VIRTUAL SYMMETRY AXES FOR THE LAYOUT OF ANALOG IC'S

Enrico Malavasi † Edoardo Charbon Gani Jusuf
Roberto Totaro † Alberto Sangiovanni-Vincentelli

Dept. of EECS - University of California, Berkeley
Berkeley CA 94720

† DEI, Università di Padova
I- 35151 Padova ITALY

Abstract

A new approach to the layout of integrated circuits with multiple symmetry axes is presented in this paper. When more than one symmetry is present, the usual approach to placement and compaction makes extensive use of hierarchy, which requires fixed positions for symmetry axes. As a result, wiring and area optimizations are poor.

The position of a virtual symmetry axis is variable, and dynamically defined by the center of a group of symmetric modules and wires called cluster*. Virtual symmetry axes have been implemented in a placement tool using the Simulated Annealing algorithm and in a compaction tool based on the constraint-graph longest-path algorithm. Results are shown proving the suitability of this approach.*

1 Introduction

Differential architectures are often used in analog circuits. They are based on electrically symmetric networks, where good matching is required between device parameters and interconnection parasitics. An effective way to meet tight matching requirements is to implement a geometrically symmetric layout. Symmetric interconnections have similar geometries and therefore their parasitics are well matched. Symmetric devices have the same size and mirrored orientation, which yields well matched characteristics [1] provided the distance between them is not too large.

In complex circuits, it is important to keep certain groups of devices clustered. The reason is not only to achieve better matching, but also to improve routability. Each cluster may have its own symmetry requirements. Different groups of devices and interconnections may be symmetric with respect to independent symmetry axes. A few CAD frameworks address this issue. In ILAC [2], symmetries are considered during placement but not during compaction. In ACACIA, [3], provision is made for one symmetry axis at a time. The only way to take into account more than one axis is by extensive use of hierarchy. This helps in simplifying the design of complex objects. However, it also makes area optimization in the layout poorer, since it introduces stylization and restrictions on geometries and wiring.
In [4] symmetric compaction is solved as a linear programming problem with a number of additional symmetry constraints. However this approach is computationally acceptable only where the number of symmetry requirements is fairly small compared with the number of elements in the circuit.

In this work we describe an original approach for non-hierarchical placement and compaction with multiple-symmetry circuits. It provides maximum flexibility to area and routability optimization, while preserving symmetries in each of the clusters in which the circuit has been partitioned. Our approach is based on the idea of *virtual symmetry axes*. A virtual axis is an axis whose location is not fixed, but is dynamically determined as the center of a cluster of symmetric objects. Details on the characteristics of virtual symmetry are given in Section 2. Based on the concept of virtual symmetry axes two algorithms have been built, one for placement and one for compaction. These algorithms are described in Section 3 and 4 respectively. They have been implemented in existing tools, which are part of a framework for integrated analog and digital synthesis under development at the University of California at Berkeley. Algorithm performances have been tested on a number of real circuits. Results are reported in Section 5.

2 Virtual Symmetry Axes

Let ξ be a symmetry axis for a circuit. Given a coordinate system, its position in the layout is defined by one coordinate z_ξ, either an abscissa, in case of a vertical axis, or an ordinate if it is horizontal. We will call *cluster of* ξ the set of all modules, wire segments and terminals linked to each other by symmetry constraints with respect to ξ. Placement, routing and compaction must account for all the symmetry relations of each cluster, by introducing and preserving proper module orientations, symmetric routing and equal distances from the axis. Moreover, the modules of the same cluster should be as close as possible to each other in order to maximize matching between electrical parameters and optimize routing of symmetric nets.

A virtual symmetry axis is an axis whose position z_ξ is not fixed and therefore not known a priori. It can be determined at any time by computing the center coordinate of its cluster. The position and shape of the objects in each cluster are modified by layout tools, hence virtual axes can move.

Virtual symmetry axes are a simple solution to the problem of multiple symmetry axes. By referring to the same axis, the modules of each cluster preserve their symmetries, although small temporary degradations may be introduced. At the same time, the goal of area optimization is not contrasted by fixed user-defined axis positions.

