of this non-linear optimization depends on the type of design and desired goals. In case of RF filters, for example, various frequencies of interest, signal loss, and noise are the main targets. The non-linear optimizer determines the value of the variables in the formulation so that the objective function is optimized. The non-linear optimizer

works as a circuit optimizer. The objective of the optimizer is to find circuit components that meet the performance objectives. The constraints for the optimizer consist of various equations that satisfies the Kirchoff's laws at various nodes based on nodal analysis. We optimize the circuit at a certain set of discrete frequency points within the range of interest. The non-linear optimizer is discussed in greater detail in the following section.

The input to the non-linear optimizer is the circuit netlist N, a set of circuit components that to be resized $C = c_1, c_2, \ldots, c_n$, and a set of frequency points $W = w_1, w_2, \ldots, w_r$ at which we want to optimize the circuit response. For each frequency point w_i, we generate a set of constraints $G_i V = I$, where G_i is the conductance matrix obtained at frequency w_i using modified nodal analysis. V and I are the node voltages and the current in the branches of the circuit. We also compute the response of the circuit at each frequency R_i that corresponds to the S or Z parameter response of the circuit at the frequency w_i. Based on the above discussion, the non-linear optimizer is formulated as follows:

$$\text{Minimize } \sum \left| R_i - R_i^{req} \right| \tag{1}$$

$$\text{Such that } G_i V = I, \ \forall i \ C_{low} \leq C \leq C_{high}. \tag{2}$$

In the above equations, R_i^{req} corresponds to the required response at each frequency w_i. Equation (2) satisfies the Kirchoff's laws at each frequency w_i. C_{low} and C_{high} correspond the lowest and the highest values of the components that can be resized.

C. Pre-Layout Optimization Algorithm

During our pre-layout optimization process, the parasitic element values of all components are fixed while the dominating element values are changing. The optimization process finds the best values for the dominating elements while optimizing the overall objective function. For every component with newly updated dominating element value, we replace it using the given library. This new component may have a different dimension than the old one. In addition, since the new component has different dominating value, the parasitics are also different in this new component. We then repeat the overall process until the maximum change in dominating element values is below a threshold or the maximum number of iterations is reached. Note that the new dominating values the non-linear optimizer determines should lie within a fixed range so that we can always find the corresponding component in the library. This range is determined by the size of the library.

The pseudo-code of our pre-layout component resizing is shown in Fig. 7, where c_j is component j in the circuit, c_j^{dom} and c_j^{para} are its dominating and parasitic elements. Line 3 fixes the parasitic element values and assigns the dominating element values as variables for the optimization engine. In line 4 we find the minimum and maximum values (= range) for dominating elements of each component based on the availability in the library. The optimization engine is called in line 5. Lines 6 and 7 update the current solution based on the values (both dominating and parasitic elements) obtained by the non-linear optimization engine. The above process is repeated until the maximum change in dominating element values is below a threshold or the maximum number of iterations is reached. Table III in Section VIII provides experimental results on the effectiveness of our pre-layout optimization step. We observe that the area and performance metrics significantly improve at the cost of 2× runtime overhead.

Pre-layout Optimization
input: netlist *NL* with initial component selection, library
output: *NL'* with resized components

1. $cnt = 0$;
2. **while** ($max\_change(c_j^{dom}) > \epsilon$ and $cnt < max\_itr$)
3. $\quad \forall c_j \in NL$, set c_j^{dom} as variable and c_j^{para} as fixed;
4. $\quad \forall c_j^{dom}$, find [min, max] range;
5. $\quad$ optimize $\forall c_j^{dom}$ to meet response;
6. $\quad$ find new components for new c_j^{dom} values;
7. $\quad$ update c_j^{para} values;
8. $\quad cnt++$;

Fig. 7. Pseudo-code of our pre-layout optimization algorithm.

VI. ILP-Based Placement and Routing

Compared with digital circuits, analog circuits are considerably smaller. However, the analog circuits are highly sensitive to layout parasitics, thus making analog placement and routing highly challenging. Our goal during the placement and the routing step is to focus on geometry-related objectives such as area and wirelength. The objective is to keep the parasitics due to wirelength a minimum while reducing area. Reducing wirelength ensures that interconnect parasitic is minimized.[1] In addition, during layout generation we perform circuit extraction and simulation-instead of EM simulation-occasionally to make sure that our layout obtains the desired quality while meeting the given design specifications. Our strategy is to perform placement and routing simultaneously to optimize the layout area and wirelength effectively. Our optimization engine choice is ILP, enhanced with various speedup techniques. Compared with the popular rectangle packing-based method such as sequence pair [14], [15], our ILP-based method obtains significantly better quality layouts as demonstrated in Section VIII. The runtime overhead is not significant because of our strategy to reduce the problem size while maintaining the solution quality. In the following sections, we discuss the details of our ILP formulation.

A. Issues in ILP-Based Method

The layout generation problem can be considered as a rectangle placement problem since even interconnects can be considered as rectangles. In order to solve this kind of

[1]Note that we leave enough keep-out zones around each RLC component so that the EM coupling between components, when they are tightly packed for area minimization, is still negligible.

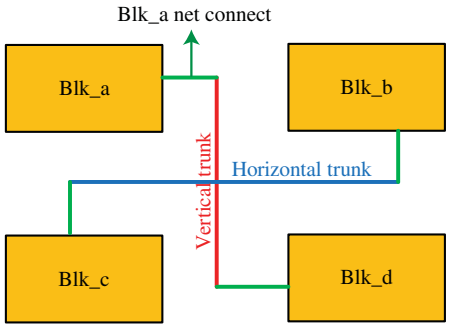

Fig. 8. Topology of a dual trunk net with one horizontal and one vertical main trunk that intersect. Each block has a net connect rectangle which extends from the pin on the block to either the vertical or horizontal trunk.

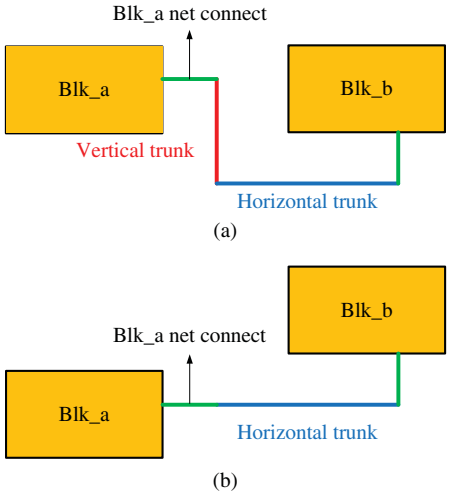

Fig. 9. Possible configurations of dual trunk net topology for two-pin net. (a) Horizontal and vertical trunk form a L-shape. (b) Vertical trunk reduces to length zero, resulting in a net topology that looks like a single trunk.

problem, we need to address three issues: 1) the number of rectangles to be used to model interconnects; 2) the number of integer variables; and 3) loops in the design. The first two issues have a direct impact on the complexity of the resulting ILP formulation. In case of the loop issue, we construct our layout iteratively by considering one net (and the components it connects) at a time. Specifically, we place and route the components in the current net, and then we move onto the next net, etc. If there exists a loop between the components already placed and the components to be placed, we need to generate additional constraints to connect them together. This requires more variables and constraints compared with cyclic netlist. Thus, we first propose a technique to perform placement and routing for acyclic netlists. We later show how our formulation can be modified to consider circuits containing loops.

