Symmetry Within the Sequence-Pair Representation in the Context of Placement for Analog Design

Florin Balasa, *Member, IEEE,* and Koen Lampaert

***Abstract*—This paper addresses the problem of device-level placement for analog layout, focusing mainly on symmetry-related aspects. Different from most of the existent analog placement approaches, employing basically simulated annealing optimization algorithms operating on flat (absolute) spatial representations [4], our model uses a more recent topological representation called *sequence-pair* [14], 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 complex symmetry constraints (for instance, an arbitrary number of symmetry groups of cells), 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.**

***Index Terms*—Analog placement, device matching, nonslicing floorplan, sequence pair, symmetry, topological representation.**

I. 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; and 3) the existence of a (built-in) library of predefined module generators and the ability to exploit their reshaping capabilities during the placement process.

Besides these specific features of analog placement, the main goal of optimally packing arbitrarily sized modules is similar to that of other very large scale integrated circuits (VLSI) placement problems—chip floorplanning, standard cell and macro cell digital placement. Due to the complexity of the basic problem,[1] several heuristic classes of placement techniques have been attempted.

Manuscript received August 25, 1999; revised December 9, 1999. This paper was recommended by Associate Editor M. Sarrafzadeh.

F. Balasa is with the Univ. of Illinois at Chicago, Chicago, IL 60607 USA (e-mail: fbalasa@eecs.uic.edu).

K. Lampaert is with the Conexant Systems Inc., Newport Beach, CA 92660 USA.

Publisher Item Identifier S 0278-0070(00)05853-X.

[1]The decision whether a given set of fixed-oriented rectangles, having widths and heights real numbers, could be packed onto a chip of known width and height was proven to be NP-complete [1] while the problem of finding a minimum area packing is NP-hard.

A. Overview of Analog 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 [8]; Mehranfar suggested a schematic-driven approach, using a constructive scheme based on connectivity and relative positioning in the input schematic [15]. 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 [17] is six), as the number of visited configurations grows exponentially with the size of the problem. The related integer linear programming (ILP) placement models [21] suffer the same scaling drawback (see also Section VI), 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).

More recently, a placement technique iteratively combining min-cut partitioning and force-directed placement (DLP) has been employed in an interactive environment for full-custom designs [13].

For the time being, the simulated annealing [9] and genetic algorithms [5] 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 software packages both in digital (TimberWolfSC v7.0 [20]) and analog design: ILAC [19], KOAN/ANAGRAM II [3], PUPPY-A [12], and LAYLA [10].

B. Absolute and Topological Representations

A simulated annealing algorithm can equally operate with two distinct spatial representations of placement configurations. The first is the so-called *flat* or *absolute* representation introduced by Jepsen and Gellat [7], 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,[2] 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 [3], PUPPY-A [12], and LAYLA [10] systems.

However, this representation has also revealed a drawback, for instance, in [20]. 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 [20] used a sophisticated negative control scheme to determine the optimum values of the cost term weights.

The flat representation approach trades off a larger number of annealing moves for easier and quicker to build layout configurations—which may not be always physically realizable, though. On the other hand, a second class of placement representations—named *topological*—allows trading off more complex, physically correct, layout constructions per each annealing move for a smaller number of moves.

In contrast to the flat representation where cell positions are defined in terms of absolute coordinates, in the topological representations cell positions are specified *relatively*, based on topological relations. The most popular representations are based on the slicing model, first introduced by Otten [18], assuming 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* [22]. 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 [11], which is a less efficient solution. Although the ILAC system [19] employs this model, it is widely acknowledged that slicing placement is not a good choice for high-performance analog design.

More recently, several novel topological representations, not restricted to slicing floorplan topologies, have been proposed. Murata *et al.* suggested to encode the "left-right" and "up-down" positioning relations between cells using two sequences of cell permutations, named *sequence-pair* [14]. Nakatake *et al.*devised a meta-grid structure (called *bounded-sliceline grid* (BSG) without physical dimensions to define orthogonal relations between modules [16]. Very recently, Guo *et al.* proposed the *ordered tree* (*O-tree*) data structure to reduce the drawback of redundancies in the two previous representations [6].

The goal of our paper is to show that sequence-pair—the most popular nonslicing topological 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. The analog examples shown in [2] contain only one symmetry group of cells; this paper displays a complex frequency divider with selectable ratio having five symmetry groups. In addition, a complete proof of the theoretical computation of the solution space size is also included.

This paper is organized as follows. Section II will briefly describe the sequence-pair representation. Section III will explain the nature of the symmetry constraints in analog design, while Section IV will thoroughly analyze the symmetry handling during simulated annealing, when this optimization operates on sequence-pair representations. Section V briefly reviews the device-matching constraints and Section VI proves the effectiveness of our analog placement solution. Then, Section VII concludes with final remarks, while the Appendix presents a thorough evaluation of the solution space size for placement problems with symmetry constraints.

II. 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 [14], a method of deriving such an encoding is

[2]In analog layout, cells can overlap not only in *legal* but also *beneficial* ways ("device merging" or "geometry sharing" [3]).

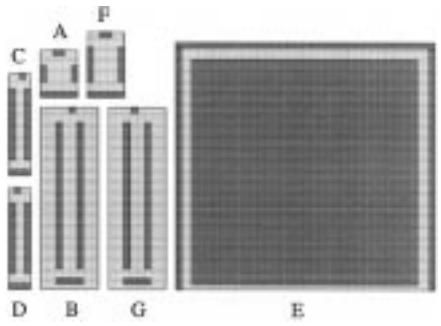

Fig. 1. Placement configuration encoded by the sequence-pair $(CDAFBGE, DCBGAFE)$.

informally described.[3] In particular, there exists at least a sequence-pair encoding corresponding to (one of) the optimal rectangle packing.[4] 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[5] , 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,[6] the topological relations between two cells A and B are given by the following:

if $\alpha_A^{-1} < \alpha_B^{-1}$ and $\beta_A^{-1} < \beta_B^{-1}$ then cell A is to the left of cell B; (T)

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 seven- cell placement configuration in Fig. 1 is $(\alpha, \beta) = (CDAFBGE, DCBGAFE)$ (see Section IV). With the notations employed, we have, for instance, $\alpha_1 = C, \alpha_2 = D$, and also, $\alpha_C^{-1} = 1, \alpha_D^{-1} = 2$, etc. 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 [14]. In order to apply this topological representation to analog placement, handling symmetry is an essential requirement. Section IV will show how to handle the symmetry constraints within the sequence-pair topological representation.