3 Placement

Virtual symmetry axes have been introduced into a placement tool called PUPPY [5], based on the Simulated Annealing (S.A.) algorithm [6]. S.A. is a non-deterministic, iterative minimization algorithm. At each iteration, the variation of the objective function, or *cost function*, deriving from random modifications of the configuration is evaluated. State transitions leading to smaller cost function values are always accepted. The ones leading to higher costs are accepted with probability

$$\text{Prob} = e^{-\frac{\triangle E}{T}}$$

where $\triangle E$ is the cost increase. The variable parameter T is called *temperature* and assumes decreasing values following a schedule depending on applications.

In PUPPY the cost function is a weighted sum given by the following expression:

$$\begin{aligned} F(s) = \quad & \alpha_{wl} f_{wl}(s) + \alpha_v f_v(s) + \alpha_a f_a(s) + \alpha_{ov} f_{ov}(s) + \\ & + \alpha_w f_w(s) + \alpha_m f_m(s) + \alpha_s f_s(s) \end{aligned} \tag{1}$$

where s is the state, f_{wl} and f_v are *routability* functions, given respectively by maximum net length and number of violations of constraints on maximum net lengths. Functions f_a and f_{ov} (*area* functions), evaluate respectively the total circuit area and overlapping area between modules. Function f_w is used with transistor-level devices and penalizes configurations where equipotential wells are separated. Function f_m evaluates device mismatches. Function f_s is the *symmetry* function, which evaluates the overall asymmetry of the placement. The symmetry function expresses the distance between the given configuration and a symmetric one in the state space. Parameters $\alpha_w l, \alpha_v, \alpha_a, \alpha_{ov}, \alpha_w, \alpha_m$ and α_s are user-defined weights. Temperature decreases by steps with logarithmic schedule. At each step T remains constant for a large number of iterations (typically 100 to 1000).

For each cluster, every displacement from rigorously symmetric position gives a positive contribution to the cost function. Such displacements are possible, since S.A. can perform cost-increasing transitions. Moreover, the symmetry function in is only one of the contributions to the cost function (1), competing with other objectives. Transitions leading to better area or routability may reduce the cost function value even if the degree of symmetry is decreased. Consequently, limited symmetry degradations are introduced by placement iterations. The asymmetry in one or more of the device pairs in a cluster determine a shift of the center of the cluster itself. The virtual symmetry axis dynamically is also shifted by the same amount.

At low temperature, the probability of accepting a transition with increasing cost is small. The movements of the modules of a cluster to non-symmetric positions at low temperature are mostly due to the effect of the other components of the cost function, contrasting with the symmetry function. Each symmetry axis follows the center of its cluster, trying to restore symmetries. In this way, area and wirability optimization remain a primary concern during placement, and symmetries are preserved.

Axis positions must be calculated before each iteration of the S.A. algorithm. As a consequence, the running time performance with virtual symmetry axes is worse than that with fixed axes. Degradation can be substantially reduced if axes are not moved before each iteration, but every n iterations, with $n \gg 1$. The value of n is the same used for temperature steps in the cooling schedule. We calculate axis positions after each temperature decrement. The principle of virtual symmetry axes still holds and convergence of S.A. is preserved.

4 Compaction

An algorithm for analog compaction using virtual symmetry axes has been implemented in a tool called SPARCS [7], based on the mono-dimensional longest-path search compaction algorithm [8]. The algorithm is alternatively repeated in orthogonal directions in an iterative approach.

SPARCS is a constraint-based symbolic layout spacer. Hence it can deal with one symmetry axis by means of a set of *constraints* that can be specified to require symmetries between modules and wire segments. Let ξ be the only axis of a circuit. We suppose, without any loss of generality, that ξ is vertical and x_ξ is its abscissa. Let a_l and a_r be two elements symmetric with respect to this axis. Let $x(a_l)$, $y(a_l)$, $x(a_r)$, $y(a_r)$ be their coordinates. When an iteration of the compaction algorithm is performed in the horizontal direction (orthogonal to the axis), the constraint is

$$x(a_l) - x_\xi = x_\xi - x(a_r). \tag{2}$$

When an iteration is carried out in the vertical direction (parallel to the axis), the constraint is

$$y(a_l) = y(a_r). \tag{3}$$

These constraints on object locations are used together with the ones coming from the layout design rules. If there is one axis, no restriction is introduced by fixing its position x_ξ. In fact the left and right sides of the circuit are free to shift horizontally, thus defining the width of the layout. Their final distance is determined by the length of the longest horizontal path.

