An Efficient Methodology for Symbolic Compaction of Analog IC's with Multiple Symmetry Constraints

Eric Felt, Edoardo Charbon, Enrico Malavasi†, 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

An efficient approach to the symbolic compaction of analog integrated circuits is presented. A fast graph-based algorithm performs a preliminary compaction taking into account a set of basic spacing constraints. The obtained configuration provides the starting point for a linear program, which optimizes the layout introducing multiple device and wire symmetry constraints. The efficiency and robustness of this technique allows the use of our compactor for very complex analog circuits with multiple symmetries and other performance constraints.

1 Introduction

New technologies allowing on-chip integration of mixed analog/digital circuits and the increasing importance of ASIC's has determined the emergence of new tools for analog synthesis. Due to a higher sensitivity of the electrical performances to layout details, special tools with modified objectives with respect to their digital counterpart must be created. Custom design requires great flexibility, and layout synthesis is often a multiple-objective optimization problem. Along with area, wiring length and delay, two groups of relevant issues must be taken into account:

1. Topological constraints, i.e. symmetries and matching.
2. Parasitics associated with devices and interconnects.

A flexible design method is commonly used in the digital domain based on technology independent placement and routing followed by a spacing (compaction) step. The purpose of the latter phase is to make the final adjustments to the layout, optimizing the area while enforcing all design rules, performance specifications and topological constraints.

The issue of analog compaction has been addressed only recently [1, 2, 3]. In these approaches, only symmetry constraints have been considered. Several techniques have been proposed to carry out compaction. The most general and effective approach for digital compaction is based on the computation of the critical paths of a constraint graph [4]. In [5] a method based on the constraint graph algorithm was proposed for analog compaction. In a preliminary step the graph is solved totally ignoring the symmetry. In a second phase the graph is slightly perturbed by gradually introducing the symmetry as an additional constraint set. In the final phase the compaction is completed by redistributing the wire slack. This approach shows good results only when the number of symmetry constraints is small. Furthermore, with two or more symmetry axes, overconstraints frequently arise due to the interaction between symmetry constraints. In order to overcome this shortcoming, mobile axes must be allowed, using the method of *Virtual Symmetry Axes*, proposed in [6]. In [2] an integer programming (IP) approach to the problem has been presented. With this method a good level of optimization can be achieved but its time complexity strongly limits its feasibility in practical circuits.

More recently a new method [3] which combines the previous two algorithms has been proposed. Constraint graph techniques are used to collapse all objects not in immediate contact with those on which a symmetry constraint has been imposed. New "super-constraints" are generated and solved using a classical linear programming (LP) approach. This constraint reduction scheme, while inducing a relative efficiency improvement, has some fundamental limitations. The constraint set has to be reduced in order to control the time complexity of the collapse operation, and thus important constraints such as maximum spacing constraint are not supported. In addition, the circuit complexity which can be considered is small compared with the needs of today's analog designs.

In this paper we propose an efficient methodology targeted toward symbolic compaction accounting for the complete set of analog constraints. The proposed methodology is described below:

1. Solve the spacing problem with no symmetry constraints using graph techniques.
2. By means of an LP algorithm based on the simplex method, solve the symmetry constraints, using the graph solution as starting point.
3. Round off the coordinates of the elements which do not lie on the critical path.
4. Verify that all constraints are satisfied.

The use of the graph solution as starting point is the key to obtain a significant speed-up in the solution of the LP. When compared to previous approaches making use of a pure LP solver [3], this algorithm represents a substantial improvement. In fact for most practical circuits our LP solver starts from a feasible configuration which is already close to the final solution. The range of cases that can be managed with acceptable computational complexity is therefore significantly expanded.

The constraint graph technique [7] and [8, Ch. 10] used in phase 1 of our method has been chosen for its robustness, efficiency and for the quality of the resulting layout. The optimal solution of mono-dimensional compaction can be obtained in polynomial time with the longest-path algorithm. Heuristics [9] exist that interrelate efficiently the two dimensions and incorporate wire balancing into mono-dimensional compaction algorithms. In particular, in our approach we have chosen a mono-dimensional constraint-graph longest-path compactor called SPARCS[1].

The paper content is organized as follows. In section 2 the general compaction problem is formulated. In section 3 the classical graph based solution for digital compaction is presented. In section 4 the procedure for the generation of the analog constraints is described. In section 5 the details of the LP algorithm are outlined and in section 6 the feasibility of the method is shown through examples.

