Symmetry-Aware Placement with Transitive Closure Graphs for Analog Layout Design

Lihong Zhang
ECE, Memorial University of Newfoundland
St. John's, NL, A1B 3X5, Canada

C.-J. Richard Shi
Electrical Engineering Department
University of Washington
Seattle, WA 98105, USA

Yingtao Jiang
ECE, University of Nevada
Las Vegas, NV
89154-4026, USA

Abstract - A new scheme is proposed to use transitive closure graph (TCG) to explore the full symmetry solution space in analog layout design. We define a set of TCG symmetric-feasible conditions and show that it is extremely useful in reducing the solution space. A method is presented for generating random symmetric-feasible TCGs in *O(n)* time preserving the TCG closure property. Experimental results have confirmed the effectiveness of the proposed symmetry-aware TCG placement algorithm.

I. Introduction

Logical and physical synthesis tools revolutionized digital IC designs over the past two decades, resulting in a huge improvement in design productivity and chip functionality. The need for analog synthesis is particularly important for ever growing mixed-signal system-on-a-chip (SoC) designs that comprise both digital and analog circuitry. Unfortunately, analog synthesis is yet a far more difficult operation, as it has to explore a much larger design space. Therefore, up to date, the analog-design portion of a mixed-signal chip has to be still routinely hand-crafted by experienced human designers, which costs extraordinarily disproportional amount of effort and time compared to only a small fraction of the chip area.

The analog device-level or macro-cell *placement*, whose purpose is to assign exact locations of various circuit components within the chip, is one of the most significant stages in the analog layout synthesis [1][2]. The core of the placement is the rectangle *packing* problem: given a set of rectangular cells with arbitrary size, place them without overlap on a plane within a rectangle of minimum size under certain constraints. There are two general placement strategies, absolute placement and relative placement. The absolute placement [2]-[4] used to be considered as the most effective solution to the analog placement problem for decades. Within the recent half a decade, a few new representations for relative placement [5][9], also called topological placement, have been proposed to solve the analog placement problem. Their applications have been changing the traditional views on how topological representations can help satisfy special constraints in analog layout design.

However, recent studies have shown that those representations are partial in exploring feasible solution space. In this paper, we attempt to apply another topological representation, i.e., transitive closure graph (TCG) [6], to the analog placement problem. We will illustrate how TCG can be used to handle the complex constraints pertaining to analog layouts. In particular, we shall show how to efficiently explore the symmetry solution space using the TCG representation.

The rest of the paper is organized as follows. Section II reviews prior work. In Section III, the TCG symmetric feasibility conditions are defined. In Section IV, we discuss how to construct a symmetric placement from a symmetric-feasible TCG. Section V describes a strategy for generating random symmetric-feasible TCG representations while keeping TCG valid. The experimental results are listed in Section VI and the conclusion is drawn in Section VII.

II. Prior Work and Our Focus

Murata *et al.* [7] proposed sequence-pair (SP) as a general non-slicing floorplan representation. After that, quite a few other topological representations, e.g., Bounded-Sliceline Grid (BSG), O-tree, B*-tree, Corner Block List (CBL), TCG [6], Q-sequence and TCG-S [8], have been developed for digital layout design.

In the analog layout domain, quite a few well-known layout automation systems, such as [2]-[4], applied the absolute-coordinate representation in their placement tools. Balasa and Lampaert [5] applied SP representation in the context of symmetry placement for analog designs. Afterwards, O-tree, B*-tree and TCG-S [9] were employed to satisfy the symmetry constraints in the analog placement problem. These investigations have clearly demonstrated that the topological representations are capable of handling very complex analog constraints.

In an analog circuit, there are two typical symmetry cases: i) *symmetric pairs*: a pair of cells have identical geometry and mirror (or identical) orientations placed on the opposite sides of a common axis; ii) *self-symmetric*: a cell can be partitioned into two symmetric halves, and it shares the same axis with other symmetric devices. In Fig. 1(a), (d, d') and (e, e') are symmetric pairs, while b and f are self-symmetric cells. The symmetric pairs and self-symmetric cells with respect to a common symmetry axis form a *symmetry group*. An analog layout may contain multiple symmetry groups. In this paper, our analysis is only focused on a symmetry group. Without loss of generality, the symmetry axis is assumed vertical.

According to the classification defined in [6], O-tree, B*-tree, CBL and Q-sequence intrinsically have a smaller solution space and low packing cost due to lack of the definition of geometric relationship between each pair of cells. On the other hand, the representations, such as BSG, SP, TCG and TCG-S, can represent definite geometric relationship between each pair of cells. However, BSG incurs many redundancies, which lead to a much larger solution space. SP encodes any placement as an ordered pair of sequences (α, β), which can be called α-sequence and β-sequence, respectively. Here we use α_a^{-1} (β_a^{-1}) to denote the position of cell a in α–sequence (β–sequence). According to the definition of SP [7], the topological relationship between two cells a and b can be derived from an SP:

- if $\alpha_a^{-1} < \alpha_b^{-1}$ and $\beta_a^{-1} < \beta_b^{-1}$, cell a is to the left of cell b;
- if $\alpha_a^{-1} < \alpha_b^{-1}$ and $\beta_b^{-1} < \beta_a^{-1}$, cell a is on the top of cell b.