III. 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 [4].

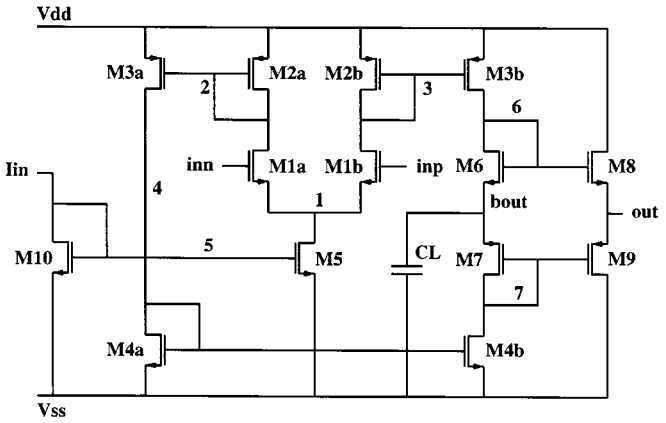

Fig. 2. Schematic of a high-speed CMOS comparator.

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. The typical forms of symmetry which should be handled by an analog placement tool are as follows [4]:

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. This type of symmetry is sometimes required in order to meet very stringent matching requirements;
3) self-symmetry—characteristic for devices presenting a geometrical symmetry and sharing the same axis with other pairs of symmetric devices.

A subset of cells is called a *symmetry group* if all cells are exhibiting a form of symmetry and, in addition, they all share a common symmetry axis.

Usually, the analog circuits have a mix of symmetric and asymmetric components. Fig. 2 shows the schematic of a high-speed CMOS comparator, circuit used in a CMOS A/D converter. This comparator converts a differential input signal into a single-ended output value. Therefore, the input stage is symmetric while the single-ended output stage introduces an asymmetric part.

IV. 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 II example should be symmetric relative to a vertical axis: the encoding $(\alpha, \beta) = (CDAFBGE,$

[3]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 IV.

[4]This property is not valid for the normalized Polish expressions [22] encoding the slicing structures, as the optimal (in terms of area) packing may not have a slicing topology.

[5]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.

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

$DCBGAFE$) 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 (see Appendix for a proof)

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

where p is the number of symmetric pairs, and s is the number of self-symmetric cells in the group.

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).

A better strategy is to explore *only* those sequence-pairs which comply with the symmetry constraints. This section will show 1) how to recognize such sequence-pairs and 2) 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.[7] We denote by $\mathrm{sym}(x)$ the symmetric of cell x, and we consider by convention that $\mathrm{sym}(x) \equiv x$ when x is a self-symmetric cell.

Definition: The sequence-pair (α, β) is called *symmetric-feasible* if

$$(S) \quad \alpha_x^{-1} < \alpha_y^{-1} \Leftrightarrow \beta_{\mathrm{sym}(y)}^{-1} < \beta_{\mathrm{sym}(x)}^{-1}, \quad \forall \text{ cells } x, y \text{ in } \gamma, \quad x \neq y.$$

Choosing $y = \mathrm{sym}(x)$ and taking into account that $\mathrm{sym}(\mathrm{sym}(x)) = x$, condition (S) shows that any symmetric pair of cells appears in the same order in both sequences α and β

$$\alpha_x^{-1} < \alpha_{\mathrm{sym}(x)}^{-1} \Leftrightarrow \beta_x^{-1} < \beta_{\mathrm{sym}(x)}^{-1}, \quad \forall \text{ cells } x \text{ in } \gamma.$$

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

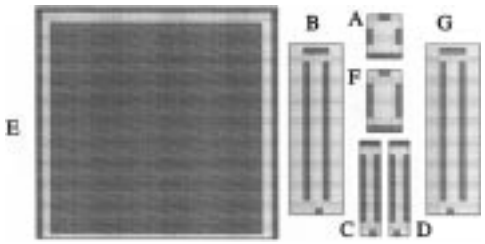

Fig. 3. Placement configuration with symmetry group $\gamma = ((C, D), (B, G), A, F)$.

According to condition (S) of symmetric-feasibility, two cells x, y belonging to distinct symmetric pairs and, respectively, their symmetric cells appear in reversed order in the two sequences of the encoding. In addition, any two self-symmetric cells appear in reversed order in the sequences α and β, as it can be easily seen taking $x = \mathrm{sym}(x)$ and $y = \mathrm{sym}(y)$ in (S).

Example (Continued): 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 $(\alpha, \beta) = (CDAFBGE, DCBGAFE)$ (see Fig. 1) is not symmetric-feasible: for instance, the symmetric cells C and D do not satisfy condition (S) as they do not appear in the same order in the sequences α and β ($\alpha_C^{-1} < \alpha_D^{-1}$, but $\beta_C^{-1} > \beta_D^{-1}$).

On the other hand, the encoding $(\alpha, \beta) = (EBAFCDG, EBCDFAG)$, derived from the placement configuration in Fig. 3, is symmetric-feasible. Taking, for instance, the self-symmetric cell A and comparing its positions α_A^{-1} and β_A^{-1} in the sequences α and, respectively, β to the corresponding positions of the other cells in the symmetry group, it may be concluded, from the pairs of valid inequalities $(\alpha_A^{-1} < \alpha_F^{-1}, \beta_F^{-1} < \beta_A^{-1})$, $(\alpha_A^{-1} < \alpha_C^{-1}, \beta_D^{-1} < \beta_A^{-1})$, $(\alpha_A^{-1} < \alpha_D^{-1}, \beta_C^{-1} < \beta_A^{-1})$, $(\alpha_B^{-1} < \alpha_A^{-1}, \beta_A^{-1} < \beta_G^{-1})$, $(\alpha_A^{-1} < \alpha_G^{-1}, \beta_B^{-1} < \beta_A^{-1})$, that condition (S) is satisfied whenever cell A is involved. Similar comparisons involving the positions of the other cells in γ will conclude the verification.[8]

The following two lemmas will prove that, in order to find a cell packing of minimum area, containing a symmetry group of cells, it is sufficient to explore the subspace of *symmetric-feasible* sequence-pairs, rather than the entire sequence-pair space, which is significantly larger (as mentioned above, e.g., 24 times larger for only two pairs of symmetric cells).