2 Problem Formulation

The primary objective of a compactor is to minimize total chip area subject to a set of constraints which guarantee proper circuit functionality. This can be formulated as:

$$minimize \;\; (x_R - x_L)(y_T - y_B) \;\; subject\ to\ \{C\}, \qquad (1)$$

where
$x_{R(L)}$ = right-most (left-most) circuit element
$y_{T(B)}$ = top-most (bottom-most) circuit element
$\{C\}$ = set of constraints

Mono-dimensional compactors perform this optimization indirectly by separating the horizontal and the vertical components of the minimization and iterating until the total chip area remains constant. The mono-dimensional algorithm can be represented as:

repeat {
  minimize $(x_R - x_L)$ subject to $\{C_X\}$
  minimize $(y_T - y_B)$ subject to $\{C_Y\}$
} **until** $(x_R - x_L)(y_T - y_B)$ does not change

where
$\{C_X\}$ = set of horizontal constraints
$\{C_Y\}$ = set of vertical constraints

3 Classical Constraint Graph Solution

In digital compactors the set of constraints generally consists of minimum and maximum spacing constraints which arise from process limitations. For example, the metal layers must be separated by a certain distance to avoid short circuits. In classical graph algorithms these constraints are mapped into a weighted, directed graph. Two graphs are actually used, one for the vertical and one for the horizontal direction. Layout elements map to nodes in the graph and spacing constraints to weighted edges. The edge directions specify the relative ordering of the corresponding nodes, while the weights model the separation requirements. A simple plane sweep algorithm is generally used to generate the constraints [5]. The constraint graph is then solved using standard longest path algorithms. Note that negative weight arcs, which represent maximum spacing constraints between nodes, are generally allowed and the resulting graph can still be solved or determined infeasible in polynomial time. The constraint graph algorithms can thus handle constraints of the following forms:

$$x_2 - x_1 \geq k \qquad (2)$$

$$x_2 - x_1 \leq k \qquad (3)$$

where x_1 and x_2 are the locations of two circuit elements (nodes) and k is a maximum or minimum spacing constraint (arc).

In the case of analog circuits, the set of constraints must be expanded to encompass circuit performance considerations. Because of parasitic capacitances and non-linear behavior, the performance of an analog circuit is often dependent on the details of the physical implementation. Thus an analog compactor must insure that the final layout be both design-rule correct and properly functional. Because of the nature of these analog performance constraints, traditional constraint graph approaches are not suitable for analog circuits.

4 Constraint Generation

To guarantee proper analog circuit functionality, a set of new additional constraints has been developed to account for:

1. Device symmetry preservation. Analog circuits are placed symmetrically to minimize electrical offsets and noise.
2. Wiring symmetry preservation. Same motivation as 1.
3. Minimum wire segment spacing. Minimum spacing between critical wires is often required to reduce cross-coupling.
4. Shield preservation. Shields, consisting of grounded metal lines, are occasionally introduced into the layout to reduce the cross-coupling between interconnect lines.

Each of these additional constraints will be considered in turn.

Assuming the constraints are properly generated, an analog layout compactor which can accommodate these additional classes of constraints can guarantee that the final compacted layout will meet all high-level performance specifications.

From close examination of the special constraints required when compacting analog layouts, it is evident that the graph perturbation method is not suitable because it cannot accommodate all classes of constraints, most notably device and wire symmetry constraints. The LP approach, applied as suggested in our methodology, is the key to an efficient solution to the general analog compaction problem.

4.1 Device Symmetry Preservation

Analog circuit designers frequently introduce topological symmetry to minimize noise and offset. A circuit with symmetry constraints is illustrated in Figure 1. It is essential that layout compaction preserve device symmetries. For a vertical symmetry axis, the symmetry constraints take the form:

$$x_a - x_s = x_s - x_b \qquad (4)$$

$$y_a = y_b \qquad (5)$$

where
x_s = position of the symmetry axis
$x_{a(b)}$ = horizontal position of device a (b)
$y_{a(b)}$ = vertical position of device a (b)

