PERFORMANCE-DRIVEN COMPACTION FOR ANALOG INTEGRATED CIRCUITS

Eric Felt, Enrico Malavasi†, Edoardo Charbon, Roberto Totaro†,
and Alberto Sangiovanni-Vincentelli

Department of Electrical Engineering
and Computer Sciences
University of California
Berkeley, CA 94720

† Dip. di Elettronica e Informatica
Università di Padova
via Gradenigo, 6/A
35131, Padova, Italy

ABSTRACT

This paper describes a new approach to layout compaction of analog integrated circuits which respects *all* of the performance and technology constraints necessary to guarantee proper analog circuit functionality. Our approach consists of two stages: a *fast* constraint graph critical path algorithm followed by a *general* linear programming algorithm. Circuit performance is guaranteed by mapping high-level performance constraints to low-level bounds on parasitics and then to minimum spacing constraints between adjacent nets.

1. INTRODUCTION

The automatic generation of circuit layout consists of three major phases: placement, routing, and compaction. The compaction phase can be viewed as an optimization problem in which one wishes to minimize layout area subject to a set of constraints necessary to insure proper circuit functionality. These constraints are of three general types: technology constraints, topology constraints, and performance constraints.

In digital systems most approaches to layout compaction consider only technology constraints, i.e. object-to-object minimum and maximum spacing requirements. This approach is sufficient because in digital layouts the dependence of electric performances on the details of physical implementation is limited to logic functions and delay requirements. As long as connectivity is not disrupted and interconnection length is not significantly increased, timing specifications are preserved.

In analog systems, however, all three types of functionality constraints are critical issues during all phases of layout synthesis. Topological constraints and performance constraints are essential because analog circuit performances are extremely sensitive to parasitics. Variations in device symmetries (topological constraints) or interconnect length (performance constraints), for example, affect cross-coupling capacitances and stray resistances, which can dramatically degrade circuit performance.

The symbolic layout compaction problem is NP-hard [1]. However, heuristics have been developed which provide good quality layouts. In particular, the use of mono-dimensional compaction phases alternately performed in orthogonal directions is one of the most popular approaches. The mono-dimensional topological compaction problem is typically formulated as a longest path problem on a directed graph, which is solvable in polynomial time. However, not all analog circuit constraints can be expressed in a form which can be solved by a constraint-graph algorithm.

Several approaches to specific portions of the general analog compaction problem have been previously described. In [2] symmetries are enforced by using an iterative graph perturbation technique. Unfortunately false overconstraints frequently arise with this approach because of the order in which the symmetric objects are processed, especially when multiple symmetry axes are present. In [3] a linear programming (LP) algorithm was proposed. Objects not directly interacting with symmetric items in the layout are collapsed, thus generating "super-constraints" which are solved using linear programming. This technique is particularly appropriate when hierarchical compaction is performed [4]. Even after reducing the set of constraints considered in the LP problem, however, this algorithm is severely limited in the maximum circuit complexity which can be considered and in the classes of constraints which can be accommodated. In [5] symmetries are enforced with a combination graph and LP algorithm which uses constraint graph longest path techniques to arrive at a good starting point for an LP solver.

These prior approaches address the problem of enforcing topological constraints (symmetry requirements, in particular), but no provision is made for performance constraints. In this paper we propose an efficient approach to symbolic compaction which accounts for *all* analog constraints. The size and complexity of the circuits with which it can cope are large enough to make the algorithm suitable for industrial-strength applications. The work presented in [5] served as a basis for the results presented here, with the primary contributions of this paper being: 1) a new aggressive technique for controlling parasitics and 2) an original technique for *global* wire length minimization.

Our approach is part of a general top-down, constraint-driven analog circuit design methodology [6]. Top-level constraints are propagated down the design hierarchy from step to step until the final layout is reached. Since early verification techniques are employed to insure that the performance specifications are satisfied at each stage of the layout, the layout which is fed into the compactor is guaranteed to be feasible in the sense that it meets the performance specifications. This methodology has been successfully used for placement [7] and routing [8, 9]. In this paper we extend the same methodology to compaction, thus ensuring that the results obtained from constraint-driven placement and routing are not spoiled during compaction. Not using constraint-driven compaction would cause the whole methodology to fail by permitting uncontrolled constraint violations.