In order to use integer programming for routing, we need to know how many rectangles should be used for a given net and how those rectangles can be connected. To address this problem, we define a fixed topology for each net. Every net topology in the design has two trunks, one horizontal and other vertical that intersect each other. In addition, each block that needs to connect to the net connects to either the horizontal trunk or vertical trunk of the net. These trunks can be of zero length if necessary. Fig. 8 describes how this net topology is defined. It can be seen that our trunk-based routing topology can handle multiple pin nets easily. For a two-pin net, the length of horizontal and vertical trunk reduces such that the trunks do not extend beyond the point of intersection. This results in a routing topology for two pin nets that resemble a single trunk or a L-shaped trunk as shown in Fig. 9.

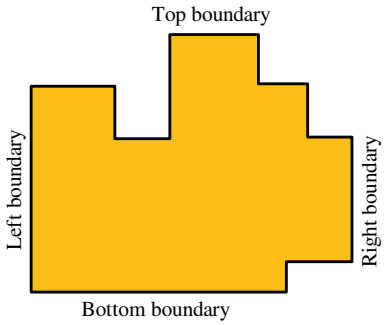

Fig. 10. Shape of a general polygon with top, bottom, left, and right boundaries.

The second major concern is to reduce the number of integer variables. We observed that in order to perform placement for n rectangles, we need $n \times (n-1)$ integer variables. To tackle the problem of large number of integer variables, we use two approaches.

1) Incremental placement and routing: We perform the placement and routing of the circuit one net at a time.
2) Super block-based integer programming: We make use of our knowledge of the layout topology to reduce the number of integer variables.

In the following sections, we discuss in detail how these techniques are used to generate layouts for our circuits.

B. ILP-Based Incremental Placement and Routing

Our approach is net-by-net, where at each iteration of the algorithm we perform placement and routing of the current net and the blocks connected to it. This process is done on top of the nets and blocks that are already laid out from previous nets. Thus, the total number of place and route iteration is equal to the number of nets in the design. To successfully implement such a scheme, we need to ensure that at each iteration no overlap exists between the current and previous nets and blocks. We use ILP method to perform this placement and routing. The objective of our ILP formulation is to minimize wirelength and area of the newly created layout at each iteration. In the following sections, we show how to generate the ILP constraints to ensure that the layout generated has no overlaps while optimizing the objective function.

1) Overlap Constraints Between Rectangles and Polygon: At any given iteration of our incremental ILP-based approach, the current layout solution constructed so far takes the form of a general polygon. Fig. 10 shows how a general polygon may look like. To ensure that the rectangles to be newly added during the new iteration do not overlap with the existing layout, we generate overlap constraints between a set of rectangles and a polygon.

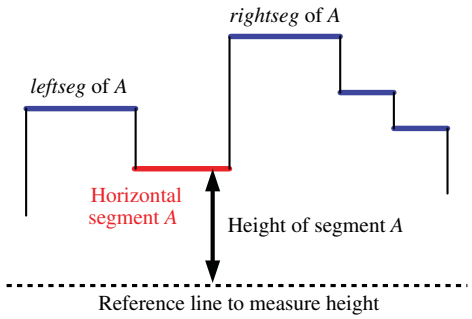

Fig. 11. Top boundary of a polygon, where all horizontal segments are highlighted in bold. A given horizontal segment A is shown with its *leftseg and rightseg*.

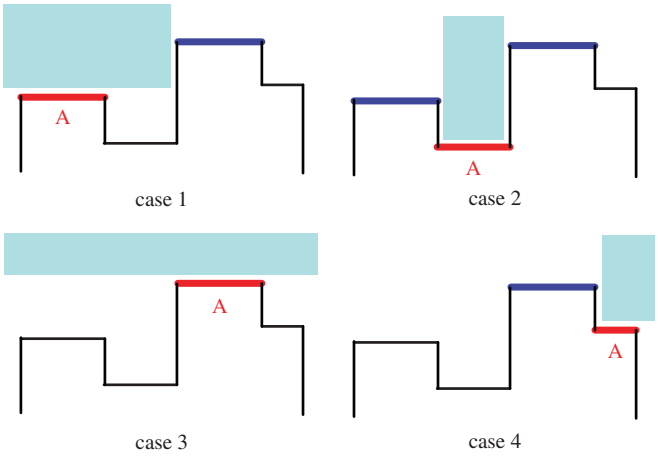

Fig. 12. Available placement area (shown in blue) identified based on a given horizontal segment A on the top polygon boundary.

To ensure that two rectangles do not overlap, (8)–(15) as discussed in Appendix A are used. In case of rectangles and a polygon, however, different equations are required. In Fig. 10, we see that there exists a large number of regions where a rectangle can fit into. To solve this problem, we generate constraints separately for each of the four sides of the polygon boundary marked as top, bottom, left, and right. In what follows, we show how the constraints are generated for the top boundary of the polygon. The constraints for other sides can be generated similarly.

In order to obtain high quality solutions, we need to ensure that the constraints we generate allow the newly added rectangle to be placed in every possible region in the polygon while keeping the number of integer variables at minimum. Fig. 11 shows various "horizontal segments" one side of the polygon. Each horizontal segment is defined by its left and right coordinate and the height of the segment. For a given horizontal segment A, we define *leftseg* and *rightseg* as the first horizontal segments that are to the left and right of A and are above the height of A. The height of a segment is measured with respect to a large negative reference base value as shown in Fig. 11.

We have the following lemma.

Lemma 1: Given a horizontal segment A on a polygon boundary, the valid placement constraint that ensures a rectangle R does not overlap with existing polygon is obtained as follows: R must be above the height of A and to the left of *rightseg* and to the right of *leftseg*.

Since the rectangle is to be placed above the current horizontal segment, we observe that R will not overlap with the segments that are below A. In addition, since it is to the left of *rightseg* and to the right of *leftseg*, it does not overlap with other horizontal segments which may be above A. This is because *leftseg* and *rightseg* are the first segments to the left of, right of, and above A. Such constraints can be generated for each horizontal segment, which in turn covers the total placeable area for the given rectangle R on one side of the polygon as shown in Fig. 12. Repeating this process for all four sides ensures the rectangle is placed outside the polygon without any overlap. In addition, our area objective function minimizes the overall area of the newly expanded polygon. To further reduce the number of integer variables used in our ILP formulation, we make use of our knowledge of the layout topology. The details of this method are discussed in Appendix B.