The vertical constraint (5) is introduced into the constraint graph. The horizontal constraint (4), however, cannot be expressed in any simple constraint graph form. It is therefore introduced as a linear programming constraint of the form:

$$x_a + x_b - 2x_s = 0 \qquad (6)$$

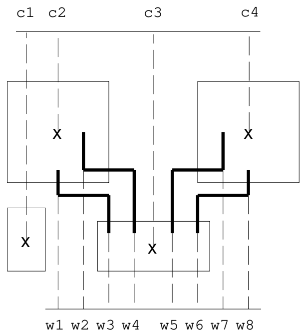

Figure 1: *Example of symmetry constraints (vertical constraints omitted for clarity).*

Device Symmetries:

$$c_4 - c_S = c_S - c_2$$
$$c_3 = c_S$$
$$c_S = \text{position of symmetry axis.}$$

Wiring Symmetries:

$$w_8 - w_6 = w_3 - w_1$$
$$w_8 > w_6$$
$$w_3 > w_1$$
$$w_7 - w_5 = w_4 - w_2$$
$$w_7 > w_5$$
$$w_4 > w_2$$

4.2 Wiring Symmetry Preservation

Wire symmetry constraints are very similar to device symmetry constraints. They are introduced to minimize the differences between matched components of a circuit. The corresponding horizontal constraint for two wires labeled 1 and 2 is represented as:

$$|x_{1,end} - x_{1,start}| = |x_{2,end} - x_{2,start}| \tag{7}$$

where

$x_{1,end}$ = rightmost ending of wire segment 1
$x_{1,start}$ = leftmost ending of wire segment 1
$x_{2,end}$ = rightmost ending of wire segment 2
$x_{2,start}$ = leftmost ending of wire segment 2

A similar equation can be used for the vertical constraint. In general (7) is a nonlinear constraint. However, it can be alternatively expressed in the following more restrictive form:

$$x_{1,end} - x_{1,start} = x_{2,end} - x_{2,start} \tag{8}$$
$$x_{1,end} \geq x_{1,start} \tag{9}$$
$$x_{2,end} \geq x_{2,start} \tag{10}$$

In this form the constraint can be directly incorporated into a standard LP approach and it is handled in the same fashion as device symmetry constraints.

The LP form for the wiring symmetry constraints restricts wire ending crossing during the compaction; the rightmost ending of a wire segment before compaction will never become the leftmost ending after compaction. As a result the total chip area may not be as small as it would have been if endings had been allowed to cross. However, for layouts in which the relative positions of devices are already close to their optimum the area increase due to this ending crossing restriction is negligible.

4.3 Minimum Wire Segment Spacing

Capacitive coupling between adjacent wire segments is a significant concern in many analog circuits. The coupling between two segments labeled 1 and 2 is controlled by introducing additional minimum spacing constraints of the form:

$$x_{segment2} - x_{segment1} \geq d_{min}. \tag{11}$$

where

$x_{segment2}$ = location of segment 2
$x_{segment1}$ = location of segment 1
d_{min} = minimum spacing between segment 1 and segment 2

the parameter d_{min} is set by our automatic parasitic constraint generator PARCAR[10] or it can be imposed by the user. The resulting equations are then introduced in the constraint graph.

4.4 Shield Preservation

Figures 3 and 2 illustrate a lateral and a vertical shield created during placement and routing to reduce coupling between critical nets. The spacing requirement for the lateral shield between two nets labeled 1 and 2 is:

$$x_{shield} \geq x_{net1} \quad OR \quad x_{shield} \geq x_{net2}. \tag{12}$$

where

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

Because of the OR condition this is a nonlinear constraint. An overly-restricted form of the constraint, however, is introduced into the constraint graph:

$$x_{shield} \geq x_{net1} \quad AND \quad x_{shield} \geq x_{net2}. \tag{13}$$

During the linear programming, a less overly-restricted form of the constraint is used:

$$x_{shield} = AVERAGE\ (x_{net1}, x_{net2}) \tag{14}$$

Vertical shield constraints are simply expressed as:

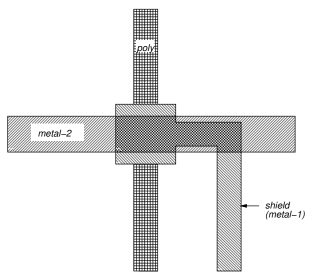

Figure 2: *Vertical shielding.*

$$x_{m1} = x_{m2} \tag{15}$$
$$y_{m1} = y_{poly} \tag{16}$$

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.

These requirements are translated into four inequality constraints and introduced into the initial graph formulation of the problem.