We believe that compaction is an essential step in any analog layout synthesis framework. The synthesis step immediately prior to compaction is usually routing. While the layout from the router is guaranteed to meet all high-level performance specifications, compaction is still essential to insure that the final layout is: 1) as small as possible and 2) design rule correct. Area minimization is an important target and it is difficult to achieve with only placement and routing, especially when area routing is used. Moreover, constraint-driven compaction makes it possible to relax the requirements on routing and accept slight design rule spacing violations by relying on the possibility of fixing the violations during compaction. Thus compaction is not merely an optional post-processing step; in addition to minimizing the area of the chip, compaction also improves the entire analog synthesis process by permitting more aggressive placement and routing.

The algorithm we propose to accommodate the analog performance constraints during compaction is as follows:

1. Generate a set of minimum spacing constraints between wire segments which satisfy all high-level performance constraints. Add these additional constraints to the constraint graph.
2. Solve the modified graph.
3. Introduce shield and symmetry constraints and use the solution achieved in step 2 as the starting point for an LP solver. The LP solver resolves all constraints and produces a final layout of minimum width (height) and minimum interconnect length.

The algorithm is illustrated in Figure 1. If overconstraints are detected in steps 2 or 3 then feedback is provided to step 1 so that an alternative set of spacing constraints can be generated.

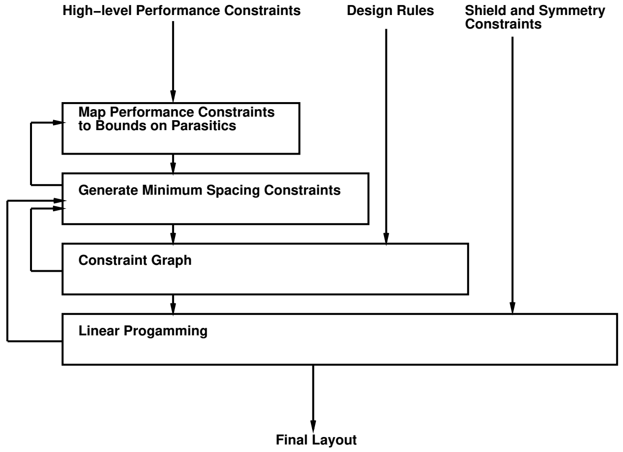

Figure 1: *Compaction algorithm which accommodates all analog constraints.*

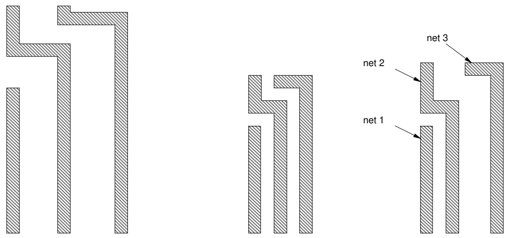

Figure 2: *Coupling constraints. From left to right: original layout, layout compacted without coupling constraints, layout compacted with coupling constraints.*

The use of the graph solution obtained in step 2 as the starting point for the LP solver in step 3 is the key to obtaining a significant speed-up in the solution of the LP, compared to previous approaches using pure LP [3]. In most observed practical cases the configuration yielded by step 2 is close enough to the final solution to require the LP solver to carry out only a small number of steps, yielding 20% to 50% reductions in cpu time over pure LP algorithms. The range of cases that can be managed with acceptable computational complexity is therefore significantly expanded. The details of using the constraint graph and LP solver together to enforce symmetries are reported in [5]. With an LP solver it is also possible to perform global wire length minimization, which is described in section 4.5.

2. ANALOG CONSTRAINTS

Compaction is one of the last steps in the layout synthesis of analog integrated circuits, so the methodology used to synthesize the circuit layout determines the set of constraints which the analog layout compactor must resolve. Under our top-down, constraint-driven methodology, high-level performance specifications are mapped onto low-level constraints which are made available to the compactor. The design methodology insures that the compactor will be provided with *all* of the necessary layout constraints and that no information except interconnects must be inferred directly from the layout. The methodology also insures that the layout and routing tools will provide a reasonable starting point for the compactor, and that if all of the constraints can be satisfied then the circuit will operate within all of its high-level specifications.