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

Proof: According to [14], any placement configuration can be encoded with a sequence-pair. In particular, this is true also for those configurations containing symmetry groups.

The procedure described below is similar to the *Gridding* encoding in [14]. However, *ConstructSequenceAlpha* avoids the ambiguity of the *Gridding* encoding by providing priority to the horizontal relation. In addition, the resulting sequence-pair has

[8]Checking the symmetric-feasibility of a given sequence-pair is not a necessary operation in the context of our placement method. As it will be explained in more detail toward the end of Section IV, constructing an initial symmetric-feasible sequence-pair and performing only transformations which preserve the symmetric-feasibility entail an efficient exploration of placement configurations satisfying the given symmetry constraints.

the property (S) when the procedure is applied to a placement configuration with a symmetry group.

```
sequence procedure ConstructSequenceAlpha (set of m fixed,
nonoverlapping cells)                                          {
// a similar procedure ConstructSequenceBeta builds
   sequence β (differences are notified)
   sequence α (resp. β) is initially zeroed;
   for k = 1 to m{//k is the current position in the
    sequence
    let j be any cell which is not yet in the sequence α
    (resp., β);
    for (every cell i(≠ j) not in the sequence) {// looking
    for a candidate for position k
     if the horizontal projections of cells i and j are
     overlapping
     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) // α_j^{-1} = k (resp., β_j^{-1} = k)
   }
 return sequence α (resp., β);
 }
```

First, it must be noticed that the topological relations (T) specific to the sequence-pair representation (see Section II) are satisfied due to lines (*) and (**). In particular, for a pair of symmetric cells (x, y), x being placed to the left of y, the routines *ConstructSequenceAlpha* and *Beta* yield, due to line (**), a sequence-pair (α, β) where $\alpha_x^{-1} < \alpha_y^{-1}$ and, respectively, $\beta_x^{-1} < \beta_y^{-1}$. Therefore, cells x and y satisfy condition (S).

It can be verified by analyzing all the possible relative positions between two pairs of symmetric cells that the encoding generated by the procedures described above satisfies condition (S). For the sake of clarity, the pairs (B, G) and (C, D) in the example from Fig. 3 will be considered. There are two cases of relative positions of the two pairs (less the possible cell interchanges within the same pair, or interchanges between pairs) as follows.

1) The pair (C, D) lies between cells B and G, as shown in Fig. 3. Due to line (**), the procedures *ConstructSequenceAlpha* and *Beta* generate a sequence-pair (α, β) such that

$$\alpha_B^{-1} < \alpha_C^{-1} < \alpha_D^{-1} < \alpha_G^{-1}$$

and

$$\beta_B^{-1} < \beta_C^{-1} < \beta_D^{-1} < \beta_G^{-1}.$$

2) The pair (C, D) lies below the pair (B, G), such that the horizontal projections of cells C and B and, respectively, D and G, are overlapping. Due to line (*) in the procedures, $\alpha_B^{-1} < \alpha_C^{-1}, \alpha_G^{-1} < \alpha_D^{-1}$ and $\beta_C^{-1} < \beta_B^{-1}, \beta_D^{-1} < \beta_G^{-1}$. In addition, due to line (**)

$$\alpha_B^{-1} < \alpha_C^{-1} < \alpha_G^{-1} < \alpha_D^{-1}$$

and

$$\beta_C^{-1} < \beta_B^{-1} < \beta_D^{-1} < \beta_G^{-1}.$$

The two inequalities above and their equivalent ones obtained by interchange operations show that any two cells in the subset $\{B, C, D, G\}$ satisfy condition (S).

The relative positions between two self-symmetric cells, or between one symmetric pair and one self-symmetric cell, may be similarly analyzed: in all cases, condition (S) is satisfied for every two cells in the symmetry group. □

Example (Continued): Applying the encoding procedure to the illustrative placement configurations in Fig. 1 and Fig. 3, the sequence-pairs $(\alpha, \beta) = (CDAFBGE, DCBGAFE)$ and, respectively, $(\alpha, \beta) = (EBAFCDG, EBCDFAG)$ are obtained. The latter encoding is symmetric-feasible, as already shown.

Lemma 1 shows that there is at least one symmetric-feasible sequence-pair corresponding to a placement configuration with a symmetry group, optimal in terms of a cost function (for instance, area).

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 $i (i = 1, \ldots, m)$ 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), symmetryAxis = 0;
//computation of the x coordinates
for i = 1 to m                                                 {
 j = α_i; //choose cell j having the position i in sequence α
 for l = i + 1 to m{
  k=α_l; //for all cells k positioned after j in sequence α
  if β_j^{-1} < β_k^{-1} // if cell k is positioned after j also in
  sequence 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(=sym(j)) such that α_k^{-1} ≤ i
  // if cell j has a symmetric which is to its left (thus,
  x_k is known), or it is self-symmetric
 then symmetryAxis = symmetryAxis 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 (symmetryAxis > 0), 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 for the cells to the right of the symmetry axis, including 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 the position i in sequence α
 if cell j has a symmetric cell k(=sym(j)) such that α_k^{-1} ≤ i
 then {
    // if cell j has a symmetric which is to its left, find
    the potential displacement for j
    d = 2 · symmetryAxis − (x_k + width_k) − x_j; //          (*)
    if k=j then d=d/2; //if j is self-symmetric, the dis-
     placement is only half
    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
    }
   }
 }
}
for i = m downto 1 {// the sweep to the left
 j = α_i; //choose cell j having the position i in sequence α
 if cell j has a symmetric cell k(=sym(j)) such that i < α_k^{-1}
 then {
    // if cell j has a symmetric which is to its right,
    find the potential displacement for j
    d = 2 · symmetryAxis − (x_k + width_k) − x_j; //          (*)
   if d < 0 then {// displacement indeed
    x_j = x_j + d; // push cell j to the left                (**)
    for l = 1 to i − 1{
     k = α_l; // all cells k to the left of j
     if β_k^{-1} < β_j^{-1} then x_k = x_k + d; // are equally pushed
      to the left
    }
   }
 }
}
```

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 $(x, \mathrm{sym}(x))$ and $(y, \mathrm{sym}(y))$ 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 $\mathrm{sym}(y)$ is pushing $\mathrm{sym}(x)$ to the left achieving the symmetry of $(y, \mathrm{sym}(y))$ but destroying again the symmetry of $(x, \mathrm{sym}(x))$. 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 such that the positioning constraints (compatible with the given sequence-pair) are satisfied.