C. Net Ordering

Net ordering decides which net and the blocks in it to be placed and routed next in our net-by-net approach. This net ordering can have a big impact on the final layout quality. We implemented a net-ordering scheme, where the nets are placed and routed such that the layout grows in a star-shape fashion. To do this, we start with a net which is at the center of the netlist tree having the maximum distance from the input/output ports. Since the algorithm routes the nets in breadth-first fashion, the order of the remaining nets is almost fixed. Any ties between nets which may be expanded simultaneously under the breadth-first order is resolved based on the degree of the nets, where the net that contains more blocks gets expanded first.

D. Sizing at Each Place-And-Route Iteration

Since we perform place-and-route incrementally at each iteration, we know the exact amount of wire parasitics added at a given iteration. Thus, at each iteration we perform a layout optimization that is based on its circuit representation, where the new interconnect parasitics are added to the existing circuit representation. Incremental optimization helps us to compensate for the effect of parasitics added by a given net at each iteration. We observe that using such an approach helps us in getting better quality results because we can take care of various wire-parasitics added during the placement and routing iteration. Our post-layout optimization method presented in Section VII is used in between the net-by-net place-and-route iteration to resize the components in the polygon and compact the overall polygon layout.

E. Summary of the Overall Design Flow

In this section we summarize how integer programming can be used for layout generation. To generate the layout we follow the following steps.

TABLE I
AREA (mm^2), WIRELENGTH (MILS), AND RUNTIME (min) COMPARISON BETWEEN ILP VERSUS SEQUENCE PAIR-BASED METHOD

	ILP method			Sequence pair		
	area	WL	cpu	area	WL	cpu
10-comp BPF	31.1	205	46	29.9	298	37
11-comp dual-BPF	27	182	143	22.1	263	88
15-comp BPF	36.5	248	123	32.2	356	97
21-comp HPF	73.1	341	247	68.3	465	189
Ratio	1.0	1.0	1.0	0.91	1.58	0.74

1) Start with net and identify the various super blocks present.
2) Formulate ILP including various constraints and objective functions.
3) Optimize layout while considering the new net parasitics added.
4) Keep repeating the above steps until all the nets and blocks are placed and routed.

The above technique works well for circuits whose netlists have no loops present. In case there are loops present, we first extract the largest sub-netlist from the existing netlist that forms a tree. ILP-based placement and routing is performed on the sub-netlist. Finally the pins that need to be connected to complete the loop are connected using a maze router.

F. Sequence Pair Placement and Maze Routing

In this section we discuss the details of our sequence pair [14] based layout generation scheme. The purpose is to compare our ILP-based method presented in Section VI with this popular approach as shown in Table I.

We first generate the placement of various components in the design using this simulated annealing-based [14] optimization. The routing is completed using an area router with corner stitching data structure. The details of the approach are discussed in the following sections.

1) Component Placement Using Sequence Pair: We discuss briefly the sequence pair algorithm described in [14]. A sequence-pair of a set of modules is a pair of sequences of the module names. For example, $s = (abcd, bacd)$ is a sequence-pair of the module set $\{a, b, c, d\}$. We can derive the relative positions between the modules from a sequence-pair by the following rules (see Fig. 13).

1) If $s = (\ldots a \ldots b \ldots, \ldots a \ldots b \ldots)$, then module b is to the right of module a.
2) If $s = (\ldots a \ldots b \ldots, \ldots b \ldots a \ldots)$, the module b is below module a.

Given a candidate solution $c = (s1, s2)$, a pair of constraint graphs (G_h, G_v) are generated according to the sequence pair to represent the relative positions between the modules. In the horizontal (vertical) constraint graph the $G_h(G_v)$, the vertices represent the modules and the edges represent the relationship between the modules in the horizontal (vertical) direction, e.g., if A is to the left of B (A is below B), there will be an edge from A to B in $G_h(G_v)$ with weight $w(A)(h(A))$, where $w(A)$ is the width and $h(A)$ is the height of module A. The final position of all modules is determined by running the shortest path algorithm on G_h and (G_v), which determine the locations of the modules in x and y, respectively. Fig. 13 shows packing for sequence pair $s = (abcd, bacd)$.

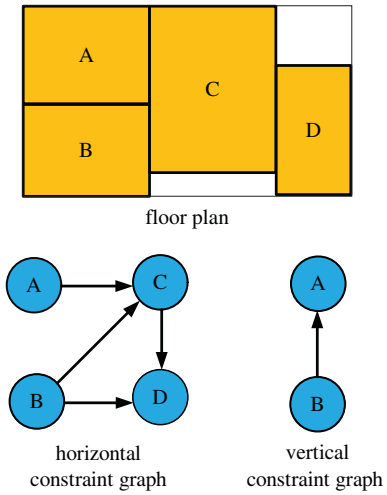

Fig. 13. Example of sequence pair $(abcd, bacd)$.

Simulated annealing [16] is a general optimization strategy based on iterative improvement with controlled hill climbing. This hill climbing allows annealing to avoid many local minima and reach better global solutions. Sequence pair is used to represent various solutions in the annealing frame work. In our case of embedded passive component placement, the metrics we use for evaluating a given sequence pair are: 1) area; 2) wirelength; and 3) coupling. Coupling as a cost metric is considered only for inductors in the circuit. The objective is to place two inductors far apart or diagonally apart thus ensuring that there is low coupling between them.

2) Area Router: In order to complete routing for the sequence-pair based placement solution, we employ a line expansion router [7]. The router is built on top of tile plane data structure [17]. Such a data structure helps to remove the limitations of a grid-based routing graph. The data structure is based on representing both the used and unused spaces in the layouts as tiles. The tiles are limited to be rectangles with sides parallel to the axis. Various tiles are linked by a set of pointers at their corners called corner stitches. Such a representation of the layout has distinct advantages because the size of tiles may vary: the structure adapts naturally to variations in sizes of used tiles. Efficient algorithms exist for manipulating and querying the data structure.

The router routes one net at a time. The router works by storing partial paths sorted by cost. The least cost path is selected and expanded. This expansion results in new partial paths which are added into the list of existing partial paths. All paths are maintained in heap kind of data structure. The process of expanding continues until all the pins in the given net are connected. The objective is to ensure that routing for all nets is completed and total wire length for all nets is optimized.