As an example, the corresponding SP of the placement in Fig. 1(a) is (d, a, f, e, e', d', b, c) (a, b, d, e, e', f, c, d').

978-1-4244-1922-7/08/$25.00 ©2008 IEEE

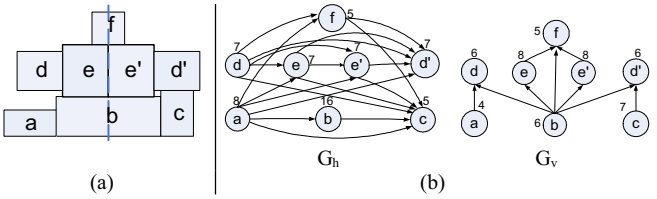


Fig. 1. (a) a placement, and (b) its corresponding TCG representation. (d, d') and (e, e') are symmetric pairs, while b and f are self-symmetric cells.

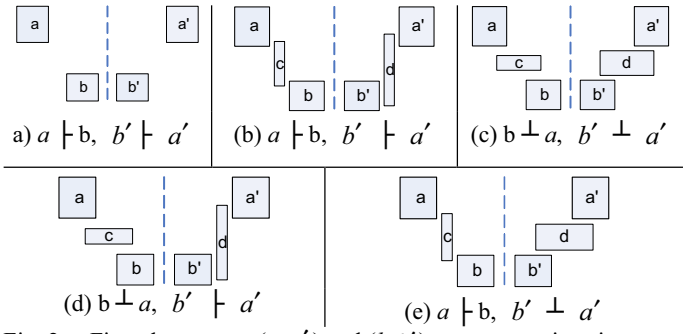


Fig. 2. Five placements, (a, a') and (b, b') are symmetric pairs.

In [5], an SP symmetric-feasibility condition is proposed as follows. Assume Γ is a symmetry group. An SP (α, β) is called symmetric-feasible if Eq. (1) holds:

$$\alpha_a^{-1} < \alpha_b^{-1} \text{ and } \beta_{b'}^{-1} < \beta_{a'}^{-1}, \forall (a,a') \in \Gamma, (b,b') \in \Gamma, a \neq b. \quad (1)$$

According to the condition above, the symmetric cells should appear in a mirror form in two sequences of SP. But this observation is partial [10]. For instance, the SP of a symmetric placement depicted in Fig. 2(b) is (a, c, b, b', d, a') (a, c, b, b', d, a'), which seems in line with the proposed Eq. (1). Here, (a, a') and (b, b') are two symmetric pairs. However, although another placement depicted in Fig. 2(d) (similarly for Fig. 2(e)) is also symmetric for (a, a') and (b, b'), its SP of (a, c, b, b', d, a') (b, c, a, b', d, a') does not follow Eq. (1). Thus, searching in a sub-set of the symmetric-feasibility space as [5] may lead to a non-optimal solution.

Recently TCG-S was proposed to realize the symmetry placement [9]. However, the proposed symmetry constraints in [9] are only of the same capability as Eq. (1). The placement shown in Figs. 2(d) or 2(e) cannot be accessed with that proposed TCG-S method. And the symmetric cells are only limited to symmetric pairs (i.e., excluding self-symmetric cells). In Table I, we summarize the features of different approaches, including SP, TCG-S and TCG (this work). The third column *Completeness* means if a complete solution space can be explored by each individual approach.

S. Kouda proposed an improved method of cell placement with symmetry constraints [10]. The SP used in that method is only for the perturbation purpose. No symmetry condition on representation itself is deployed. Instead, any perturbed solutions represented in SP are converted to linear equations/inequalities and finally solved with linear programming. In spite of generality, that proposed method may experience inefficiency due to the time-consuming feature of linear programming. More recently, Y. Tam *et al.* proposed a placement algorithm considering symmetry and other constraints simultaneously [11]. Since that work employs Eq. (1), the limitation of only exploring part of symmetric solution space is inevitable, as [5][9]. Moreover, although SP is used as the representation, constraint graphs have to be derived in order to append edges or adjust edge weights for inclusion of symmetry and other constraints.

Table I: Time complexity comparison of different topological representations in the context of symmetric-aware placement.

