Module Placement for Analog Layout Using the Sequence-Pair Representation

Florin Balasa Koen Lampaert
Conexant Systems, Newport Beach, CA 92660
E-mail: florin.balasa@conexant.com

Abstract

This paper addresses the problem of device-level placement for analog layout. Different from most of the existent approaches employing basically simulated annealing optimization algorithms operating on flat Gellat-Jepsen spatial representations [2], we are using a more recent topological representation called *sequence-pair* [7], which has the advantage of not being restricted to slicing floorplan topologies. In this paper, we are explaining how specific features essential to analog placement, as the ability to deal with symmetry and device matching constraints, can be easily handled by employing the sequence-pair representation. Several analog examples substantiate the effectiveness of our placement tool, which is already in use in an industrial environment.

1 Introduction

In order to automatically produce analog device-level layouts matching in density and performance the high-quality manual layouts, a placement tool must not only provide a good *rectangle packing* functionality (which must be common to any placement method) but, additionally, it must include also analog-specific capabilities. Such specific features are, for instance, (1) the ability to deal with topological constraints for symmetry and device matching; (2) the ability to arrange devices such that critical structures are shared in common (also known as *device merging*) in order to reduce both layout density and induced parasitics; (3) the existence of a (built-in) library of predefined module generators and the ability to exploit their reshaping capabilities during the placement process.

1.1 Brief overview of placement methods

The constructive placement techniques, which consist in evolving gradually the placement solution by selecting one module at a time and positioning it in the "best" available location, were among the first developed for VLSI layout. Several systems for analog placement employ constructive methods: Kayal *et al.* developed an expert knowledge base to guide the placement [3]; Mehranfar suggested a schematic-driven approach, using a constructive scheme based on connectivity and relative positioning in the input schematic [8]. Although these methods are fast, scaling well with the problem size, the results can be poor due to the order dependence, lacking of global view in dealing with a variety of interacting quality measures.

Branch-and-bound placement techniques use a controlled enumeration of all possible layout configurations in the search space, where a lower bound of the chosen cost function is used to prune the search. The branch-and-bound algorithms eventually find the optimal solution as they explore exhaustively the search space. However, they are effective only for problems of very small size (the manageable number of blocks in [9] is 6), as the number of visited configurations grows exponentially with the size of the problem. The related ILP placement models [13] suffer the same scaling drawback (see also Section 6), as most ILP packages are based on branch-and-bound approaches. Even if the placement problems are tackled hierarchically, the branch-and-bound methods are less attractive for analog device placement due to usually a much larger search space than digital problems of similar size (for instance, due to the presence of "soft" capacitors which can be implemented in a large number of versions).

For the time being, the simulated annealing (e.g., [2]) and genetic algorithms are the most effective choice for solving industrial analog placement problems. These algorithms use stochastically controlled hill-climbing to avoid local minima during the optimization process. In addition, they do not impose severe constraints on the size of the problems or on the mathematical properties of the cost function. While efficiently trading-off between a variety of layout factors as area, total net length, aspect ratio, maximum chip width and/or height, cell orientation, "soft" cell shape, etc., they are very flexible – supporting incremental addition of new functionality, and they are relatively easy to implement (although good tunning needs more time). This is why simulated annealing, the most mature of the stochastic techniques, provided the engine for effective tools both in digital (TimberWolfSC v7.0 [12]) and analog design: ILAC [11], KOAN/ANAGRAM II [1], LAYLA [4], PUPPY-A [6].

1.2 Motivation of the research

A simulated annealing algorithm can equally operate with two distinct spatial representations of placement configurations. The first is the so-called *flat* representation introduced by Jepsen and Gellat [2], where the cells are specified in terms of *absolute* coordinates on a gridless plane. The moves are simple coordinate shifts or changes in cell orientation. Cells are allowed to overlap in possibly illegal

ways[1], as no restriction is made referring to the relative position of a cell with respect to another cell. A (weighted) penalty cost term is associated with infeasible overlaps, and this penalty must be driven to zero in the optimization process. The flat representation is well-suited to handle device matching and symmetry constraints, typical to analog layout; it also allows to explore the beneficial device overlaps. For these reasons, the flat representation was the choice for KOAN/ANAGRAM II [1], LAYLA [4], and PUPPPY-A [6] systems.

