A Routing Methodology for Analog Integrated Circuits

Enrico Malavasi † Umakanta Choudhury
Alberto Sangiovanni-Vincentelli

Department of Electrical Engineering and Computer Sciences
UC Berkeley, CA 94720

Abstract

A general methodology for the design of the interconnections of analog circuits respecting high-level constraints on performances is described in this work. In our approach, sensitivities of performances to parasitics are computed, and a set of bounding constraints for parasitics is determined. Sensitivities are then used to generate the weights for a new cost function-driven analog area router. After the routing is completed, the actual values of critical parasitics are used to check if the user-defined constraints on circuit performances are met. If the requirements have not been satisfied, the bounding constraints generated on the parasitics are used to increase the weights associated to the parasitics which violated the constraints, and the circuit rerouted. Results are reported validating the effectiveness of this approach for layout-design automation of analog circuits.

1 Introduction

Circuit performances are often sensitive to parasitics, such as line-to-line coupling capacitances and stray resistances, which introduce crosstalks and undesired voltage drops. Before the implementation of the layout these parameters are difficult to evaluate, since they depend heavily on layout configuration, in particular on interconnections. The layout design must be followed by a verification phase. If performance specifications have not been satisfied the layout must be re-designed. This gives rise to a large number of

†On leave from the "Dipartimento di Elettronica ed Informatica" of the University of Padova, Italy

time-consuming iterations. Hence the automation of analog layout synthesis is becoming an important issue.

The goal of analog CAD tools should not be simply to achieve a layout in minimum area, but also to meet the user-defined performance constraints. Digital-targeted tools in general are not suitable for performing this task. Some tools for the synthesis of analog layout have recently been developed. In [1], a standard-cell design style is proposed in which nets having critical coupling capacitances are routed in different channels; however, this approach is very restrictive. In [2], channel routing driven by bounds on capacitances is described, but the issue of obtaining these bounds is not addressed. In ILAC [3] emphasis is given to module generation, but performances are not directly considered while routing. The problem of analog routing is not specifically dealt with in [4], where a digital router [5] is used in a module generator for CMOS operational amplifiers. In ANAGRAM [6], a cost function in terms of routing parasitics is used, but with no explicit reference to performance specifications of the circuit. In all the previous approaches, performance constraints of a general analog circuit can not be handled directly.

Both placement and routing phases contribute significantly to introducing parasitics in the layout. Although the placement imposes minimum bounds on stray resistances and capacitances to ground, their final values are determined by routing. Moreover the routing phase is almost entirely responsible for line-to-line couplings. In this paper we present a general approach for the routing of analog circuits driven by user's constraints on circuit performances. Our framework is organized as shown in figure 1. A constraint generator generates performance sensitivities and a set of bounding constraints for the critical parasitics. Sensitivities are used to define a set of *weights* for an area router. The rôle of weights in area routing is to define a cost function driving the router in order to penalize the most critical parasitics. Next, a check phase follows the routing attempt, in which parasitics are extracted from the layout and the degradation of performances due to their existence is computed. If constraints on performances have been violated, weights are modified based on the knowledge of performance sensitivities and of the contribution of the associated parasitics to constraint violations, and the circuit is routed again with the new weight set. The procedure ends after a finite number of iterations. If still any constraint has been violated, it is possible to check the effective amount of the violation and the contribution of each parasitic to such violation.

In section 2 details of our framework are illustrated: the constraint generator and the analog area router are described. In section 3 the algorithms for the generation and modification of weights are presented. Some remarks about the iterative behavior are reported in section 4. An example showing the suitability of our approach is illustrated in section 5.

2 The Performance-Driven Routing Framework

The constraint generator PARCAR[7] computes the sensitivities of performance functions to all possible parasitics in the circuit and detects the critical parasitics. It also generates a set of bounds on critical parasitics to satisfy performance constraints while providing maximum flexibility to the router. In differential circuits, a set of matching constraints on parasitics based on matched-node-pair information is evaluated, and worst-case sensitivities are computed taking mismatch into account.

