# Analog Placement with Symmetry and Other Placement Constraints *

Yiu-Cheong Tam Evangeline F.Y. Young Chris Chu
Department of CSE Department of ECE
The Chinese University of Hong Kong Iowa State University
02654543@alumni.cse.cuhk.edu.hk fyyoung@cse.cuhk.edu.hk cnchu@iastate.edu

## ABSTRACT

In order to handle device matching in analog circuits, some pairs of modules are required to be placed symmetrically. This paper addresses this device-level placement problem for analog circuits and our approach can handle symmetry constraint and other placement constraints simultaneously. The problem of placing devices with symmetry constraint has been extensively studied but none of the previous works has considered symmetry constraint with other placement constraints simultaneously. Instead of handling the constraints by having a penalty term in the cost function to penalize violations, a unified method is proposed that, by adjusting the edge weights in a pair of constraint graphs, can try to satisfy all the placement and symmetry constraints simultaneously in a candidate floorplan solution. The maximum distance of the modules in a symmetry group from the corresponding symmetry axis will be minimized in this weight adjusting step, in order to minimize the total packing area. We have compared our method with the most updated results on this problem [2] when there are only symmetry constraints and results show that our approach can give solutions of better quality, in an acceptable amount of run time. We will also demonstrate the effectiveness of our approach in handling different types of constraints simultaneously by testing on data sets with both symmetry and other placement constraints, and the results are very promising.

## Categories and Subject Descriptors

B.7.2 [**Integrated Circuits**]: Design Aids - Layout, Placement and routing; J.6 [**Computer-Aided Engineering**]: Computer-aided design(CAD)

## General Terms

Algorithms, Design, Theory

## Keywords

Analog circuits, Placement, Symmetry constraints, Sequence-Pair

*This work was supported by the Direct Grant for Research of the Chinese University of Hong Kong, under Project 2050321.


*ICCAD'06,* November 5-9, 2006, San Jose, CA


## 1. INTRODUCTION

In the design of analog circuits, it is often required to have some devices (modules) to be placed symmetrically with respect to one or several common axes. The main reason is to match the layout-induced parasitics. Failure to do so may lead to higher offset voltages and degraded power-supply rejection ratio [5]. Placing devices symmetrically can also reduce the circuit sensitivity to thermal gradients.

The problem of placing devices with symmetry constraint has been extensively studied [6, 8, 10, 1, 12, 2]. Most of them used simulated annealing as an optimization engine based on a packing representation. We can classify these representations into two categories: (1) absolute representation, and (2) topological representations. In absolute representation [6, 8, 10], modules are represented by their absolute coordinates on the chip plane. Since no restrictions is made to the relative positions between modules, illegal overlaps will occur. A penalty term in the cost function will be associated with those infeasible overlaps, which will be driven to zero in the optimization process. However the size of the solution space is huge in this absolute representation, which will affect the solution quality given a limited amount of search time. In topological representation, the relative positions between the modules are encoded. The solution space is much smaller in comparison with that of absolute representation, but complicated computations are needed for checking symmetry feasibility and adjusting the module positions to satisfy the constraints. Topological representations like sequence-pairs [11], O-tree [4], B*-trees [3] have been applied to handle symmetry constraints in [1, 12, 2]. Comparisons in [2] have shown that the segment tree approach [2] has out-performed others in both solution quality and run time. TCG-S [9] has also been applied to handle symmetry constraints in [13] and the comparisons in [13] have shown that this TCG-S approach is comparable with the segment tree approach in solution quality but the run time is longer especially when the size of the data set increases.

None of the previous works has considered symmetry constraints and other general placement constraints simultaneously. In placement of analog circuits, there are also other placement constraints like device separation constraint (an upper bound on the separation distance between pairs of critically matched devices), alignment constraint, abutment constraint, boundary constraint, preplaced constraints and range constraint, etc. In this paper, we try to address this analog placement problem with both symmetry and other general placement constraints simultaneously. Sequence pair is used as the representation in the simulated annealing engine, but the approach can be applied to any other representations as long as they pack by constructing constraint graphs. Instead of handling the constraints by having a penalty term in the cost function to penalize violations, we will try to adjust the

edge weights in the constraint graphs in a unified way so that all the placement and symmetry constraints can be satisfied simultaneously During this process, we will try to minimize the maximum distance of the modules in a symmetry group from the corresponding symmetry axis, in order to minimize the total packing area. We have compared our method with the segment tree approach [2] when there are only symmetry constraints and results show that our approach can give solutions of better quality. We have also demonstrated the effectiveness of our approach in handling different types of constraints by testing on some data sets with both symmetry and other placement constraints, and the results are all very promising.

In the following, we will first define the problem (section 2), then we will discuss the methodology used in our system (section 3). In section 4, we will describe the simulated annealing process employed, and the experimental results will be reported in section 5.

