Analog Layout Synthesis - Recent Advances in Topological Approaches

H. Graeb[§], F. Balasa[*], R. Castro-Lopez[†], Y.-W. Chang[‡], F.V. Fernandez[†], P.-H. Lin[‡], M. Strasser[§]
[*]Dept. of Computer Science and Information Systems, Southern Utah University, USA
[†]Institute of Microelectronics of Sevilla, CSIC and University of Sevilla
[‡]Graduate Institute of Electronics Engineering, National Taiwan University, Taiwan
[§]Institute for EDA, Technische Universitaet Muenchen, Germany

***Abstract*—This paper gives an overview of some recent advances in topological approaches to analog layout synthesis and in layout-aware analog sizing. The core issue in these approaches is the modeling of layout constraints for an efficient exploration process. This includes fast checking of constraint compliance, reducing the search space, and quickly relating topological encodings to placements. Sequence-pairs, B*-trees, circuit hierarchy and layout templates are described as advantageous means to tackle these tasks.**

I. Introduction

Due to the increasing functional complexity of system-on-chips, the difficulties in analog design and the lack of design automation support for analog circuits continually increase the bottleneck character of analog components in chip design. Eminently critical is the layout synthesis part of the analog design flow. Although there have been a lot of very good works from university over the years, some of which even found their way to commercial EDA tools, industrial application of analog layout synthesis is still in its infancy compared to its digital counterpart. But it seems that this situation is about to change: library alliances for layout transfer between platforms are initiated, EDA start-ups as well as major leaders are announcing new automated layout tools. In this exciting scenario, academia continues to strive for new approaches to analog layout and has recently produced new solutions. This paper gives an overview of these solutions.

A major issue in analog layout synthesis is the consideration of constraints. All four approaches treated in the following provide different views and contributions to the handling of layout constraints in topological approaches to analog synthesis.

Section II introduces into the topic of device-level topological placement with layout constraints [13] and statistical solution approaches. The advantages of topological placement compared to methods based on absolute coordinates are given, and the usage of sequence-pairs to model and verify symmetry constraints within the iterative placement process is explained.

Section III presents hierarchical placement with layout constraints [17]. Hierarchy is applied as a means to further reduce the search space and leads to complex symmetry constraints across different levels of hierarchy. Hierarchical B*-trees are introduced to capture these hierarchical symmetry constraints.

Section IV sketches enhanced shape functions for deterministic placement [25]. Enhanced shape functions provide white-box shape models of building blocks, which allow to interleave their shapes while preserving all symmetry constraints. This allows for a deterministic bottom-up hierarchical placement.

Section V introduces an approach to layout-aware synthesis [4]. Layout templates are used to consider the impacts of the physical layout during the sizing process in a very efficient way. This avoids the costly re-iterations over sizing, layout extraction, simulation steps.

Section VI summarizes the main conclusions of this joint work.

II. Device level topological placement with symmetry constraints

After the initial phase of constructive techniques, the simulated annealing and genetic algorithms proved to be the most effective choice for solving industrial analog device-level placement problems. These algorithms use stochastically controlled hill-climbing to avoid local minima during the optimization process. In addition, they do not impose severe constraints on the size of the problems or on the mathematical properties of the cost function. While efficiently trading-off between a variety of layout factors as area, total net length, aspect ratio, maximum chip width and/or height, cell orientation, "soft" cell shape, etc., they are very flexible – supporting incremental addition of new functionality, and they are relatively easy to implement (although good tuning needs more time). This is why simulated annealing, the most popular of the stochastic techniques, provided the exploration engine for effective placement tools for analog design: ILAC [24], KOAN/ANAGRAM II [7], PUPPY-A [20], LAYLA [14]. More recently, a two-phase approach using both a genetic algorithm and simulated annealing with dynamic adjustment of the parameters has been reported [28]. These software synthesis systems for analog layout approached the device-level placement problems in the traditional way initiated by Jepsen and Gellat for macrocell placement [11], that is, to explore within a combinatorial optimization framework the search space of both feasible and unfeasible solutions. Assuming the cells represented by means of their *absolute* coordinates, this exploration style allows cell overlaps as the cells move (by translations and changes of orientation) in the chip plane.