Sensitivities are used to generate the weights for the cost function driving the area router. Area routing is inherently driven by a cost function, and it may not be easy to directly incorporate bounding constraints into it. However, its flexibility makes its use possible with very complicate geometries, as it is often required in full custom design. The tool used in our framework is a maze router called ROAD, derived by the router described in [8] with a number of substantial improvements. Maze routing is particularly appropriate for circuits with intricate geometries, although it is applicable to relatively small systems, because of its high computational cost. But since most of the analog parts of complex systems are either small or deeply hierarchical, this approach is quite acceptable for a large variety of applications. The tool described in [8] was based on the Lee algorithm [9] [10] and controlled by a cost function defined on the edges of a relative grid with dynamic allocation. In ROAD, the Lee algorithm has been substituted with the A* algorithm [11,Ch.3][12] which uses a "depth-first" approach. The A* algorithm is based on a heuristic estimation of the cost of a path connecting a grid node and a target terminal. In ROAD such estimate is given by the manhattan distance between the node and the target. If they are on different layers the cost of a via is added. Such heuristic is always an optimistic estimate of the real path length and therefore the algorithm is admissible [11,Ch.3], [13,Ch.3].

An optimal path for a net is defined as the one minimizing the integral of the cost function along its path. In ROAD, the cost function is a weighted sum of a number of non-homogeneous items. Let $\mathcal{N}$ be the set of all nets. On a given grid edge x of length $L(x)$, associated to layer l, the cost function for a given net $N \in \mathcal{N}$ assumes the following expression:

$$\begin{aligned} F(x) = \quad & L(x) \cdot \Bigg(1 + \frac{Cr(x)}{Cr_0} + w_R \frac{R_u(l)}{R_0} + \\ & + w_{C_u} \frac{C_u(l)}{C_0} + \frac{1}{C_P} \sum_{n \in \mathcal{N} - \{N\}} w_{C_n} C_n(x) \Bigg) . \end{aligned} \tag{1}$$

$Cr(x)$ is a measure of local area crowding. $R_u(l)$ is the resistance of a minimum-width unit-length wire segment on layer l. $C_u(l)$ is the capacitance to bulk of a minimum-width unit-length wire segment on layer l. $C_n(x)$ is the capacitance between a unit-length wire segment located across edge x and the wire implementing net n. w_{R_u}, w_{C_u} and w_{C_n} are weights regulating the relative importance of each item. Cr_0, R_0, C_0, C_P are reference parameters providing dimensional homogeneousness to the addenda and a meaning to their comparison.

The effect of weights on wires is evident in the example shown in figure 2, a simple op-amp, where the coupling between the two nets indicated by arrows was given different weights, low in figure 2a, higher in figure 2b. In the first case, crossing could not be avoided; in the case of figure 2b, on the contrary, due to a higher cost function "hill" surrounding the first wire, a completely different path was chosen with no crossings.

A proper definition of weights can maximize the probability that the most critical parasitics be kept as small as possible. By determining the relative importance of critical parasitics, weights provide a flexible way to limit their magnitude, although this does not insure that such parasitics don't exceed their bounds.

In ROAD, an efficient scheduler sorts the nets to be routed on the ground of a priority list. The priority of a given net N is given by a weighted sum of the cost function weights for that net. The nets with the tightest electrical requirements are routed first. Additional contributions for symmetric nets are added, such that symmetric nets are also given high priority.

After performing the weight-driven routing, the actual parasitics are evaluated and the performance degradation is estimated and compared with the given constraints.