```
initialize y_i = 0 (i = 1, ..., m); // computation of the y coor-
 dinates
for i = m downto 1 {
 j = α_i; //choose cell j having the position i in sequence α
 for l = i − 1 downto 1 {
  k = α_l; //for all cells k positioned before j
  in sequence α
  if β_j^{-1} < β_k^{-1} then {// if cell k is positioned after j
   in sequence β
    y_k = y_k max (y_j + height_j); // then k is above cell j
    if cell k has a symmetric cell k'(≠k) then y_{k'} = y_k;
                                                          (***)
  }
 }
}
```

In order to prove the statement of *Lemma 2*, it must be shown that the construction described above yields a *valid* placement configuration (i.e., cells do not overlap), satisfying the positioning constraints (T) specific to the given sequence-pair, and verifying also the conditions of symmetry; in addition, the resulting placement has minimum width and minimum height subject to the topological and symmetry constraints.

During the "computation of x coordinates", we have $\alpha_j^{-1} < \alpha_k^{-1}$ (as $i < l$). If, in addition, $\beta_j^{-1} < \beta_k^{-1}$ then cell k will be placed to the right of j, as $x_k \geq x_j + \mathrm{width}_j$: these cells do not overlap. The "sweep to the right" will not alter this situation: when a cell j is pushed to the right, all the cells k such that $\alpha_j^{-1} < \alpha_k^{-1}$ and $\beta_j^{-1} < \beta_k^{-1}$ are equally displaced to the right. The "sweep to the left" preserves the left-right relations as well.

During the "computation of y coordinates", we have $\alpha_k^{-1} < \alpha_j^{-1}$ (as $l < i$). If, in addition, $\beta_j^{-1} < \beta_k^{-1}$ then cell k will be positioned above j, as $y_k \geq y_j + \mathrm{height}_j$: neither these cells do overlap. As any two cells j and k appear in the sequences α and β either in the same order $(\alpha_j^{-1} < \alpha_k^{-1}, \beta_j^{-1} < \beta_k^{-1})$ or in reversed order $(\alpha_k^{-1} < \alpha_j^{-1}, \beta_j^{-1} < \beta_k^{-1})$, at least one of the relation types $x_k \geq x_j + \mathrm{width}_j, y_k \geq y_j + \mathrm{height}_j$ holds. It follows that no overlap occurs in the resulting placement configuration; moreover, the topological constraints (T) derived from the given sequence-pair are satisfied.

Due to lines (*) and (**), together with (***), for any pair of symmetric cells (k, j), cell j is pushed in order to make it satisfy the symmetry conditions $x_j = 2 \cdot symmetryAxis - (x_k + \mathrm{width}_k)$ and $y_j = y_k$. (For self-symmetric cells $(k = j)$, the symmetry condition is $x_j = symmetryAxis - \mathrm{width}_j/2$.) As explained earlier, when the value of x_j is trimmed in order to satisfy the symmetry constraints, no alteration of the previously trimmed x coordinates can occur due to the symmetric-feasibility of the given sequence-pair. It follows that the resulting placement configuration satisfy also the symmetry constraints.

Finally, it must be shown that the resulting placement configuration has minimum width and height, while satisfying the constraints.

Suppose for the time being that there are no symmetry constraints and let us construct the horizontal- and vertical-constraint graphs G_H and G_V as described in [14]. These are two vertex-weighted directed acyclic graphs, having a *source* and a *sink*, and m nodes labeled with the cell names. In G_H, the arcs represent the left-right relations between cells, while the vertex-weights are the cell widths; in G_V the arcs represent top-down relations between cells, and the vertex-weights are the cell heights (the weights of *source* and *sink* being zero). It must be noticed that the first part of the construction algorithm ("computation of x coordinates") determines in fact the longest path lengths (denoted x_k) from *source* to the nodes k in G_H. Similarly, the "computation of y coordinates" determines the longest path lengths (denoted y_k) from *source* to the vertices k in G_V. Due to the absence of symmetry constraints, the "sweep to the right/left" operations are not taking place.

The width and the height of the placement configuration result to be the longest path lengths between *source* and *sink* in G_H and, respectively, in G_V. The width and height have independently minimum values (relative to the positioning constraints derived from the sequence-pair): lesser values would imply violations of positioning constraints in G_H and/or G_V.