Since this strategy may exhibit a slow convergence due to the, typically, huge size of the search space, the alternative approach is to use topological representations in order to explore only *feasible* placement solutions. While placement techniques based on the absolute representation trade-off a larger number of moves in a combinatorial optimization framework for easier- and quicker-to-build layout configurations not always physically realizable, the techniques adopting topological representations trade-off a smaller number of moves for more complex-to-build, feasible layouts. The choices for the latter strategy are either to adopt the *slicing layout model* – where the cells are organized in a set of slices whose direction and nesting are recorded in a *slicing tree* or, equivalently, in a *normalized Polish expression* – or to use *nonslicing* topological representations like, for instance, sequence-pairs [22], ordered trees and B*-trees [5], or transitive closure graphs [15]. Although the ILAC system [24] employs the slicing layout model, today it is widely acknowledged that this is not a good choice for high-performance analog design since the slicing representations limit the set of reachable layout topologies, degrading the layout density especially when cells are very different in size – which is often the case in analog layout [7].

978-3-9810801-5-5/DATE09 © 2009 EDAA

A major difficulty of using topological representations for the exploration of feasible placements is satisfying and maintaining the layout design constraints. For instance, in high-performance analog circuits it is often required that groups of devices are placed symmetrically with respect to one or several axes. Differential circuit techniques are used extensively to improve the accuracy, power supply rejection ratio, and dynamic range of many analog circuits. The full performance potential of many of these circuits cannot be achieved unless special care is taken to match the layout parasitics in the two halves of the differential signal path. Failure to match these parasitics in, for instance, differential analog circuits can lead to higher offset voltages and degraded power-supply rejection ratio [7]. The main reason of symmetric placement (and routing, as well) is to match the layout-induced parasitics in the two halves of a group of devices. Placement symmetry can also be used to reduce the circuit sensitivity to thermal gradients. Some VLSI devices (the bipolar devices, in particular) exhibit a strong sensitivity to ambient temperature. If two such devices are placed randomly relative to the isothermal lines, a temperature-difference mismatch may result. Failure to adequately balance thermal couplings in a differential circuit can even introduce unwanted oscillations. In order to combat potentially-induced mismatches, the thermally-sensitive device couples should be placed symmetrically relative to the thermally-radiating devices. Since the symmetrically placed sensitive components are equidistant from the radiating component(s), they see roughly identical ambient temperatures and no temperature induced mismatch results.

Dealing with symmetry constraints in the framework of topological representations implies addressing three problems: (a) how to recognize encodings complying with the given symmetry constraints, without building the corresponding layout, (b) how to restrict the exploration of the solution space of the representation only to "symmetric-feasible" (S-F) codes, and (c) how to build efficiently the feasible placement.

In the case of the sequence-pair representation [22], a possible solution to these problems was proposed in [13]. Let (α, β) be the sequence-pair of a placement configuration containing a number of symmetry groups (each group composed of pairs of symmetric devices and self-symmetric devices relative to a common vertical axis). Denoting α_A^{-1} the position of a cell A in the sequence α (as α can be viewed as a one-to-one mapping, α^{-1} is well-defined) and defining similarly β_A^{-1}, and also denoting $sym(x)$ the symmetric cell of x, the sequence-pair (α, β) will be called *symmetric-feasible* (S-F) if for any distinct cells x, y in any of the symmetry groups

$$\alpha_x^{-1} < \alpha_y^{-1} \iff \beta_{sym(y)}^{-1} < \beta_{sym(x)}^{-1} \quad (1)$$

The property (1) shows that any symmetric pair of cells appears in the same order in both sequences α and β, whereas two cells belonging to distinct symmetric pairs appear in one sequence in reversed order as do their symmetric cells in the other sequence; it also works neatly also when self-symmetric cells (having $x = sym(x)$) are involved. For instance, the sequence-pair $(EBAFCDG, EBCDFAG)$ satisfies (1) relative to the cells in the symmetry group $\{ (C, D), (B, G), A, F \}$ composed of two symmetric pairs and two self-symmetric cells A and F, and the corresponding placement is shown in Fig. 1. It must be emphasized that a sequence-pair is not automatically symmetric-unfeasible if the property (1) is not satisfied. The property (1) is only a *sufficient* condition: it ensures the building of a valid placement in symmetry point of view.

The exploration of the solution space of placement problems with symmetry constraints can be reduced to the exploration of those sequence-pairs satisfying property (1) relative to every symmetry group of cells. The benefit is a significant reduction in size of the search space. The magnitude of this reduction is given by the following

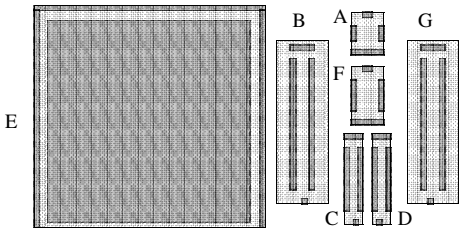


Fig. 1. Placement with symmetry group $\gamma = \{ (C, D), (B, G), A, F \}$.