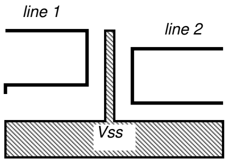

Figure 3: *Lateral shielding.*

5 Linear Programming Solution

5.1 Theoretical Considerations

The formulation of the problem and the nature of the starting point allow us to enforce the symmetry constraints using linear programming rather than integer programming. In fact, under the following conditions the analog compaction problem is unimodular, i.e. the optimum solution contains only integer coordinates:

Theorem:
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,

Then there exists a solution to the LP problem where all coordinates are integer.

Proof:

- Let us first consider the case of a node on the critical path. Suppose that some node n_0 is on the critical path but does not converge to an integer in the optimum solution. The critical path between the leftmost node and n_0 must contain at least one *unnested* symmetry constraint with a non-integer separation between the symmetric nodes, since all other constraints are integers and any sum of integers is an integer. Call the nodes of this symmetry constraint n_{s1} and n_{s2} [1]. Since n_{s1} and n_{s2} are unnested, the only constraints between them are minimum or maximum spacing constraints. All minimum spacing constraints are integers, so the total minimum distance between n_{s1} and n_{s2} must be an integer. If the minimum separation between the two symmetric nodes in an integer and the actual separation is not an integer distance, then either the symmetric nodes are not on the critical path or the solution is not optimal because the area could be decreased by moving the two symmetric nodes closer together. These conclusions contradict the original assumption. Thus if n_0 is on the critical path then it must converge to an integer value.
- Consider now nodes not on the critical path. These nodes may or may not converge to an integer value. We can always obtain a solution with the same cost, where the non-critical nodes are at integer coordinates.

To adjust non-integer to integer coordinates, the nodes have to be "packed" against nodes on the critical path. Since each non-critical node is separated from critical nodes by only integer spacing constraints, it can be moved to the left by extending the linear programming objective function to be:

$$minimize \quad \sum_{i=1}^{N}(x_i - x_0) \qquad (17)$$

where
x_i = location of node i.
x_0 = location of leftmost node.
N = number of nodes.

This objective function forces every node to be as close as possible to the leftmost node, x_0.

Another solution can be obtained by rounding off the coordinates of any nodes which do not converge to integers. There is always sufficient "slack" in the constraints so that these non-critical nodes can be rounded off to the nearest integer without violating any constraints and without increasing the area of the circuit. These propositions will be considered in turn:

- Rounding off a non-critical node can never increase the area of the circuit. Rounding off never moves a node by more than 0.5 units, and if the constraints on that node were such that there was not at least 1 unit of slack on the node, then the node would have been on the critical path. This arises from the fact that all non-symmetry constraints are integer-valued.
- Rounding off a non-critical node can never violate a constraint. Since all non-symmetry constraints are integer-valued, any node which is not located at an integer coordinate has at least enough slack to move to the nearest integer coordinate. If it did not, then it's original non-integer coordinate would not have satisfied all of the spacing constraints.

To minimize computational complexity it is desirable to keep the linear programming objective function as simple as possible, hence the round-off approach is superior to the "packing" approach for most problems. Moreover, all elements on the left may result in an "inferior" layout.

5.2 Linear Programming Algorithm

The analog compactor described in this work uses both graph-based techniques and linear programming techniques, taking advantage of the high speed of the former and the generality of the latter.

The LP algorithm used in this implementation is a standard simplex algorithm which finds the optimum solution by sequentially visiting the vertices of the polytope bounding the feasible region until the optimum solution is found. Thus providing the linear programming with a good initial solution minimizes the number of required iterations, since the optimum solution will be found sooner than if one starts with a random initial state. In our methodology we use constraint graph techniques to obtain this important starting point.

Constraint graph perturbation approaches to preserving device symmetries have been investigated [5]. In general these methods only work for circuits with one fixed axis of symmetry. The principle advantages of the linear programming approach over graph perturbation approaches are:

1. linear programming works for multiple symmetry axes, and
2. linear programming allows the symmetry axes to move during compaction.

As a result of these advantages linear programming guarantees a globally optimum solution, which the graph perturbation approaches do not.