Approaches		*Packing*	*Perturbation*	*Completeness*
SP	General	*O(nlglgn)*	*O(1)*	*Yes*
	Symm.	*O(n^2)* [5]	*O(1)*	*No*
TCG-S	General	*O(nlgn)* [8]	*O(n)*	*Yes*
	Symm.	*O(n^2)* [9]	*O(n^2)*	*No*
TCG	General	*O(n^2)* [6]	*O(n^2)*	*Yes*
	Symm. (this work)	*O(n^2)*	*O(n)*	*Yes*

Although SP and TCG exhibit an inherent equivalence to each other [8], their distinctions are exhibited in the following aspects. For SP, many of the theoretical results from the studies of sequence permutation/combination can be applied to ease the analysis of the related solution space. Meanwhile, as only sequences are included in the SP representation, additional constraint graphs have to be resorted to in order to handle placement with constraints (e.g., boundary constraints, pre-placed cells, etc). For TCG, the explicit geometric relationship between cells makes it possible that complex constraints pertaining to the analog circuits can be evaluated and/or satisfied at the graph level even before the real packing operation takes place. Moreover, in spite of a topological representation, TCG has a potential to handle device merging or device separation constraints that can help increase the layout density or minimize the induced parasitic effects.

With those considerations, in this paper, we shall study the symmetry placement using TCGs, and focus on the exploration of the placement solution space with device-symmetry constraints.

III. Symmetric-Feasible TCG

A. TCG Representation

Consider a directed acyclic graph $G = (V, E)$, where V is the set of vertices, and E is the set of edges. The *transitive closure* of G is defined as the graph $G' = (V, E')$ with $E' = (v_i, v_j)$ iff ∀ v_i, $v_j \in V$, there is a path from vertex v_i to vertex v_j in G.

A TCG representation of a placement consists of two graphs: *horizontal transitive closure graph* G_h and *vertical transitive closure graph* G_v [6]. In a graph, a vertex (e.g., v_i) corresponds to a cell (e.g., c_i). Throughout the paper, the terminologies of vertex and cell can be used interchangeably. In G_h (G_v), a directed edge <v_i, v_j> represents that cell c_i is to the left of (below) cell c_j, denoted as $c_i \vdash c_j$ ($c_i \perp c_j$). For instance, Fig. 1(a) depicts a placement with eight cells. Fig. 1(b) shows the corresponding representation $TCG=(G_h, G_v)$.

The *weight* associated with a vertex in G_h (G_v) corresponds to the width (height) of the corresponding cell, and edge <v_i, v_j> in G_h (G_v) denotes the horizontal (vertical) relationship between c_i and c_j. For an edge <v_i, v_j> in G_h (G_v), v_i (c_i) is called a *fan-in vertex* (*fan-in cell*) of v_j (c_j), while v_j (c_j) is called a *fan-out vertex* (*fan-out cell*) of v_i (c_i).

B. TCG and SP

As shown in [8], given a TCG, its corresponding SP can be constructed in $O(n^2)$ time. In the following, we show how this can be performed in $O(n)$ time. Consider a TCG. Every two vertices are connected by one and only one directed edge,

either in G_h or G_v. An SP is composed of α-sequence and β-sequence. If there is a directed edge $<v_i, v_j>$ in G_h, the corresponding SP will be (v_i, v_j) (v_i, v_j); while if there is a directed edge $<v_i, v_j>$ in G_v, the corresponding SP will be (v_j, v_i) (v_i, v_j). Following the observations above, we can see that the vertices in α-sequence are ordered incrementally according to the sum of in-degree in G_h and out-degree in G_v of each vertex, and the vertices in β-sequence are ordered incrementally according to the sum of in-degrees in both G_h and G_v of each vertex. So we have:

Lemma 1: Given a TCG, its corresponding SP can be constructed in *O(n)* time, where *n* is the number of cells.

Proof: Based on the operation described above, since it takes *O(1)* to obtain in-degree and out-degree of a vertex in TCG, the time complexity of the operation above is *O(n)*. ∎

As an example, in Fig. 1, in-degrees of *d*, *a*, *f*, *e*, *e′*, *d′*, *b*, *c* in G_h are 0, 0, 2, 2, 3, 5, 1, 6, respectively. And out-degrees of *d*, *a*, *f*, *e*, *e′*, *d′*, *b*, *c* in G_v are 0, 1, 0, 1, 1, 0, 5, 1, respectively. Thus, based on the sum of two arrays above, we can obtain the corresponding α-sequence of (*d*, *a*, *f*, *e*, *e′*, *d′*, *b*, *c*). Similarly, the corresponding β-sequence of (*a*, *b*, *d*, *e*, *e′*, *f*, *c*, *d′*) can also be derived.

C. TCG Symmetric-Feasibility Conditions

Let (G_h, G_v) be the TCG representation of a placement containing a symmetry group Γ. For $(a,a')\in\Gamma$, if $a\neq a'$, (a,a') is a symmetric pair consisting of two distinct cells *a* and a', and if $a=a'$, *a* (or equivalently a') is a self-symmetric cell.