3 Weight Generation and Modification

A list of notations and definitions used throughout this section follows:

N_c	The number of performances
N_p	The number of critical parasitics
K_i	The i-th performance, for $i = 1, \ldots, N_c$
$\triangle K_i^-, \triangle K_i^+$	Respectively the maximum negative and positive variation allowed on performance K_i
p_j	The j-th critical parasitic, for $j = 1, \ldots, N_p$
$p_j^{(b)}$	The bounding constraint on parasitic p_j
$S_{i,j}$	The sensitivity of performance K_i to parasitic p_j
$S_{i,j}^-$	$= -S_{i,j}$ if $S_{i,j} < 0$, 0 otherwise
$S_{i,j}^+$	$= S_{i,j}$ if $S_{i,j} > 0$, 0 otherwise

Weights express the criticality of the contribution of parasitics to performances. If a performance function is constrained in the positive (negative) direction, then each parasitic having nonzero $S_{i,j}^+$ ($S_{i,j}^-$) will degrade that performance. Each performance function contributes to the criticality of a parasitic. Such contribution is proportional to the associated sensitivity and inversely proportional to the maximum variation range allowed to that performance. Since $S_{i,j}^-, S_{i,j}^+ \geq 0$, this contribution is always non-negative.

The weight w_j associated to the j-th parasitic p_j is defined as follows:

$$w_j = w_0 \sum_{i=1}^{N_c} \left(\frac{S_{i,j}^-}{\triangle K_i^-} + \frac{S_{i,j}^+}{\triangle K_i^+} \right) P_0 \qquad (2)$$

where w_0 is a normalization factor, such that, if sensitivities are not all zero, at least one weight is set to 1, and the others are all in $[0, 1]$. p_0 is a dimensional parameter equal to R_0, C_0 or C_P, depending on the type of parasitic. With this definition, the parasitic items in (1) can be interpreted as a linear approximation of the performance degradation due to the wire element on x. After routing, a check over constraint satisfaction is carried out considering the expressions:

$$\overline{\triangle K_i^-} = \sum_{j=1}^{N_p} (S_{i,j}^- \cdot p_j) \ , \qquad \overline{\triangle K_i^+} = \sum_{j=1}^{N_p} (S_{i,j}^+ \cdot p_j)$$

and comparing them with the performance ranges specified by the user. If for each performance K_i,

$$\overline{\triangle K_i^-} \leq \triangle K_i^- \quad \text{and} \quad \overline{\triangle K_i^+} \leq \triangle K_i^+,$$

the specifications have been satisfied, otherwise it is necessary to modify the weights and repeat a part or all of the routing. Linear approximations for sensitivities are acceptable, because our goal is to keep each performance within a small tolerance specified by the user.

The way parasitic bounding constraints are generated is such that if the value of each critical parasitic remains within its bound, then all the performance variations remain within their respective bounds too. Therefore, if a performance constraint is violated, at least one parasitic must have exceeded its bound as well. Hence, one possible approach to modify automatically the weights is to increase the ones associated to the parasitics whose values resulted larger than the bounding constraints and to decrease the ones associated to the parasitics whose values were smaller. But with such an approach weights could oscillate indefinitely through iterations. In our approach, only weight increases are allowed. The weights of parasitics whose values remained under the specified bounds are not modified. Weights are increased based on the relative contributions of the violations of parasitic constraints to the violations of performance constraints.

Weight modifications must be carried out with care: if an increase is too small it does not significantly affect the router behavior, and the procedure may require several iterations to achieve a satisfactory result; if on the contrary it is too large, it decreases the relative importance of the other parasitics, and eventually lets some other critical parameter rise.

Let K_i be a performance whose constraints were violated, that is $\triangle(\triangle K_i^-) = \overline{\triangle K_i^-} - \triangle K_i^- > 0$ or $\triangle(\triangle K_i^+) = \overline{\triangle K_i^+} - \triangle K_i^+ > 0$. Let p_j be a parasitic whose value exceeded its bounding constraint, that is $\triangle p_j = p_j - p_j^{(b)} > 0$.