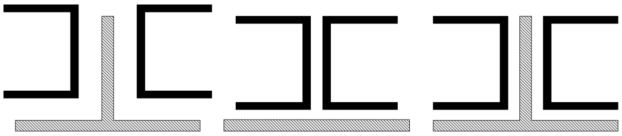

Figure 3: *Shield requirements. From left to right: original layout, layout compacted without shield preservation constraints, layout compacted with shield-preservation constraints.*

Low-level constraints fall into the following categories: 1) maximum cross-coupling, 2) shield preservation, and 3) symmetry enforcement. Each of these constraints will be considered in turn.

- **Maximum Cross-Coupling**

The minimum design rule spacing requirements used when compacting digital layouts are often not sufficient for analog circuits because of capacitive cross-coupling between adjacent wires. When two critical wires are routed closely together, the capacitive coupling between them can seriously degrade the circuit performance. To prevent this detrimental coupling, additional constraints must be added during compaction to insure that the signal coupling does not cause the circuit to violate high-level performance specifications. An example is shown in Figure 2. Without maximum cross-coupling constraints the compacted circuit may not meet the high-level performance requirements because of excessive coupling between nets 2 and 3.

- **Shield Preservation**

When routing an analog circuit it is sometimes impossible to reduce the coupling between two critical lines below the required maximum because of the circuit topology. In these cases the router may introduce a shield between the critically coupled nets. Special constraints must be passed from the router to the compactor to insure that shields are properly preserved. An example is shown in Figure 3.

- **Symmetry Requirements**

Analog circuit designers frequently introduce topological symmetries in differential circuits to optimize offset, differential gain, and noise. Special care must be taken during the compaction step to insure that those symmetries are preserved.

3. ALGORITHM

3.1. Technology Constraints

If only technology constraints are considered, mono-dimensional compaction can be solved efficiently with the constraint graph longest path algorithm [10] [1, Ch. 10]. The pattern of component connectivity and minimum separations required by the technology is described as a weighted, directed graph. Each component is represented in the graph by a node. Maximum and minimum spacing constraints of the form:

$$x_2 - x_1 \geq K \tag{1}$$

are represented as edges of weight K. Note that K is negative for maximum spacing constraints.

A simple plane sweep algorithm is used to generate the minimum spacing constraints. The longest path in this graph provides the minimum width (height).

3.2. Coupling Constraints

High-level performance specifications can be mapped efficiently onto a set of low-level bounds on parasitics [11]. If every parasitic is kept below its bound, all performance constraints are guaranteed to be met. Bounds are set so as to be feasible, on the basis of reasonable estimates of parasitic values drawn from analysis of the layout.

The algorithm to determine the minimum spacings between parallel wire segments is an iterative procedure:

1. Compute the longest path on the current graph.
2. Extract actual parasitics and compare to bounds. If all bounds are met, exit. Otherwise
3. Compute the actual performance degradations, based on extracted parasitics and sensitivities. If performance constraints are met, exit. Otherwise

procedure modify-graph
/* Add constraints to the graph accounting for capacitive decoupling */
/* In what follows C_j indicates the j-th cross-coupling capacitance */
/* and C_j^b is the bound on its maximum value. */
/* Δ_{max} is the maximum distance at which wire coupling is considered. */
/* K is a constant depending on the model used to estimate parallel */
/* wire capacitances. */
/* Since this procedure is called only if some performance violation */
/* has been found, we know that at least one bound has been exceeded. */
for each cross-coupling C_j **such that** $C_j > C_j^b$:
  let δ_j = current min. distance between any two parallel segments
    contributing to C_j
  if $\delta_j \leq \Delta_{max}$ **then**
    $\delta_j = \delta_j + K \cdot (C_j - C_j^b)$
    for each pair P_i of parallel segments:
      let d_i = current minimum distance between the segments of pair P_i;
      if $d_i < \delta_j$ **then**
        Add a min-spacing constraint δ_j between the segments
    endfor
  endif
endfor

Figure 4: *Constraint generation for capacitive decoupling.*

4. Modify the graph by increasing the spacings between critically coupled wire segments. Then go to step 1.

In step 3 performance degradations are computed based on their sensitivities to each parasitic. This step substantially improves the quality of the algorithm's output by permitting the layout to violate some parasitic bounds as long as high-level performance constraints are still satisfied. This situation can occur when one or more parasitics lay below their upper bounds, since the resultant performance improvement can offset the performance degradation due to the parasitics that violate their upper bounds. The algorithm thus guarantees that if a feasible spacing exists then it will be found, even if a non-feasible set of bounds is used by the previous steps in the synthesis process. In addition, area minimization is not artificially constrained by the exact set of parasitic bounds used in the previous steps; if an alternative set of bounds which yields more area improvement exists, it will be found and used.