However, this representation has also a drawback revealed, for instance, in [12]. Due to the complexity of the cost function, the total (infeasible) overlap in the final placement solution is not necessarily equal to zero: a final step eliminating the gaps and overlaps must be performed, degrading the computation time and the solution optimality (in terms of the cost function). Moreover, the weight of the overlap term must be carefully chosen: if it is too small, the cells may have the tendency to collapse; if it is too large, the search ability of the annealer for a good placement (in terms of area, total net length, etc.) may be impeded. To combat this effect, an earlier version of the TimberWolf system [12] used a sophisticated negative control scheme to determine the optimum values of the cost term weights.

The second placement representation uses a *topological* or a graph-based model. The most popular is the slicing model, first introduced by Otten [10]. In contrast to the flat representation, cell positions are specified *relatively*, based on the topological relations between them. The cells are organized in a set of slices which recursively bisect the layout horizontally and vertically. The direction and nesting of the slices is recorded in a slicing tree or, equivalently, in a normalized Polish expression [14]. The annealing algorithm does not move the cells explicitly: it rather alters their relative positions, modifying the slicing tree or Polish expression. In this representation, cells cannot overlap, which may lead to an improved efficiency in the placement optimization.

However, the slicing representation limits the set of reachable layout topologies. This can degrade layout density, especially when cells may be very different in size, which is often the case in analog layout. Furthermore, symmetry and matching constraints are difficult to maintain: for instance, slicing style placement tools have to implement symmetry constraints in the cost function through the use of virtual symmetry axes [5], which is a less efficient solution. Although the ILAC system [11] employs this model, it is widely acknowledged that slicing placement is not a good choice for high-performance analog design.

More recently, a novel topological representation – called *sequence-pair* – has been proposed by Murata *et al.* [7]. This representation is not restricted to slicing floorplan topologies. The goal of our paper is to show that the sequence-pair representation is adequate for high-performance analog layout, as symmetry and device matching constraints can be easily handled. In this way, the superior efficiency of the topological representations is combined with the completeness of the search space of placement configurations and the ability to handle typical analog constraints. The results obtained so far with our placement tool, already active in an industrial environment, will substantiate the validity of these claims.

This paper is organized as follows. Section 2 will briefly describe the sequence-pair representation. Section 3 will explain the nature of the symmetry constraints in analog design, while Section 4 will thoroughly analyze the symmetry handling during simulated annealing, when this optimization operates on sequence-pair representations. Section 5 briefly reviews the device-matching constraints and Section 6 proves the effectiveness of our analog placement solution. Then, Section 7 concludes with final remarks.

[1] In analog layout, cells can overlap not only in *legal* but also *beneficial* ways ("device merging" or "geometry sharing" [1]).

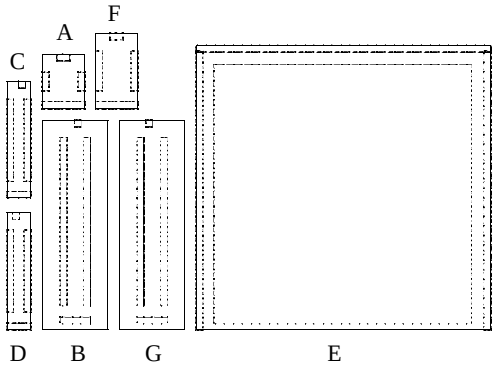

Figure 1: Placement configuration encoded by the sequence-pair $(CDAFBGE, DCBGAFE)$

2 The sequence-pair representation

The basic idea of the sequence-pair representation, briefly described below for the sake of consistency, is to encode any rectangle packing as an ordered pair of cell sequences (α, β) . In [7] a method of deriving such an encoding is informally described[2]. In particular, there exists at least a sequence-pair encoding corresponding to (one of) the optimal rectangle packing[3]. In addition, all the encodings are feasible in the sense that a placement configuration can be derived from any encoding; moreover, the solution is optimal in terms of area and it can be constructed in $O(m^2)$ time, where m is the number of placeable cells. As the total number of encodings is finite although large[4], the solution space can be effectively explored employing simulated annealing or genetic algorithms.