Definition 1: For $(a,a')\in\Gamma$ and $(b,b')\in\Gamma$, a TCG (G_h and G_v) representation is symmetric-feasible if both of the following conditions are satisfied:

$$\text{in } G_h\text{: } a \vdash b \;\leftrightarrows\; a' \vdash b', \qquad (2)$$

$$\text{in } G_v\text{: } a \perp b \;\leftrightarrows\; b' \perp a', \qquad (3)$$

where ⇆ denotes that the two cases before and after this symbol cannot simultaneously appear in the same TCG. As an example, the TCG depicted in Fig. 1(b) is symmetric-feasible. Based on Eqs. (2) and (3) in Definition 1, we have the following observations:

(i) if *b* is a self-symmetric cell (i.e., $b=b'$), we have:
in G_h, $a \vdash b \;\leftrightarrows\; a' \vdash b$; in G_v, $a \perp b \;\leftrightarrows\; b \perp a'$.

(ii) if both *a* and *b* are self-symmetric cells (i.e., $a=a'$, $b=b'$), we have: in G_v, $a \perp b$ or $b \perp a$.

(iii) if $a=b'$ and $a'=b$, we have: in G_h, $a \vdash a'$ or $a' \vdash a$.

The statement above governs the relationship between two cells in a symmetric pair.

Now we show that a constructed TCG can satisfy the symmetric-feasibility conditions defined in Definition 1.

Lemma 2: Any placement containing a symmetry group can be represented with a symmetric-feasible TCG.

Proof: The relationship among multiple symmetric cells can be viewed between each two symmetric pairs/cells. There are all together four cases for two symmetric pairs/cells: (1) two symmetric pairs; (2) one symmetric-pair and one self-symmetric cell; (3) two self-symmetric cells; (4) two cells in one symmetric pair. Once we analyze case (1), the rest of cases can be studied in a similar way.

For any of two symmetric pairs $(a,a')\in\Gamma$ and $(b,b')\in\Gamma$, there are three possible relative positions:

```
Y-Dimensional Symmetric Packing
Begin
1   construct the topological order of the TCG; // Lemma 1
2   calculate the vertical longest path of the TCG based on the
    topological order;
3   for (each cell c_i in the topological order) {
4       if (c_i has a symmetric counterpart c_j) {
5           if (ΔY = c_i.y - c_j.y < 0) {
6               shift c_i and its G_v fan-out cells for -ΔY; }
7           else if (ΔY = c_i.y - c_j.y > 0 ) {
8               shift c_j and its G_v fan-out cells for ΔY; }}}
End
```

Fig. 3. Y-dimensional symmetric packing flow.

(i) (a,a') sit within (b,b') along the horizontal direction, and their vertical projections overlap. Thus, no relationship in G_v exists, and in G_h: $b \vdash a$ and $a' \vdash b'$. Clearly, they are in harmony with Eq. (2) in Definition 1.

(ii) (a, a') sit below (b, b') along the vertical direction, and their horizontal projections overlap, respectively. Thus, there is no relationship in G_h, and in G_v: $a \perp b$ and $a' \perp b'$. They are in line with Eq. (3) in Definition 1.

(iii) if both horizontal and vertical projections of (a, a') and (b, b') do not overlap, according to the definition of TCG [6], the horizontal relationships dominate (unless there exists a chain of vertical relationships in between), as indicated in Fig. 2(a) or Fig. 2(b), which can be handled in the same way as case (i). Otherwise, there exist three other possible cases, as depicted in Figs. 2(c)-(e), where edge relationships in the corresponding TCGs are also marked. It can be seen that there is no violation from the conditions in Definition 1.

Thus, for any of two symmetric pairs, Eqs. (2) and (3) in Definition 1 always hold. The same conclusion can be drawn if we study cases (2)-(4) in a similar way. ∎

IV. Symmetric-Feasible TCG and Symmetric Placement

In this section, we will study how to construct a symmetric placement from a symmetric-feasible TCG. We shall introduce a complete X/Y dimensional packing strategy to satisfy positioning and symmetry constraints based on a given symmetric-feasible TCG.

According to [8], β-sequence (also called *topological order* in this paper) of an SP represents the packing sequence of both G_h and G_v in the corresponding TCG. Let $\{c_1, c_2, \ldots, c_n\}$ be a set of *n* rectangular cells, and the *i*-th cell's width (height) is denoted as W_i (H_i), where $1 \le i \le n$, and *i* is the index of cell c_i in the topological order of the given TCG's corresponding SP. Assume each cell is allowed to be rotated or mirrored. Let ($c_i.x$, $c_i.y$) and ($c_i.rx$, $c_i.ry$) represent the coordinates of the bottom-left and the right-top corners of cell c_i.

We will first explain Y-dimensional symmetric packing flow as listed in Fig. 3. Since there is no directed-edge chain between a symmetric pair in G_v, the shift of a symmetric cell described in Fig. 3 will not cause cyclic move between the two symmetric cells. Line 2 initially satisfies the positioning constraints, and the steps described in Lines 3-8 finalize the cell positions while satisfying the Y-dimensional symmetry constraints. Of a total of *p* symmetric pairs, the time complexities of Lines 1, 2, and 3-8 are *O(n)* (Lemma 1), $O(n^2)$, and *O(p*n)*, respectively. Therefore, the complexity of Fig. 3 is still $O(n^2)$, the same as that in the absence of symmetry constraints.