## 2. PROBLEM FORMULATION

Given a set of $n$ blocks of areas $A_i$ and aspect ratio bounds $[l_i, u_i]$ where $i = 1 \dots n$, a set of $m$ nets $N_1, N_2 \dots N_m$ between the $n$ blocks, a set of $p$ symmetry groups $G_1, G_2 \dots G_p$ where each symmetry group $G_i$ is consisted of $self(G_i)$ self-symmetry blocks and $pair(G_i)$ symmetry pairs, and a set of $q$ placement constraints $C_1, C_2 \dots, C_q$ where each placement constraint $C_i$ denotes a constraint in placement between two arbitrary blocks, the objective is to construct a floorplan $F$ satisfying all the symmetry and placement constraints and minimizing a cost function $cost(F) = area(F) + \alpha \times wire(F)$ where $\alpha$ is a user given weight, $area(F)$ is the total area of $F$ and $wire(F)$ is the total wire length of $F$ measured by the half-perimeter estimation. Placement constraint can be alignment, abutment, maximum separation, boundary, preplace or range constraint and their detailed definitions will be given in the next section. Please note that each symmetry group can be symmetric horizontally or vertically, unless stated specifically by the users.

## 3. METHODOLOGY

In our approach, sequence pair [11] is used to represent a placement. In sequence pair, a packing is represented by a pair of permutations of the module names $(s_1, s_2)$. If the relative positions of two modules $A$ and $B$ in $s_1$ and $s_2$ are $\dots A \dots B \dots$ and $\dots A \dots B \dots$ ($\dots A \dots B \dots$ and $\dots B \dots A \dots$) respectively, $A$ is on the left of $B$ ($A$ is below $B$). In each step of the annealing process, a candidate solution $x = (s_1, s_2)$ will be generated. We will first have an initial scan to check if $x$ will be a feasible solution satisfying all the constraints. Details of this scanning process to check the feasibility of a sequence pair will be given in a later section. After this initial scan, a pair of constraint graphs $(H_h, H_v)$ will be built according to the sequence pair to represent the relative positions between the modules. In the horizontal (vertical) constraint graph $H_h$ ($H_v$), the vertices represent the modules and the edges represent the relationship between the modules in the horizontal (vertical) direction, e.g., if $A$ is on the left of $B$ ($A$ is below $B$), there will be an edge from $A$ to $B$ in $H_h$ ($H_v$) with weight $w(A)$ ($h(A)$) where $w(A)$ ($h(A)$) is the width (height) of module $A$. Additional nodes and constraint edges will be inserted into $H_h$ and $H_v$ to enforce the required symmetry and placement constraints in some later stages. Some of those newly inserted edges have variable weights and we need to determine their weights to minimize the packing area and to ensure that no positive cycles will be created. Finally, if no positive cycles exist in the graphs, all constraints can be satisfied simultaneously and we will pack accordingly to obtain one feasible candidate solution. The whole

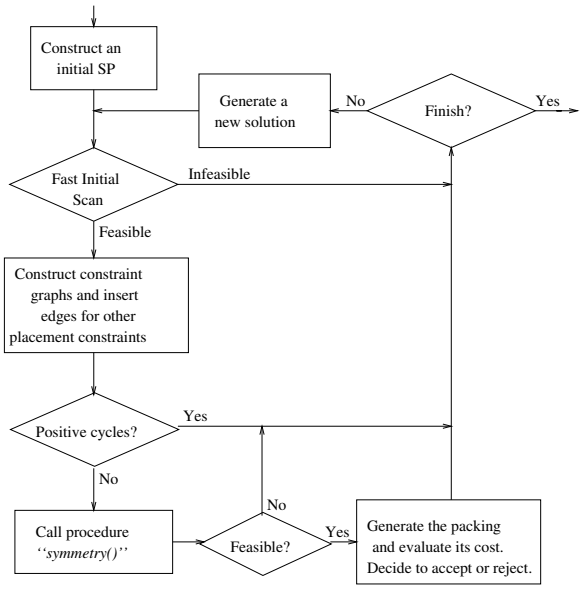


**Figure 1: An Overview of Our Floorplanner.**

flow of our system is shown in fig. 1 and details of each step will be given in the following sections.

### 3.1 Fast Initial Scan

The purpose of this fast initial scan is to identify those infeasible solutions early by just looking at the sequence pair representation. Notice that these checkings are only used to screen out some infeasible solutions but those remaining may still be infeasible and we cannot identify them until the constraint graphs are built. In our implementation, we will only generate those sequence pairs satisfying the symmetry condition $Q4$ because this is the most complicated one while checkings for violation of the other conditions can be done very effectively.