Denoting by α_i the cell occupying the position i in sequence α, and by α_A^{-1} the position of the cell A in the sequence[5], the topological relations between two cells A and B are given by: if $\alpha_A^{-1} < \alpha_B^{-1}$ and $\beta_A^{-1} < \beta_B^{-1}$ then cell A is to the left of cell B; if $\alpha_A^{-1} < \alpha_B^{-1}$ and $\beta_B^{-1} < \beta_A^{-1}$ then cell A is above cell B.

Example The sequence-pair encoding of the 7 cell placement configuration in Fig. 1 is $(\alpha, \beta) = (CDAFBGE, DCBGAFE)$ (see Section 4). As $\alpha_F^{-1} < \alpha_B^{-1}$ and $\beta_B^{-1} < \beta_F^{-1}$ $(4 < 5$ and $3 < 6)$, it follows that cell F is positioned above cell B.

Murata *et al.* have conceived the sequence-pair representation as a general rectangle packing method [7]. In order to aply this topological representation to analog placement, handling symmetry is an essential requirement. Section 4 will show how to handle the symmetry constraints within the sequence-pair topological representation.

[2] A procedure which builds the encoding (α, β) from the configuration of m placeable cells in $O(m^2)$ time is also given in Section 4.

[3] This property is not valid for the normalized Polish expressions [14] encoding the slicing structures, as the optimal (in terms of area) packing may not have a slicing topology.

[4] The search space for m fixed-oriented "hard" cells has the size $(m!)^2$. If the cells can be rotated, or rotated and mirrored, the size is $(m!)^2 4^m$ or $(m!)^2 8^m$, respectively.

[5] α and β are one-to-one mappings and, therefore, the inverse mappings are well-defined.

3 Symmetry constraints in analog layout

In high-performance analog circuits it is often required that groups of devices are placed symmetrically with respect to one or several (vertical) axes. The main reason of symmetric placement and routing is to match the layout-induced parasitics in the two halves of a group of devices. Failure to match these parasitics in, for instance, differential analog circuits can lead to higher offset voltages and degraded power-supply rejection ratio [1]. Placement symmetry can also be used to reduce the circuit sensitivity to thermal gradients. Failure to adequately balance thermal couplings in a differential circuit can even introduce unwanted oscillations. In order to combat potentially-induced mismatches, the thermally-sensitive device couples should be placed symmetrically relative to the thermally-radiating devices.

Usually, the analog circuits have a mix of symmetric and asymmetric components. The typical forms of symmetry which should be handled by an analog placement tool are [1]: (1) mirror symmetry – which consists in placing a symmetry group of cells about a common axis such that the cells in every pair have identical geometry and mirror-symmetric orientation; (2) perfect symmetry – which differs from the previous by the identical (rather than mirror-symmetric) orientations of the paired devices; (3) self-symmetry – characteristic for devices presenting a geometrical symmetry and sharing the same axis with other pairs of symmetric devices.

4 Handling symmetry constraints with the sequence-pair representation

Assuming that a given subset of the placeable cells must constitute a symmetry group, not all the sequence-pair codes are feasible any more. For instance, suppose the cell couple (C, D) in the Section 2 example should be symmetric relative to a vertical axis: the encoding $(\alpha, \beta) = (CDAFBGE, DCBGAFE)$ is not feasible as it leads to a placement configuration where cell C lays above D.

At a first glance, one would be tempted to perform minor changes to the search space exploration: if the current encoding proves to be consistent with the symmetry constraints then the cost of the placement configuration is evaluated and the annealing algorithm operates normally; otherwise, the current encoding is infeasible (in symmetry point of view) and, therefore, disregarded. Unfortunately, such a simple solution is not effective: taking into account that the size of the search space without symmetry constraints is $(m!)^2$ (the total number of sequence-pairs), the size of the solution space becomes significantly smaller if the placement configuration must contain a symmetry group. Indeed, the size of this new search space is given by the formula (which, due to lack of space, will be proven elsewhere):

$$\left[C_m^{m-2p-s}(m-2p-s)!\right]^2 \cdot N_{p,s} \qquad (1)$$

where p is the number of symmetric pairs, and s – the number of self-symmetric cells in the group. $N_{p,s}$ can be computed recurrently as follows:

$$N_{p,s} \;=\; 6p \cdot N_{p-1,s} \;+\; 2s \cdot N_{p,s-1}$$

where $N_{p,0} = 6^p p!/3$, $p \geq 1$ and $N_{0,s} = s!$, $s \geq 0$.