If its weight w_j is already 1, no increase is possible, therefore the procedure ends with an overconstraint condition. All performances whose constraints were not satisfied and having nonzero sensitivity to the j-th parasitic are indicated as non-feasible. However, if $w_j < 1$, it is increased by an amount proportional to the relative contribution of $\triangle p_j$ to the total violation in performance constraints:

$$w_j^{(new)} = w_j + \delta w_j \tag{3}$$

where $\delta w_j = \dfrac{(1-w_j)}{\lambda_0} \left(\sum_i \dfrac{S_{i,j}^-}{\triangle(\triangle K_i^-)} + \sum_i \dfrac{S_{i,j}^+}{\triangle(\triangle K_i^+)} \right) \triangle p_j.$

The two sums are extended to the performances which exceeded respectively their lower and upper bound. λ_0 is a normalization factor such that at least one of the increased weights, the one associated to the parasitic which gave the most important contribution to the constraint violation, is brought to 1, while the others $w_j \in [0,1], \ \forall j$.

4 Priority Factors and Iteration Control

It is easy to show that the iterative approach based on weight modification terminates. In fact the number of steps is necessarily finite, because at each iteration at least one of the weights is brought to its maximum value of 1 due to (2) and (3) and is not changed any more.

The router can be regarded as an operator transforming a vector of weights into a vector of parasitics; but the relationship between weights and parasitics is so complicate that it can not be defined formally. Weights are used to determine the routing schedule and cost functions, but the final values of parasitics depend upon the whole layout structure. We have adopted the following representation of parasitic dependence on weights. If a weight has a small value, its effect on the cost function becomes negligible compared to considerations on congestion and area minimization. If, on the contrary, it assumes a significantly high value, its rôle becomes important and its contribution to priority in scheduling makes the satisfaction of the related constraints more likely.

The contribution of a weight to the definition of the value of the associated parasitic can be expressed by means of the *priority factor*, given by the following expression:

$$\pi_j = \frac{w_j}{\sum_{i=1}^{N_p} c_{i,j} w_i} \tag{4}$$

where the $c_{i,j}$ are correlation factors, depending on the distribution of terminals for the nets related to parasitics p_i and p_j. The priority factor π_j of a given parasitic p_j assumes a value close to 1 when w_j is significantly large while the weights of other correlated parasitics are all negligible. In this case, the nets related to p_j will be given the highest priority and the corresponding cost functions will be mainly concerned with the minimization of p_j. Therefore the probability of keeping p_j near its minimum possible value is close to 1. When the weight for a parasitic is small, or several correlated parasitics are given similar weights, little control can be exercised over the effective value of the parasitic. In this case the priority of each of the correlated parasitics may vary over a more or less wide range.

On the ground of these considerations, priority factors can be usefully interpreted as the probability that parasitics have, at the end of the routing, to assume values close to their minimum.

All priority factors are modified by weight updating. The ones corresponding to parasitics exceeding their bounds are increased, the others are decreased. By lowering the priority factors we provide a compensation for the parasitics whose values did not exceed the specified bounds and whose weights were not modified.

5 Results

The algorithm described above was implemented in the UNIX environment using the C language and tested on a number of circuits. The op-amp shown in figure 3 was routed with our approach, using different performance specifications on closed loop gain, band-width and phase margin.
Table 1 reports the results for two sets of different specifications on such op-amp. With relatively relaxed constraints on the maximum degradation, the first routing phase satisfied all the requirements.

With a set of tighter requirements on performances, two of the constraints (band-width and phase margin) were not satisfied after the first routing attempt. An analysis of the extracted parasitics showed that all critical, heavily weighted parameters assumed very small values compared to their bounding constraints. The requirements on performances were not satisfied due to a few less critical parasitics which were given very low weights (under 0.2) and were consequently badly controlled by the cost function. After weight redefinition, the whole circuit was re-routed. Two routing phases were sufficient to bring all constraints within the required bounds. Figure 4 displays the final layout of the routed circuit. Arrows indicate two wires implementing critical nets, whose coupling was limited by means of proper spacing. An alternative solution would be to introduce a shielding wire saving some area between them. However this would require an additional interconnection between the shield and an appropriate net in the circuit.