VII. POST-LAYOUT OPTIMIZATION

The goal of the placement and routing phase was to minimize the parasitic effects of interconnect with our wirelength

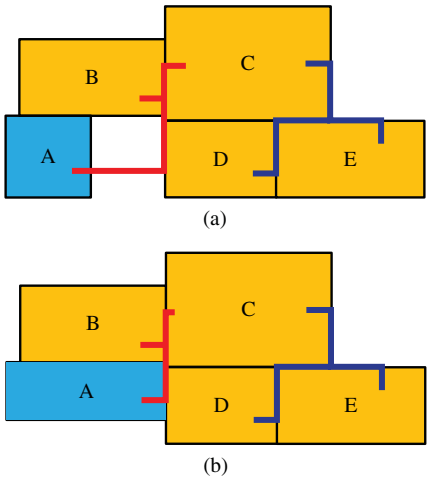

Fig. 14. (a) Initial placement and routing solution. (b) Change in routing and block locations, where the size of component A is changed.

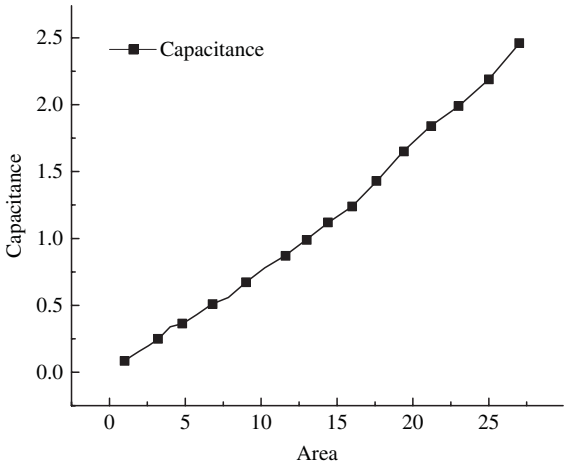

Fig. 15. Change in capacitance as a function of area. We observe that capacitance can be largely approximated as varying linearly with the area in regions close to the current capacitance value.

objective so that the overall RF embedded passive designs show responses that are closer to the desired responses. However, the response of layout may not always match the desired response. In addition, component sizing done during pre-layout stage may not be ideal for the given layout topology simply because the interconnect length is not determined during that stage.

During our post-layout optimization stage, we perform component resizing while considering interconnect parasitic such that overall layout topology is preserved as much as possible and the area is minimized. We propose a new way to perform post-layout optimization named relative place-and-route that is based on sequential linear programming. We then compare this method with another approach we develop: iterative feedback.

A. Relative Place-And-Route

To optimize a given layout topology for both area and performance, we make the following observations.

1) Interconnects can have a huge impact on performance. Thus, minimizing interconnect length while performing layout optimization is critical to obtain best layout results.
2) The shape of the component is critical based on the current layout topology. Choosing the right shape of the component helps to reduce the overall layout area.

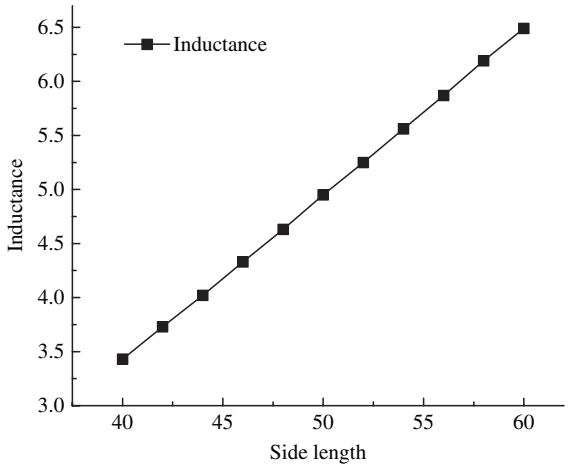

Fig. 16. Change in the main inductance based on its side-length for inductors. We observe that inductance can be approximated as a linear function of its side-length when other parameters are fixed.

To model interconnect length accurately based on the size of components, we introduce the concept of relative place-and-route. Previous work in [4] and [18] enable us to perform rectangle compaction horizontally or vertically. We employ a similar technique to generate a set of constraints, where locations and dimension of each component and interconnect can be determined based on the width and height of each component. Fig. 14 shows the effect on routing solution based on resizing of a related component. The equations required to ensure layout topology is maintained after resizing are as follows (we show equations only for horizontal axis. Similar equations exist along vertical axis):

$$cmp^i_{xr} = cmp^i_{xl} + cmp^i_w \;\forall\; cmp^i \tag{3}$$

$$wc^i_{xl} \geq cmp^i_{xr} + const \;\forall\; wc^i \tag{4}$$

$$cmp^i_{xl} \geq wc^i_{xr} \;\forall\; cmp^i \tag{5}$$

$$lw^i = wc^i_{xr} - wc^i_{xl} \forall wc^i. \tag{6}$$

Equation (3) ensures that the right coordinate of a component cmp_i is equal to the left coordinate of the component plus the width cmp^i_w of the component. Equation (4) ensures that the left coordinate of a given wire wc^i_{xl} is greater the right coordinate cmp^i_{xr} of all blocks to the left of the given wire. Similarly, (5) ensures that all blocks to the right of a given wire have their left coordinates cmp^i_{xl} greater than the right coordinate of the wire wc^i_{xr}. Equation (6) computes the wirelength of a given wire. If desired, additional constraints, such as specifying the minimum separation between blocks and wires, can be easily added to this ILP-based frame work.

B. Area-Based Component Sizing

To effectively resize components for area and performance optimization, component resizing is done based on the

Post-layout component resizing
input: initial layout and its circuit presentation NL
output: optimized layout with resized components

```
1.  partition the initial circuit;
2.  for (each sub-circuit sub_i ∈ NL)
3.      cnt = 0;
4.      do
5.          find circuit model ckt_i^sub of layout;
6.          resize ∀c_j^sub component ∈ ckt_i^sub;
7.          cnt ++;
8.      while (error (ckt_i^sub) > ∈ and cnt < max_iter)
9.  intialize ∀c_i ∈ ckt such that c_i = c_i^sub;
10. cnt = 0;
11. do
12.     find circuit model ckt of total layout;
13.     resize ∀c_j component ∈ ckt;
14.     cnt ++;
15. while (error (ckt) > ∈ and cnt < max_iter))
16. return resized layout;
```

Fig. 17. Pseudo-code of our Iterative Feedback resizing algorithm.

sensitivity of performance parameters to the component dimensions. We make the following two critical observations.

1) Capacitance of a component can be approximated as a function of its area. The change in dominating capacitance as a function of area is shown in Fig. 15.
2) The model for inductance as a function of parameters such as line-width, number of turns, spacing, etc., is a non-trivial function. However, after place and route, we fix all other parameters except for the side-length of inductor that can vary along horizontal and vertical directions. We observe that the dominating inductance changes linearly with the change in its side-length when other parameters are fixed as shown in Fig. 16.