Formula (1) shows that the size of the search space is $(m!)^2/24$ if $(p = 2,\ s = 0)$. Therefore, when there are only two pairs of symmetric cells more than 95.83% of the full sequence-pair search space contains symmetric-infeasible solutions. This has been confirmed experimentally when trying to place the cells in Fig. 1 such that the pairs of cells (C, D) and (B, G) are respectively symmetric relative to a common axis: the CPU time was several times higher and, in addition, the final solution was poor because the number of feasible codes investigated during the simulated annealing was insufficient (most of the codes being rejected).

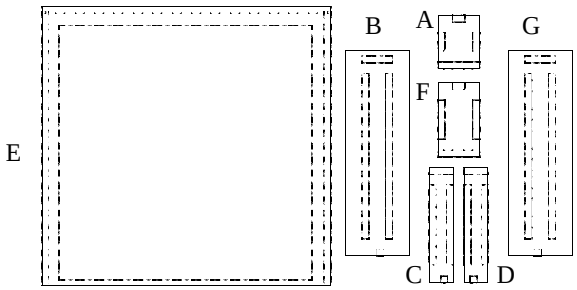

Figure 2: Placement configuration with symmetry group $\gamma = ((C, D)\ ,\ (B, G)\ ,\ A\ ,\ F)$

A better strategy is to explore *only* those sequence-pairs which comply with the symmetry constraints. This section will show (a) how to recognize such sequence-pairs, and (b) how to efficiently restrict the annealer exploration only to their subspace.

Let (α, β) be the sequence-pair of a placement configuration containing a symmetry group γ composed of several pairs of (mirrored/perfect) symmetric cells and self-symmetric cells relative to a common vertical axis[6]. We denote by $sym(x)$ the symmetric of cell x, and we consider by convention that $sym(x) \equiv x$ when x is a self-symmetric cell. The sequence-pair (α, β) is called *symmetric-feasible* if for any distinct cells x, y in γ :

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

Choosing $y = sym(x)$ and taking into account that $sym(sym(x)) = x$, condition (S) shows that any symmetric pair of cells appears in the same order in both sequences α and β. At the same time, the cells belonging to distinct symmetric pairs appear in reversed order in the two sequences of the encoding.

Example(cont'd) Assumming there is a symmetry group $\gamma = (\ (C, D)\ ,\ (B, G)\ ,\ A,\ F)$ composed of two symmetric pairs and two self-symmetric cells A and F, the encoding $(\alpha, \beta) = (CDAFBGE, DCBGAFE)$ is not symmetric-feasible: the (S) condition is not valid for any of the pairs of cells $(C, D), (C, B), (C, G), (C, A), (C, F), (D, B), (D, G), (D, A), (D, F), (A, F)$. On the other hand, the encoding $(\alpha, \beta) = (EBAFCDG, EBCDFAG)$ is symmetric-feasible; it has been derived from the placement solution in Fig. 2.

Lemma 1 Any placement configuration containing a symmetry group can be encoded with a symmetric-feasible sequence-pair.

Proof: According to [7], any placement configuration can be encoded with a sequence-pair. In particular, this is true also for those configurations containing symmetry groups. If the encoding is done with the procedure described below, then the resulting sequence-pair has the property (S) when applied to a placement configuration with a symmetry group:

sequence ConstructSequenceAlpha(m non-overlapping cells)
{ // a similar procedure builds sequence β
sequence α (resp., β) is initially empty (zeroed);
for $k = 1$ to m { // k is the crt. position in sequence

[6] The case of multiple symmetry groups can be similarly approached, as it will be shown further.

```
    let j be a cell not yet in sequence α (resp., β);
    for (every cell i (≠ j) not in the sequence) {
      if the horizontal projections of cells i and j overlap
      then if cell i is above (resp., below) cell j then j = i
      else if cell i is to the left of cell j then j = i;
    }
  } α_k = j; (resp., β_k = j)
} return sequence α (resp., β);
```

It can be easily verified by analyzing all the possible relative positions between two pairs of symmetric cells (e.g., the second pair between the cells of the first pair, and so on), or two self-symmetric cells, or one symmetric pair and one self-symmetric cell, that the encoding generated as described above satisfies condition (S).□

Applying the encoding procedure to the illustrative placement configurations in Fig. 1 and Fig. 2, the sequence-pairs $(\alpha,\beta) = (CDAFBGE, DCBGAFE)$ and, respectively, $(\alpha,\beta) = (EBAFCDG, EBCDFAG)$ are obtained. The latter encoding is symmetric-feasible.

