Block Placement with Symmetry Constraints based on the O-tree Non-Slicing Representation

Yingxin Pang† Florin Balasa‡ Koen Lamparter† Chung-Kuan Cheng†
†Dept. of CSE, University of California, San Diego, La Jolla, CA 92093
‡Dept. of EECS, University of Illinois, Chicago, Chicago, IL 60607-7053
††Conexant Systems Inc., 4311 Jamboree Road, Newport Beach, CA 92660

ABSTRACT
The ordered tree (O-tree) representation has recently gained much interest in layout design automation. Different from previous topological representations of non-slicing floorplans, the O-tree representation is simpler, needs less computation effort to generate a corresponding layout, and exhibits a smaller upper-bound of possible configurations. This paper addresses the problem of handling symmetry constraints in the context of the O-tree representation. This problem arises in analog placement, where symmetry is often used to match layout-induced parasitics and to balance thermal couplings in differential circuits. The good performance of our placement tool dealing with several analog designs taken from industry proves the effectiveness of our technique.

1. INTRODUCTION
The ability of placement tools to optimize complex layout-related objectives, while having the flexibility to handle a large variety of specific constraints, is crucial in order to automatically produce high-quality layouts in terms of density and electrical performance. In device-level analog placement dealing with symmetry constraints is essential, as analog circuits use very often differential architectures based on electrically symmetric networks. Symmetry is widely used in analog layout to match interconnection parasitics and device parameters, or to balance thermal effects.

The issue of symmetry has been addressed so far in the context of two distinct classes of analog placement solutions. The first class of tools, exploring the absolute representation of placement configurations with simulated annealing algorithms, has proven to be successful when dealing with industrial examples [2],[5-7]. However, they may converge slowly, due to the huge size of the search space which contains also infeasible placement configurations (as cells are allowed to overlap). In addition, the tuning of these tools is usually difficult, requiring a significant amount of testing effort.

The second class of solutions employs topological representations of placement configurations, where cell positions are specified based on encoded topological relations. In the ILAC system [9], symmetry is handled in a slicing floorplan model. Recently, several non-slicing topological representations have been proposed. Symmetry constraints can be efficiently handled within the sequence-pair representation as shown in [1].

The O-tree representation [4] has recently gained an increasing interest: different from the other topological representations, O-tree needs a smaller amount of encoding storage and linear time computation effort to generate each placement configuration. In addition, the upper-bound of possible encodings is smaller, which entails a reduced representation redundancy. These important advantages are strong incentives to address the symmetry constrained placement problem in the context of the O-tree representation.

In this paper, we present a novel placement technique based on the O-tree representation, in the presence of positioning and symmetry constraints. The good performance of our placement tool when dealing with several analog designs taken from industry proves the effectiveness of our approach.

2. BRIEF OVERVIEW OF THE O-TREE REPRESENTATION
An n-node O-tree is a tree with n+1 nodes encoded by (T, π), where T is a 2n-bit string that identifies the branching structure of the tree, and π is a permutation of the n node labels (excluding the root).

When traversing the tree, we write a '0' for descending an edge and a '1' for subsequently ascending that edge. Given the 7-node O-tree in Figure 1(a), we can represent it as T= 00101001010110011, π=abcdefgh.

An O-tree where nodes represent rectangular blocks imposes both horizontal and vertical positioning constraints:
(a) each parent block must be placed to the left of its children;
(b) if two blocks are overlapping along their x-coordinate projection, the block having a higher index in permutation π must be placed on top of the one having a lower index.

Figure 1(b) displays the minimum area placement of the O-tree(a).

Figure 3: An O-tree and its corresponding block placement
3. RECTANGLE PACKING WITH SYMMETRY CONSTRAINTS IN THE CONTEXT OF THE O-TREE REPRESENTATION

Symmetry constraints can be formulated in terms of symmetry pairs and symmetry groups. A symmetry pair is a couple of blocks having the same dimensions, which have to be placed symmetrically with respect to an axis. A symmetry group is a set of symmetry pairs which share a common axis. Assuming the common axes are horizontal in all symmetry groups, then the symmetry constraints can be formulated as follows: if \((a_1, b_1), (a_2, b_2), \ldots, (a_k, b_k)\) is a symmetry group, and \((a_i, b_i)\) is a symmetry pair, for \(i = 1, \ldots, k\) then

\[
\frac{x_{a_i}}{y_{a_i}} = \frac{x_{b_i}}{y_{b_i}} \tag{1}
\]

\[
y_{a_i} + y_{b_i} + h_{a_i} = 2y_s \tag{2}
\]

where \(y_s\) defines the position of the common symmetry axis.