With multiple symmetry axes, symmetry constraints would define the distance between axes. Therefore they cannot be used without fixing a priori axis relative positions. Let $\xi_1, \ldots, \xi_N$ be N symmetry axes. Let z_i be the position of axis ξ_i (its abscissa if it is a vertical axis, its ordinate if it is horizontal). At each iteration we apply mono-dimensional compaction N times as follows:

for $i = 1$ to N

1. For all the objects in ξ_i's cluster, apply the symmetry constraint (2) with $x_{\xi_i} = z_i$ if the on the direction of compaction is orthogonal to the direction of ξ_i. Apply constraint (3), if they are parallel.

2. For each of the other axes ξ_j, with $j \neq i$, substitute its cluster with its hierarchic view. In this way the whole cluster can translate rigidly, but its shape and the distances between each pair of objects in it cannot change.

3. Execute the compaction iteration.

4. Remove the additional constraints added in the previous steps.

No constraints are introduced for objects with no symmetry requirements, since they don't belong to any cluster.

At the beginning of each step, the location of each axis is determined by the center of its cluster. During the i-th step, the shape of the cluster of axis ξ_i may change, but the position of ξ_i is fixed. The other axes are dynamically defined by their clusters, since they are all virtual.

Longest-path worst-case running time performance is $O(LU)$ per iteration [9], where L and U are the numbers of the lower- and upper-bound branches respectively in the constraint graph. Each symmetry constraint introduces at most two branches, a lower-bound and an upper-bound. With N fixed axes and hierarchical compaction the time required for each iteration would be $O(N(L+S)(U+S))$, where S is the maximum number of symmetry constraints within one cluster; L and U are the max numbers of lower- and upper-bound branches between the elements of any cluster. We suppose $N \ll L, U$. With N virtual symmetry axes the algorithm runs N times as many iterations as in the previous case. However, at each iteration the time complexity is only $O((L+S)(U+S))$. Consequently the running time is still $O(N(L+S)(U+S))$.

5 Results

The usefulness of non-hierarchical layout is evident in the following example. Two similar folded-cascoded op-amps share bias and supply lines. The two op-amps don't need to be matched with each other, but both require symmetry. The layout shown in Figure 1 was obtained by placing by hand two symmetric circuits. The hierarchical approach provides the best matching between the two op-amps. On the other hand, the space used by the circuit depends on the effective shape of the two modules. Since each module was placed independently of the other, routing can be hard, especially if sensitive nets are present. Also, the modules could not be used with different aspect ratios.

In figure 2, the two op-amps have been placed non-hierarchically, with virtual symmetry axes, in the same area. Symmetry has been preserved for each op-amp. Slight differences are present between the two devices. Interconnections between the op-amps were taken into account during placement. The layout was routed with ROAD [10], an analog router able to control symmetries and parasitics.

In figure 3 the circuit is shown with the same area, but different aspect ratio. The axes were aligned during placement and the two devices merged in a configuration defined mainly on the ground of aspect ratio and routability considerations. Although the two op-amps are no longer physically distinct, their symmetries are preserved, as well as the device matching. Symmetric routing was performed by ROAD. Over-the-device routing was carefully controlled in order to match the parasitics on symmetric nets.

6 Conclusions

An original approach for the synthesis of layouts with multiple symmetry axes has been illustrated. It allows variable spacing between axes, therefore it provides layout synthesis tools with greater flexibility. Circuits can fit into quite different aspect ratios, which makes this approach particularly useful when alternative floor-planning configurations have to be considered. Area optimization and routability remain primary issues, while symmetries are preserved. Examples have been reported showing the flexibility of the results which can be obtained with this methodology.

