

# Topological Placement with Multiple Symmetry Groups of Devices for Analog Layout Design

Karthik Krishnamoorthy  
Freescale Semiconductor, Inc.  
Phoenix, AZ

Sarat C. Maruvada  
Microsoft, Inc.  
Redmond, WA

Florin Balasa  
Univ. of Illinois at Chicago  
Chicago, IL

**Abstract**— This paper presents an improved topological algorithm for device-level analog placement with symmetry constraints. Based on the exploration of symmetric-feasible sequence-pairs [1], the technique employs an efficient model of priority queue [3]. The use of this data structure entails a complexity of  $O(G \cdot n \log \log n)$  for each code evaluation, where  $n$  and  $G$  are the numbers of devices and symmetry groups, which is better than the complexity of other existent topological placement algorithms supporting symmetry constraints. The computation times exhibited by this approach are significantly better than those of the algorithms using an exploration strategy based on the absolute representation, as well as those of other previous topological algorithms.

## 1. INTRODUCTION

In high-performance analog circuits it is often required that groups of devices are placed symmetrically with respect to one or several axes. Differential circuit techniques are used extensively to improve the accuracy, power supply rejection ratio, and dynamic range of many analog circuits. The full performance potential of many of these circuits cannot be achieved unless special care is taken to match the layout parasitics in the two halves of the differential signal path. Failure to match these parasitics in, for instance, differential analog circuits can lead to higher offset voltages and degraded power-supply rejection ratio [2]. The main reason of symmetric placement (and routing, as well) is to match the layout-induced parasitics in the two halves of a group of devices.

Placement symmetry can also be used to reduce the circuit sensitivity to thermal gradients. Some VLSI devices (the bipolar devices, in particular) exhibit a strong sensitivity to ambient temperature. If two such devices are placed randomly relative to the iso-thermal lines, a temperature-difference mismatch may result. 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.

A subset of cells is called a *symmetry group* if all the cells in the group are exhibiting a form of symmetry and, in addition, they all share a common symmetry axis. Note that all along this paper, the symmetry axes will be considered vertical since this is the typical way most layouts are designed. The symmetry constraints for a pair of rectangular devices ( $B_i, B_j$ ) in the  $k$ -th symmetry group have the form:  $(x_i + w_i) + x_j = 2 \cdot x_{axis_k}$  and  $y_i = y_j$ ,

where  $(x_i, y_i)$  are the left-bottom coordinates of the device  $B_i$ ,  $w_i$  denotes its width, and  $x_{axis_k}$  is the abscissa of the symmetry axis of the  $k$ -th group. Self-symmetric devices are cells having internal symmetry, their individual axes coinciding with the axis of the whole group. A self-symmetric device  $B_i$  must satisfy the constraint:  $x_i + w_i/2 = x_{axis_k}$ .

This paper presents an enhanced algorithm for solving analog placement problems exploring within a simulated annealing framework [2] a subset of sequence-pairs called *symmetric-feasible* [1], where the typical presence of symmetry groups of devices is *directly* taken into account during the exploration. The technique employs an efficient model of priority queue introduced by Johnson [3], achieving a complexity of  $O(G \cdot n \log \log n)$  per code evaluation, where  $n$  is the number of devices and  $G$  is the number of symmetry groups. The model of priority queue used in this paper was brought to the attention of the CAD community by Tang and Wong who used it for the computation of the longest common subsequence in a pair of weighted sequences in the evaluation of sequence-pairs [11]. Their technique influenced our algorithm; however, there are essential differences due to the presence of symmetry groups in the design.

The contributions of this paper are the following:

(1) the present evaluation algorithm exhibits a better complexity of the code evaluation than other existent topological algorithm supporting symmetry constraints: for instance, in [1, 10, 6] the evaluation complexity is quadratic and in [8] it is  $O(n \log n)$  for each symmetry group;

(2) different from the evaluation algorithm in [1] which sometimes could produce non-optimal placements in terms of area, the current algorithm was improved with a mechanism which keeps tightly the symmetry groups about their axes, without affecting the global complexity either.