Lemma 2 Given a set of placeable cells containing a symmetry group and a symmetric-feasible sequence-pair, then one can build in polynomial time an optimal placement configuration (in terms of area) satisfying the positioning and symmetry constraints.

Proof: Denoting x_i, y_i the coordinates of the left-bottom corner of cell i (i=1,...,m) of width $width_i$ and height $height_i$, and given a symmetric-feasible sequence-pair (α,β), a construction with the properties stated in *Lemma 2* is described below.

First, the x coordinates of the cells are computed such that the positioning constraints (compatible with the given sequence-pair) are satisfied:

```
initialize x_i = 0 (i = 1,m) and symAxis = 0;
for i = 1 to m {
  j = α_i; // choose cell j having position i in seq. α
  for l = i+1 to m {
    k = α_l; // for cells k positioned after j in seq. α
    if β_j^-1 < β_k^-1 // if k is positioned after j also in β
    then x_k = x_k max (x_j + width_j)
  } // then k is to the right of cell j
  if cell j has a symmetric cell k such that α_k^-1 ≤ i
     // thus cell k is to the left of j, or j is self-symmetric
  then symAxis = symAxis max ((x_k + width_k) + x_j)/2;
}
```

This part of the algorithm determines in $O(m^2)$ time the x coordinates of the cells such that the horizontal positioning constraints resulting from the sequence-pair (α,β) are satisfied. In the absence of a symmetry group, the computation of the x coordinates is over; otherwise, the x coordinates must be trimmed in order to satisfy also the conditions of symmetry. This is done in two steps: first, a "sweep to the right" which finds the final x positions of the cells to the right of the symmetry axis, and also of the self-symmetric ones; second, a "sweep to the left" which determines the x positions of the cells to the left of the symmetry axis.

```
for i = 1 to m {   // the sweep to the right
  j = α_i; // choose cell j having position i in seq. α
  if cell j has a symmetric cell k such that α_k^-1 ≤ i then
  { // if cell k is to the left of j, or j is self-symmetric
    d = 2 · symAxis − (x_k + width_k) − x_j;
    if k = j then d = d/2; // if j is self-symmetric
    if d > 0 then {
      x_j = x_j + d; // push cell j to the right
      for l = i+1 to m {
        k = α_l; // all cells k to the right of j ...
        if β_j^-1 < β_k^-1 then x_k = x_k + d;
      } //                ... are equally pushed to the right
    }
  }
}
```

The "sweep to the left" works according to a similar scheme. It must be noticed that the symmetric-feasibility condition is sufficient in order to ensure the correct x-positioning after only two sweeps – one to the right and one to the left. If the sequence-pair (α,β) does not satisfy the (S) condition, it could happen that two pairs of symmetric cells $(x\ ,\ sym(x))$ and $(y\ ,\ sym(y))$ may prevent each other from being simultaneously symmetric about the axis if, for instance, x is pushing y to the right when achieving the symmetry of the first pair, and afterwards $sym(y)$ is pushing $sym(x)$ to the left achieving the symmetry of $(y\ ,\ sym(y))$ but destroying again the symmetry of $(x\ ,\ sym(x))$. Moreover, the complexity of the algorithm is still $O(m^2)$, the same as in the absence of any symmetry constraint.

Finally, the y coordinates of the cells are computed in the inverse order of sequence α, taking also into account that symmetric couples have equal y coordinates.

A placement configuration built as described above has a minimum width and height while satisfying the constraints. Indeed, employing the graphs of horizontal and vertical constraints introduced in [7], one can show in a similar way that the width and height of the chip is determined by the longest paths length between source and sink in the two graphs which are minimal under the constraints. The only difference is that the weights of the edges in the two graphs must take into account also the constraints of symmetry.□

The two lemma's presented above show that, in order to find an optimal cell packing containing a symmetry group, it is not necessary to explore the entire space of $(m!)^2$ sequence-pairs; it suffices to explore the significantly smallerspace of *symmetric-feasible* sequence-pairs. The annealing algorithm can be easily adapted to explore the space of symmetric-feasible sequence-pairs taking the following caveats:

(1) The initial sequence-pair must be symmetric-feasible. Such a sequence-pair corresponds, for instance, to a configuration where the pairs of symmetric cells are disposed in line, like several embedded brackets, surrounding the self-symmetric cells which are disposed one on the top of the other.