- **Alignment Condition** $Q1$ - If block $A$ is required to align with block $B$ horizontally (vertically), the order of $A$ and $B$ in $s_1$ and $s_2$ must be the same (reversed).
- **Abutment Condition** $Q2$ - If block $A$ is required to abut with block $B$ horizontally with $A$ on the left (right), $s_1$ and $s_2$ must be of the form $s_1 = \dots A \dots B \dots$ ($\dots B \dots A \dots$) and $s_2 = \dots A \dots B \dots$ ($\dots B \dots A \dots$) respectively. Similarly, we can derive the condition for vertical abutment.
- **Boundary Condition** $Q3$ - If block $A$ is required to abut with the left (right) boundary of the chip, there should not be any block $B$ such that $B$ is before (after) $A$ in both $s_1$ and $s_2$. Similarly, if block $A$ is required to abut with the bottom (top) boundary, there should not be any block $B$ such that $B$ is before (after) $A$ in $s_1$ and after (before) $A$ in $s_2$.
- **Symmetry Condition** $Q4$ - It has been given in [1] that a sufficient symmetry feasible condition [7] in sequence pair $(s_1, s_2)$ is:

  $$s_1^{-1}(A) < s_1^{-1}(B) \Leftrightarrow s_2^{-1}(sym(B)) < s_2^{-1}(sym(A))$$

  for horizontal symmetric groups where $A$ and $B$ are any two distinct blocks in the group, $s_1^{-1}(X)$ ($s_2^{-1}(X)$) denotes the position of block $X$ in $s_1$ ($s_2$) and $sym(X)$ denotes the symmetry block of $X$ ($sym(X)$ of a self sym-

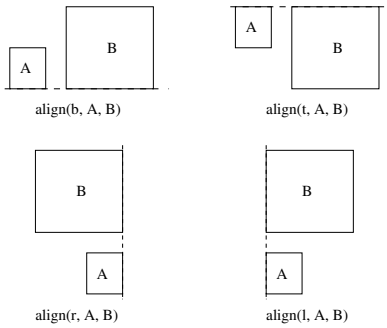


**Figure 2: Alignment Constraints.**

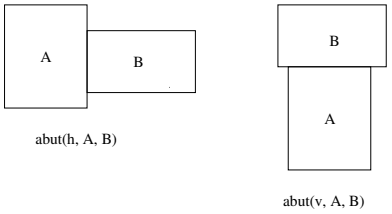


**Figure 3: Abutment Constraints.**

metry block $X$ is $X$ itself). Notice that the above condition holds for any two blocks in the group, e.g., we can put $B$ as $sym(A)$ and the condition requires that $A$ is on the left of $sym(A)$. According to this symmetry condition, the blocks of a horizontal symmetry group will appear in a *mirror* form in a sequence pair, e.g., $s_1 = \ldots A_1 \ldots A_2 \ldots A_x \ldots$ and $s_2 = \ldots sym(A_x) \ldots sym(A_2) \ldots sym(A_1) \ldots$. Similarly, a sufficient symmetry feasible condition for vertical symmetric groups is:

$$s_1^{-1}(A) < s_1^{-1}(B) \Leftrightarrow s_2^{-1}(sym(A)) < s_2^{-1}(sym(B))$$

where $A$ and $B$ are any two distinct blocks in the group. According to this symmetry condition, the blocks of a vertical symmetry group will appear in an *ordered* form in a sequence pair, e.g., $s_1 = \ldots A_1 \ldots A_2 \ldots A_x \ldots$ and $s_2 = \ldots sym(A_1) \ldots sym(A_2) \ldots sym(A_x) \ldots$.

## 3.2 Handling General Placement Constraints

In our problem, we will handle the following placement constraints:

- **Alignment** - We use the notation $align(x, A, B)$ where $x \in \{l, r, t, b\}$ to denote that two blocks $A$ and $B$ are required to align vertically along the left ($x = l$) or the right ($x = r$) side, or to align horizontally along the top ($x = t$) or the bottom ($x = b$) side (fig. 2).

- **Abutment** - We use the notation $abut(x, A, B)$ where $x \in \{v, h\}$ to denote that two blocks $A$ and $B$ are required to abut horizontally ($x = h$) with $A$ on the left and $B$ on the right, or to abut vertically ($x = v$) with $A$ at the bottom and $B$ on top (fig. 3). In our definition of abutment constraint, the shorter abuting side must abut completely with the longer abuting side.

- **Maximum Separation** - We use the notation $maxsep(x, A, B, y)$ where $x \in \{v, h\}$ and $y$ is a positive real number to denote that two blocks $A$ and $B$ can at most be separated by a distance $y$ horizontally ($x = h$) or vertically ($x = v$).