The paper is organized as follows. Section 2 reviews the concept of *symmetric-feasible* (S-F) sequence-pair [1] and describes a fast evaluation algorithm in the presence of symmetry constraints. Finally, Section 3 presents an overview of the experimental results and Section 4 summarizes the conclusions of this research.

## 2. PLACEMENT WITH SYMMETRY CONSTRAINTS

### 2.1 Symmetric-feasible sequence-pairs

Dealing efficiently with symmetry constraints in the framework of topological representations implies addressing two problems: (a) how to recognize encodings complying with the given symmetry constraints, without building the corresponding layout, and (b)

how to restrict the exploration of the solution space of the representation only to such “symmetric-feasible” (S-F) codes.

Let  $(\alpha, \beta)$  be the sequence-pair of a placement configuration containing a number of symmetry groups (each group composed of pairs of symmetric devices and self-symmetric devices relative to a common vertical axis). Denoting  $\alpha_A^{-1}$  the position of a cell  $A$  in the sequence  $\alpha$  (as  $\alpha$  can be viewed as a one-to-one mapping,  $\alpha^{-1}$  is well-defined) and defining similarly  $\beta_A^{-1}$ , and also denoting  $\text{sym}(x)$  the symmetric cell of  $x$ , the sequence-pair  $(\alpha, \beta)$  is called *symmetric-feasible* (S-F) [1] if for any distinct cells  $x, y$  in any of the symmetry groups

$$\alpha_x^{-1} < \alpha_y^{-1} \iff \beta_{\text{sym}(y)}^{-1} < \beta_{\text{sym}(x)}^{-1} \quad (1)$$

The property (1) is a *sufficiency*<sup>1</sup> condition: it ensures the building of a valid placement in symmetry point of view, as it will be shown in Section 2.2. Intuitively, the property (1) prohibits the situation when two cells from distinct symmetric pairs are in an “above-below” topological relation, while their pairs are in the reverse relation, the two pairs preventing each other to align horizontally. It also prohibits the situation when two symmetric pairs are intertwined, rather than embedded, preventing each other to align vertically about the same symmetry axis.

The exploration of the solution space of placement problems with symmetry constraints can be reduced to the exploration of those sequence-pairs which are *symmetric-feasible*, i.e., satisfy property (1), relative to every symmetry group of cells. The benefit is a significant reduction in size of the search space – initially,  $(n!)^2$ . The magnitude of this reduction is given by the following

*Lemma* The number of symmetric-feasible sequence-pairs corresponding to a placement configuration with  $n$  cells and  $G$  symmetry groups, each group  $k$  containing  $p_k$  pairs of symmetric cells and  $s_k$  self-symmetric cells ( $k = 1, \dots, G$ ), is upper-bounded by  $\frac{(n!)^2}{(2p_1+s_1)! \cdots (2p_G+s_G)!}$  (no proof is given due to lack of space).

## 2.2 Evaluation algorithm for S-F sequence-pairs

This section will present an algorithm using the *longest common subsequence* (LCS) approach [11] building a placement *subject to symmetry constraints* from a given *symmetric-feasible* sequence-pair in each inner-loop iteration of the simulated annealing optimization. The analog devices to be placed on the chip area are represented by  $n$  rectangular blocks  $B_1, \dots, B_n$ , each block  $B_i$  having the width  $w_i$  and the height  $h_i$ , and having  $(x_i, y_i)$  as coordinates of its left-bottom corner. The algorithm presented in this section assumes for the time being that all the devices subject to symmetry constraints belong to a single symmetry group. The extension to multiple symmetry groups is addressed in Section 2.3 .

<sup>1</sup>Kouda *et al.* stated in [4] that in [1] it was claimed that condition (1) is “necessary and sufficient.” Their statement is untrue: since the sequence-pair typically presents redundancies, one can find sequence-pairs whose evaluation yield a valid placement in symmetry point of view, but still do not satisfy the property (1). We called the sequence-pairs having property (1) *symmetric-feasible* since they can certainly generate symmetrically-valid placements. But it must not be construed that a sequence-pair not satisfying property (1) is automatically symmetric-unfeasible.