(2) The move-set of the annealer must be selected such that the property of symmetric-feasibility is preserved for all the visited sequence-pairs. The modifications of cell positions or cell interchanges in sequence α and/or β are valid moves for the cells outside the symmetry group. However, if two cells from symmetric couples are interchanged in sequence α, then their symmetric counterparts must be also interchanged in sequence β.

The requirement of several symmetry groups – that is, groups of cells having distinct symmetry axes – can be modeled in a similar way: the cells within every group must comply with the (S) condition; in addition, the construction of the placement solution from a given sequence-pair must be refined, along with the move-set of the annealing algorithm.

5 Handling device-matching constraints

The degree to which the electrical properties of identically specified components fail to match can often limit the circuit performance in analog design. Device mismatches are due both to random events in the manufacturing process – leading to small unpredictable variations in the electrical characteristics of devices – and to dissimilar geometrical choices for matching devices during the design process.

The latter systematically-induced mismatches are handled during placement. For instance, in order to reduce the degree of electrical mismatch due to area effects, matching groups of cells can be defined, all members being constrained to the same orientation and device variant. The matching groups are handled as constraints at the move-set level of the annealer [8, 1, 4].

In addition to handling matching constraints, the shape optimization of parametric cells with a discrete number of possible implementations [4] and of "soft" cells – with the aspect ratio varying continuously between given limits – is also performed during placement.

6 Overview of the main results

The placement tool is currently implemented in Mainsail – object-oriented language which is a trademark of Xidak, Inc. The tool is embedded in ROSE, an in-house retargetable object symbolic environment for layout design, which is in use for industrial purpose and is available on HP UX and SunOS platforms. The results presented below have been obtained on an HP 9000/777 workstation.

The simple illustrative example in Fig. 2 has been processed by our tool in 4 seconds. In contrast, the same example with only 7 cells was processed in over 15 minutes by an ILP-based placer, implementing for testing purpose the analytic model described in [13]. This result shows clearly that the branch-and-bound and related ILP-based techniques are ineffective (unless the number of blocks on each hierarchical level is at most 6 [9], which is not common).

The final layout of the amplifier is presented in Fig. 3. As already mentioned, Murata's rectangle packing algorithm based on sequence-pair [7] cannot handle directly this example due to the symmetry constraints. Fig. 4 shows a telescopic opamp with gain-boost amplifiers (one of which is displayed in Fig. 3). The placement has been performed in 7.8 minutes. Although the theoretical complexity is not affected, the symmetry constraints increase the computation time as they affect the construction of the placement configuration corresponding to a sequence-pair encoding (see Section 4). Without any symmetry constraint, the placement for the 36 cell telescopic opamp took only 3.2 minutes and the area was about 15% smaller.

Fig. 5 shows a programmable capacitor block used in a continuous-time filter. This circuit with 28 cells has no symmetric devices, but it contains 6 large "soft" capacitors which aspect ratio can vary between given limits while preserving the same area. The initial shapes of the 6 capacitors were different from the ones in Fig. 5. After the placement tool determined the "optimal" aspect ratios of the soft capacitors, the device generator incorporated in ROSE [4] was automatically invoked and the 6 capacitors were regenerated with the required new shape, as shown in Fig. 5.

Table 1 displays the results obtained for several benchmark analog circuits. These examples are relevant due to the fact that analog circuits usually contain no more than 100 cells per hierarchical level. The computation time is dependent not only on the circuit size and on the number and type of constraints, but also on the schedule employed during the simulated annealing. This is why comparative time evaluations with other annealing-based placement techniques are somewhat difficult to make. However, because of the schedule similitude, we can report that our experiments indicate a better speed performance by a factor of $1.2 \div 1.4$ compared to the analog placement tool described in [4] which operates on flat Gellat-Jepsen spatial representations [2].