The X-dimensional symmetric packing flow is depicted in Fig. 4. Different from the two-sweep method developed in [5], here we have proposed a one-stage packing strategy, that is, the cells that form up as a symmetric pair are positioned simultaneously. Therefore, only one iteration loop is enough to satisfy the symmetry constraints. The proposed flow is composed of the following major steps:

- Step 1: satisfy the positioning constraints in the absence of symmetry (Lines 1-2).
- Step 2: determine the symmetry axis (Line 3). For a symmetric pair (c_i, c_j), $i \neq j$, or a self-symmetric cell (c_i, c_j), $i=j$, (i and j are indices in the topological order), the symmetry axis is derived as follows:

$$X_{symmAxis} = (c_i.x + c_j.rx)/2 \quad , \qquad (4)$$

$|i-j| = min\{|l-k|, \forall(c_l, c_k)$ are symmetric counterparts, l and k are indices in the topological order.$\}$.

Thus, it is ensured that no other symmetric pairs or self-symmetric cells will be located between the innermost symmetric pair. As a result, the redundant empty space around the symmetry axis as shown in Fig. 5(d), which may happen when the two-sweep method [5] is applied, will be avoided.

- Step 3: symmetric shift (Lines 4-19). This step is divided into two stages: symmetric cells encompassed by c_i and c_j are shifted as stated in Lines 4-7, and symmetric cells outside c_i and c_j are shifted as stated in Lines 8-19. If the symmetry axis is determined by a self-symmetric cell, due to $i=j$, only the latter stage is performed.

Lines 6, 10 and 16 shift the symmetric pair (c_s, c_t), where c_s (c_t) and its fan-in (fan-out) cells are moved leftwards (rightwards) with respect to the symmetry axis. Since the related fan-in or fan-out cells in the C_h always shift along with the symmetric cells, the positioning constraints already being satisfied by Step 1 will be preserved, while the symmetry constraint between a symmetric pair is also realized. In a similar manner, Lines 12 and 18 shift self-symmetric cell c_s.

Since Eq. (2) holds for the given TCG, cycling of horizontal shifts caused by multiple embedded symmetric cells will not happen. Thus, a valid horizontal placement can surely be built following the scheme detailed in Fig. 4. Lines 1-3 take $O(n)$, $O(n^2)$ and $O(p+s)$ time, respectively, where p is the number of symmetric pairs, and s is the number of self-symmetric cells. The time complexity of Lines 4-19 is $O((p+s)*n)$.

As an example, consider a TCG=(G_h, G_v) shown in Fig. 5(a) and Fig. 5(b), where (a, a') and (b, b') are two symmetric pairs. Fig. 5(c) depicts the corresponding placement in the absence of symmetry constraints. Fig. 5(d) is the placement using the two-sweep packing method described in [5]. Fig. 5(e) is the placement using the proposed one-stage packing method outlined in Fig. 4. In Fig. 5(d) and Fig. 5(e), the symmetry axes are shown in dash lines. Thus, we have:

Lemma 3: Given a symmetric-feasible TCG containing a symmetry group, one can build a placement satisfying the positioning and the symmetry constraints in $O(n^2)$ time.

V. Perturbation of Symmetric-Feasible TCG

Following Lemmas 2 and 3, we can see that an optimal symmetric placement solution can be obtained by exploring the symmetric-feasible TCGs. The problem is now converted to designing a proper scheme that can preserve the symmetric-feasibility and keep TCG valid after random perturbations. Now we first introduce two definitions.

Definition 2: Any two symmetric pairs, two symmetric cells in a pair, two self-symmetric cells, or two asymmetric cells in a TCG representation are allowed to swap their geometric positions and orientations. Those swapping operations are called symmetric-swap.

Definition 3: In a TCG, moving an edge from G_h to G_v, or vice versa, is called edge move operation; moving an edge from G_h to G_v, or vice versa, and also changing the direction of the edge after the edge move is called edge move-reverse operation. A set of edge move and edge move-reverse operations is called edge change operation.

We have designed three operations for TCG perturbation: (i) vertex rotation, (ii) symmetric-swap, (iii) edge change.

A. TCG Vertex Rotation

Given a symmetric-feasible TCG, for an asymmetric or self-symmetric cell, the rotation operation can be done without any constraints. However, for a symmetric pair, the rotation of one cell should be accompanied by the mirror/identical rotation of its symmetric counterpart.