(a) *The computation of the device ordinates.*

**Step 1y:** Given a sequence-pair  $(\alpha, \beta)$ , the maximal (i.e., which cannot be enlarged) subsequences common to  $\alpha^R$  (sequence  $\alpha$  reversed) and  $\beta$  represent paths [11] in the vertical constraint graph of  $(\alpha, \beta)$  [9]. If the cells are weighted with their heights, the longest common subsequence (LCS) represents the height of the whole placement. Based on these concepts, the device ordinates are computed using a priority queue model introduced by Johnson [3]. This data structure is implemented as a partially complete binary tree whose leaves correspond to a doubly linked list of queue elements called *buckets* [3]. Similarly to [11], each bucket in the priority queue has associated two keys (*index, length*), where the *index* key is the cell position in sequence  $\beta$  and *length* is the length of the LCS till that cell in the sequence-pair. Since the number of buckets is at most  $n + 1$ , the insertion and deletion operations take  $O(\log \log n)$  time [3]. Hence, the complexity of *Step 1y* is  $O(n \log \log n)$ .

The basic difference from [11] is that the equality of symmetric devices’ ordinates must be enforced, so the ordinates of symmetric cells are *simultaneously* updated. If a  $y$ -coordinate that was previously computed must be subsequently increased due to symmetry, then *Step 1y* must be repeated since, otherwise, some vertical topological constraints may be violated. *Step 1y* may need to be executed as many as  $\Theta(p)$  times, where  $p$  is the number of symmetric pairs in the group. The computation of the  $y$ -coordinates may hence take  $O(p \cdot n \log \log n)$  time. However, in most of the practical cases, no more than two iterations are necessary to fix the vertical symmetry and topological constraints.

(b) *The computation of the device abscissae.*

**Step 1x:** The first step, called *initialization*, computes the block abscissae leaving aside for the time being the symmetry constraints. Given a sequence-pair  $(\alpha, \beta)$ , the maximal subsequences common to  $\alpha$  and  $\beta$  represent paths in the horizontal constraint graph of  $(\alpha, \beta)$ . If the cells are weighted with their widths, the LCS represents the width of the block placement [11].

**Example:** Consider the S-F sequence-pair  $(F_1 E_1 B A_1 A_2 X E_2 Y D_1 Z C_1 C_2 D_2 F_2, F_1 Y D_1 Z C_1 C_2 D_2 E_1 A_1 A_2 B X E_2 F_2)$  relative to a symmetry group  $\{(A_1, A_2), B, (C_1, C_2), (D_1, D_2), (E_1, E_2), (F_1, F_2)\}$  consisting of five pairs of symmetric devices and one self-symmetric cell. The placement after the computation of the  $y$ -coordinates and the execution of *Step 1x* is shown in Fig. 1(a).□

**Step 2x:** Next, the position of the symmetry axis is chosen. Different from [1], we employ a scheme for selecting the symmetry axis and initializing the abscissae of the devices in the symmetry group such that the  $x$ -span of the group is kept minimal (relative to the topological constraints) by the end of the evaluation algorithm. This axis selection scheme is described below.

The general idea of this step is to align the individual axes of the innermost symmetric pairs and to position the other pairs to leave enough space (but not more than necessary) to satisfy the other horizontal topological constraints. First, a directed acyclic graph (DAG) is built from the S-F sequence-pair, showing the embedding relation between pairs along the  $Ox$  axis: each symmetric pair or self-symmetric device is a node in this DAG, the node of an