- **Boundary** - We use the notation $boundary(x, A)$ where $x \in \{l, r, t, b\}$ to denote that block $A$ is required to abut with the left ($x = l$), right ($x = r$), top ($x = t$) or bottom ($x = b$) boundary of the whole chip.

- **Preplace** - We use the notation $preplace(x, y, A)$ where $x, y$ are real numbers to denote that block $A$ is required to be placed with its lower left corner at the coordinates $(x, y)$.

- **Range** - We use the notation $range(x, y, x_1, y_1, A)$ where $x, x_1, y, y_1$ are real numbers and $x_1 \geq x$ and $y_1 \geq y$ to denote that block $A$ is required to be placed with its lower left corner lying in the range from $(x, y)$ to $(x_1, y_1)$.

We will make use of the approach in [14] of adding pairs of edges in the constraint graphs to handle these placement constraints. Details can be found in [14] and the methodology will not be repeated here again.

## 3.3 Handling Symmetry Constraint

In order to handle symmetry constraint and other placement constraints simultaneously in a unified framework, we will also augment the constraint graphs to enforce symmetry constraint. For each symmetry group $G_i$ containing $r_i = self(G_i)$ self symmetry blocks $Z_1, Z_2 \ldots Z_{r_i}$ and $s_i = pair(G_i)$ symmetry pairs $(X_1, Y_1), (X_2, Y_2) \ldots (X_{s_i}, Y_{s_i})$, we will first check if $G_i$ should be symmetric horizontally or vertically in a candidate sequence pair solution, according to the $Q4$ condition as stated in section 3.1. W.l.o.g., we assume that $G_i$ should be symmetric horizontally in the following discussion. First of all, we need to add constraint edges to the vertical constraint graph to align the symmetry pairs in $G_i$ horizontally. A dummy node $d_i$ will then be added to the horizontal constraint graph to represent the symmetry axis of $G_i$. In order to enforce the equidistant constraint between the symmetry pairs, four edges, $e(X_j, d_i)$, $e(d_i, X_j)$, $e(d_i, Y_j)$ and $e(Y_j, d_i)$, will be added with weights $x_{ij}$, $-x_{ij}$, $x_{ij} - w(Y_j)$ and $-(x_{ij} - w(Y_j))$ respectively for each $j = 1 \ldots s_i$ where $w(Y_j)$ is the width of block $Y_j$ (notice that $w(X_j) = w(Y_j)$) and $x_{ij} \geq w(Y_j)$ is a positive real number. For each self symmetry block $Z_j$ where $j = 1 \ldots r_i$, a pair of edges, $e(Z_j, d_i)$ and $e(d_i, Z_j)$ of weights $w(Z_j)/2$ and $-w(Z_j)/2$ will be added to ensure that $Z_j$ will be lying symmetrically on the axis. After adding these dummy nodes and constraint edges, we will determine the value of $x_{ij}$ for $j = 1 \ldots s_i$ such that no positive cycles exists in the graph and $\max_{1 \leq j \leq s_i} x_{ij}$ is minimized.

### 3.3.1 Bounds on the Variable Edge Weights

Fig. 4 shows the scenario of a simple symmetry group $G_i$ with only two symmetry pairs $(X_1, Y_1)$ and $(X_2, Y_2)$, and one self symmetry block $Z_1$. Now, we want to determine the values of $x_{i1}$ and $x_{i2}$ such that no positive cycles will be created and the value $\max\{x_{i1}, x_{i2}\}$ is minimized in order to obtain a more compacted solution. Consider any positive cycle *possibly forming*, the cycle must contain the dummy node $d_i$. In the following, we will enumerate all these potential positive cycles. The variable $dist(A, B)$ denotes the longest path length (can be negative) from node $A$ to node $B$ in the original constraint graph before adding those dummy nodes and additional constraint edges for symmetry groups and is equal to $-\infty$ if there is no such paths. There are totally four types of positive cycles possibly forming as enumerated as follows:

- A cycle $X_j \rightarrow Y_j \rightarrow d_i \rightarrow X_j$ ($Y_j \rightarrow X_j \rightarrow d_i \rightarrow Y_j$) for some $j = 1 \ldots s_i$ of total weight $dist(X_j, Y_j) - (x_{ij} - w(Y_j)) - x_{ij}$ ($dist(Y_j, X_j) + x_{ij} + (x_{ij} - w(Y_j))$) may be

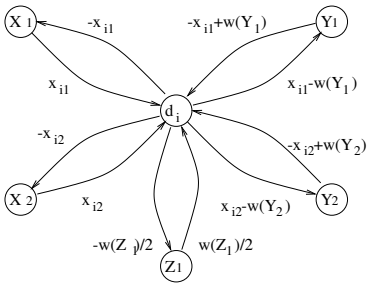


**Figure 4: Dummy Nodes and Additional Constraint Edges for a Symmetry Group.**

formed. To avoid positive cycles, it is required to have $2x_{ij} \geq dist(X_j, Y_j) + w(Y_j)$ ($2x_{ij} \leq w(Y_j) - dist(Y_j, X_j)$).