Lemma The number of symmetric-feasible sequence-pairs corresponding to a placement configuration with n cells and G symmetry groups, each group k containing p_k pairs of symmetric cells and s_k self-symmetric cells ($k = 1, \ldots, G$) , is upper-bounded by

$$\frac{(n!)^2}{(2p_1 + s_1)! \cdot \ \cdots \ \cdot (2p_G + s_G)!}.$$

The number of S-F sequence-pairs for the example in Fig. 1, where $n = 7$ and $p_1 = s_1 = 2$, is $(7!)^2/6! = 35,280$, whereas the total number of sequence-pairs is $(n!)^2$=25,401,600 , therefore, a reduction of the search space of 99.86%.

The algorithm building the placement from an S-F sequence-pair [13] employs an efficient model of priority queue [26] which entails a complexity of $O(G \cdot n \log \log n)$ for each code evaluation, where n is the number of devices and G is the number of symmetry groups. The move-set of the simulated annealing algorithm can be adapted to restrict the exploration of the sequence-pairs to the subset of those having the property (1). To do this it is sufficient to start the exploration with an initial sequence-pair which is symmetric-feasible relative to all the symmetry groups, and to design the move-set such that the property (1) is preserved after each move. For instance, if two cells from distinct symmetric pairs are interchanged in the sequence α, then their symmetric counterparts must be interchanged as well in the sequence β.

III. Hierarchical placement with layout constraints

For modern analog layout synthesis, it is desirable to consider layout design hierarchy to reduce the large search space. The layout design hierarchy may contain both exact and virtual hierarchies of analog circuit design. The exact hierarchy is the same as the circuit hierarchy, while the virtual hierarchy consists of hierarchical clusters [17]. Each cluster contains some devices and sub-circuits which are gathered based on device models, sub-circuit functionality [9], [21], and/or other specific constraints [6]. Fig. 2 shows an example layout design hierarchy, where each sub-circuit corresponds to a specific constraint.

A. Hierarchical Layout Constraints

According to [7] and [10], the basic analog layout constraints include *common-centroid*, *symmetry*, and *proximity* constraints, illustrated in Fig. 3. The common-centroid constraint is usually applied to a sub-circuit of a current mirror or a differential pair to reduce process-induced mismatches among the devices. The symmetry constraint is always required in the layout design of the whole differential sub-circuit. It helps reduce the the parasitic mismatches between two identical signal flows in the differential sub-circuit. The proximity

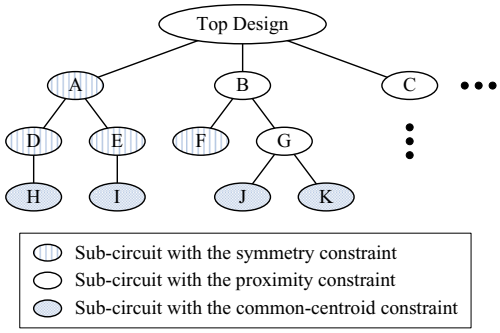


Fig. 2. Layout design hierarchy and the corresponding constraint in each sub-circuit.

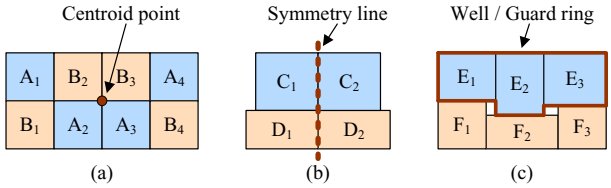


Fig. 3. Basic analog layout constraints. (a) Common-centroid constraint. (b) Symmetry constraint. (c) Proximity constraint.

constraint is widely used in the sub-circuit of a common device model or a certain circuit functionality. It helps form a connected placement of a sub-circuit so that the sub-circuit can share a connected substrate/well region or be surrounded by a common guard ring to reduce the layout area, the interconnecting wire length, and the substrate coupling effect. In particular, it is not necessary for the placement outline of each sub-circuit with the proximity constraint to be rectangular for better area utilization. Fig. 3(c) shows an example placement of two sub-circuits, $\{E_1, E_2, E_3\}$ and $\{F_1, F_2, F_3\}$, with the proximity constraint.