X-Dimensional Symmetric Packing
Begin
1 construct the topological order of the TCG; // Lemma 1.
2 calculate the horizontal longest path of the TCG based on the topological order;
3 determine the symmetry axis, and label the symmetric cells, which are chosen to calculate the symmetry axis, as c_i and c_j, ($i \leq j$); // see Eq. (4)
4 for (any unprocessed symmetric cell c_s, $i<s<j$) {
5 if (c_s has a symmetric counterpart c_t) {
6 shift symmetric pair (c_s, c_t);
7 mark c_s and c_t as processed; }}
8 for (any unprocessed symmetric cell c_s, $j<s \leq n$) {
9 if (c_s has a symmetric counterpart c_t) {
10 shift symmetric pair (c_s, c_t); }
11 else if (c_s is a self-symmetric cell) {
12 symmetrically shift c_s; }
13 mark c_s(c_t) as processed; }
14 for (any unprocessed symmetric cell c_s, $0 \leq s<i$) {
15 if (c_s has a symmetric counterpart c_t) {
16 shift symmetric pair (c_s, c_t); }
17 else if (c_s is a self-symmetric cell) {
18 symmetrically shift c_s; }
19 mark c_s (c_t) as processed; }
End

Fig. 4. X-dimensional symmetric packing flow.

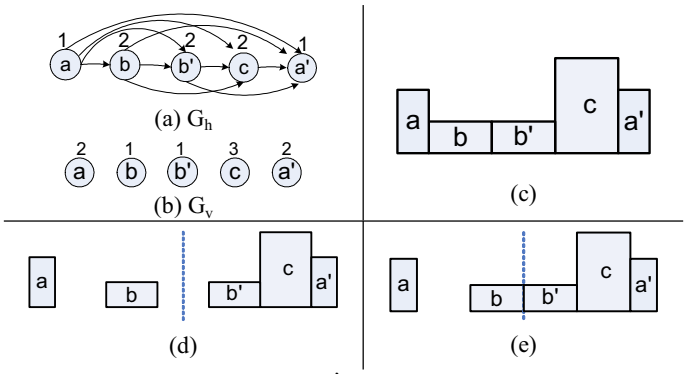


Fig. 5. A packing example. (a, a') and (b, b') are symmetric pairs.

Lemma 4: Given a symmetric-feasible TCG, the perturbed TCG is still symmetric-feasible and valid under the vertex-rotation operation, and this operation takes *O(1)* time.
Proof: Since the vertex set and the edge set of TCG remain unchanged after the rotation process, the resulting graphs are still symmetric-feasible and valid. Exchanging the weights of the related vertices in G_h and G_v only takes *O(1)* time. ∎

B. TCG Symmetric-Swap

In a symmetric-feasible TCG, if two cells in two symmetric pairs are swapped, their corresponding symmetric counterparts have also to be swapped.
Lemma 5: Given a symmetric-feasible TCG, the resulting TCG after a symmetric-swap operation is still symmetric-feasible and valid, and it takes *O(1)* time.
Proof: Since the symmetric-swap operation only exchanges two or four vertices in both G_h and G_v without changing the topology of the original TCG, the resulting graphs are still symmetric-feasible and valid. Exchanging the corresponding pointers of two or four vertices only takes *O(1)* time. ∎

C. TCG Edge Change

According to the definition of SP, the TCG edge move operation is equivalent to the reversal of two vertices in the α-sequence of the corresponding SP. For instance, if $v_a \vdash v_b$ in TCG, the corresponding SP is (v_a, v_b) (v_a, v_b). After the edge move operation, the edge becomes $v_a \perp v_b$ and the corresponding SP becomes (v_b, v_a) (v_a, v_b). Similarly, the TCG edge move-reverse operation is equivalent to the reversal of two vertices in the β-sequence of the corresponding SP.

We have designed a scheme as described in Fig. 6 to randomly change TCG edges while keeping valid. There is no need to check the symmetric-feasibility by packing the perturbed TCG. Instead, without packing operation, the proposed scheme itself will report failure (according to our symmetric-feasibility conditions introduced in Section III.C) if there exists any violation of symmetric-feasibility. Here the explanation is given following the term α- or β-sequence. However, the explicit representation of the corresponding SP of a TCG is not required. All the operations only make use of the information provided by the given TCG.

In Line 1 of Fig. 6, one vertex *a* is chosen randomly. The *slack range* of symmetric-vertex *a* (used in Line 3) is defined as the largest span of moving *a* in the α- (β-) sequence without changing the horizontal relationship between *a* and *a′* (if (*a*, *a′*) is a symmetric pair) or without changing the vertical relationship between *a* and any other self-symmetric vertex (if *a* is self-symmetric). For instance, in Fig. 1, for vertex *d′* in the α-sequence (*d*, *a*, *f*, *e*, *e′*, *d′*, *b*, *c*), the slack range is from *a* to *c* (due to the symmetric counterpart *d*), while for self-symmetric vertex *b* in the β-sequence (*a*, *b*, *d*, *e*, *e′*, *f*, *c*, *d′*), the slack range is from *a* to *e′* (due to the closest self-symmetric vertex *f*).