Design	# cells	CPU[min]
Bias current generator	85	24
Gain-boost amplifier	17	0.8
Telescopic opamp (gain-boost amp.)	36	7.8
Programmable capacitor block	28	1.7
Amplifier with selectable gain	79	21
Charge pump	98	42
Limiter ($17 \div 500$ MHz)	111	58
Frequency divider (selectable ratio)	116	64

Table 1: Placement results

7 Conclusions

This paper has addressed the problem of device-level placement for analog layout. Different from most of the existent tools based on a simulated annealing algorithm employing flat representations, this paper has advocated the use of sequence-pair, a topological representation not restricted to slicing structures. Handling of symmetry constraints, essential requirement for any analog placement method, has been thoroughly studied in the context of the sequence-pair representation. The effectiveness of our placement tool, already operational in an industrial environment, has been demonstrated by typical examples from analog design.

References

[1] J. Cohn, D. Garrod, R. Rutenbar, L. Carley, *Analog Device-Level Automation*, Kluwer Academic Publishers, 1994.

[2] D.W. Jepsen, C.D. Gellat Jr., "Macro placement by Monte Carlo Annealing", *Proc. IEEE Int. Conf. on Comp. Design*, pp. 495-498, Nov. 1984.

[3] M. Kayal, S. Piguet, M. Declerq, B. Hochet, "SALIM: a layout generation tool for analog ICs," *Proc. IEEE Custom Integrated Circuits Conf.*, pp. 7.5.1-4, 1988.

[4] K. Lampaert, G. Gielen, W. Sansen, "A performance-driven placement tool for analog integrated circuits," *IEEE J. of Solid-State Circ.*, Vol. SC-30, No. 7, pp. 773-780, July 1995.

[5] E. Malavasi, E. Charbon, G. Jusuf, A. Sangiovanni-Vincentelli, "Virtual symmetry axes for the layout of analog IC's," *Proc. 2nd ICVC*, pp. 195-198, Seoul, Korea, Oct. 1991.

[6] E. Malavasi, E. Charbon, E. Felt, A. Sangiovanni-Vincentelli, "Automation of IC layout with analog constraints," *IEEE Trans. on Comp.-Aided Design of IC's and Systems*, Vol. 15, No. 8, pp. 923-942, Aug. 1996.

[7] H. Murata, K. Fujiyoshi, S. Nakatake, Y. Kajitani, "VLSI module placement based on rectangle-packing by the sequence-pair," *IEEE Trans. on Comp.-Aided Design of IC's and Systems*, Vol. 15, No. 12, pp. 1518-1524, Dec. 1996.

[8] S.W. Mehranfar, "STAT: a schematic to artwork translator for custom analog cells," *Proc. 1990 IEEE Custom Integrated Circuits Conf.*, pp. 30.2.1-3, 1990.

[9] H. Onodera, Y. Taniguchi, K. Tamaru, "Branch-and-bound placement for building block layout," *Proc. 28th ACM/IEEE Design Automation Conf.*, pp. 433-439, 1991.

[10] R. Otten, "Complexity and diversity in IC layout design," *Proc. IEEE Intn'l Symp. Circuits and Computers*, 1980.

[11] J. Rijmenants, J.B. Litsios, T.R. Schwarz, M. Degrauwe, "ILAC: an automated layout tool for analog CMOS circuits," *IEEE J. of Solid-State Circuits*, Vol. SC-24, No. 2, pp. 417-425, April 1989.

[12] W.-J. Sun, C. Sechen, "Efficient and effective placement for very large circuits," *IEEE Trans. on Comp.-Aided Design of IC's and Systems*, Vol. 14, No. 3, pp. 349-359, March 1995.

[13] S. Sutanthavibul, E. Shragowitz, J.B. Rosen, "An analytical approach to floorplan design and optimization," *IEEE Trans. on Comp.-Aided Design of IC's and Systems*, Vol. 10, No. 6, pp. 761-769, June 1991.

[14] D.F. Wong, C.L. Liu, "A new algorithm for floorplan design," *Proc. 23rd ACM/IEEE Design Automation Conf.*, pp. 101-107, 1986.