- A cycle $X_j \rightarrow Y_k \rightarrow d_i \rightarrow X_j$ ($Y_k \rightarrow X_j \rightarrow d_i \rightarrow Y_k$) for some $j, k = 1 \ldots s_i$ and $j \neq k$ of total weight $dist(X_j, Y_k) - (x_{ik} - w(Y_k)) - x_{ij}$ ($dist(Y_k, X_j) + x_{ij} + (x_{ik} - w(Y_k))$) may be formed. To avoid positive cycles, it is required to have $x_{ij} + x_{ik} \geq dist(X_j, Y_k) + w(Y_k)$ ($x_{ij} + x_{ik} \leq w(Y_k) - dist(Y_k, X_j)$).

- A cycle $X_j \rightarrow Z_k \rightarrow d_i \rightarrow X_j$ ($Z_k \rightarrow X_j \rightarrow d_i \rightarrow Z_k$) for some $j = 1 \ldots s_i$ and $k = 1 \ldots r_i$ of total weight $dist(X_j, Z_k) + w(Z_k)/2 - x_{ij}$ ($dist(Z_k, X_j) + x_{ij} - w(Z_k)/2$) may be formed. To avoid positive cycles, it is required to have $x_{ij} \geq dist(X_j, Z_k) + w(Z_k)/2$ ($x_{ij} \leq w(Z_k)/2 - dist(Z_k, X_j)$). Similarly, there may be cycles between $Y_j$ and $Z_k$ resulting in the constraints $x_{ij} \geq dist(Z_k, Y_j) - w(Z_k)/2 + w(Y_j)$ and $x_{ij} \leq w(Y_j) - w(Z_k)/2 - dist(Y_j, Z_k)$.

- A cycle $X_j \rightarrow d_i \rightarrow X_k \rightarrow X_j$ ($Y_j \rightarrow d_i \rightarrow Y_k \rightarrow Y_j$) for some $j, k = 1 \ldots s_i$ and $j \neq k$ of total weight $x_{ij} - x_{ik} + dist(X_k, X_j)$ ($-x_{ij} + w(Y_j) + x_{ik} - w(Y_k) + dist(Y_k, Y_j)$) may be formed. To avoid positive cycles, it is required to have $x_{ik} - x_{ij} \geq dist(X_k, X_j)$ ($x_{ij} - x_{ik} \geq w(Y_j) - w(Y_k) + dist(Y_k, Y_j)$).

### 3.3.2 Computations of the Variable Edge Weights

From the above analysis, we can obtain upper and lower bounds for a single variable $x_{ij}$, for the sum of two variables $x_{ij} + x_{ik}$ and for the difference of two variables $x_{ij} - x_{ik}$. The bounds involve some pair-wise longest paths in the original acyclic constraint graph and the widths of some blocks, and they can be computed effectively. Our goal is to evaluate all $x_{ij}$'s satisfying these bounds and minimizing $\max_{1 \leq j \leq s_i} x_{ij}$. This can of course be solved optimally by a linear solver but it will be too expensive to invoke a solver in every iteration of the annealing process. Therefore, we will solve this system of linear equation directly. Our approach can obtain the optimal solution when there are only lower bound constraints, e.g., when there are only symmetry constraints and no other general placement constraints. When there are both upper and lower bound constraints as in general cases, the solution obtained by our method may be sub-optimal sometimes but this occurs very rarely as verified by the experiments (8 out of 2626 trials).

When there are only lower bounds, we can basically increase the values of the variables until all the lower bounds are satisfied. Now we also want to minimize the value $\max_{1 \leq j \leq s_i} x_{ij}$. This can be achieved by carefully accounting a *slack* for each variable (how much a variable can be increased without increasing the value of the objective function). When there are both upper and lower bounds, we will keep account of a *largest possible slack* for each variable due to the upper bound constraints. For example, consider two upper bound constraints $x_{ij} \leq a$ and $x_{ij} + x_{ik} \leq b$ (notice that a difference constraint can always be written as a lower bound constraint), the largest possible slack of $x_{ij}$ will be $\min\{a - x'_{ij}, b - x'_{ij} - x'_{ik}\}$ where $x'_{ij}$ and $x'_{ik}$ are the current values of $x_{ij}$ and $x_{ik}$ respectively. Then by adjusting the values of the variables according to the slacks and the largest possible slacks (which are updated dynamically), we can obtain a solution for the system of linear equation efficiently.

### 3.3.3 Summary

The pseudocode below shows a summary of the steps to handle symmetry constraint.