	$\triangle$ C.L.G.	$\triangle$ P.M.	$\triangle$ B.W.
Relaxed constraints	10^{-4} (0.01%)	0.5° (1%)	100 KHz (1%)
After 1 routing phase	0	0.069°	17.5 KHz
Tight constraints	10^{-4} (0.01%)	0.05° (0.1%)	10 KHz (0.1%)
After 2 routing phases	0	0.037°	4.5 KHz

Table 1 *Performance degradations with different sets of constraints. Closed Loop Gain, Phase margin and Band Width are considered. Constraints express the maximum degradation allowed for such performances.*

6 Conclusions and future work

An original approach for routing analog VLSI circuits respecting user's performance constraints has been presented. Using a cost-function based area router it is possible to address the problem of constraint-driven design for analog circuits, characterized by a strong emphasis on the control over parasitics, and to obtain a solution matching a set of high-level specifications. In this way, the designer is not concerned with low-level details regarding the layout. The router takes care of such details, and in case the requirements are not satisfied, detection of overconstraints and of the critical parasitics responsible for them is automatically handled.

Actual experience using our approach has shown that its efficiency in handling constraints characterized by difficult trade-offs is very acceptable, although convergence to a feasible solution, if one exists, can not be proved.

More work has still to be done in the field of high-performance analog design. In particular, the constraint-driven placement of analog modules is an important issue. The possibility of automatic placement modification in case of failure in constraint satisfaction should be considered. Finally, more accurate and computationally efficient models should be developed for estimating parasitics.

7 Acknowledgements

This research has been partially supported by DARPA. The first author has been supported by "Progetto Finalizzato MADESS" of the Italian National Council of Research.

References

1. C.D.Kimble et al. "Autorouted Analog VLSI". *Proc. CICC*, 1985.

2. R.S.Gyurcsik and J.-C. Jeen. “A Generalized Approach to Routing Mixed Analog and Digital Signal Nets in a Channel”. *IEEE JSSC*, 24, N.2:436–442, April 1989.

3. J. Rijmenants et al., “ILAC: An Automated Layout Tool for Analog CMOS Circuits”. *Proc. IEEE CICC*, pages 761–764, May 1988.

4. H. Y. Koh, C. H. Séquin, and P. R. Gray, “Automatic Layout Generation for CMOS Operational Amplifiers”. *Proc. IEEE ICCAD*, pages 548–551, November 1988.

5. H. Shin and A. Sangiovanni-Vincentelli, “A Detailed Router Based on Incremental Routing Modifications: Mighty”. *IEEE Trans. on CAD*, CAD-6, N.6:942–955, November 1987.

6. D. J. Garrod, R. A. Rutenbar, and L. R. Carley, “Automatic Layout of Custom Analog Cells in ANAGRAM”. *Proc. IEEE ICCAD*, pages 544–547, November 1988.

7. U. Choudhury and A. Sangiovanni-Vincentelli, “Constraint Generation for Routing Analog Circuits”. *Proc. DAC*, June 1990.

8. E. Malavasi, M. Chilanti, and R. Guerrieri, “A General Router for Analog Layout”. *Proc. COMPEURO '89, Hamburg*, pages 549–551, May 1989.

9. C. Lee, “An algorithm for path connections and applications”. *IRE Trans. Electron. Computer*, EC-10:346–365, September 1961.

10. T. Ohtsuki, “Maze-Running and Line-Search Algorithms”. *Layout Design and Verification*, chapter 3, pages 99–131. T.Ohtsuki Ed.North Holland, 1986.