In the presence of a symmetry group, the constraint graphs will be modified in order to encompass the additional symmetry constraints. Let j,k be two nodes corresponding to symmetric cells. Then, for every arc (i,j) in G_V, an additional arc (i,k) must be added (unless it already exists), and vice versa. In this way, the longest paths from the *source* of G_V to j and to k will have same lengths $(y_k = y_j)$.

At the same time, an extra node A of zero weight, corresponding to the symmetry axis, along with the new arcs (k,A) and (A,j) for every pair of symmetric cells (k,j) are added to G_H. The first part of the construction algorithm determines as before the longest path lengths x_i from the *source* of G_H to every vertex i. (*symmetryAxis* will be the length of the longest path to node A.) Afterwards, in topological order, for every cell j having a symmetric cell k to the left, whenever $w_{j'} \stackrel{\text{not}}{=} symmetryAxis - (x_k + \text{width}_k) > x_j - symmetryAxis$ (i.e., $d > 0$ in "sweep to the right"), the arc (A,j) is replaced by an additional node j' of weight equal to $w_{j'}$, and the arcs (A,j') and (j',j). The longest path lengths from the *source* to the descendents of j in G_H are also updated in topological order. A similar operation, but in reversed topological order, has to be done for every cell k having a symmetric cell j to the right, new nodes k' being added between k and A in order to compensate the symmetry violations.[9]

In the modified graphs G_H and G_V, which model both the topological and symmetry constraints, x_k and y_k still represent the longest path lengths in G_H and G_V from the respective *source* to node k. The width and height of the placement configuration are still the longest path lengths between the *source* and *sink* in the two constraint graphs. Due to similar arguments, they have minimum values subject to both types of constraints. □

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 smaller space 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 is, for instance

$$(\alpha,\beta) = (a_1 \ldots a_p\, c_1 \ldots c_s\, b_p \ldots b_1 \ldots,\ a_1 \ldots a_p\, c_s \ldots c_1\, b_p \ldots b_1 \ldots)$$

where $(a_i,b_i), i = 1,\ldots,p$ are the pairs of symmetric cells and $c_j, j = 1,\ldots,s$ are the self-symmetric cells.[10] The initial sequence-pair can be built in different ways: our tests show that the choice is irrelevant with respect to CPU time and the quality of the final solution.

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 distinct 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. This extension is quite straightforward and it will be subsequently exemplified in Section VI (Fig. 7).

The symmetry about a horizontal symmetry axis can be equally modeled starting from the modified (S) condition

$$(S') \quad \alpha_x^{-1} < \alpha_y^{-1} \Leftrightarrow \beta^{-1}_{\text{sym}(x)} < \beta^{-1}_{\text{sym}(y)},\ \forall \text{ cells } x,y \text{ in } \gamma, \quad x \neq y$$

and inverting the construction of the x and y coordinates presented in the proof of *Lemma 2*. Actually, the last remarks lead to the conclusion that, within the sequence-pair representation, one can model symmetry relative to any number of vertical and horizontal axes.

V. 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

[9]If the weights of the "compensation" nodes j', k' are chosen larger with the same amount for any pair of symmetric cells (j,k), the topological and symmetry constraints will still be satisfied: the width of the placement will result larger, though.

[10]This sequence-pair corresponds 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.

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 [15], [3], [10].

In addition, device proximity constraints which offer the designer the possibility to specify devices to be placed in close proximity are modeled both with an imaginary proximity net of a relatively high weight, similar to the method employed by other layout systems (e.g., [3]), and also restricting the move-set by keeping together the cells in the matching group in the sequences α and β.

The device separation constraints which allow the designer to specify a maximum separation distance between pairs of critically matched devices are modeled with a penalty quadratic term in the annealer's cost function. Performance constraints specifying the maximum capacitance of certain critical nets are handled similarly.