Besides the basic layout constraints, there exist *hierarchical symmetry* and *hierarchical proximity* constraints as shown in Fig. 2. A sub-circuit with the hierarchical symmetry constraint may contain some devices together with other sub-circuits with the common-centroid and (hierarchical) symmetry constraints. Fig. 4 shows an example hierarchical symmetric placement of several hierarchical

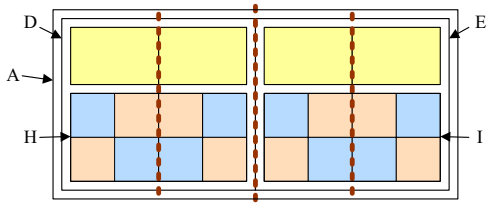


Fig. 4. An example placement of the sub-circuits A, D, and E with the hierarchical symmetry constraint, and the sub-circuits H and I with the common-centroid constraint in Fig. 2.

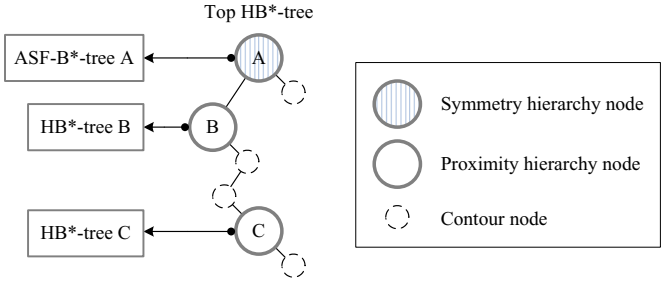


Fig. 5. Example HB*-trees modelling the hierarchical floorplan of the design in Fig. 2.

sub-circuits in Fig. 2. Similarly, a sub-circuit with the hierarchical proximity constraint may contain some devices together with other sub-circuits with the common-centroid, (hierarchical) symmetry, and (hierarchical) proximity constraints.

B. Hierarchical Analog Placement

Based on the concept of the layout design hierarchy illustrated in Fig. 2, it is essential to synthesize analog layout hierarchically for better efficiency and effectiveness. Modern analog placement techniques often simultaneously optimize the placement in different hierarchical sub-circuits, *e.g.*, [18], [16], [17], [19], [25], instead of bottom-up integration, because the optimal placement of a sub-circuit may not lead to the globally optimal placement. Most of them apply simulated annealing [12] based on the topological floorplan representations, such as sequence-pair [22] and B*-tree [5], while the latest one [25] adopts a full deterministic approach which will be introduced in Section IV. Among these works, the one based on the hierarchical B*-tree (HB*-tree) in [17] discusses how to handle the hierarchical symmetry and hierarchical proximity constraints.

Derived from the B*-tree, the HB*-tree introduces a hierarchy node with a variable number of contour nodes at its right child to model each hierarchical sub-circuit and its top rectilinear outlines. Each hierarchy node further links to an HB*-tree that models the floorplan of the device modules in the hierarchical sub-circuit modelled by the hierarchy node. Figure 5 shows example HB*-trees modelling the hierarchical floorplan of the design in Fig. 2. Consequently, the number of the HB*-trees will be equal to that of the sub-circuits plus the one modelling the top design. When perturbing the HB*-trees, one of the HB*-trees should be selected first, and then any perturbation operation for the B*-tree can be applied to the selected HB*-tree. When converting the HB*-trees to a hierarchical placement, the packing procedure is also similar to that for the B*-tree which adopts a pre-order tree traversal. Once a hierarchy node is traversed, the nodes in the HB*-tree linked by the hierarchy node will be traversed before traversing the next node in the HB*-tree which the hierarchy node belongs to. During the HB*-tree packing, the properties of the proximity constraint should also be considered [17].

The hierarchical framework based on the HB*-tree can easily integrate other placement approaches for different sub-circuits with different placement requirements. For example, the automatically symmetric-feasible B*-trees (ASF-B*-trees) which models a symmetric placement of a sub-circuit with the symmetry constraint as a *symmetry island* [16] can be linked by a symmetry hierarchy node in an HB*-tree, as shown in Fig. 5. Based on the symmetry-island formulation, the ASF-B*-trees and the HB*-trees can be further

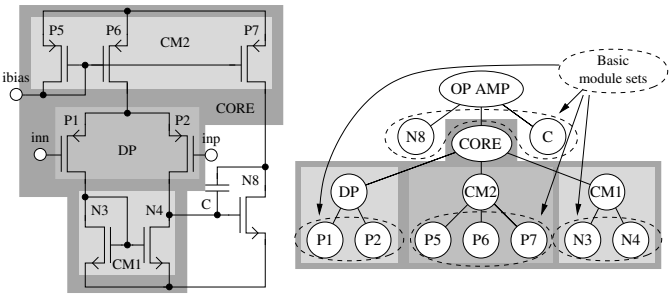


Fig. 6. Miller op amp schematic and hierarchy tree.

extended to handle the hierarchical symmetric placement of a sub-circuit with the hierarchical symmetry constraint. Besides integrating the ASF-B*-tree, the HB*-tree can also integrate both the Corner Block List (CBL) and the grid-based approach in [19] for a common-centroid placement, the signal-flow driven approach [18] for the placement of a specific sub-circuit with clear signal flows, and other placement approaches.