Pseudo-code for the graph modification procedure implemented at step 4 is shown in Figure 4. For each cross-coupling C_j exceeding its bound C_j^b, δ_j represents the distance between the closest parallel wire segments, namely the segments whose coupling provides the largest contribution to the overall capacitance. The distance between such segments is increased by an amount proportional to the parasitic bound violation. Notice that the distance is increased not only for the closest segments, but also for all the segment pairs whose distances are below δ_j, after this parameter has been increased.

The spacing step implemented by procedure **modify-graph** can introduce overconstraints, making the graph unsolvable. When an overconstraint is detected, a *pruning* procedure is invoked which relaxes the newly-added spacing constraints contained in positive weight loops. In such situations the task of decoupling the two nets is left to the remaining segment pairs. If a feasible solution involving the remaining segment pairs does not exist, then an error is reported because in that event at least one high-level performance constraint cannot be met.

3.3. Shield Constraints

Figure 5 illustrates a lateral and a vertical shield created during routing to reduce coupling between critical nets. The spacing requirement for the lateral shield between two nets labeled 1 and 2 is:

$$x_{shield} \geq x_{net1} \ AND \ x_{shield} \geq x_{net2}. \tag{2}$$

where
x_{shield} = location of top of shield
x_{net1} = location of top of net 1
x_{net2} = location of top of net 2

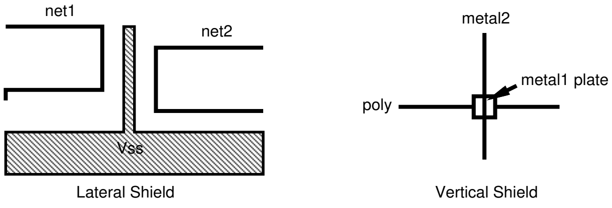

Figure 5: *Horizontal and vertical shields.*

Vertical shield constraints are expressed as:

$$x_{m1} = x_{m2} \tag{3}$$

$$y_{m1} = y_{poly} \tag{4}$$

where
x_{m1}, y_{m1} = center of shielding sheet (usually on metal1).
x_{m2} = location of upper conductor (usually metal2) in horizontal direction.
y_{poly} = location of lower conductor (usually poly) in vertical direction.

Note that shields are inserted between two nets by the router only when absolutely necessary, i.e. when increased spacing between the nets cannot sufficiently reduce the coupling. If the compactor were to remove those shields then the coupling control algorithm outlined in section 4.2 would report an overconstraint and fail.

3.4. Symmetry Constraints

Consider two devices a and b between which a symmetry constraint is required with respect to a vertical axis s (the case with respect to a horizontal axis is perfectly dual). The constraints which must be enforced are:

$$x_a - x_s = x_s - x_b \tag{5}$$

$$y_a = y_b \tag{6}$$

where
x_a = horizontal position of a
x_b = horizontal position of b
y_a = vertical position of a
y_b = vertical position of b
and x_s is the axis position.

While the vertical constraint (6) can be easily resolved in the constraint graph, the horizontal constraint (5) has no graph representation and hence cannot be enforced. It is therefore introduced as a linear programming constraint:

$$x_a + x_b - 2x_s = 0. \tag{7}$$

Symmetry constraints between complete wire segments can be reduced to object symmetry constraints on the wire endings. The horizontal constraints for the two wires labeled 1 and 2 in Figure 6 are formulated as:

$$x_{2,left} - x_s = x_s - x_{1,right} \tag{8}$$

and

$$x_{2,right} - x_s = x_s - x_{1,left} \tag{9}$$

where
$x_{1,right}$ = rightmost ending of wire segment 1
$x_{1,left}$ = leftmost ending of wire segment 1
$x_{2,right}$ = rightmost ending of wire segment 2
$x_{2,left}$ = leftmost ending of wire segment 2.