**Pseudocode** *Symmetry()*
// Given a pair of acyclic constraint graphs $(H'_h, H'_v)$ which are
// already augmented with edges to handle other types of placement constraints, this procedure either announces that the
// symmetry constraints cannot be satisfied simultaneously or
// further augments them to take into account the symmetry
// constraints.

1. For each symmetry group $G_i$
2. Determine the longest path between every pair of blocks in $G_i$ in the constraint graphs $(H'_h, H'_v)$.
3. For each symmetry group $G_i$
4. Insert a dummy node $d_i$ to $H'_h$ ($H'_v$).
   /* Assume that $G_i$ is symmetric horizontally (vertically). */
5. Insert additional constraint edges between $d_i$ and the blocks in $G_i$ according to section 3.3.
6. Determine the weights of the additional constraint edges.
7. If no solutions is obtained, return(fail).
8. Check for positive cycles in $(H'_h, H'_v)$.
9. If positive cycles found in $(H'_h, H'_v)$, return(fail).
10. Return$(H'_h, H'_v)$.

Notice that step 9 above is needed since there may be cycles formed between different symmetry groups after inserting those dummy nodes and additional constraint edges. For efficiency purpose, we have chosen to determine the edge weights of each group separately and check for positive cycles once at the end. An alternative will be solving a system of linear equation involving *all* the variable edge weights. In that case, we must invoke a solver since the upper and lower bound constraints will be very general, e.g., involving many variables. Notice that if different symmetry groups do not interleave and no negatively weighted paths exist between different symmetry groups in $(H'_h, H'_v)$, e.g., no other placement constraints between symmetry groups, the two approaches are the same, i.e., the variable edge weights in different groups will not affect each other.

# 4. ANNEALING PROCESS

## 4.1 Set of Moves

We employ the following set of moves that, starting with a sequence pair satisfying the symmetry condition $Q4$ in section 3.1, can generate another candidate sequence pair satisfying $Q4$:

1. **Swapping two symmetry groups** - Two symmetry groups are picked randomly and swapped, For example, if group $G_i$ has three blocks occupying positions 15, 17 and 20 in $s_1$ and group $G_j$ has two blocks occupying positions 26 and 28 in $s_1$. After the swap, the blocks in $G_j$ will occupy the positions 15 and 17 in $s_1$, and the blocks in $G_1$ will occupy the positions 20, 26 and 28 in $s_1$, without changing the relative ordering between the blocks of the same group. We will do similarly for the blocks of $G_1$ and $G_2$ in $s_2$. Notice that we do not consider interleaving of symmetry groups in our implementation, so this operation is well-defined.

2. **Swapping two blocks of the same symmetry group** - Two blocks $A$ and $B$ which are not symmetry pair of each other are picked randomly from a symmetry group. Then we swap $A$ and $B$ in $s_1$ and swap $sym(A)$ and $sym(B)$ in $s_2$. Notice that the blocks $A$ and $B$ can be self symmetry or belong to a symmetry pair.

3. **Moving an asymmetric block** - Since the positions of the asymmetry blocks do not affect the symmetry-feasibility of a sequence pair and they can be moved freely. In this move, an asymmetry block is picked randomly and its position in the sequence $s_1$ or $s_2$ is modified.

4. **Rotating a symmetry group** - A symmetry group is picked randomly and its orientation is changed (from horizontal to vertical, or vice versa). To perform this rotation, we only need to reverse the order of the related blocks in $s_2$.

5. **Changing the shape of a soft block** - A soft block $A$ is picked randomly and its aspect ratio is changed. If $A$ belongs to a symmetry pair, we also need to make the corresponding change to $sym(A)$.

## 4.2 Initial Solution

The initial solution of the annealing process is obtained by first generating sub-sequences of $s_1$ and $s_2$ for the blocks in a symmetry group according to the condition $Q4$ in section 3.1. This sub-sequences of $s_1$ and $s_2$ are then concatenated with the remaining asymmetric blocks appended at the end to form an initial sequence pair candidate solution.

## 4.3 Annealing Schedule

In our annealing engine, the initial temperature is set to $10^6$ and will drop at a rate of 0.9. At each temperature, an *IterNum* number of iterations are performed.

## 4.4 Cost Function

As stated in the problem formulation in section 2, the objective is to construct a floorplan $F$ satisfying all the symmetry constraints and placement constraints and minimizing a cost function $cost(F) = area(F) + \alpha \times wire(F)$ where $\alpha$ is a user given weight, $area(F)$ is the total area of $F$ and $wire(F)$ is the total wire length of $F$ measured by the half-perimeter estimation.