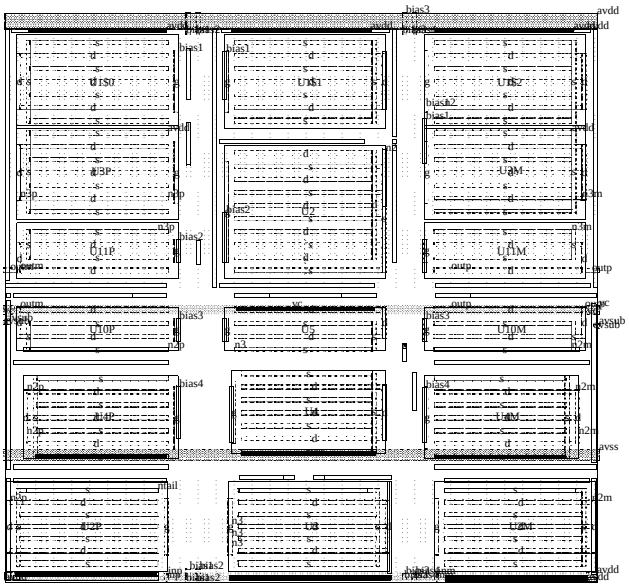

Figure 3: Final layout of the gain-boost amplifier

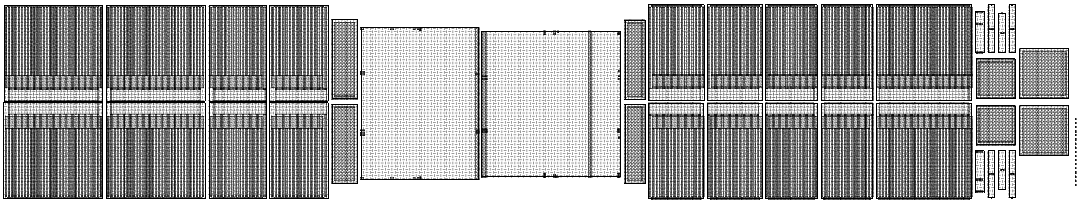

Figure 4: Placement for a telescopic opamp with gain-boost amplifiers

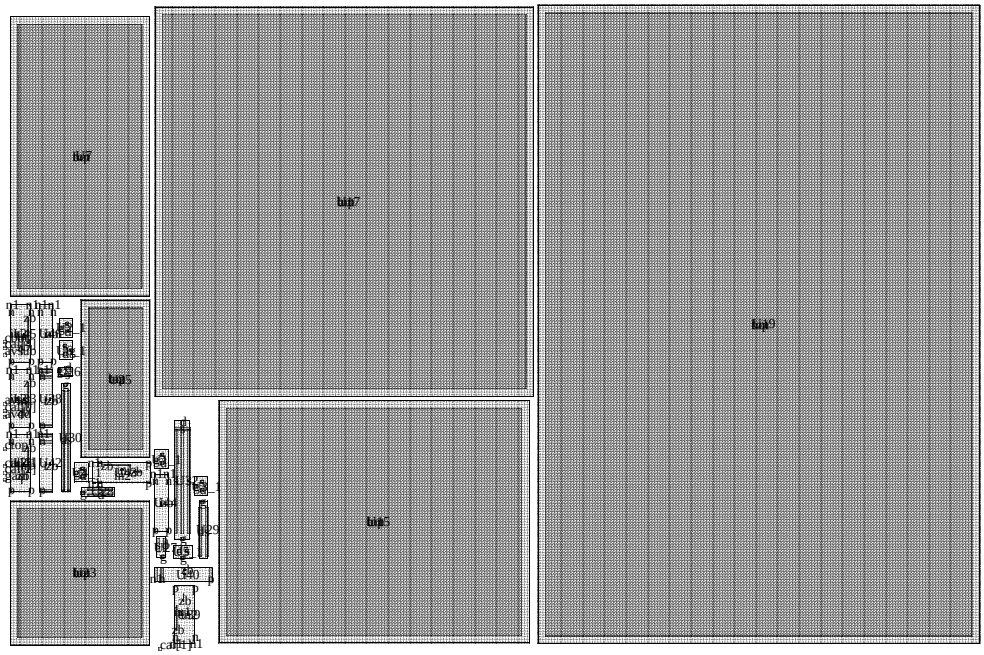

Figure 5: Placement for a programmable capacitor block from a continuous-time filter