The first type equalities are called horizontal symmetric constraints; the second type are vertical symmetric constraints.

3.1 Symmetric X-feasible O-trees

If an O-tree can lead to a placement which satisfies both the horizontal positioning constraints and the horizontal symmetric constraints (1), then the O-tree is called symmetric x-feasible.

In order to determine whether an O-tree is symmetric x-feasible or not, an horizontal constraint graph \(G_x\) is built. This directed graph has the same nodes as the given O-tree, and the same directed edges as horizontal positioning constraints. In addition, for each symmetric pair of blocks \((a, b)\), two new arcs \((a, b)\) and \((b, a)\) are introduced (unless positioning constraint paths between \(a\) and \(b\) already exist).

The edges of \(G_x\) are weighted: if the edge \((u, v)\) corresponds to a positioning constraint, its weight is \(w(u, v) = w_u\), the width of the block represented by node \(u\); if the edge corresponds to a symmetric constraint, then \(w(u, v) = 0\).

The existence of positive cycles in the directed graph \(G_x\) prevents the symmetric x-feasibility of the O-tree.

**Theorem 3.1.1:** If the horizontal constraint graph \(G_x\) does not contain positive cycles, then the corresponding O-tree is symmetric x-feasible. In addition, one can build a minimum width placement satisfying both horizontal positioning and symmetric constraints in \(O(n^2)\) time.

**Proof:** In a minimum width placement the x-coordinate of each block must be equal to the longest path length from the "root" node to its corresponding node in \(G_x\). Finding a minimum width placement results to be a single-source longest path problem, which can be solved with the Bellman-Ford algorithm [3]. If \(G_x\) contains no positive cycles, the longest path problem is provably consistent and the Bellman-Ford algorithm converges.

As the x-coordinates of blocks are longest path lengths, the horizontal positioning constraints being satisfied, the horizontal symmetric constraints (1) are satisfied as well.

The complexity of the Bellman-Ford algorithm is \(O(|V|E)\), where \(V\) is the number of nodes and \(E\) is the number of edges [3]. Since the O-tree has \(n\) edges, the number of edges in \(G_x\) is \(O(n)\). Taking also into account that \(G_x\) has \(n+1\) nodes, the complexity results to be \(O(n^2)\).

**Remark:** The proof of Theorem 3.1.1 relies on the Bellman-Ford algorithm [3]. In practice, it is possible to exploit the special structure of the O-tree and perform the same tasks in linear time. The use of Bellman-Ford algorithm in the proof of Theorem 3.1.1 is due only to clarity reasons.

![Figure 2: A symmetric x-feasible O-tree and its placement](image)

**Example:** Assuming the pairs of blocks \((b, h)\) and \((e, f)\) are symmetric, the horizontal constraint graph \(G_x\) of the O-tree displayed in Figure 1(a) can be visualized in Figure 2(a), where the broken arcs model the horizontal symmetric constraints (1). Figure 2(b) displays the minimum width placement.

**Theorem 3.1.2:** If there is a positive cycle in \(G_x\), then the O-tree is not symmetric x-feasible.

**Proof:** If \(G_x\) contains a positive cycle, this cycle must include at least one newly added edge between nodes corresponding to symmetric blocks. If \((a, b)\) is such a directed edge of zero weight, it follows that \(x_a > x_b\), as there is a path of positive length from \(b\) to \(a\) formed by all the edges of the given positive cycle, excepting \((a, b)\). It follows that the O-tree is not symmetric x-feasible and the horizontal symmetric constraints conflict with the O-tree constraints.

Theorems 3.1.1 and 3.1.2 prove the equivalence between the existence of positive cycles in the horizontal constraint graph \(G_x\) and the O-tree property of symmetric x-feasibility.

3.2 Symmetric Y-feasible O-trees

If an O-tree can lead to a placement which satisfies the vertical symmetric constraints (2) and, at the same time, which does not violate the vertical positioning constraints (see Section 2), then the O-tree is called symmetric y-feasible.

In order to determine whether an O-tree is symmetric y-feasible or not, we construct a vertical constraint graph \(G_y\).