In this fashion the problem of enforcing one wire symmetry constraint is reduced to satisfying two simple object symmetries. The wire symmetry constraints can thus be directly incorporated into the LP solver and handled in the same way as the device symmetry constraints.

The formulation of the problem and the nature of the starting point allow us to achieve the optimum integer solution using linear programming (LP)

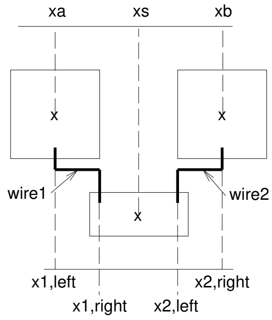

Figure 6: *Wire symmetry constraints.*

rather than integer programming (IP). This simplification is possible because if the nodes of each symmetric pair are also constrained by design-rule minimum spacing requirements, all design-rule minimum spacing requirements are integers, and the leftmost node is located at an integer coordinate. Under these conditions the optimum solution to the LP problem contains only integer coordinates [5].

The LP solver used in this implementation is a simplex algorithm which finds the optimum solution by sequentially visiting the vertices of the polytope bounding the feasible region. Because of the nature of this search algorithm, providing the LP solver with a good initial solution significantly reduces the number of iterations with respect to the case of a random initial basis.

4. WIRE LENGTH MINIMIZATION

Once a minimum area solution which satisfies all of the performance requirements is obtained, a secondary optimization is performed to minimize total interconnect length. This wire length minimization is performed by the LP solver so that the complete set of compaction constraints (including shield and symmetry constraints) can be considered during the optimization. The objective function is formulated as follows:

$$minimize\ L = \sum_{i \in \{all\ wire\ segments\}} (x_{i,right} - x_{i,left}) \qquad (10)$$

where

$x_{i,right}$ = rightmost ending of wire segment i
$x_{i,left}$ = leftmost ending of wire segment i
$x_{i,right} \geq x_{i,left}$

This formulation of the objective function permits the LP solver to find the global minimum horizontal (vertical) wire length. While not required by the analog design methodology, optimizing on this secondary objective improves circuit performance by further reducing wire resistances and parasitic capacitances.

One alternative approach [12] performs heuristic local wire length minimization, but no previous approach addresses the issue of global wire length minimization. This important secondary optimization is only possible in this implementation because of the use of the LP solver as a post-processor of the constraint-graph solution.

5. RESULTS

Three example circuits were compacted with the new tool, SPARCS-A. Two are folded cascode CMOS operational transconductance amplifiers (OTA) and the third is a CMOS comparator. For "ota1," high-level performance constraints were specified for unity gain bandwidth and low frequency gain. For "ota2," constraints were specified for unity gain bandwidth, low frequency gain, and offset. For "comp," a high-performance clocked comparator, the high-level constraints were decision time and offset. Circuit performance was measured by extracting all capacitances and resistances from the layout and running SPICE3 with level 2 transistor models. The "comp" circuit, which