IV. Enhanced shape functions for deterministic analog placement

Placement algorithms for analog circuits are facing severe complexity problems, because the number of possible placements is huge. For example, when B*-trees [5] are used to encode the placement, the number of possible placements for 8 modules is already 57,657,600 [3]. Thus, a complete enumeration of all possible placements is impracticable.

Since analog circuits show a hierarchical structure, the hierarchy can be used as a bound for the enumeration. The hierarchical structure can be automatically detected [9], [21] and formulated as a hierarchy tree. Figure 6 shows a folded cascode op amp and its hierarchy tree.

In the approach [25], leaf nodes of the hierarchy tree represent the modules of the circuit. The modules which share the same parent node in the hierarchy tree form so-called basic module sets. A basic module set only contains a small number of modules, e.g., the transistors of a differential pair or a current mirror. Thus, a full enumeration of all possible placements for a basic module set is practicable.

The deterministic approach consists of two steps: Firstly, all placements of the basic module sets are generated by enumeration. The results of the enumerations can be considered as partial solutions of the placement problem. Secondly, the results of the enumerations are combined, guided by the hierarchy tree. The hierarchy determines the enumeration as well as the combination. Since the hierarchy is determined by the electrical function, the search space is limited to the electrically relevant part.

A. Enhanced Shape Functions

For the enumeration of placements of basic module sets, it is necessary to store the placements efficiently. This is done by enhanced shape functions, presented in [25]. In short, an enhanced shape function is defined similarly to the well-known shape functions [23]. A shape function consists of an ordered set of shapes, being a tuple of width and height of the bounding rectangle of a placement. Placements which have a greater height, while having the same or even a greater width than some other shape in the function are considered to be redundant and therefore removed. This reduces the computational as well as memory effort in subsequent steps. In addition to the tuple of width and height, enhanced shape functions store the B*-tree corresponding to the placement. This allows for

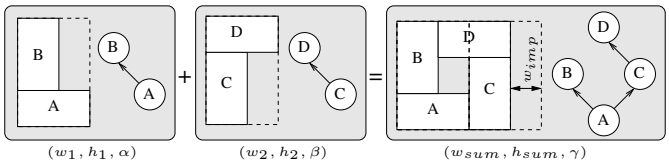


Fig. 7. Example of an enhanced shape addition.

efficient combinations of placements by adding the B*-trees when stepping up the hierarchy tree. Figure 7 illustrates a horizontal enhanced shape addition. The dashed rectangle in the result denotes the size of the resulting shape, if only the bounding rectangles are used for addition. Since the B*-trees α and β of the placements are used to add the two shapes, the width of the resulting placement is w_{imp} smaller than the one which could be achieved by a regular shape addition. More details on enhanced shape additions can be found in [25]. In general, more compact solutions can be found when using enhanced shape functions. But since the addition of B*-trees consumes more time than adding bounding boxes, enhanced shape functions cause bigger computational effort.

B. Results

To demonstrate the benefits, different experiments have been conducted, using enhanced shape functions (ESF) and using regular shape functions (RSF). The circuits are similar to the ones used in [25]. The results are summarized in Table I. The area of the placements using enhanced shape functions is on average 4.4% smaller than using shape functions, at cost of ten times the runtime on average. But the run times using enhanced shape functions are still practical, because they range from seconds to a few minutes. It can be seen from the results, that the beneficial effect on the area usage of enhanced shape functions grows with the number of modules. In Figure 8, the enhanced shape function and the shape function of the circuit named "lnamixbias" [2] are both plotted into the same diagram for direct comparison.

V. Layout-aware synthesis

It is well known that a fundamental issue in micro and nano-electronic integrated circuits is to ensure the effective use of area. Besides carefully choosing device dimensions, best area usage can be assured by geometrically optimizing the design: by maximizing the layout regularity, optimizing the component aspect ratios, and establishing a proper floor-plan. Another critical issue is the performance degradation due to layout-induced parasitic devices. Compensating for these degradations typically requires many time-consuming and

Experiment	# of	ESF		RSF		Area im-
Criterion	mods	Area usage	Time	Area usage	Time	provement
Miller V2	13	111.74%	14	112.40%	1	0.66%
Comparator V2	10	112.50%	1	113.39%	1	0.89%
Folded casc.	22	121.03%	44	128.31%	10	7.28%
Buffer	46	111.39%	134	118.12%	7	6.73%
biasynth	65	104.96%	337	111.77%	64	6.81%
lnamixbias	110	107.68%	387	111.97%	25	4.29%

TABLE I
Summary of example circuits, using enhanced shape functions (ESF) and using regular shape functions (RSF). Area usage is the area of the bounding rectangle of the smallest shape, divided by the total module area.