7 Acknowledgments

The authors would like to thank Greg Sorkin for expert advice on Simulated Annealing analysis. This research has been partially supported by SRC grant N. 91-DC-008, MICRO project of the State of California and "Swiss National Foundation", Bern, CH.

References

[1] M. J. M. Pelgrom, A. C. J. Duinmaijer, and A. P. G. Welbers, "Matching Properties of MOS Transistors", *IEEE JSSC*, vol. 24, pp. 1433–1440, October 1989.

[2] J. Rijmenants et al., "ILAC: An Automated Layout Tool for Analog CMOS Circuits", in *Proc. IEEE CICC*, pp. 761–764, May 1988.

[3] J.M.Cohn, D.J.Garrod, R.A.Rutenbar, and L.R.Carley, "New Algorithms for Placement and Routing of Custom Analog Cells in ACACIA", in *CICC*, pp. 2761–2765, 1990.

[4] 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.

[5] Andrea Casotto, Fabio Romeo, and Alberto Sangiovanni-Vincentelli, "A Parallel Simulated Annealing Algorithm for the Placement of Macro-Cells", *IEEE Trans. on CAD*, vol. CAD-6(5), September 1987.

[6] E. H. L. Aarts and P. J. M. van Laarhoven, *Simulated Annealing: Theory and Applications*, D. Reidel Publishing, 1987.

[7] J.R.Burns and A.R.Newton, "SPARCS: A New Constraint-Based IC Symbolic Layout Spacer", in *Proc. of IEEE CICC*, pp. 534–539, 1986.

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

[9] Y. Z. Liao and C. K. Wong, "An Algorithm to Compact a VLSI Symbolic Layout with Mixed Constraints", *IEEE Trans. on CAD*, CAD-2, pp. 62–69, April 1983.

[10] E. Malavasi, M. Chilanti, and R. Guerrieri, ""A General Router for Analog Layout"", in *Proc. COMPEURO '89, Hamburg*, pp. 549–551, May 1989.