Based on these observations, we resize components using sequential LP. The basic idea is to compute the sensitivity of the performance metrics in terms of the component dimensions. For each performance metric P_i, where $P_i \epsilon P$ and P is the set of all performance goals, we compute $(\Delta P_i/\Delta W_j), \forall i, j$, where ΔW_j is the change in the width of component j. Similarly, we compute $(\Delta P_i/\Delta H_j), \forall i, j$, where ΔH_j is the change in height of component j. The sensitivity values are computed while considering the effect of current wirelength and intra-component parasitics. During each stage of the sequential LP, the objective function is formulated such that the difference in required and current performance metric is minimized. To reduce the effect of interconnects, we include wirelength as an objective in our sequential LP. In addition, area is added as another objective in the LP. The overall objective function is as follows:

$$\lambda_1 \cdot \Sigma DP_i + \lambda_2 \cdot WL + \lambda_3(prev_w \cdot \delta_H + prev_h \cdot \delta_W) \quad (7)$$

where $\lambda_1 * \Sigma DP_i$ is the sum of the difference between the current performance metric and the required performance metric, WL is the overall wirelength in the layout, δ_H and δ_H are the change in width and height of the layout, and $prev_w$ and $prev_h$ are the width and height of the layout in the previous LP stage. We multiply the change in height with the previous width and vice versa because if the previous width $prev_w$ is greater than the previous height $prev_h$, then more area reduction is achieved per unit change in height compared with per unit change in width.

Our sequential LP consists of a series of LP steps. Rather than using the circuit-based optimizer described in Section V-B, we correlate performance, area, and wirelength metric of each component to the change in component size based on (7). The constraints for the LP at each stage are based on (3)–(6). Once the LP for the current iteration is solved, we update the sensitivity values for each component based on (7). The entire process is then repeated. We perform our sequential LP for a given number of user defined iterations.

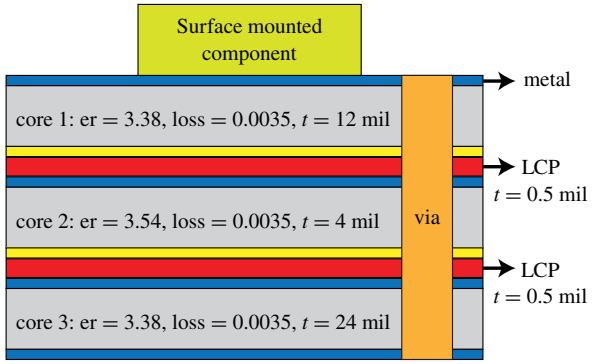

Fig. 18. Substrate stack up targeted for LCP-based embedded passive circuit layout. It contains two layers of metal-covered LCP layers. The passive elements are added into the metal layers shown in yellow. The thickness of each layer is also shown.

TABLE II
AREA (mm^2), AVERAGE RELATIVE ERROR (%), AND RUNTIME (min) COMPARISON BETWEEN ITERATIVE FEEDBACK VERSUS RELATIVE PLACE-AND-ROUTE METHOD

	Iterative Feedback			Relative P&R		
	area	error	cpu	area	error	cpu
10-comp BPF	34.1	7	14	29.9	5	37
11-comp dual-BPF	31.5	9	39	22.1	5	88
15-comp BPF	39.4	7	42	32.2	4	97
21-comp HPF	80.5	14	94	68.3	10	189
Ratio	1.0	1.0	1.0	0.83	0.65	2.2

TABLE III
AREA (mm^2), AVERAGE RELATIVE ERROR (%), AND RUNTIME (min) COMPARISON BETWEEN PRE-LAYOUT AND NO-PRE-LAYOUT OPTIMIZATION

	no optimization			w/optimization		
	area	error	cpu	area	error	cpu
10-comp BPF	36.8	7	12	29.9	5	37
11-comp dual-BPF	33.2	8	35	22.1	5	88
15-comp BPF	36.3	7	39	32.2	4	97
21-comp HPF	79.8	19	82	68.3	10	189
Ratio	1.0	1.0	1.0	0.82	0.58	2.47

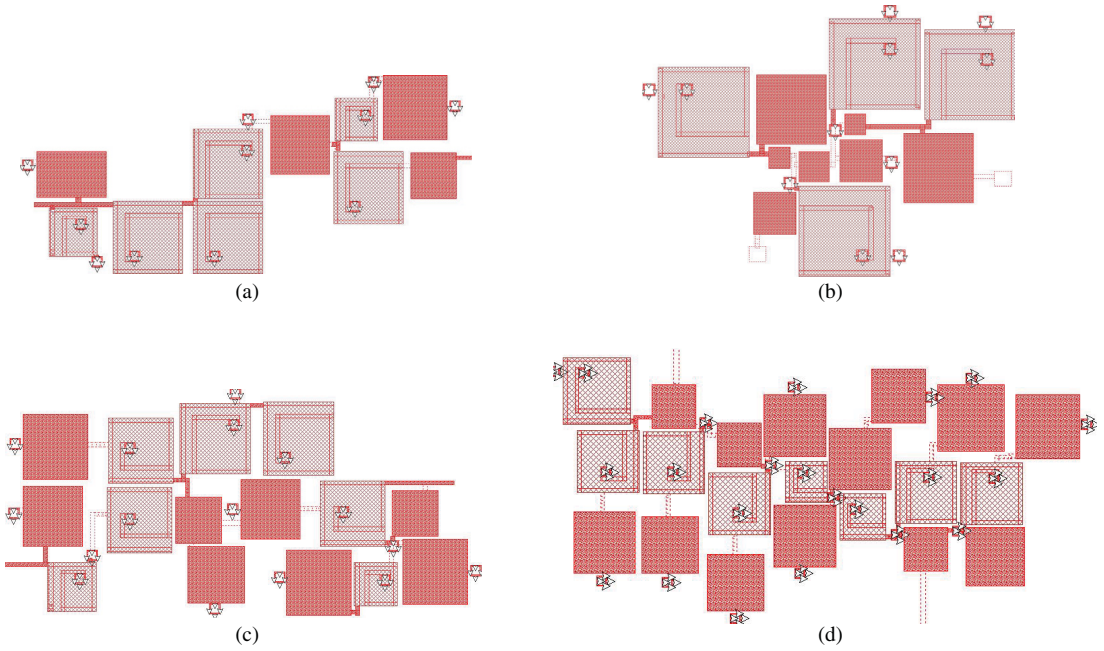

Fig. 19. Filter layouts obtained with our ILP-based method. (a) Ten-component BPF (8.9 mm ×3.5 mm, 37 min), (b) 11-component dual-BPF (6.0 mm × 4.5 mm, 88 min), (c) 15-component BPF (8.9 mm × 4.1 mm, 97 min), and (d) 21-component HPF (10.3 mm × 7.1 mm, 189 min).

C. Iterative Feedback Approach

We describe our Iterative Feedback approach in this section. This method serves as a baseline to compare our relative place-and-route method against as shown in Table II. Our Iterative Feedback approach is similar to our pre-layout optimization presented in Section V, where the components in the design are resized to improve the response of the final layout. The main difference is that during iterative feedback, we are given an initial layout of the whole design, where the length of all interconnects are fixed. This allows more accurate component resizing and layout optimization compared with pre-layout optimization.