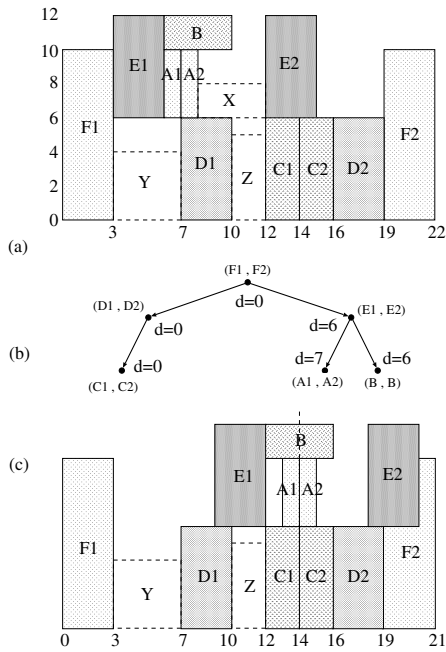


Figure 1: (a) Intermediate placement for the S-F sequence-pair in the illustrative *Example* at the end of *Step 1x*. (b) Construction of the DAG showing the embedding relations in the symmetry group. (c) Update of the device abscissae in the symmetry group at the end of *Step 2x*. (The possible overlaps will eventually be fixed.)

inner pair being the successor of an outer one. The DAG obtained for this *Example* is shown in Fig. 1(b).

Afterwards, the position of the symmetry axis is selected such that  $x_{axis} = \max \left\{ \frac{x_j + (x_k + w_k)}{2} \right\}$ , for all nodes without successors  $(B_j, B_k)$  in the DAG. In our *Example*,  $x_{axis} = 14$ , corresponding to the pair  $(C_1, C_2)$ . Then, each node without successors  $(B_j, B_k)$  will receive a value  $d = x_{axis} - \frac{x_j + (x_k + w_k)}{2} \geq 0$  and the abscissae of  $B_j$  and  $B_k$  computed at *Step 1x* will be updated:  $x_j = x_j + d$ ,  $x_k = x_k + d$ , thus shifted to the right. Proceeding bottom-up in the embedding DAG, each node will receive a  $d$  value equal to the minimum one among node's children (Fig. 1(b)); the abscissae of the node's cell(s) are similarly updated (Fig. 1(c)). The complexity of this step is dominated by the construction of the embedding DAG, which takes  $O((p + s) \log \log H)$  since it is done with a priority queue ( $p$  is the number of symmetric pairs,  $s$  is the number of self-symmetric cells, and  $H$  is the height of the placement).

**Step 3x:** This step is similar to the *initialization* step, but before inserting the bucket  $(l, x_j + w_j)$  into the queue, if  $(B_k, B_j)$  is a symmetric pair and  $d = 2x_{axis} - x_j - (x_k + w_k) > 0$ , then  $x_j$  is increased with  $d$ . For the pair  $(D_1, D_2)$ , for instance,  $d = 2 \times 14 - 16 - (7 + 3) = 2$ , therefore  $D_2$ 's abscissa will be increased with 2 becoming 18 (see Fig. 2(a)), such that  $(D_1, D_2)$  satisfy the symmetry constraint.

**Step 4x:** Some horizontal symmetry constraints may still be unsatisfied since some rightmost cells in the pairs may have been pushed further to the right: see the pair  $(E_1, E_2)$  in Fig. 2(a). Then,

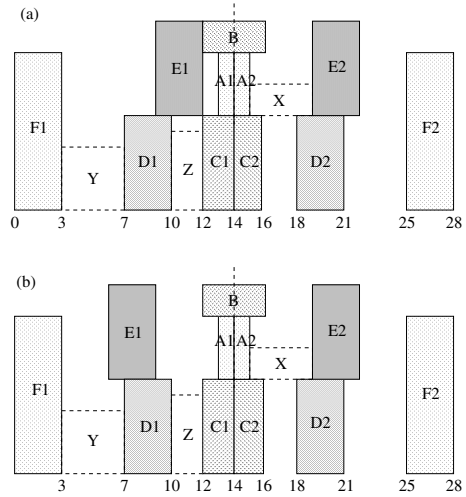


Figure 2: Placements (a) after *Step 3x* and (b) after *Step 4x* (final).

an additional traversal of the sequence  $\alpha$  in reverse order will fix all the symmetry constraints: if  $(B_j, B_k)$  is a symmetric pair and  $d = -2x_{axis} + x_j + (x_k + w_k) > 0$ , then  $x_j$  is decreased with  $d$ . Both *Step 3x* and *Step 4x* take  $O(n \log \log n)$  time. In the absence of symmetry constraints, the evaluation algorithm reduces to *Step 1y* and *Step 1x*, being identical to [11]. It can be proven that the evaluation algorithm yields a correct placement in both topological and symmetry point of view, and having also a minimum area.