In addition to handling matching constraints, the shape optimization of parametric cells with a discrete number of possible implementations [10] and of "soft" cells—with the aspect ratio varying continuously between given limits—is also performed during placement.

VI. 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. 3 has been processed by our tool in 4 s. In contrast, the same example with only seven cells was processed in over 15 min by an ILP-based placer, implementing for testing purpose the analytic model described in [21]. 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 six [17], which is not common).

Fig. 4 shows a more realistic example which allows to evaluate the packing capability of the placement tool. The 116-block example represents a frequency divider with selectable ratio (two or four). Without specifying any symmetry constraint, the placement has been performed in 50 min. This example is relevant due to the fact that analog circuits seldom contain more than 100 cells per hierarchical level. The placement of analog cells is very demanding not because of their size, but due to the characteristics and constraints explained in the previous sections, like symmetry, existence of "soft" cells, device matching.

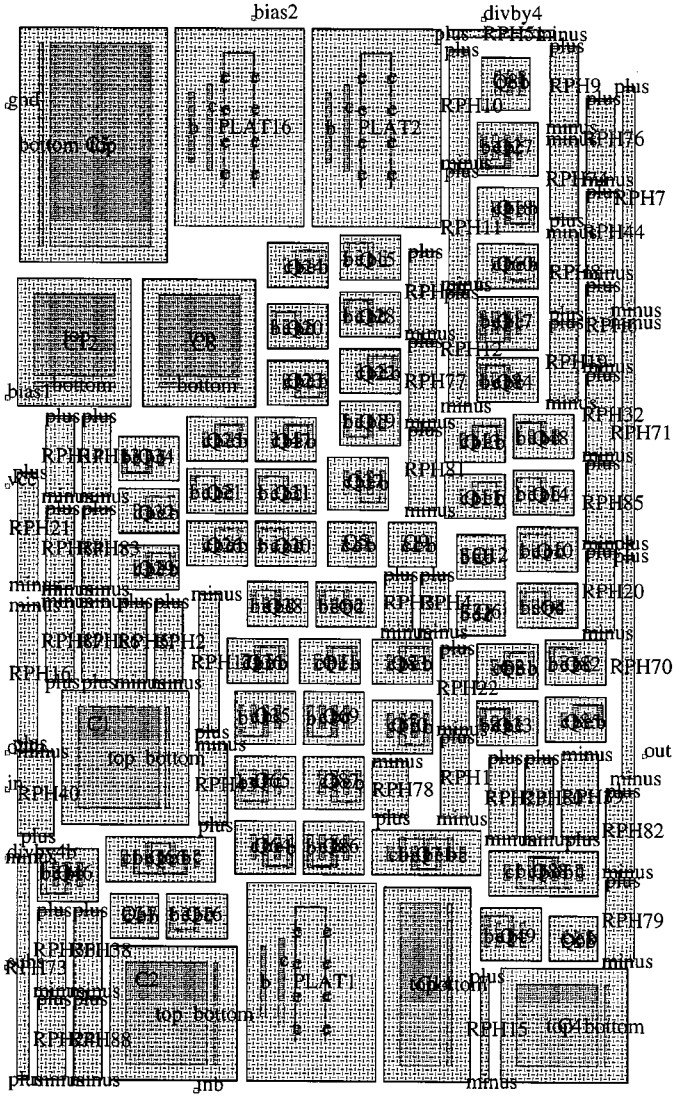

Fig. 4. Placement for a frequency divider with selectable ratio (no symmetry constraints).

Fig. 5 shows the final layout of a gain-boost amplifier containing several self-symmetric and pairs of symmetric devices. As already mentioned, Murata's rectangle packing algorithm based on sequence-pair [14] cannot handle directly this example due to the symmetry constraints.

Fig. 6 shows a telescopic opamp with gain-boost amplifiers (one of which is displayed in Fig. 5). The placement has been performed in 7.8 min. 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 IV). Without any symmetry constraint, the placement for the 36 cell telescopic opamp took only 3.2 min and the area was about 15% smaller.

Finally, the frequency divider—which placement had been initially performed without symmetry constraints (see Fig. 4)—was processed once again, the second time deriving the five symmetry groups of cells specified in the attribute section of the netlist. This time the placement took (as expected!) longer (68 min) and the area was about 26.7% larger. The final

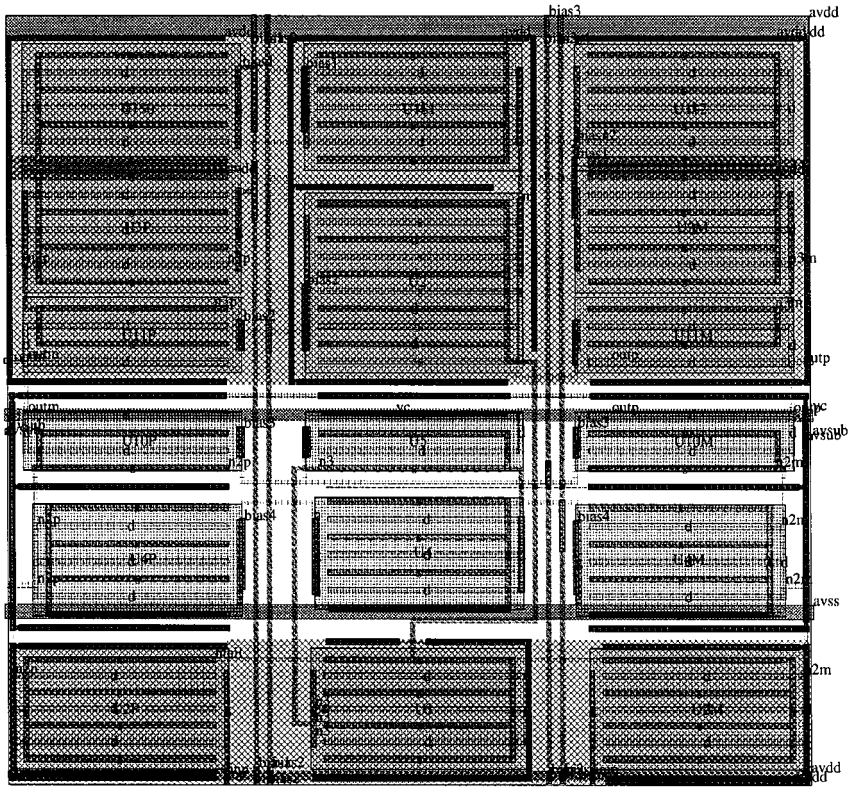

Fig. 5. Final layout of a gain-boost amplifier.

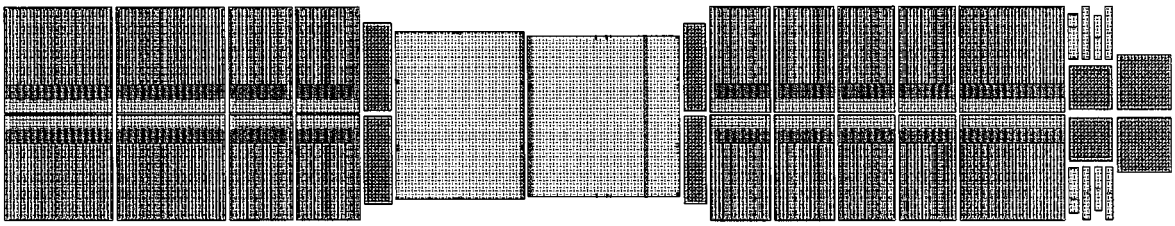

Fig. 6. Placement for a telescopic opamp with gain-boost amplifiers.

solution—displayed in Fig. 7—clearly proves the ability of the placement tool to deal simultaneously with several symmetry groups.

Table I displays the results obtained for several benchmark analog circuits. 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. [11] However, because of the schedule similitude, we can report that our experiments indicate a better speed performance by a factor of 1.2–1.4 compared to the analog placement tool described in [10] which operates on flat Gellat-Jepsen spatial representations [7]. Due to the good capability in handling symmetry, device alignment and matching, the sequence-pair representation proves to be very effective for device-level placement of analog circuits.

[11]Murata *et al.* reported three results [14]: the *ami49* example (from the MCNC benchmark) processed in 31.36 min; a 146 cell example solved in 29.9 min, and a large example containing 500 blocks processed in 18.83 h. No constraints (like symmetry or device matching) are mentioned for any of those examples.

VII. Conclusion

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, absolute 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 employed in an industrial environment, has been demonstrated by typical examples from analog design.

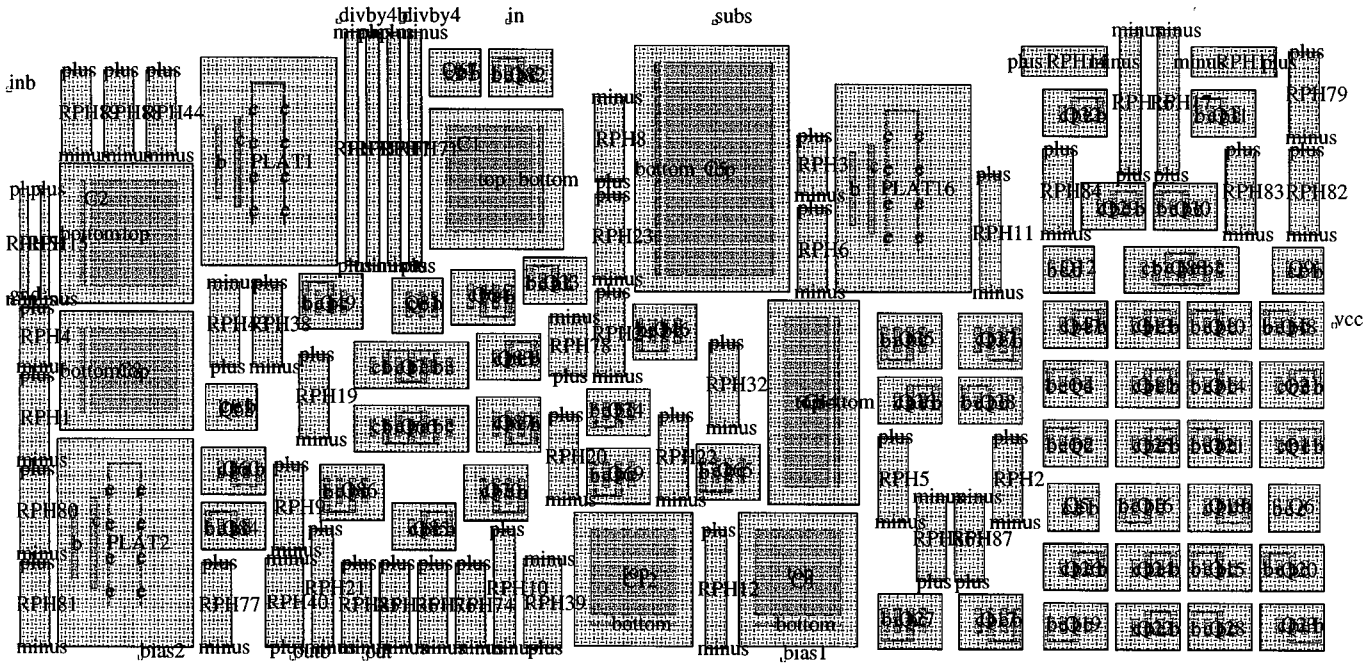

Fig. 7. Placement for a frequency divider with selectable ratio (five symmetry groups).

TABLE I
PLACEMENT RESULTS

Design	Constraints	Nr. cells	Area [$\mu m \times \mu m$]	Time [min]
Gain-boost amplifier*	sym.,dev. matching	17	71.2 × 68.0	0.8
Telescopic opamp* with gain-boost amplifiers	sym.,dev. matching, and 4 soft cells	36	527.2 × 96.0	7.8
Programmable capacitor block 1	6 soft cells	28	175.2 × 115.2	1.7
Programmable capacitor block 2	12 soft cells	34	220.8 × 186	3.4
15 MHz buffer	–	64	189.5 × 250.5	12
Amplifier with selectable gain	–	79	191.5 × 251.0	21
Bias current generator	–	85	237 × 186	24
Charge pump	sym.,dev. matching	98	220.5 × 333.0	42
Limiter (17 ÷ 500 MHz)	sym.,dev. matching, and 14 soft cells	111	177.5 × 375.0	58