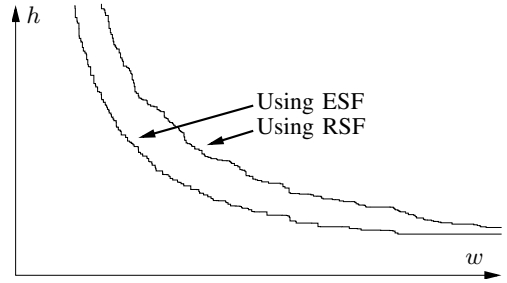


Fig. 8. The ESF and RSF of lnamixbias [2].

unsystematic iterations between electrical sizing and layout generation. Because of the complete separation of these two steps in the design flow, electrical and physical synthesis, the evaluation of the impact of layout-induced parasitics early in the design flow becomes a crucial factor: overestimation leads to power and area waste, underestimation leads to a (perhaps fatal) performance degradation. It is also necessary to stress the importance of the relationship between optimizing the design geometrically and enhancing the design with respect to parasitics. Any solution to either aspect unavoidably affects the other; if one of the two is left unresolved, there is a risk to end up with designs that do not use silicon area effectively or fail to cope with layout-induced degradations [1].

To deal with these issues, the concept of layout-aware electrical sizing has been proposed. Among the reported implementations, the approaches [27] and [8] were the first that simultaneously included some kind of geometric optimization together with certain parasitic considerations. These two solutions, however, perform electrical design by resorting to knowledge-based techniques, which ultimately takes away some accuracy in evaluated performances and, therefore, may yield to inadequate sizing. A more complete layout-aware sizing technique was proposed in [4]. This implementation of the technique relies on numerical simulation and optimization and can be thus applied to any analog circuit that can be efficiently simulated at the device level. The technique developed has two components: the parasitic-aware sizing technique and the geometrically-constrained sizing technique. The essence of the layout-aware concept is the inclusion of physical information right into the electrical design of the circuit. There are two main points here. First, the electrical sizing process is carried out by using a simulation-based optimization approach. This is an iterative process in which the design space of the circuit is explored (and thousands of different circuit sizings are evaluated) to find the sizing that best fits all performance specifications (like dc-gain higher than 50dB) and design objectives (such as minimizing area and power consumption). Second, to enable the use of layout data and details, it is necessary that all layout information can be accessible for each sizing that is being evaluated in each iteration of the electrical sizing process. Furthermore, this access has to be rapid enough so as to prevent the optimization process to become prohibitive in terms of CPU time. For this reason, layout generation is carried out by resorting to the template-based technique. Though templates are known to lack some flexibility to adapt any sizing, they are however a more suitable solution (when compared to optimization-based layout generation) for several reasons : (1) layout generation turnaround times (a few seconds) are considerably smaller than those of optimization-based approaches; (2) layout templates permit searching for optimal block parameters while a priori revealing the knowledge needed for evaluation of layout parasitics, which turns out essential for minimizing the number of

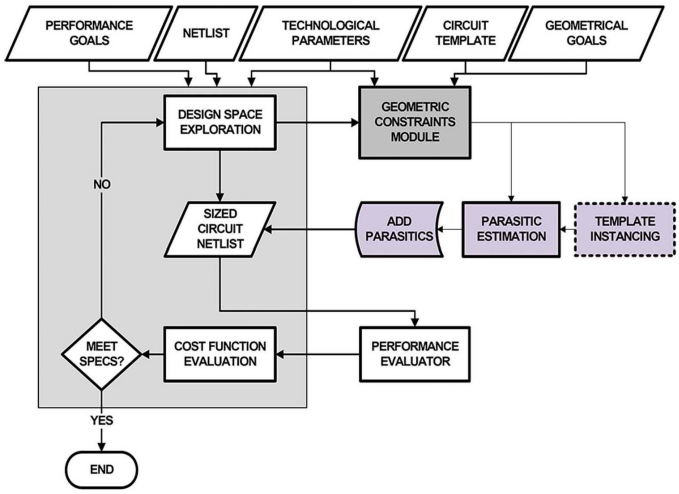
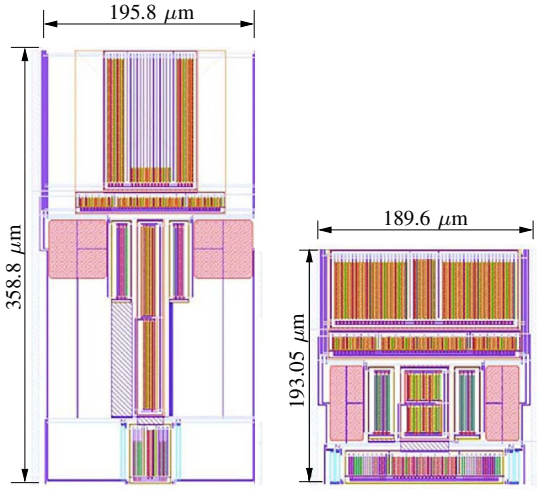