### 2.3 Handling multiple symmetry groups

When dealing with an arbitrary number of symmetry groups, the technique is similar to the single-group case, but more traversals of the (S-F) sequence-pair are necessary. The symmetry groups are processed one by one, after each iteration the symmetry constraints relative to one symmetry group being fixed, and the relative positions of the devices in that group being "frozen." If  $G$  is the number of symmetry groups, the complexity of the evaluation algorithm is basically  $O(G \cdot n \log \log n)$ . In the case of embedded symmetry groups, the inner groups must be processed before the outer ones, therefore the order of the group processing does matter. (An embedding DAG of the symmetry groups is used to order the group processing.) But if the cells of the groups do not intermingle in the sequence-pair, then the processing order of the groups is irrelevant.

## 3. EXPERIMENTAL RESULTS

A placement tool for analog layout using selectable exploration algorithms has been implemented in C++ on a SUN Blade 100 workstation. This prototype tool can operate both with different topological representations (sequence-pairs, O-trees [10], and binary trees [8]) and different code evaluation algorithms. In addition, a complementary placement algorithm based on the traditional absolute representation has been embedded in the tool as well. Besides symmetry constraints, the tool handles systematically-induced device mismatches, alignment constraints, and also performs shape optimizations.

Design	Nr. cells	Symmetry groups	Absolute		O-trees [10]		Bin. Trees [8]		S-F Seq.-Pairs			
			Area	Time	Area	Time	Area	Time	Area		Time	
									[1]	JPQ	[1]	JPQ
dcservo_cmf6	66	5+4	67.7	31.0	60.5	14.8	60.4	3.6	61.2	54.0	16.9	3.1
modbias_2p4g	87	16+12+6+6+6	65.9	45.0	59.5	32.7	55.1	8.0	56.9	50.0	28.8	7.1
lnamixbias_2p4g	110	8+6+8+6+8+4	62.2	69.3	54.1	55.1	50.3	8.7	51.0	44.5	47.1	8.0
div_by_2or4	116	6+4+8+12+15	59.6	61.7	52.8	47.0	49.7	9.4	51.4	45.6	41.2	8.6

Table 1: Placement of analog blocks with symmetry constraints (Time [min], Area [ $10^3 \times \mu m^2$ ]). Column 3 shows the number of cells (self-symmetric or in symmetric pairs) in each group. E.g., the first design has two symmetry groups with 5 and, respectively, 4 devices.

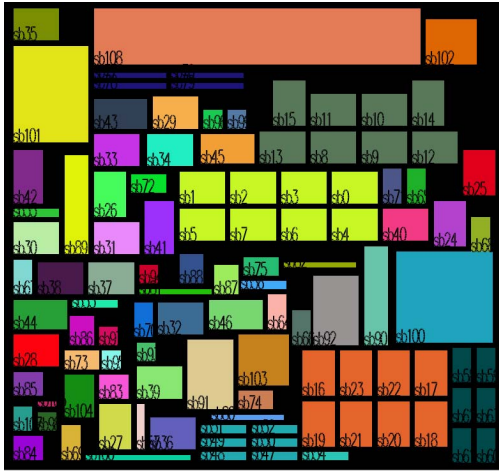


Figure 3: Placement with 6 symmetry groups of *lnamixbias\_2p4g*.