11. N.J.Nilsson, *Problem-Solving Methods in Artificial Intelligence.* McGraw-Hill, 1971.

12. G.W.Clow, “A Global Routing Algorithm for General Cells”. *Proc. 21st DAC*, pages 45–51, 1984.

13. J.Pearl, *Heuristics: intelligent search strategies for computer problem solving.* Addison Wesley, 1984.

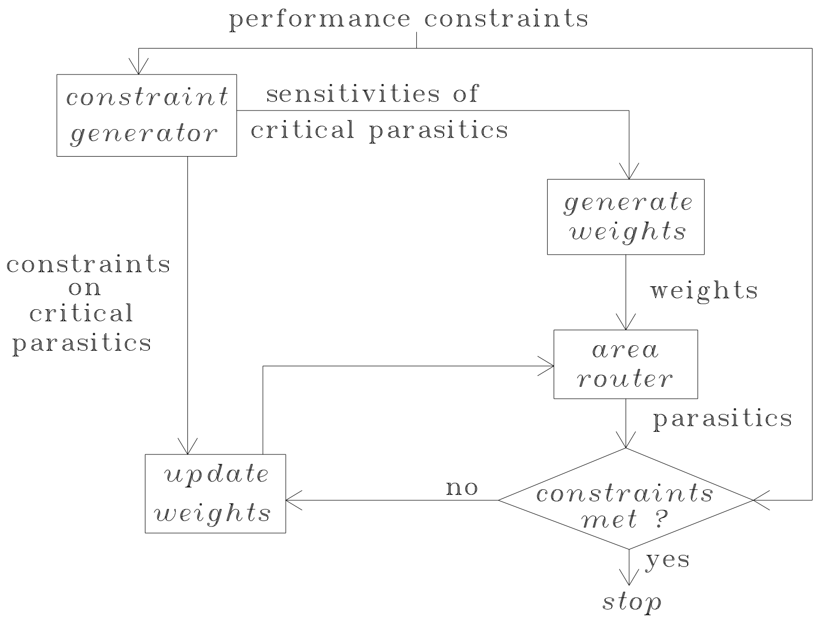

Figure 1 *The general organization of the approach for performance-driven routing of analog circuits.*

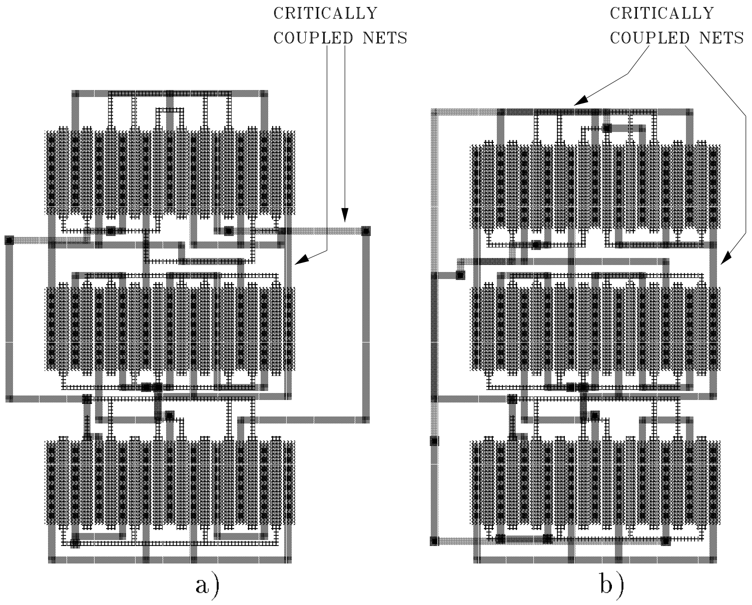

Figure 2 *Different coupling weights yield quite different results. In the case of figure 2a, crossing could not be avoided. In figure 2b, all crossings were avoided, choosing a longer path for one of the wires.*