We first extract the circuit model of the entire layout. Our circuit model extraction is done by mapping various layout features such as component and interconnect geometries to the circuit model library. The layout provides the information about the size of different components and lengths of various interconnects that are used to connect these components. The layout information is then used to build a circuit representation of the layout based on the circuit model library. The circuit model library includes the models for individual components and various interconnect wires and vias. The circuit representation of the entire layout is then optimized for performance specifications. During this optimization, the device sizes are allowed to vary. The circuit layout obtained after this resizing may have overlaps, and the routing may no longer be valid. The circuit is then replaced and re-routed while keeping the overall area and wirelength change to a minimum. The circuit model of the new layout is then extracted again and the entire process is repeated in an iterative fashion. Note that the post-layout optimization is performed using only circuit models, and thus leads to fast runtime. In order to reduce the complexity of the optimization process, we divide the whole process into two phases.

1) *Local Resizing:* In this step, the entire circuit is partitioned into a few parts first. The partitioning is done based on the user input. Typically a given netlist is partitioned into two or three smaller parts. The objective of the partition is to reduce the number of interconnects between different partitions so that the local resizing can be done for each partition efficiently. The sub-circuit in partition is optimized while considering the parasitics involved within the sub-circuit. The objective is to obtain the response of the layout so that it matches with the desired response.
2) *Global Resizing:* During this phase, all the components of the circuit are considered for resizing. In this phase, the solution from the local resizing is considered as a starting point. In addition, the interconnect parasitics of the entire layout are also considered.

The resizing is performed using methods similar to that discussed for pre-layout optimization. The pseudo-code of the resizing algorithm is given in Fig. 17. The circuit representation of the given initial layout is partitioned first (line 1). Then, local resizing is performed in lines 2–8. For each sub-circuit, we find the circuit model of the layout (lines 2–5). Each sub-circuit is then resized using a non-linear optimization engine to meet the desired response (line 6). This non-linear optimizer is based on the tool presented in Section V-B. The loop is repeated until we meet the desired response or maximum number of iterations $\max_{iter}$ is exceeded (line 8). During the global resizing (lines 9–15), the solution from local resizing is used as the starting point. The remaining steps of global resizing (lines 10–15) are similar to that of local resizing (lines 3–8). The goal is to find a new layout from the given

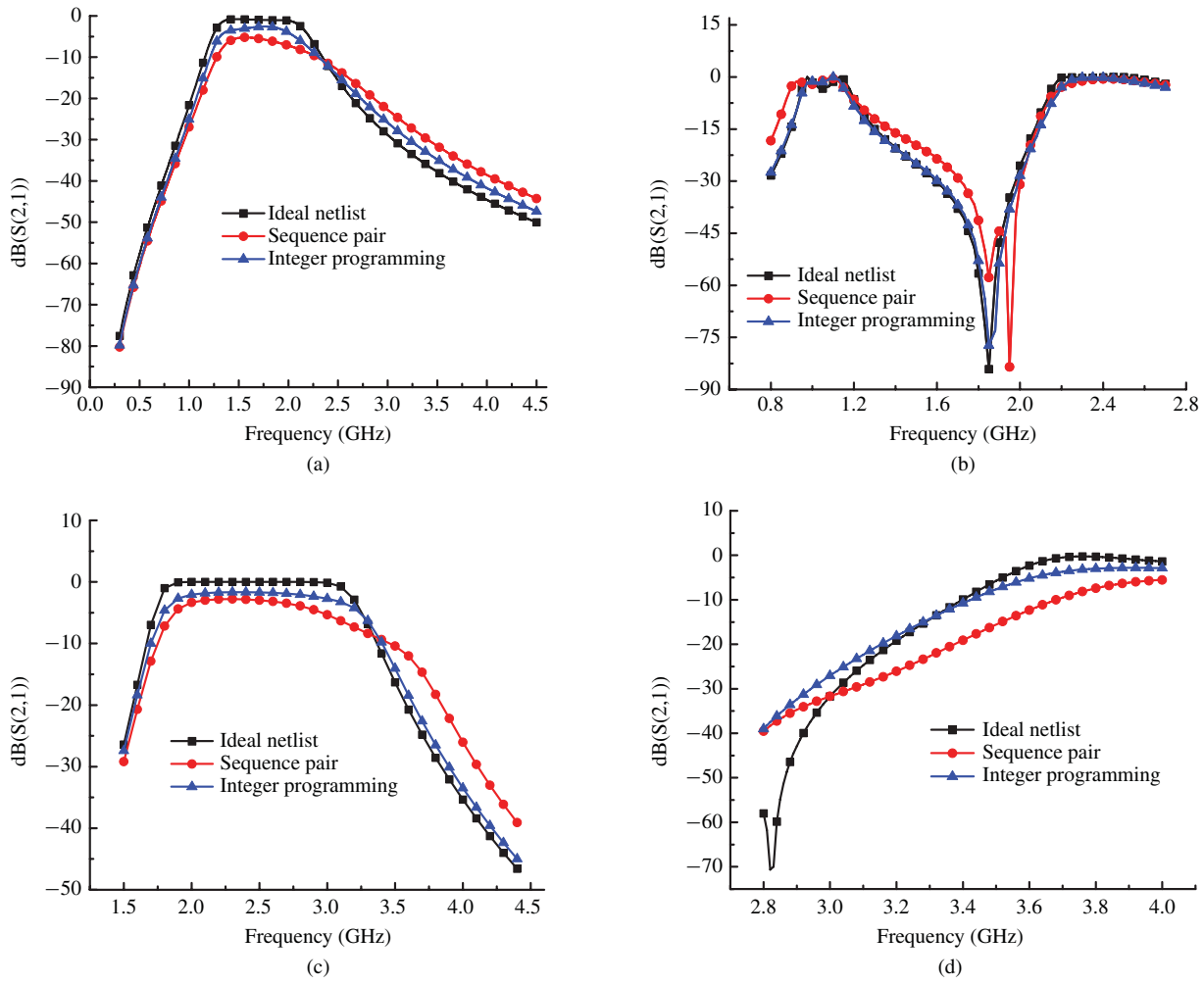

Fig. 20. Layout response. (a) Ten-component BPF, (b) 11-component dual-BPF, (c) 15-component BPF, and (d) 21-component HPF.

original layout which matches closely to the desired response of the circuit.

VIII. Experimental Results

We implemented our algorithms in C++/STL and ran our experiments on a Linux PC running at 2.5 GHz. We ran our tool on four filter circuits with varying number of lumped components. Fig. 2 shows the circuit schematic of our ten-component band pass filters. The response of the final layout for each circuit is obtained with SONNET [19]. For each circuit, a set of frequency points that were spread across the range of the required response was provided as an input. These frequency points were the points where the circuit was optimized for performance specifications. The stack-up of our packaging substrate is shown in Fig. 18.