Frequency divider with selectable ratio*	–	116	155 × 262	50
- , , - *	5 sym. groups	116	350 × 147	68

The solutions of the designs marked with * are displayed.

APPENDIX

If the number of sequence-pairs for m placeable cells is $(m!)^2$, the number of *symmetric-feasible* sequence-pairs (therefore, the size of the search space in the presence of symmetry constraints) is given by the following:

Lemma: The number of symmetric-feasible sequence-pairs corresponding to a placement configuration with m cells and G symmetry groups, each containing p_k pairs of symmetric cells and s_k self-symmetric cells $(k = 1, \ldots, G)$, is given by the formula

$$[(m-2p-s)!]^2 \cdot \prod_{k=1}^{G} \left[C_m^{2p_k+s_k}\right]^2 N_{p_k,s_k}, \quad p = \textstyle\sum_k p_k, \quad s = \textstyle\sum_k s_k \qquad (2)$$

where each factor N_{p_k,s_k}—denoting the contribution of the symmetry group k—is equal to $(2p_k + s_k)!$.

Proof: We may assume without any lack of generality that the number of symmetry groups is $G = 1$. This is possible due to the fact that the contributions of the asymmetric component of the circuit and those of each of the symmetry groups are independent and, therefore, the total number of symmetric-feasible sequence-pairs is the product of these contributions. Then, we may consider the simplified formula (1) to be proven.

The contribution of the asymmetric part (having $m - 2p - s$ cells) of the circuit is $[C_m^{m-2p-s}(m-2p-s)!]^2$ as there are C_m^{m-2p-s} sets of positions in sequence α (and the same number in sequence β) for the $m - 2p - s$ cells, and there are also $(m-2p-s)!$ possibilities of assigning these cells to the selected positions in both sequences. It may be noticed that when there are no symmetry constraints $(p = s = 0)$, the contribution results to be $(m!)^2$, which is the total number of sequence-pairs.

When the symmetry group contains p pairs of symmetric cells $\{(a_1, b_1), \ldots, (a_p, b_p)\}$, and s self-symmetric cells

$\{c_1, \ldots, c_s\}$, the set of $N_{p,s}$ symmetric-feasible sequence-pairs $\{(\alpha_{p,s}, \beta_{p,s})\}$ may be constructed recurrently as the union of the following sets:

1) $\bigcup_i \{(a_i \alpha_{p,s}^{a_i}, \beta_{p,s}^{b_i} b_i)\} \bigcup \bigcup_i \{(b_i \alpha_{p,s}^{b_i}, \beta_{p,s}^{a_i} a_i)\}$;
2) $\bigcup_j \{(c_j \alpha_{p,s}^{c_j}, \beta_{p,s}^{c_j} c_j)\}$.

where $\{(\alpha_{p,s}^{a_i}, \beta_{p,s}^{b_i})\}$ are the symmetric-feasible sequence-pairs generated by all the cells in the symmetry group except a_i in $\alpha_{p,s}$ and except b_i in $\beta_{p,s}$; similarly, $\{(\alpha_{p,s}^{c_j}, \beta_{p,s}^{c_j})\}$ are the symmetric-feasible sequence-pairs generated by all the pairs of symmetric cells and all the self-symmetric cells except c_j.

It follows that

$$\{(\alpha_{p,s}, \beta_{p,s})\} = \left\{ \bigcup_{\pi} (\pi(a_1 \ldots a_p, b_1 \ldots b_p, c_1 \ldots c_s), \mathrm{mirror}(\pi(\mathrm{sym}(a_1) \ldots \mathrm{sym}(a_p), \mathrm{sym}(b_1) \ldots \mathrm{sym}(b_p), \mathrm{sym}(c_1) \ldots \mathrm{sym}(c_s)))) \right\}.$$

where π is a permutation and $\mathrm{mirror}(\pi)$ is the reverse permutation of π. (Example: $(a_1 b_1 b_2 c_1 a_2, b_2 c_1 a_2 a_1 b_1)$.

Therefore, the number of symmetric-feasible sequence-pairs $\{(\alpha_{p,s}, \beta_{p,s})\}$ is equal to the number of permutations $\pi : N_{p,s} = (2p + s)!$, and the lemma is proven. □

Example (Continued): Given the example in Fig. 3 having the symmetry group $\gamma = ((C, D), (B, G), A, F)$, the number of symmetric-feasible sequence-pairs is

$$\left[C_7^{4+2} \cdot (7 - 4 - 2)!\right]^2 \cdot N_{2,2} = \left[\frac{7!}{6!}\right]^2 \cdot 6! = 7 \cdot 7! = 35\,280.$$

REFERENCES

[1] B. S. Baker, E. G. Coffman, and R. L. Rivest, "Orthogonal packings in two dimensions," *SIAM J. Comput.*, vol. 9, no. 4, pp. 846–855, 1990.

[2] F. Balasa and K. V. Lampaert, "Module placement for analog layout using the sequence-pair representation," in *Proc. 36th ACM/IEEE Design Automation Conf.*, New Orleans, LA, June 1999, pp. 274–279.

[3] J. Cohn, D. Garrod, R. Rutenbar, and L. Carley, "KOAN/ANAGRAM II: New tools for device-level analog layout," *IEEE J. Solid-State Circuits*, vol. 26, pp. 330–342, Mar. 1991.

[4] ——, *Analog Device-Level Automation*: Kluwer Academic Publishers, 1994.

[5] J. Cohoon and W. Paris, "Genetic placement," *IEEE Trans. Computer-Aided Design*, vol. CAD–6, pp. 956–964, Nov. 1987.