# 5. EXPERIMENTAL RESULTS

We have done two sets of experiments. In the first set, we want to compare our approach with previous works. Since most of the previous works on analog placement do not consider other general placement constraints, we only have symmetry constraints in the first set of data. According to the comparisons in [2] and [13], the segment tree approach [2] has out-performed the SP approach in [1] and the O-tree approach in [12] in both run time and solution quality, while the TCG-S approach in [13] is comparable with the segment tree approach in solution quality but the run time is longer especially when the size of the data set increases. Therefore, we have chosen to implement and compare with the segment tree approach. In the second set of experiments, we want to demonstrate the effectiveness of our approach in handling both symmetry and other general placement constraints simultaneously.

## 5.1 Comparisons with Previous Approach

Simulated annealing is used in both the segment tree approach and our floorplanner as the optimization engine. All the experiments are performed on a Sun Ultra 5/270 with a 270MHz CPU and 128MB RAM. In order to have a fair comparison, the cost functions ($\alpha$ is set to 0) and the annealing schedules are all the same, except that the variable *IterNum* (as described in section 4.3) in our floorplanner is set to 5 while it is set to 50 in the segment tree approach since the segment tree approach can run faster for each iteration and we want to make sure that the total time spent are the same in both methods so that a fair evaluation can be made. The data sets are randomly generated with block areas uniformly distributed from 4 to 1271 and all are soft blocks with aspect ratio bounds $[0.5, 2]$.

**Table 1: Comparisons with the Segment Tree Approach**

| Data Set | Block No. | Symmetry Groups | Our Approach | | Segment Tree | |
|---|---|---|---|---|---|---|
| | | | Time (s) | Area | Time (s) | Area |
| D50 | 50 | 8,7,7,4,6 | 100.4 | 18576 | 122.4 | 20100 |
| D70 | 70 | 9,4,9,5,9 | 212.5 | 22575 | 270.7 | 25730 |
| D100 | 100 | 4,12,4,11,12 | 475.4 | 45540 | 518.5 | 53424 |
| D120 | 120 | 5,4,4,7,8 | 510.3 | 49126 | 726.4 | 46530 |

Table 1 displays the results of this first set of experiments. The third column shows the number of symmetry groups and the number of blocks in each group, e.g., the first data set has 5 symmetry groups with 8, 7, 7, 4 and 6 blocks respectively. Column 4 and 6 show the run times in second while column 5 and 7 show the areas. We can see from this table that our floorplanner can perform better than the segment tree approach in three out of the four cases when given a similar amount of run time. Fig. 5 shows the placement of one of the four data sets in Table 1. If we allow the annealing process to run for about an hour, very good results can be obtained and an example is shown in fig. 6.

## 5.2 Symmetry Constraint with Other Placement Constraints

In this second set of experiments, we want to study the performance of our floorplanner when there are both symmetry and other general placement constraints. All the experiments are performed on a P4 machine with a 2.6GHz CPU and 1024MB RAM. Table 2 displays the results of this second set of experiments. The third column shows the number of symmetry groups and the number of blocks in each group as in table 1 while the fourth column shows the number and types of other placement constraints, e.g., the first data set has one symmetry group with five blocks and four alignment constraints. Column 5 and 6 show the run time in second and the dead space percentage. We can see from this table that our floorplanner can handle both the symmetry constraint and the other general placement constraints very effectively. Fig. 7 shows one result packing of these data sets.

# 6. REFERENCES


[1] F. Balasa and K. Lampaert. Symmetry within the Sequence-Pair Representation in the Context of Placement for Analog Design. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 19(7):712–731, 2000.

[2] F. Balasa, S. C. Maruvada, and K. Krishnamoorthy. On the Exploration of the Solution Space in Analog Placement with Symmetry Constraints. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 23(2):177–191, 2004.

[3] Y. C. Chang, Y. W. Chang, G. M. Wu, and S. W. Wu. B*-Trees: A New Representation for Non-Slicing Floorplans. *Proceedings of the 37th ACM/IEEE Design Automation Conference*, 2000.

[4] Pei-Ning Guo, Chung-Kuan Cheng, and Takeshi Yoshimura. An O-Tree Representation of Non-Slicing Floorplan and Its Applications. *Proceedings of the 36th ACM/IEEE Design Automation Conference*, pages 268–273, 1999.