This directed graph has the same nodes as the given O-tree. Part of the directed edges represent the positioning vertical constraints of the O-tree. These edges can be determined as follows:

for each block \(B_i\), \(i = 1, \ldots, n\), let \(\psi(i)\) be the set of block indexes \(k\) lower than \(i\) in permutation \(\pi\), which spanning intervals \((x_k, x_k + w_k)\) overlap \((x_i, x_i + w_i)\); if \(\psi(i)\) is non-empty, for each \(k \in \psi(i)\) introduce in graph \(G_y\) a directed edge \((B_k, B_i)\), unless there is no other edge \((B_k, B_j)\), with \(j \in \psi(i)\) (in order to disregard unnecessary transitive arcs). In order to handle the vertical symmetric constraints (2), for each edge \((B_k, B_i)\), we define the weight as \((h_i+h_j)/2\).

In this way, the nodes will correspond to the center of the blocks rather than the bottom of the blocks. Additional edges modeling the vertical symmetric constraints (2) must be added in \(G_y\) as well. Assum-
ing that all the symmetry pairs \((a_i, b_i)\) have their elements in topological order relative to the positioning constraints, the additional edges are introduced as follows:

for every symmetry group \(\{(a_1, b_1), (a_2, b_2), \ldots, (a_k, b_k)\}\)

for every two symmetry pairs \((a_i, b_i), (a_j, b_j)\)

if there is a path from \(a_i\) to \(a_j\) then \(\}

add to \(G_j\) a directed edge \((b_j, b_i)\), unless it exists already

let weight\((b_j, b_i)\) = the longest path length from \(a_i\) to \(a_j\)

if there is a path from \(b_i\) to \(b_j\) then \(\}

add to \(G_j\) a directed edge \((a_i, a_j)\), unless it exists already

let weight\((a_i, a_j)\) = the longest path length from \(b_i\) to \(b_j\)

Example: Assuming the pairs of blocks \((b_1, h)\) and \((c_1, f)\) are symmetric, the vertical constraint graph \(G_y\) of the O-tree displayed in Figure 1(a) can be visualized in Figure 3(a). The plain arcs represent the O-tree vertical positioning constraints. The broken arc \((b_1, e)\) was introduced during the execution of the procedure shown above, as there was a path from node \(f\) to node \(h\). This arc was added to model the vertical symmetry constraints, as it will be explained in the proof of Theorem 3.2.1.

![Figure 3: A symmetric y-feasible O-tree and its placement](image)

The existence of cycles in the directed graph \(G_y\) prevents the symmetric y-feasibility of the O-tree.

**Theorem 3.2.1:** If the vertical constraint graph \(G_y\) does not contain cycles, then the corresponding O-tree is symmetric y-feasible. In addition, one can build a minimum height placement satisfying both vertical positioning and symmetric constraints in \(O(n^2)\) time.

**Proof:** Denoting \(x_j, y_i\) the coordinates of the left-bottom corner of block \(B_i\), \(z_j = y_j + h_j/2\) the y-position of the center of \(B_i\), and given a symmetry group \(U=\{(a_1, b_1), (a_2, b_2), \ldots, (a_k, b_k)\}\), a placement which has the properties stated in Theorem 3.2.1 can be constructed as follows:

1. Let \(z_{\text{min}}=0\) and \(z_j=\text{the longest path length from root to node } B_j\).
2. Determine the position of the common symmetry axis

   \[ y_s = \max_{(a_i, b_i) \in U} \left( \frac{z_{a_i} + z_{b_i}}{2} \right) \]

3. Execute a topological sort of the nodes in \(G_y\).

   Reorder the symmetry pairs \((a_i, b_i)\) in \(U\) such that nodes \(b_i\) are ordered according to the topological sort

4. For each symmetry pair \((a_i, b_i)\) in \(U\)

   \[ d = y_s - \frac{z_{a_i} + z_{b_i}}{2} \]

if \(d > 0\) then \(\}

update \(z_{b_i} = z_{b_i} + d\)

execute the Bellman-Ford single-source longest path algorithm [3], considering node \(b_i\) as the source \(\}

Since the y-positions of the blocks are adjusted (Step 4) according to the topological order, the y-positions of the early symmetry pairs are not affected by the adjustments involving symmetry pairs processed later. After each iteration in Step 4, the current pair \((a_i, b_i)\) will get y-coordinates of block centers symmetric relative to the position \(y_s\) of the common axis. One by one, all vertical symmetric constraints are satisfied. Taking into account that the vertical constraint graph \(G_y\) was built such that the positioning constraints are inherently satisfied, the O-tree results to be symmetric y-feasible. As the position \(y_s\) of the symmetry axis has the minimum possible value, the final placement has a minimum height.