A. Filter Layout and Response

Fig. 19 shows the layouts we obtained with our ILP-based method: pre-layout optimization plus ILP-based incremental placement and routing plus relative place-and-route. We observe that the components are highly packed and the wires connecting them are minimized. Some layouts include whitespace mainly in order to reduce the overall wirelength. The total runtime used to obtain these layouts range from 37 min to 189 min, which is only a fraction of the manual design time. This demonstrates the point that our automatic method reduces the design time dramatically while generating layouts with small footprint and short interconnection length.

Fig. 20 shows the SONNET-based circuit response results. The ideal response assumes there are no parasitics associated with any component and interconnects. This serves as theoretical optimal response that can never be achieved in practice. We also show our sequence pair plus maze routing-based method for comparison. We observe that in all these cases we achieve a fairly accurate result with our ILP-based method in terms of the final response. In all cases our ILP-based method outperforms sequence pair plus maze routing method.

B. Impact of Various Optimization Methods

Table I shows more detailed comparison between ILP-based versus sequence pair plus maze routing method in terms of their layout qualities: we show the wirelength, area, and runtime metrics of the layouts. ILP-based method results in much lower interconnect length (58%) as compared with the sequence pair-based approach. The overall runtime of

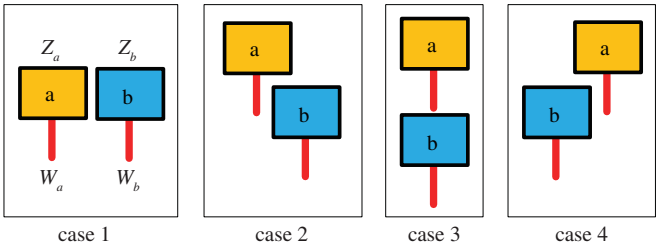

Fig. 21. Four different cases on how two super blocks $(Z_a + W_a)$ and $(Z_b + W_b)$ are placed. W_a is the wire connecting to block Z_a. Since these two rectangles always stay together, they can be combined to form a super block.

ILP-based method is 25% slower on average, showing that our ILP-based method is scalable and able to handle large-scale designs efficiently. ILP-based method shows an average 9% increase in layout area.

We also compare Iterative Feedback and relative place-and-route based post layout optimization schemes. For this comparison, we used sequence pair based placement and maze routing solution as the starting point. The results are shown in Table II. The average percentage error was computed using the magnitude of S-parameter response between required and achieved results. We observe that relative place-and-route gives better area and response but needs longer runtime as compared with Iterative Feedback approach.

We show the impact of our pre-layout optimization in Table III. We observe that our pre-layout optimization helps to improve both area (18% on average) and performance (42% on average) at the cost of runtime increase. This happens as we have a better estimate of required device sizes during layout generation, leading to lower area and better responses for placed modules.

IX. Conclusion

In this paper, we show an effective methodology to automatically generate layouts for RF embedded passive designs. We make use of accurate circuit models to represent our layout and perform optimization, thus keeping time-consuming EM simulations out of the design loop. We explore various techniques for layout generation and layout optimization. At various stages of layout generation, we perform layout optimization to achieve various performance metrics while reducing the overall area and interconnect length.

Appendix

A. Rectangle Overlapping

We briefly discuss the set of equations required to ensure that two rectangles A and B do not overlap. Here are the overall ILP constraint equations

$$A_L - B_R + I_1 \cdot M + I_2 \cdot M >= 0 \quad (8)$$

$$B_L - A_R + (1 - I_1) \cdot M + I_2 \cdot M >= 0 \quad (9)$$

$$A_B - B_T + I_1 \cdot M + (1 - I_2) \cdot M >= 0 \quad (10)$$

$$B_B - A_T + (1 - I_1) * M + (1 - I_2) \cdot M >= 0 \quad (11)$$

$$A_R - A_L = A_{width} \quad (12)$$

$$A_T - A_B = A_{height} \quad (13)$$

$$B_R - B_L = B_{width} \quad (14)$$

$$B_T - B_B = B_{height}. \quad (15)$$

In the above formulation, (8)–(11) ensure that the two rectangles A and B do not overlap, where I_i are integer variables and M is a large constant. We see that for $I_1 = 0$ and $I_2 = 0$ (8) ensures that block A is to the right of block B, and for the other values of I_1 and I_2 (8) is always satisfied due to the large value of M. Similarly, (9)–(11) ensure that for valid values of integer variables, the two blocks A and B do not overlap. Equations (12)–(15) ensure the width and height constraints of the blocks. We observe that in order to perform placement for n blocks, we need $n \cdot (n-1)$ integer variables. This number can be large for large value of n.

B. Super Block Based Integer Variable Reduction

In this section, we show that our knowledge about the layout topology can help to create super blocks which can be used effectively to reduce the number of integer variables used in our ILP-based method. Consider for example two blocks Z_a and Z_b, where each of these blocks have wires W_a and W_b connecting to them, respectively (see Fig. 21). Thus, the total number of rectangles is 4, requiring 12 integer variables as shown in (8)–(15) in Appendix A. However, we observe that W_a is always connected to Z_a, and similarly W_b is connected to Z_b as shown in Fig. 21. Each of these can be considered as a super block. There are eight possible ways in which these super blocks can be placed as shown in Fig. 21. Thus, in order to place these super blocks we need only three integer variables because k integer variables can cover 2^k different disjoint cases.

To show how the integer constraint equations are written, we write the equations for cases (1–4) shown in Fig. 21 as follows:

$$Z\_b_L - Z\_a_R + (I_1 + I_2 + I_3) \cdot M >= 0 \quad (16)$$

$$Z\_b_L - W\_a_R + (I_1 + I_2 + 1 - I_3) \cdot M >= 0 \quad (17)$$

$$Z\_a_B - Z\_b_T + (I_1 + I_2 + 1 - I_3) \cdot M >= 0 \quad (18)$$

$$W\_a_B - Z\_b_T + (I_1 + 1 - I_2 + I_3) \cdot M >= 0 \quad (19)$$

$$W\_a_L - Z\_b_R + (I_1 + 1 - I_2 + 1 - I_3) \cdot M >= 0 \quad (20)$$

$$Z\_a_B - Z\_b_T + (I_1 + I_2 + 1 - I_3) \cdot M >= 0 \quad (21)$$

where, I_i are the integer variables, and M is a large constant value. Here is how the four cases are handled.

1) Equation (16) handles case 1, where we ensure Z_b is to the right of Z_a.
2) Equations (17) and (18) handle case 2, where Z_b is to the right of Z_{acon} and Z_b is below Z_a.
3) Equation (19) handles case 3, where Z_b is below W_a.
4) Equations (20) and (21) handle case 4, where Z_b is to the left of W_a and Z_b is below Z_a.