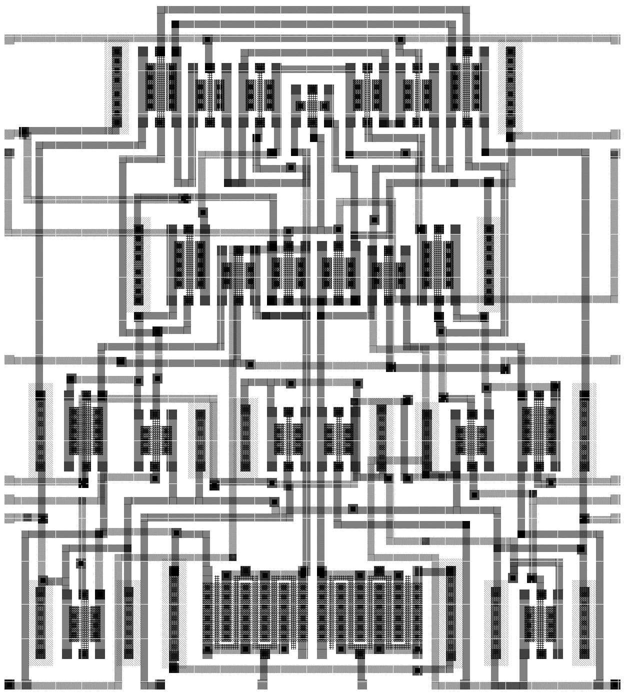

Figure 7: *Layout of the circuit "ota" after compaction.*

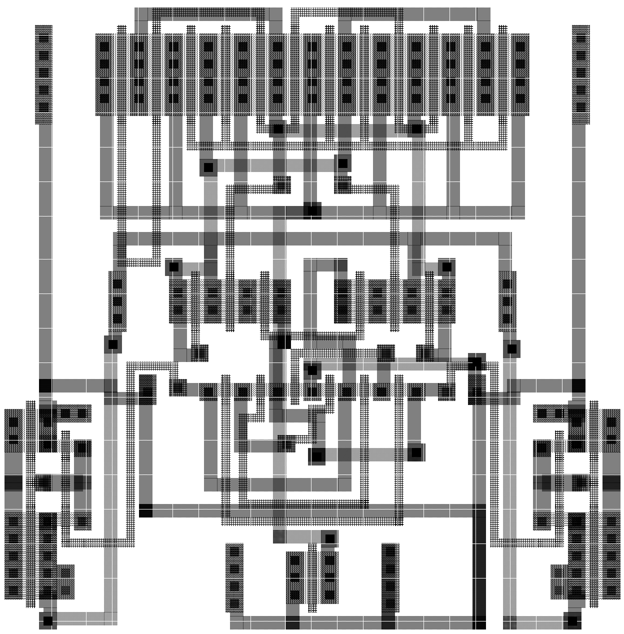

Figure 8: *Layout of the circuit "comp" after compaction.*

ota1, 39 objects, 61 critical parasitics cpu time for compaction: **106.2 sec** 2 performance constraints	ota1 before compaction	 after compaction
area	102480	69918 (68.2%)
wiring length	7333	5964 (81.3%)
unity gain bandwidth–nominal value	$6.015\ MHz$	
maximum permissible degradation (constraint)	$-180\ kHz$	
actual degradation	$-180\ kHz$	$-165\ kHz$
low frequency gain–nominal value	$63.42\ dB$	
maximum permissible degradation (constraint)	$+/-\ 1\ dB$	
actual degradation	$+0.27\ dB$	$+0.32\ dB$

Table 1: *Reduction and performance statistics for the circuit "ota1."*

ota2, 23 objects, 69 critical parasitics cpu time for compaction: **44.3 sec** 3 performance constraints	ota2 before compaction	 after compaction
area	40086	27250 (68.0%)
wiring length	3870	2870 (74.2%)
unity gain bandwidth–nominal value	$6.015\ MHz$	
maximum permissible degradation (constraint)	$-180\ kHz$	
actual degradation	$-188\ kHz$	$-143\ kHz$
low frequency gain–nominal value	$63.42\ dB$	
maximum permissible degradation (constraint)	$+/-\ 1\ dB$	
actual degradation	$+0.04\ dB$	$+0.04\ dB$
offset voltage–nominal value	$0\ \mu V$	
maximum permissible degradation (constraint)	$+100\ \mu V$	
actual degradation	$+81\ \mu V$	$+74\ \mu V$

Table 2: *Reduction and performance statistics for the circuit "ota2."*

comp, 26 transistors, 8 critical parasitics cpu time for compaction: **36.2 sec** 2 performance constraints	comp before compaction	 after compaction
area	38982	20338 (52.5%)
wiring length	2758	2144 (77.7%)
decision time–nominal value	$6.80\ ns$	
maximum permissible degradation (constraint)	$+3.00\ ns$	
actual degradation	$+2.25\ ns$	$+2.17\ ns$
offset voltage–nominal value	$756\ \mu V$	
maximum permissible degradation (constraint)	$+10\ \mu V$	
actual degradation	$+8\ \mu V$	$-13\ \mu V$

Table 3: *Reduction and performance statistics for the circuit "comp."*

first appeared in [13], was simulated with a 1 pF load capacitance. Note that the final area which SPARCS-A reports for "comp" is almost identical to the area reported in [13].