Steps 1 and 4 contain the most expensive operations. Due to the Bellman-Ford algorithm, step 1 requires \(O(n^2)\) time [3]. Step 4 has \(k\) iterations; in each iteration the single-source longest path algorithm may be executed. Actually, each edge in \(G_y\) is examined no more than once and, therefore, the complexity is \(O(n^2)\). In conclusion, the algorithm described above runs in \(O(n^2)\) time.

**Theorem 3.2.2:** If there is a cycle in \(G_y\), then the O-tree is not symmetric y-feasible.

**Proof:** For every symmetry pair \((a_i, b_i)\), as we may assume \(x_a = x_b\), there is a path between the nodes \(a\) and \(b\) containing only edges corresponding to positioning constraints. Let \((a_1, b_1)\) and \((c, d)\) be two symmetry pairs belonging to the same group, and suppose there is such a path from \(a\) to \(c\) and another one from \(c\) to \(d\). A cycle in \(G_y\) must contain a "broken" arc (corresponding to a vertical symmetric constraint), as the positioning constraints determine an acyclic graph. Let \((a, b)\) be such a "broken" arc contained in a cycle: it follows there is a path from node \(b\) to node \(d\) in \(G_y\) and, consequently, \(z_b > z_d\). But this "broken" arc \((a, b)\) could be introduced in \(G_y\) only due to the existence of a path from node \(a\) to node \(c\). This implies \(z_a > z_c\), which together with the other inequality \(z_c > z_b\), implies that the symmetry pairs cannot have a common symmetry axis (as \((z_a + z_b)/2 > (z_c + z_b)/2\) ), which contradicts the symmetry group assumption.

### 3.3 Symmetric Feasible O-trees

An O-tree is called symmetric feasible if it is both symmetric x- and y-feasible. Given a rectangular blocks, a set of symmetric constraints, and a symmetric feasible O-tree, one can build a minimum area placement as described in subsections 3.1 and 3.2, where the positions of the O-tree is the following O-tree constraints and the given symmetric constraints. On the other hand, there is a reciprocal property.

**Theorem 3.3.1:** A minimum area rectangle packing with symmetric constraints can be represented by a symmetric feasible O-tree.

Our placement tool employs the simulated annealing algorithm as the exploration engine for the O-tree search space. According to theorem 3.3.1, only the symmetric feasible O-trees are taken into account, as they lead to placement configurations satisfying the symmetry constraints. The test of symmetric feasibility and the placement construction can be done simultaneously.
4. EXPERIMENTS

The placement algorithm described in this paper has been implemented in C on a SUN ULTRA-60 workstation.

Table 1 displays the placement results obtained from several test cases. SP denotes symmetry pairs, SS denotes self-symmetric cells. The self-symmetric cells are cells presenting geometrical symmetry and sharing the same (horizontal) axis with the other symmetry pairs in the group. The smallest example having 14 cells runs in 10 seconds, while the largest example - with 110 cells - has been successfully placed in about 25 minutes.

Figure 4 shows the block placement of a circuit containing a symmetry group, consisting of 3 pairs of symmetric devices and 2 self-symmetric cells, which can be noticed near the bottom of the figure. The 47 block example has been processed in 4.37 minutes. The layout area is only about 12% larger than the total area of all the blocks in the circuit, which proves the good rectangle packing capability of the placement tool.

Figure 5 shows the layout (after placement) of another circuit characterized by a larger number of symmetric constraints. Our placement tool has processed this example (having 11 symmetry pairs and 1 self-symmetric cell) in only 5.21 minutes. The speed exhibited by the tool can be justified as follows: the upper-bound of the number of configurations in the O-tree representation [4] is smaller than the upper-bounds in other topological representations.

Our experiments suggest that the speed performance of our placement tool based on the O-tree representation is better relative to similar tools where placement configurations are represented employing absolute coordinates [5], or tools based on the sequence-pair topological representation [1].

5. CONCLUSIONS

This paper has addressed the problem of taking into account symmetry constraints when non-slicing floorplans are represented with O-tree structures. The necessity of handling symmetry emerges when, for instance, the O-tree topological representation is applied to device-level placement for analog layout. The good performance of our ordered tree-based placement tool when dealing with several analog designs taken from industry substantiates the effectiveness of our novel techniques.

6. REFERENCES