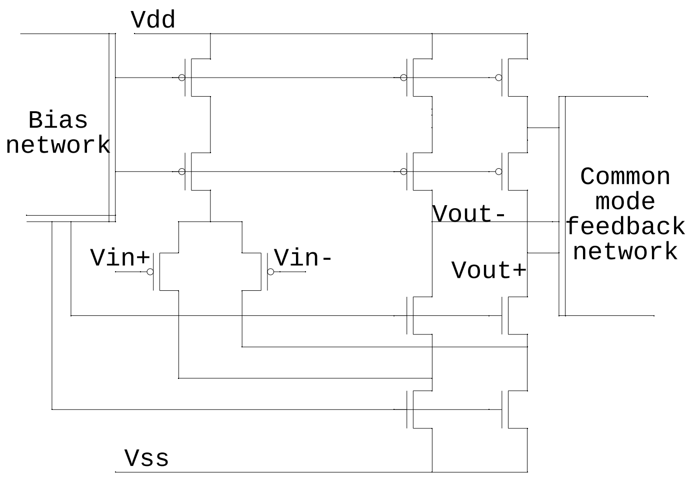

Figure 3 *Op-amp Schematic*

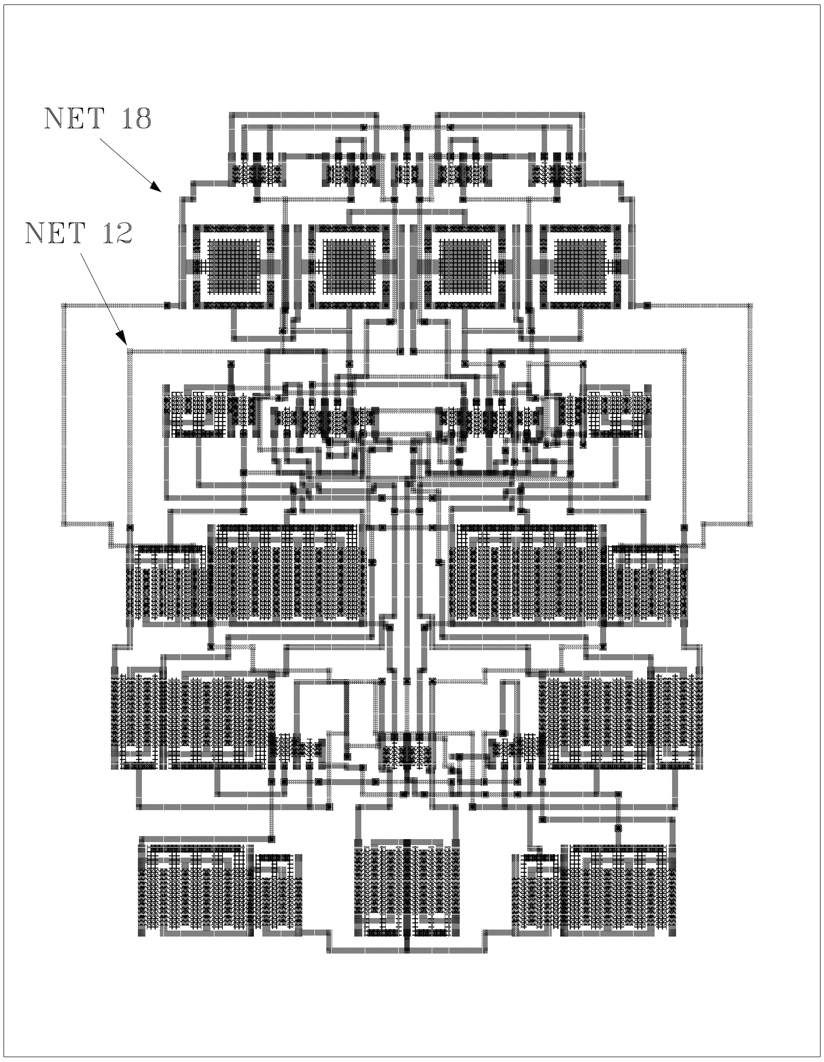

Figure 4 *Final layout of the op-amp, routed after weight redefinition.*