Similar equations can be written for the other cases. It is thus seen that by using a super block representation, much fewer integer variables are needed. Similar tricks can be applied to other cases in our integer formulation, for example, where the two main trunks of a net always intersect and can be considered as a super block.

REFERENCES

[1] L. Golanka, K. Wolter, A. Dziedzic, J. Kita, and L. RebenKlau, "Embedded passive components for MCM," in *Proc. 24th Int. Spring Seminar Electron. Technol.*, Calimanesti-Caciulata, Romania, May 2001, pp. 73–77.

[2] S. Mukherjee, B. Mutnury, S. Dalmia, and M. Swaminathan, "Layout-level synthesis of RF inductors and filters in LCP substrates for Wi-Fi applications," *IEEE Trans. Microw. Theory Tech.*, vol. 53, no. 6, pp. 2196–2210, Jun. 2005.

[3] H. Koh, C. Sequein, and P. Gray, "OPASYN: A compiler for CMOS operational amplifiers," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 9, no. 2, pp. 113–125, Feb. 1990.

[4] S. Bhattacharya, N. Jangkrajarng, and C.-J. R. Shi, "Template-driven parasitic-aware optimization of analog integrated circuit layouts," in *Proc. 42nd Design Autom. Conf.*, Jun. 2005, pp. 644–647.

[5] N. Jangkrajarng, L. Zhang, S. Bhattacharya, N. Kohagen, C.-J. R. Shi, "Template-based parasitic-aware optimization and retargeting of analog and RF integrated circuit layouts," in *Proc. IEEE Int. Conf. Comput.-Aided Design*, San Jose, CA, Nov. 2006, pp. 342–348.

[6] J. Rijmenants, J. Litsios, T. Schwarz, and M. Degrauwe, "ILAC: An automated layout tool for analog CMOS circuits," *IEEE J. Solid State Circuits*, vol. 24, no. 2, pp. 417–425, Apr. 1989.

[7] J. Cohn, D. J. Garrod, R. Rutenbar, and L. Carley, "KOAN/ANAGRAM II: New tools for device-level analog placement and routing," *IEEE J. Solid-State Circuits*, vol. 26, no. 3, pp. 330–342, Mar. 1991.

[8] M. Aktuna, R. A. Rutenbar, and L. R. Carley, "Device-level early floorplanning algorithms for RF circuits," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 18, no. 4, pp. 375–388, Apr. 1999.

[9] E. Malavasi and A. S. Vincentelli, "Area routing for analog layout," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 12, no. 8, pp. 1186–1197, Aug. 1993.

[10] E. Charbon, E. Malavasi, U. Choudhary, A. Casotto, and A. S. Vincentelli, "A constraint-driven placement methodology for analog integrated circuits," in *Proc. IEEE Custom Integr. Circuits Conf.*, May 1992, pp. 28.2.1–28.2.4.

[11] U. Choudhary and A. S. Vincentelli, "Automatic generation of parasitic constraints for performance-constrained physical design of analog circuits," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 12, no. 2, pp. 208–224, Feb. 1993.

[12] A. Agarwal and R. Vemuri, "Layout-aware RF circuit synthesis driven by worst case parasitic corners," in *Proc. IEEE Int. Conf. Comput. Design*, Oct. 2005, pp. 444–449.

[13] G. Sommer, W. John, and H. Reichl, "Layout synthesis algorithm of embedded passive components for RF and EMC reliable system design," in *Proc. 8th IEEE Workshop Signal Propagat. Interconn.*, May 2004, pp. 201–204.

[14] H. Murata, K. Fujiyoshi, S. Nakatake, and Y. Kajitani, "Rectangle-packing-based module placement," in *Proc. IEEE Int. Conf. Comput.-Aided Design*, San Jose, CA, Nov. 1995, pp. 472–479.

[15] E. F. Young, C. C. Chu, and M. Ho, "Placement constraints in floorplan design," *IEEE Trans. Very Large Sacle Integrat.*, vol. 12, no. 7, pp. 735–745, Jul. 2004.

[16] S. KirkPatrick, C. Gelatt, and M. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, no. 4598, pp. 671–680, May 1983.

[17] J. Ousterhout, "Corner stitching: A data-structuring technique for VLSI layout tools," *IEEE. Trans. Comput. Aided Design Inetegr. Circuits Syst.*, vol. 3, no. 1, pp. 87–100, Jan. 1984.

[18] S. L. Lin and J. Allen, "Minplex - A compactor that minimizes the bounding rectangle and individual rectangles in a layout," in *Proc. 23rd ACM Design Autom. Conf.*, Cambridge, MA, Jun. 1986, pp. 123–130.

[19] *Sonnet High Frequency Electromagnetic Software* [Online]. Available: http://www.sonnetsoftware.com/

Mohit Pathak (S'05) received the B.Tech. degree in computer science and engineering from the Indian Institute of Technology, Kharagpur, India, in 2004. He is currently pursuing the Ph.D. degree with the Georgia Institute of Technology, Atlanta.

He was with Magma Design Automation, Uttar Pradesh, India, for a year, where he worked as a Technical Staff Member. He is a Graduate Research Assistant with the School of Electrical and Computer Engineering, Georgia Institute of Technology. He is currently with Cadence Design Systems, San Jose, CA. His current research interests include physical design automation for stacked 3-D-integrated circuits, timing optimization, placement and routing algorithms, design for manufacturing techniques, and very large scale integration designs.

Sung Kyu Lim (S'94–M'00–SM'05) received the B.S., M.S., and Ph.D. degrees from the Computer Science Department, University of California, Los Angeles, in 1994, 1997, and 2000, respectively.

He joined the School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, in 2001, where he is currently an Associate Professor. He is the author of *Practical Problems in VLSI Physical Design Automation* (New York: Springer, 2008). His current research interests include architectures, circuits, and physical designs for 3-D integrated circuits and 3-D system-in-packages.

Dr. Lim received the National Science Foundation Faculty Early Career Development Award in 2006. He was on the Advisory Board of the Association for Computing Machinery (ACM) Special Interest Group on Design Automation (SIGDA) from 2003 to 2008 and received the ACM SIGDA Distinguished Service Award in 2008. He was an Associate Editor of the IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION SYSTEMS from 2007 to 2009 and served as a Guest Editor for the *ACM Transactions on Design Automation of Electronic Systems*. He has served the Technical Program Committee of several conferences on electronic design automation including the ACM Design Automation Conference (DAC) and the IEEE International Conference on Computer-Aided Design (ICCAD). His work was nominated for the Best Paper Award at ISPD'06, ICCAD'09, CICC'10, and DAC'11. He is a member of the Design International Technology Working Group of the International Technology Roadmap for Semiconductors. He has been leading the Cross-Center Theme on 3-D Integration for the Focus Center Research Program and the Semiconductor Research Corporation since 2010.