[1] Unnested refers to the fact that there is no pair of symmetrically-constrained nodes on the critical path between n_{s1} and n_{s2}.

	ota			newOta		
	original	no symmetry	symmetry	original	no symmetry	symmetry
# instances	39	39	39	23	23	23
# symmetric obj.	39	39	39	23	23	23
area	102480	78375	79800	40086	29700	29737
area compression	100%	76%	77.9%	100%	74.1%	74.2%
wiring length	7333	5877	5883	3869	2856	2878
CPU time	-	5.5 sec	56.3 sec	-	6.3 sec	24.3 sec
# constraints	-	-	2946	-	-	1878
# variables	-	-	573	-	-	328

	newOta		
	original	no symmetry	symmetry
unity gain bandwidth	6.53 MHz	6.08 MHz	6.58 MHz
phase margin	82.8^o	95.9^o	83.32^o
offset voltage	81 μ V	1,080 μ V	74 μ V

Table 1: *(a) Different types of compaction on a set of examples. (b) Electrical performance in compacted circuits.*

A C++ class library is used for the problem formulation. The library interfaces with various linear programming packages which perform the actual optimization. The solution contains the optimum values for each variable and a flag to indicate whether the solution is optimum, underconstrained, or overconstrained. A flag for each variable indicates whether that variable is critical, at its upper bound, at its lower bound, free, overconstrained, etc. All of the specific details about how to interface to a particular optimizer are abstracted from the user. The C++ class supports two optimization packages: MINOS [11], a general nonlinear optimization package which has been under continuous development at Stanford since 1981, and MINIT [12], a public domain dual simplex linear programming package. The primary difference between these two packages is that MINOS uses sparse matrices and MINIT full matrices, so MINOS requires less memory on large problems but MINIT is faster on small problems.

6 Results

The tool has been implemented in C++ and consists of about 20,000 lines of code. Considering the generality of the compaction problems that the algorithms can solve, the performance is quite reasonable. The largest feasible problem which we have solved is 4000 constraints and 800 variables; on that large problem the execution time is still only 2.5 minutes on a DECsystem 5900.

Two examples of a folded cascode CMOS op amp were placed, routed, and compacted with the new tool. In both examples significant reduction of chip area were obtained in a very short CPU time. In Figure 4 the circuit newOta is depicted after placement and routing. Figure 5 shows the layout after compaction with no enforcement on any analog constraints. Finally in Figure 6 the compacted layout with enforcement of symmetry constraints is shown. Notice the better utilization of silicon and the reduction of highly resistive paths. The reduction statistics for both this layout and a second, larger op amp are reported in Table 1. Furthermore parasitics were extracted from the layouts and various analyses were performed. Table 1 reports several performance data obtained with SPICE simulations. With no enforcement of symmetry constraints, the compactor failed to control mismatch-sensitive circuit non-linearities. On the contrary using our methodology it was possible to preserve and even to improve the original circuit performance.