[6] P.-N. Guo, C.-K. Cheng, and T. Yoshimura, "An O-tree representation of nonslicing floorplan and its applications," in *Proc. 36th ACM/IEEE Design Automation Conf.*, June 1999, pp. 268–273.

[7] D. W. Jepsen and C. D. Gellat Jr., "Macro placement by Monte Carlo annealing," in *Proc. IEEE Int. Conf. Computer Design*, Nov. 1983, pp. 495–498.

[8] M. Kayal, S. Piguet, M. Declerq, and B. Hochet, "SALIM: A layout generation tool for analog ICs," in *Proc. IEEE Custom Integrated Circuits Conf.*, 1988, pp. 7.5.1–7.5.4.

[9] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, no. 4598, pp. 671–680, May 1983.

[10] K. Lampaert, G. Gielen, and W. Sansen, "A performance-driven placement tool for analog integrated circuits," *IEEE J. Solid-State Circuits*, vol. 30, pp. 773–780, July 1995.

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

[12] E. Malavasi, E. Charbon, E. Felt, and A. Sangiovanni-Vincentelli, "Automation of IC layout with analog constraints," *IEEE Trans. Computer-Aided Design*, vol. 15, pp. 923–942, Aug. 1996.

[13] E. Malavasi, J. L. Ganley, and E. Charbon, "Quick placement with geometric constraints," in *Proc. IEEE Custom Integrated Circuits Conf.*, 1997, pp. 561–564.

[14] H. Murata, K. Fujiyoshi, S. Nakatake, and Y. Kajitani, "VLSI module placement based on rectangle-packing by the sequence-pair," *IEEE Trans. Computer-Aided Design*, vol. 15, pp. 1518–1524, Dec. 1996.

[15] S. W. Mehranfar, "STAT: A schematic to artwork translator for custom analog cells," *Proc. 1990 IEEE Custom Integrated Circuits Conf.*, pp. 30.2.1–30.2.3, 1990.

[16] S. Nakatake, K. Fujiyoshi, H. Murata, and Y. Kajitani, "Module packing based on the BSG-structure and IC layout applications," *IEEE Trans. Computer-Aided Design*, vol. 17, pp. 519–530, June 1998.

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

[18] R. Otten, "Complexity and diversity in IC layout design," presented at the IEEE Int. Symp. Circuits and Computers, 1980.

[19] J. Rijmenants, J. B. Litsios, T. R. Schwarz, and M. Degrauwe, "ILAC: An automated layout tool for analog CMOS circuits," *IEEE J. Solid-State Circuits*, vol. SC-24, no. 2, pp. 417–425, Apr. 1989.

[20] W.-J. Sun and C. Sechen, "Efficient and effective placement for very large circuits," *IEEE Trans. Computer-Aided Design*, vol. 14, pp. 349–359, Mar. 1995.

[21] S. Sutanthavibul, E. Shragowitz, and J. B. Rosen, "An analytical approach to floorplan design and optimization," *IEEE Trans. Computer-Aided Design*, vol. 10, pp. 761–769, June 1991.

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

Florin Balasa (S'93–M'95) received the M.Sc. and Ph.D. degrees in computer science from the Polytechnical University of Bucharest, Bucharest, Romania, in 1981 and 1994, respectively. He received the M.Sc. degree in mathematics from the University of Bucharest in 1990 and the Ph.D. degree in electrical engineering from the Katholieke Universiteit Leuven, Leuven, Belgium, in 1995.

He is currently an Assistant Professor of Electrical Engineering and Computer Science at the University of Illinois, Chicago. He worked over seven years at the R&D Institute for Electronic Components, Bucharest, Romania. From 1990 to 1995, he worked at the VLSI System Design Methodology division, the Interuniversity Microelectronics Center (IMEC), Leuven, Belgium. In the fall of 1995, he joined as a Senior Design Automation Engineer the Advanced Technology division, Conexant Systems (former Rockwell Semiconductor Systems), Newport Beach, CA. He was also a Lecturer at the University of California, Irvine. He coauthored *Custom Memory Management Methodology: Exploration of Memory Organization for Embedded Multimedia System Design* (Norwell, MA: Kluwer Academic, 1998). His research interests include high-level synthesis, physical design, combinatorial optimization and mathematical programming.

Koen Lampaert was born in Roeselare, Belgium, in 1968. He received the M.Sc. and Ph.D. degrees in electrical engineering from the Katholieke Universiteit Leuven, Belgium, in 1992 and 1998, respectively.

From 1992–1996, he was a Research Assistant with the ESAT-MICAS laboratories of the Katholieke Universiteit Leuven, Leuven, Belgium, working in the field of analog design automation. Since 1996, he is working for Conexant Systems Inc., Newport Beach, CA, where he is responsible for physical design of analog, radio-frequency and high-performance digital integrated circuits. He has authored or coauthored one book and more than 20 papers in edited books, international journals, and conference proceedings.