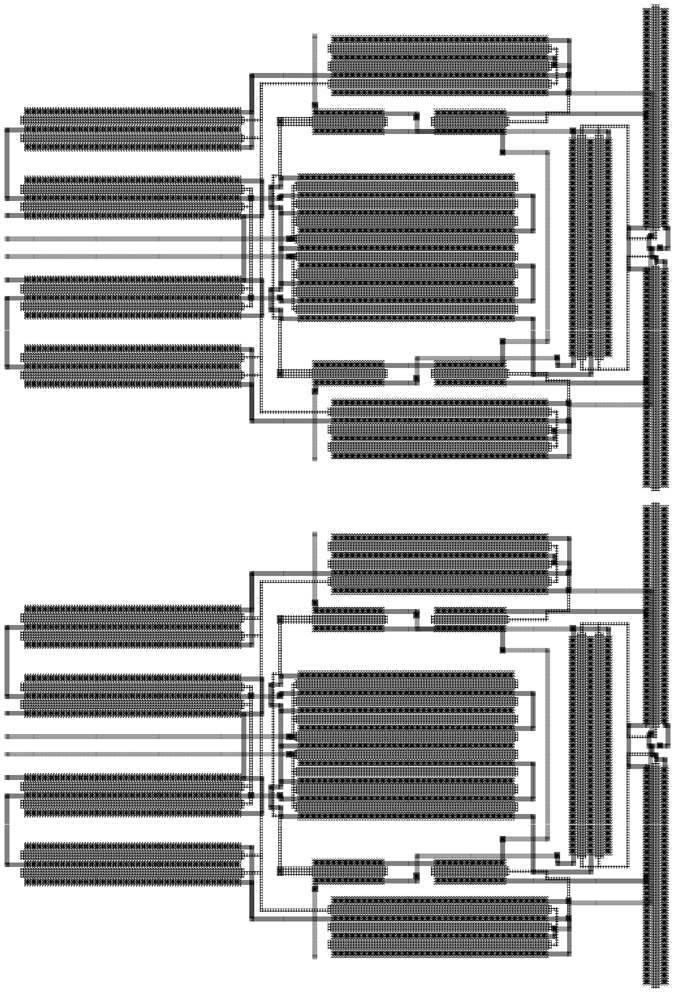

Figure 1: Hierarchically placed layout

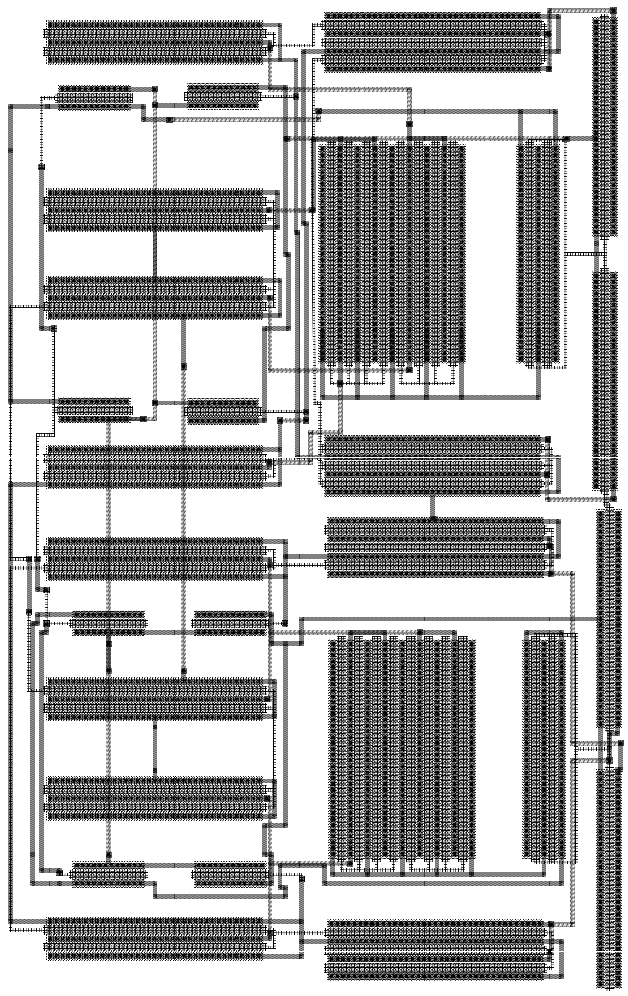

Figure 2: Layout with wide aspect ratio

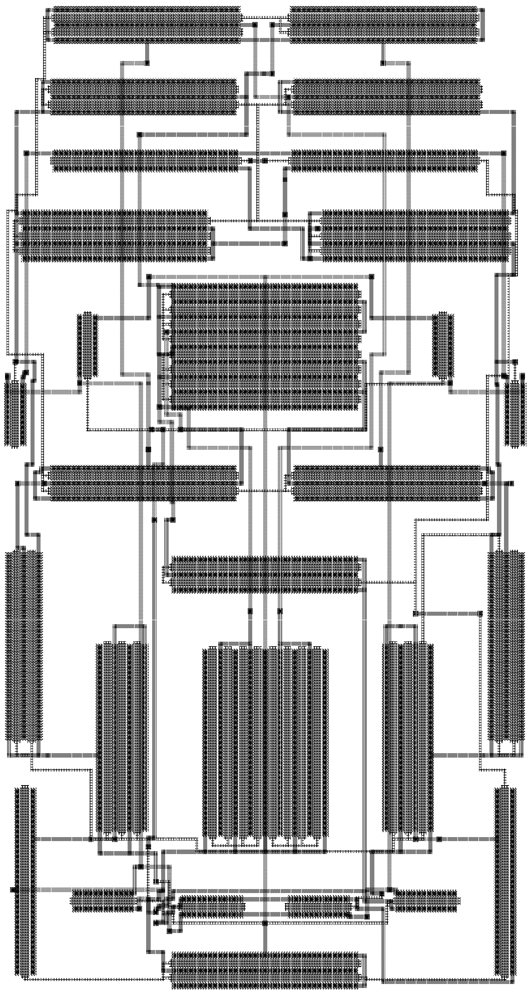

Figure 3: Layout with tall aspect ratio