Fig. 9. Flow diagram of the layout-aware sizing technique.

iterations between sizing and layout; (3) layout templates are very efficient at encapsulating design expertise, both for placement and routing ; (4) layout templates can be straightforwardly ported, because template generators can be made independent of the fabrication process. Moreover, the geometrically-constrained sizing technique described below together with a careful design of layout templates contribute to palliate their flexibility problems to a large extent. In the case of the layout-aware technique developed, the template-based approach has been implemented using the Cadence's PCELLS technology and SKILL programming.

Fig. 9 shows the flow diagram of our implementation of the layout-aware sizing technique. All in all, the goal of layout-aware sizing is to avoid any iterations between electrical and physical design. Such a goal is accomplished by finding during the sizing process, the values of geometric parameters, like the number of folds of MOS transistors, that yield optimal geometric features. This optimization can be defined either as a restriction (e.g., a pre-defined layout aspect ratio, or a maximum width or height for the whole circuit layout) or as a design objective (i.e., area minimization). In parasitic-aware sizing, the values of layout parasitics are computed concurrently with sizing, by using specific layout information (e.g., the possible implementation style of a group of MOS transistors) and actual device sizes. The final attained sizing guarantees a robust performance when considering layout-induced parasitic effects. These two techniques have been combined since different geometric variables cause different layout-induced effects (e.g., different foldings change the junction capacitances of a MOS transistor).

Fig. 10 shows two layout instances for a fully-differential folded-cascode amplifier. Instance (a) is the result of an electrical sizing process with no geometrical or parasitic considerations at all; many of the electrical specifications of this experiment are unfulfilled when layout parasitics are considered. Instance (b) is the result of a layout-aware sizing process. Not only is this solution geometrically much better (more compact, less unused area, less area occupation), but all specifications are met even when including all parasitics. There is also an important detail regarding CPU time required by this layout-aware technique: the results demonstrate that at least for analog cells with a few tens of devices, extraction of parasitics is reasonably fast so as to be included within the iterative optimization loop since it takes only 17% of the total sizing time.



(a) (b)

Fig. 10. Layout-aware experiments: (a) Sizing without geometrical or parasitic considerations, (b) layout-aware sizing process.

A few conclusions can be drawn from this experience in layout-aware sizing of analog circuits:

- Geometric aspects and the impact of parasitics can and should be taken into account simultaneously for a successfully and robust design of analog circuits.
- Layout templates are an efficient solution for layout-aware sizing.
- Said flexibility of templates can be improved because area aspects are included in the electrical sizing process of the layout-aware technique.
- Extraction within sizing is not as expensive as it has been traditionally considered. Actually, other approaches that use estimation instead of extraction incur accuracy errors while attaining only a very small CPU time improvement.

VI. Conclusion

In this paper, some recent topological approaches to analog layout synthesis have been surveyed. Especially, modeling approaches for layout constraints concerning, e.g., symmetry or alignment, are subject of recent research works. Specifically, sequence-pairs, B*-trees and hierarchical B*-trees have been used to model and verify these constraints during the placement process, which can be done by statistical or deterministic solution approaches. In addition, a template-based approach to consider layout effects during the sizing process has been given.

References

[1] Medea+ design automation roadmap, 5th edition. http://www.medeaplus.org/, 2005.

[2] F. Balasa and K. Lampaert. Symmetry within the sequence-pair representation in the context of placement for analog design. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 19(7):721–731, July 2000.

[3] F. Balasa, S. C. Maruvada, and K. Krishnamoorthy. On the exploration of the solution space in analog placement with symmetry constraints. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 23(2):177–191, Feb. 2004.