In Line 4, one vertex *b* is randomly picked up and Set-A (Set-B) is accordingly shrunk to only include the vertices between *a* and *b* (including *b*). The operations in Lines 6-9 are equivalent to move vertex *a* from the current position to the one before (after) *b* in the corresponding α-(β-) sequence of the TCG. Thus, we have the following lemma:
Lemma 6: Given a symmetric-feasible TCG, the perturbed TCG is still symmetric-feasible and valid under the certain

randomChangeTcgEdges
(Input: a symmetric-feasible TCG, Output: TRUE for a successful random edge-change operation and FALSE for a failure random edge-change operation)
Begin
1 randomly choose one vertex *a*;
2 if (*a* is symmetric) {
3 obtain the slack range of vertex *a* in the α- (β-) sequence and save them into Set-A (Set-B);
4 in Set-A (Set-B), randomly pick up one vertex *b* (*a*≠*b*) and only keep the vertices between *a* and *b* (including *b*) in the corresponding set;
5 randomly choose to operate on Set-A or Set-B;
6 for (each vertex *c* in Set-A (Set-B)) {
7 move (move-reverse) the edge between *a* and *c*;
8 if (*c* is a symmetric vertex and the updated (*a*, *c*) or (*a′*, *c′*) violates symmetric-feasibility Eqs. (2) and (3)) {
9 the random edge change fails and return FALSE;}}}
10 else {
11 randomly pick up one vertex *b* (*a*≠*b*);
12 obtain the vertices lying between *a* and *b* in the α- (β-) sequence and save them into Set-A (Set-B);
13 for (each vertex *c* in Set-A (Set-B)) {
14 move (move-reverse) the edge between *a* and *c*;}}
15 the random edge change is successful and return TRUE;
End

Fig. 6. TCG edge change scheme under symmetry constraints.

edge change operation, and such an operation takes *O(n)* time.
Proof: The edge change operation can be performed in a flow as outlined in Fig. 6. Since the corresponding SP is feasible, it is ensured that the derived TCG is valid. In Line 8, the symmetric-feasibility is checked according to the Eqs. (2) and (3) in Definition 1. If any violation is found, no further operation is performed and FALSE is returned. Otherwise, a perturbed symmetric-feasible TCG can be obtained after random edge-change operation.

Based on Lamma 1, it takes *O(n)* time to extract α-(β-) sequence of a TCG. The rest of operations, including relationship recognition of symmetric vertices and symmetric-feasibility checking, takes *O(1)* time. Therefore, the edge change operation takes *O(n)* time. ∎

In [6], a few schemes with $O(n^2)$ time complexity have been proposed to move or reverse a TCG edge in the absence of symmetry constraints. In that method, to keep the perturbed TCG valid, special care has to be taken for fan-in and fan-out vertices related to the moved/reversed edge. In contrast, our proposed edge change operation above applies a set of correlated edge move and edge move-reverse operations as a whole, which inherently guarantees the TCG closure property and sustain much lower time complexity.
Theorem 1: The solution space of symmetric-feasible TCG can be fully explored using random vertex rotation, symmetric-swap, edge change operations. The transformation of two solutions represented in TCG takes at most *O(n)* time.
Proof: We may clearly view the exploration of the TCG solution space with the aid of the corresponding SP. According to the features of the vertex rotation, symmetric-swap and edge change, a cell can be randomly changed to any valid position in both α- and β- sequences. In addition, due to Lemma 3, it is ensured that the perturbed TCG satisfying the symmetric-feasibility conditions can map to a valid symmetric placement. This means the full exploration of the solution space can be performed by the

vertex and edge operations. Obviously, the time complexity of the operations above is *O(n)*. ∎

VI. Experimental Results

We have developed a simulated annealing based symmetry-aware TCG placement algorithm for analog layout designs. According to Theorem 1, we can make use of the operations introduced in Section V to explore the solution space of valid symmetry placements. Compared with the normal symmetry-free TCG placement [6], our symmetry TCG placement algorithm can achieve faster solution perturbation (in *O(n)* time) and the same packing time complexity (i.e., $O(n^2)$). The initial TCG, which can be readily built in different ways, must be symmetric-feasible.

Our cost function used to evaluate a placement solution is a weighted function of three components as given in Eq. (5),

$$C = \alpha_{ared} C_{area} + \alpha_{nets} C_{nets} + \alpha_{size} C_{size} \quad , \qquad (5)$$

where α_* is the weight factor for the corresponding cost C_*. C_{area} is the area cost that is made up of the whole area, NWELL and PWELL region areas, and the analog and digital region areas. It could make NWELL/PWELL regions relatively concentrated and isolate analog/digital regions from each other as required in mixed-signal circuits. C_{nets} is the net-length cost, in which a priority coefficient can be specified for each net. C_{size} is the size cost, which is used to control the shape of the final layout.

To test the performance of our proposed algorithm, it has been coded in C++ and applied to several test circuits on a Sun-Ultra10 workstation. Our test circuits are collected from a variety of sources. To demonstrate the efficiency of our algorithm, two other approaches coded in C++ on the same platform have been included for comparison: (i)*AbsPlace.*, one absolute placement scheme using absolute coordinates [2]-[4]; and (ii) *SymmSP*, an implementation that imitates [5].