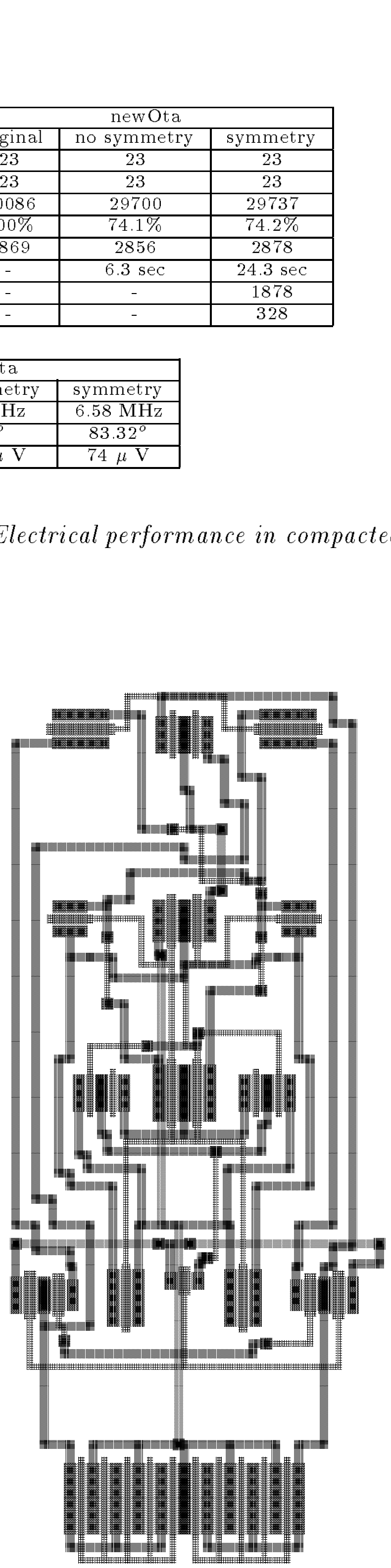

Figure 4: *Layout of the circuit "newOta" before compaction.*

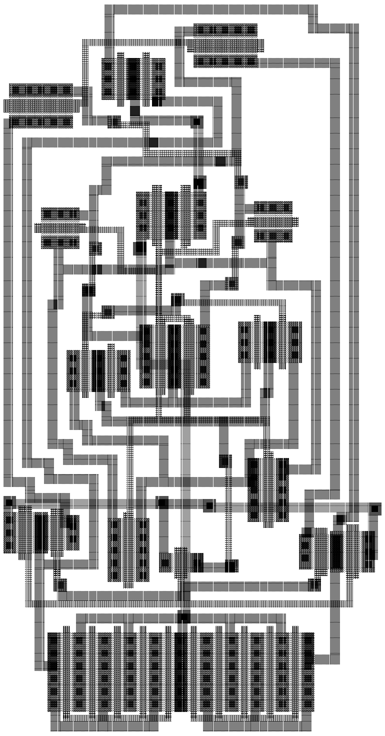

Figure 5: *Layout of the same circuit after non-symmetric compaction.*

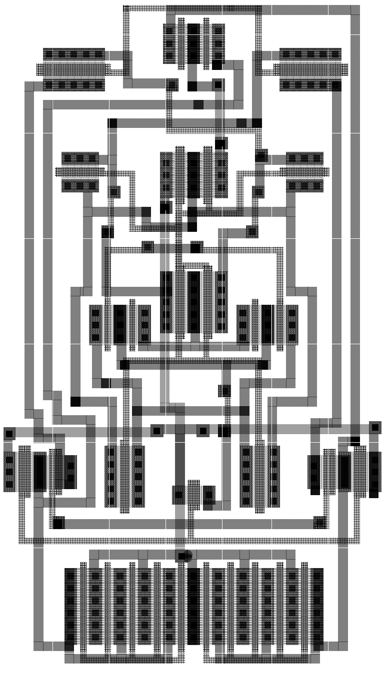

Figure 6: *Layout of the same circuit after symmetric compaction.*

7 Conclusions

An efficient methodology for solving general analog compaction problems has been developed. The combination of a graph-based and a linear programming solver has shown remarkable robustness and significant speed-up when compared to previous approaches.

Future work includes the automatic generation of the minimal spacing constraints for geometrically complex interconnect lines (see section 4.3), the study of new methods for efficient slack redistribution, and the integration of the tool into the new top-down, constraint-driven framework for analog integrated circuits currently under development [13].

8 Acknowledgements

This research has been partially supported by DARPA and SRC. The first author has been supported by a California Microelectronics Fellowship and the second author by the Swiss Science Foundation, Bern, Switzerland. The authors would like to thank Gani Jusuf for providing the examples presented in this work.

References

[1] 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.

[2] H. J. DeMan L. Rijnders, P. Six, "Design of a Process-Tolerant Cell Library for Regular Structures Using Symbolic Layout and Hierarchical Compaction", *IEEE Journal of Solid State Circuits*, vol. 23, n.3, pp. 714–721, 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] A. R. Newton, "Symbolic Layout and Procedural Design", in *Design Systems for VLSI Circuits*, pp. 65–112. De Micheli et al. Eds., Martinus Nijhoff, 1987.

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

[6] E. Malavasi, E. Charbon, G. Jusuf, R. Totaro and A. Sangiovanni-Vincentelli, "Virtual Symmetry Axes for the Layout of Analog IC's", in *Proc. 2^{nd} ICVC, Seoul, Korea*, pp. 195–198, October 1991.

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

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

[9] W. Schiele, "Improved Compaction with Minimized Length of Wires", in *Proc. Design Automation Conference*, pp. 390–395, 1983.

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

[11] B. A. Murtagh and M. A. Saunders, "MINOS 5.1 User's Guide", Technical Report Rep. SOL 83-20R, Dept. of Operations Research, Stanford University, Stanford, CA, Jan 1987.

[12] R. Salazar and S. Sen, *Collected Algorithms from CACM*, The Association, 1966.

[13] 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.