Tables 1 and 2 report the area reduction and relevant performance statistics for "ota1" and "comp." Figure 7 depicts the final layout of "ota1" and Figure 8 depicts the final layout of "comp." From the tables it is apparent that performance-driven compaction respects the circuits' high-level performance constraints and that significant reduction of chip area can be obtained in a very short CPU time.

5. CONCLUSIONS

A performance-driven compaction algorithm which accommodates *all* layout constraints associated with analog design has been presented. Because of the manner in which the set of constraints is derived, the layout produced by the compactor is guaranteed to meet all of the circuit's high-level analog performance requirements. The algorithm has been implemented and displays remarkable completeness and efficiency. When compared to previous approaches, the algorithm is the first to consider all of the analog constraints necessary to meet high-level circuit performance specifications. Furthermore, we have applied the algorithm to circuits far larger than those shown by previous approaches. The methodology performs well on practical circuits and is suitable for industrial applications.

REFERENCES

[1] T. Lengauer, *Combinatorial Algorithms for Integrated Circuit Layout*, Applicable Theory in Computer Science. John Wiley & Sons, New York, 1990.

[2] J. R. Burns and A. R. Newton, "SPARCS: A New Constraint-Based IC Symbolic Layout Spacer", in *Proc. IEEE Custom Integrated Circuits Conference*, pp. 534–539, 1986.

[3] R. Okuda, T. Sato, H. Onodera and K. Tamaru, "An Efficient Algorithm for Layout Compaction Problem with Symmetry Constraints", in *Proc. IEEE ICCAD*, pp. 148–151, November 1989.

[4] C. S. Bamji and R. Varadarajan, "Hierarchical Pitchmatching Compaction Using Minimum Design", in *Proc. Design Automation Conference*, pp. 311–317, June 1992.

[5] E. Felt, E. Charbon, E. Malavasi and A. Sangiovanni-Vincentelli, "An Efficient Methodology for Symbolic Compaction of Analog IC's with Multiple Symmetry Constraints", in *Proc. European Design Automation Conference*, pp. 148–153, September 1992.

[6] H. Chang, A. Sangiovanni-Vincentelli, F. Balarin, E. Charbon, U. Choudhury, G. Jusuf, E. Liu, E. Malavasi, R. Neff and P. Gray, "A Top-down, Constraint-Driven Design Methodology for Analog Integrated Circuits", in *Proc. IEEE Custom Integrated Circuits Conference*, pp. 841–846, May 1992.

[7] E. Charbon, E. Malavasi, U. Choudhury, A. Casotto and A. Sangiovanni-Vincentelli, "A Constraint-Driven Placement Methodology for Analog Integrated Circuits", in *Proc. IEEE Custom Integrated Circuits Conference*, pp. 2821–2824, May 1992.

[8] E. Malavasi, U. Choudhury and A. Sangiovanni-Vincentelli, "A Routing Methodology for Analog Integrated Circuits", in *Proc. IEEE ICCAD*, pp. 202–205, November 1990.

[9] U. Choudhury and A. Sangiovanni-Vincentelli, "Constraint-Based Channel Routing for Analog and Mixed-Analog Digital Circuits", in *Proc. IEEE ICCAD*, pp. 198–201, November 1990.

[10] A. Mlynsky and C.-H. Sung, "Layout Compaction", in *Layout Design and Verification*, ch. 6, pp. 199–235. T.Ohtsuki Ed., North Holland, 1986.

[11] U. Choudhury and A. Sangiovanni-Vincentelli, "Constraint Generation for Routing Analog Circuits", in *Proc. Design Automation Conference*, pp. 561–566, June 1990.

[12] J. L. Burns, "Techniques for IC Symbolic Layout and Compaction", Memorandum UCB/ERL M90/103, UCB, November 1990.

[13] J. M. Cohn, D. J. Garrod, R. A. Rutenbar and L. R. Carley, "KOAN/ANAGRAM II: New Tools for Device-Level Analog Placement and Routing", *IEEE Journal of Solid State Circuits*, vol. 26, n. 3, pp. 330–342, March 1991.