The comparison results are given in Table II, where *Cost* and *T* are the normalized percentages of mean cost and execution time, respectively. It can be seen that our proposed algorithm *SymmTCG* outperforms the other two algorithms in the search quality. On average, compared with *SymmSP*, *SymmTCG* reduces the cost by 10.7% with almost identical CPU time. When compared with *AbsPlace*, *SymmTCG* reduces the cost by 28.2% and the execution time by 40.6%.

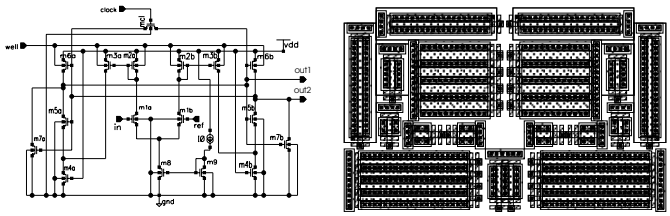
Fig. 7(a) shows the schematic of a CMOS high speed comparator abounding in symmetry and matching constraints. The placement layout result obtained using our *SymmTCG* is depicted in Fig. 7(b). In particular, all the matching devices are oriented symmetrically. The PMOS and the NMOS regions are respectively concentrated and separated.

VII. Conclusion

In this paper, the analog VLSI placement problem has been solved using transitive closure graph. We introduced a set of TCG symmetric-feasible conditions and proved that the solution space of symmetric placements can be efficiently explored by evaluating symmetric-feasible TCGs. Thus, an efficient strategy has been proposed to generate random symmetric-feasible TCG representations while keeping TCG valid in *O(n)* time. In addition, we have developed a new packing scheme for symmetric-feasible TCG in a simulated-annealing based placement algorithm. Experimental results have shown that this proposed algorithm, with very high computation efficiency, can generate higher quality placement results than two other well-known approaches.

Table II: Comparison among different algorithms. Absolute values of the cost and the CPU seconds are given only for the *SymmTCG*, the rest of *Cost* and *T* being displayed as the normalized percentages compared to the last column.

Analog Circuits		*AbsPlace*	*SymmSP*	*SymmTCG*
Rail-to-rail Opamp	Cost	119.6%	106.2%	142643
	T (sec)	134.2%	94.5%	48
Comm.-mode-feed. Opamp	Cost	126.6%	109.2%	306512
	T (sec)	136.3%	102.1%	72
Low-noise opamp	Cost	126.3%	105.7%	286559
	T (sec)	139.5%	104.2%	84
Comparator	Cost	140.1%	121.6%	19720
	T (sec)	152.5%	98.2%	118



(a) (b)

Fig. 7. (a) schematic and (b) a placement from *SymmTCG* of the CMOS comparator.

Acknowledgements

This work was supported in part by the Natural Sciences and Engineering Research Council of Canada (NSERC), Memorial University of Newfoundland, and U.S. Defense Advanced Research Projects Agency (DARPA).

References

[1] G. Gielen and R. Rutenbar, "Computer-aided design of analog and mixed-signal integrated circuits," *Proc.IEEE*,vol.88,pp.1825-1852,2000.

[2] J. Cohn, D. Garrod, R. Rutenbar, and L Carley, *Analog Device-level Layout Automation,* Boston: Kluwer Academic Publishers, 1994.

[3] 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.

[4] K. Lampaert, G. Gielen, and W. Sansen, *Analog Layout Generation for Performance and Manufacturability,* Boston: Kluwer Academic Publishers, 1999.

[5] F. Balasa and K. Lampaert, "Symmetry within the sequence-pair representation in the context of placement for analog design," *IEEE Trans. CAD*, vol. 19, pp. 721-731, July 2000.

[6] J.-M. Lin and Y.-W. Chang, "TCG: a transitive closure graph based representation for general floorplans," *IEEE Trans. VLSI Systems*, vol. 13, pp. 288-292, Feb. 2005.

[7] 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.

[8] J.-M. Lin and Y.-W. Chang, "TCG-S: Orthogonal coupling of P*-admissible representations for general floorplans," *IEEE Trans. CAD*, vol. 24, pp. 968-980, June 2004.

[9] J.-M. Lin, G.-M. Wu, Y.-W. Chang, and J.-H. Chuang, "Placement with symmetry constraints for analog layout design using TCG-S," *Proc. Asia and South-Pacific Design Automation Conf.*, 2005, pp. 1135-1138.

[10] S. Kouda, C. Kodama, and K. Fujiyoshi, "Improved method of cell placement with symmetry constraints for analog IC layout design," *Proc. International Symposium on Physical Design*, 2006, pp. 192-199.

[11] Y. Tam, E. Young, and C. Chu, "Analog placement with symmetry and other placement constraints," *Proc. IEEE/ACM Int. Conf. on CAD*, 2006, pp. 349-354.