**Table 2: Results with Both Symmetry and Other Placement Constraints**

| Data Set | Block No. | Symmetry Groups | Other Constraints | Time (s) | Dead Space (%) |
|---|---|---|---|---|---|
| D40a | 40 | 5 | 4 align | 92.4 | 8.84 |
| D40b | 40 | 5 | 4 align | 95.7 | 7.33 |
| D40c | 40 | 5 | 4 boundary | 102.1 | 9.37 |
| D70a | 70 | 9,4,9,5,9 | 1 range, 2 align, 2 abut, 1 maxsep | 127 | 7.81 |
| D70b | 70 | 9,4,9,5,9 | 1 preplace, 2 boundary, 2 align, 1 maxsep | 123 | 11.12 |
| D70c | 70 | 9,4,9,5,9 | 10 align | 90 | 7.43 |
| D70d | 70 | 9,4,9,5,9 | 1 align, 1 preplace, 1 abut, 1 maxsep, 1 boundary | 117 | 8.95 |
| D70e | 70 | 9,4,9,5,9 | 10 abut | 114 | 11.85 |
| D70f | 70 | 9,4,9,5,9 | 3 boundary, 3 maxsep, 3 range | 200 | 5.69 |
| D70g | 70 | 9,4,9,5,9 | 10 range | 282 | 6.61 |

**Figure 5: D100 with 100 Blocks and 5 Symmetry Groups.**

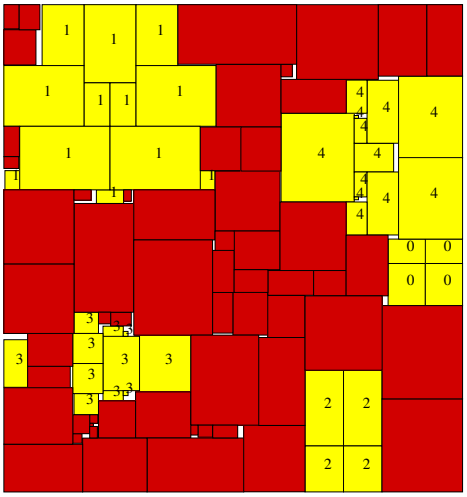

**Figure 6: D100 with 100 Blocks and 5 Symmetry Groups (anneal for about an hour).**

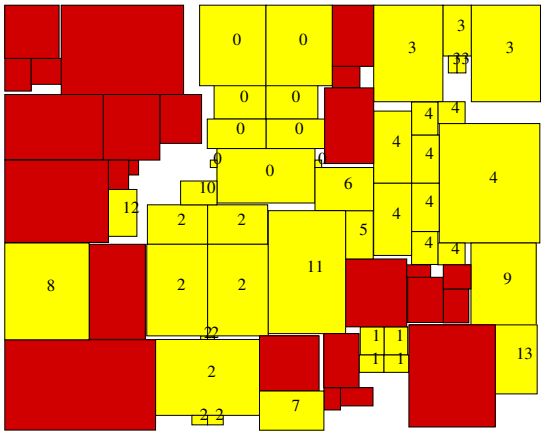

**Figure 7: Result Packing of D70d (5 symmetry groups, align(t,8,9), preplace(10,50,50), range(11,0,0,100,100), maxsep(v,12,13,50), boundary(b,7), abut(v,5,6)).**


[5] J. Cohn and D. Garrod and R. Rutenbar and L. Carley. *Analog Device-level Automation*. Kluwer Acad. Publi., 1994.

[6] J. Cohn, *et al.* KOAN/ANAGRAMII: New Tools for Device-Level Analog Layout. *IEEE J. Solid-State Circuits*, 26(3):330–342, 1991.

[7] Shinichi Kouda, Chikaaki Kodama, and Kunihiro Fujiyoshi. Improved Method of Cell Placement with Symmetry Constraints for Analog IC Layout Design. *International Symposium on Physical Design*, pages 192–199, 2006.

[8] K. Lampaert, G. Gielen, and W. Sansen. A Performance-driven Placement Tool for Analog Integrated Circuits. *IEEE J. Solid-State Circuits*, 30(7):773–780, 1995.

[9] J. M. Lin and Y. W. Chang. TCG-S: Orthogonal Coupling of P*-admissible Representations for General Floorplans. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 24(6), 2004.

[10] E. Malavasi, E. Charbon, E. Felt, and A. Sangiovanni-Vincentelli. Automation of IC Layout with Analog Constraints. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 15(8):923–942, 1996.

[11] H. Murata, K. Fujiyoushi, S. Nakatake, and Y. Kajitani. Rectangle-Packing-Based Module Placement. *Proceedings IEEE International Conference on Computer-Aided Design*, pages 472–479, 1995.

[12] Y. X. Pang, F. Balasa, K. Lampaert, and C. K. Cheng. Block Placement with Symmetry Constraints based on the O-tree Nonslicing Representation. *Proceedings of the 37th ACM/IEEE Design Automation Conference*, pages 464–467, 2000.

[13] G. M. Wu, J. M. Lin, Y. W. Chang, and R. H. Chuang. Placement with Symmetry Constraints for Analog Layout Design. *IEEE Asia and South Pacific Design Automation Conference*, pages 1135–1138, 2005.

[14] Evangeline F. Y. Young, Chris C. N. Chu, and M. L. Ho. Placement Constraints in Floorplan Design. *IEEE Transactions on Very Large Scale Integration Systems*, 12(7):735–745, 2004.