Table 1 displays the results of our experiments. The performance of the algorithm described in this paper (the columns marked with JPQ – from Johnson’s priority queue – in the header) has been evaluated in comparison to three other topological algorithms able to handle symmetry constraints: one using O-trees [10] having a quadratic complexity per code evaluation, a second one exploring symmetric-feasible (S-F) binary trees and using an evaluation technique based on red-black interval trees [8] (of complexity  $O(n \log n)$  per symmetry group), and the initial algorithm using S-F sequence-pairs [1] (of quadratic complexity). Tests with a traditional algorithm based on the absolute representation, similar to [2, 5, 7], have been performed as well. The benchmarks are analog blocks, components of a spread spectrum transceiver. Figure 3 displays the placement solution of one of these blocks.

The running times obtained with this enhanced evaluation techniques based on the highly efficient priority queue [3] are regularly better than the times obtained when using previous algorithms. For instance, the older algorithm [1] – whose evaluation complexity is quadratic – is several times slower. The experiments show that the exploration of the solution space of the placement problems is clearly more efficient when using topological representations, in contrast to the traditional absolute representation. Also the tuning of the simulated annealing optimizer was easier when using topological representations.

## 4. CONCLUSIONS

This paper has presented an improved analog placement technique operating on a subset of nonslicing topological representations of the layout, where symmetry constraints – typical in analog placement – are directly taken into account during the exploration of the solution space.

## References

- [1] F. Balasa, K. Lampaert, “Symmetry within the sequence-pair representation in the context of placement for analog design,” *IEEE Trans. CAD of IC’s and Syst.*, vol. 19, no. 7, pp. 721-731, 2000.
- [2] J. Cohn, D. Garrod, R. Rutenbar, L. Carley, *Analog Device-Level Automation*, Kluwer Acad. Publ., 1994.
- [3] D.B. Johnson, “A priority queue in which initialization and queue operations take  $O(\log \log D)$  time,” *Mathematical Systems Theory*, vol. 15, pp. 295-309, 1982.
- [4] S. Kouda, C. Kodama, K. Fujiyoshi “Improved method of cell placement with symmetry constraints for analog IC layout design,” *Proc. Int. Symp. Physical Design*, pp. 192-199, San Jose CA, April 2006.
- [5] K. Lampaert, G. Gielen, W. Sansen, “A performance-driven placement tool for analog integrated circuits,” *IEEE J. Solid-State Circ.*, vol. 30, no. 7, pp. 773-780, July 1995.
- [6] J.-M. Lin, Y.-W. Chang, “TCG-S: orthogonal coupling of P-admissible representations for non-slicing floorplans,” *Proc. 39th ACM/IEEE Design Aut. Conf.*, June 2002.
- [7] E. Malavasi, E. Charbon, E. Felt, A. Sangiovanni-Vincentelli, “Automation of IC layout with analog constraints,” *IEEE Trans. CAD of IC’s and Syst.*, vol. 15, no. 8, pp. 923-942, Aug. 1996.
- [8] S.C. Maruvada, K. Krishnamoorthy, F. Balasa, L.M. Ionescu, “Red-black interval trees in device-level analog placement,” *IEICE Trans. on Fundamentals of Electronics, Communications and Computer Sciences*, vol. E86-A, no. 12, pp. 3127-3135, Dec. 2003.
- [9] H. Murata, K. Fujiyoshi, S. Nakatake, Y. Kajitani, “VLSI module placement based on rectangle-packing by the sequence-pair,” *IEEE Trans. CAD of IC’s and Syst.*, vol. 15, no. 12, pp. 1518-1524, 1996.
- [10] Y.-X. Pang, F. Balasa, K. Lampaert, C.-K. Cheng, “Block placement with symmetry constraints based on the O-tree non-slicing representation,” *Proc. 37th ACM/IEEE DAC*, pp. 464-467, June 2000.
- [11] X. Tang, D.F. Wong, “FAST-SP: A fast algorithm for block placement based on sequence pair,” *Proc. Asia-S. Pacific Design Aut. Conf.*, pp. 521-526, Yokohama, Japan, 2001.