[4] R. Castro-Lopez, O. Guerra, E. Roca, and F. Fernandez. An integrated layout-synthesis approach for analog ICs. In *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, volume 27, pages 1179–1189, 2008.

[5] Y.-C. Chang, Y.-W. Chang, G.-M. Wu, and S.-W. Wu. B*-trees: A new representation for non-slicing floorplans. In *ACM/IEEE Design Automation Conference (DAC)*, volume 37, pages 458–463, 2000.

[6] E. Charbon, E. Malavasi, and A. Sangiovanni-Vincentelli. Generalized constraint generation for analog circuit design. In *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 408–414, 1993.

[7] J. M. Cohn, D. J. Garrod, R. A. Rutenbar, and L. R. Carley. *Analog Device-Level Layout Automation*. Kluwer Academic Publishers, 1994.

[8] M. Dessouky and M.-M. Louerat. A layout approach for electrical and physical design integration of high-performance analog circuits. In *IEEE First International Symposium on Quality Electronic Design*, pages 291–298, 2000.

[9] H. Graeb, S. Zizala, J. Eckmueller, and K. Antreich. The sizing rules method for analog integrated circuit design. In *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 343–349, 2001.

[10] A. Hastings. *The Art of Analog Layout*. Prentice-Hall, 2001.

[11] D. Jepsen and C. G. Jr. Macro placement by monte carlo annealing. In *IEEE International Conference on Computer Design (ICCD)*, pages 495–498, 1983.

[12] S. Kirkpatrick, C. D. Gelatt, Jr., and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220:671–680, 1983.

[13] K. Krishnamoorthy, S. C. Maruvada, and F. Balasa. Topological placement with multiple symmetry groups of devices for analog layout design. In *IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 2032–2035, May 2007.

[14] K. Lampaert, G. Gielen, and W. Sansen. A performance-driven placement tool for analog integrated circuits. *IEEE Journal of Solid-State Circuits SC*, 30(7):773–780, July 1995.

[15] J.-M. Lin and Y.-W. Chang. Tcg: a transitive closure graph-based representation for non-slicing floorplans. In *ACM/IEEE Design Automation Conference (DAC)*, June 2001.

[16] P.-H. Lin and S.-C. Lin. Analog placement based on novel symmetry-island formulation. In *ACM/IEEE Design Automation Conference (DAC)*, pages 465–470, June 2007.

[17] P.-H. Lin and S.-C. Lin. Analog placement based on hierarchical module clustering. In *ACM/IEEE Design Automation Conference (DAC)*, pages 50–55, June 2008.

[18] D. Long, X. Hong, and S. Dong. Signal-path driven partition and placement for analog circuit. In *Asia and South Pacific Design Automation Conference*, pages 694–699, 2006.

[19] Q. Ma, E. F. Y. Yong, and K. P. Pun. Analog placement with common centroid constraints. In *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, Nov. 2007.

[20] E. Malavasi, E. Charbon, E. Felt, and A. Sangiovanni-Vincentelli. Automation of IC layout with analog constraints. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 15(8):923–942, Aug. 1996.

[21] T. Massier, H. Graeb, and U. Schlichtmann. The sizing rules method for cmos and bipolar analog integrated circuit synthesis. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 27(12):2209 – 2222, Dec. 2008.

[22] H. Murata, K. Fujiyoshi, S. Nakatake, and Kajitani. VLSI module placement based on rectangle-packing by the sequence-pair. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 15(12):1518–1524, 1996.

[23] R. Otten. Efficient floorplan optimization. In *IEEE International Conference on Computer Design (ICCD)*, pages 499–501, Oct. 1983.

[24] J. Rijmenants, J. Litsios, T. Schwarz, and M. Degrauwe. Ilac: An automated layout tool for analog cmos circuits. *IEEE Journal of Solid-State Circuits SC*, 24(2):417–425, 1989.

[25] M. Strasser, M. Eick, H. Gräb, U. Schlichtmann, and F. M. Johannes. Deterministic analog circuit placement using hierarchically bounded enumeration and enhanced shape functions. In *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, Nov. 2008.

[26] X. Tang and D. Wong. FAST-SP: a fast algorithm for block placement based on sequence pair. In *Asia and South Pacific Design Automation Conference*, pages 521–526, 2001.

[27] P. Vancorenland, G. V. der Plas, M. Steyaert, G. Gielen, and W. Sansen. A layout-aware synthesis methodology for rf circuits. In *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 358–362, 2001.

[28] L. Zhang, R. Raut, Y. Jiang, and U. Kleine. Placement algorithm in analog-layout designs. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 25(10):1889–1903, Oct. 2006.