Analog Placement with Common Centroid Constraints

Qiang Ma, Evangeline F. Y. Young
Department of Computer Science and Engineering
The Chinese University of Hong Kong
Email: {qma,fyyoung}@cse.cuhk.edu.hk

K. P. Pun
Department of Electronic Engineering
The Chinese University of Hong Kong
Email: kppun@ee.cuhk.edu.hk

***Abstract*— In order to reduce parasitic mismatch in analog circuits, some groups of devices are required to share a common centroid while being placed. Devices are split into smaller ones and placed with a common center point. We will address this problem of handling common centroid constraint in placement. A new representation called *Center-based Corner Block List* (C-CBL) is proposed which is a natural extension of Corner Block List (CBL) [1] to represent a common centroid placement of a set of device pairs. C-CBL is complete and non-redundant in representing any common centroid mosaic packings with pairs of blocks to be matched. To address the same problem with an additional constraint that devices are required to be placed uniformly to average out the parasitic errors, a grid-based approach is proposed. Experimental results show that both approaches are fast and promising, and have high scalability that even large data sets can be handled effectively.**

I. Introduction

In today's SoC designs, both digital and analog parts of a circuit will be implemented on one chip. Placement of the analog part is an error prone and time-consuming process. In analog placement, parasitic mismatch induced by the layout will affect the circuit performance significantly. Parasitic mismatch may lead to higher offset voltage and degraded power-supply rejection ratio [3]. Considerations of symmetry and common centroid constraints during placement can help to reduce these errors. For symmetry constraint, pairs of cells are required to be placed symmetrically with respect to a horizontal or a vertical axis and this problem has been extensively studied [4], [5], [6], [7], [8], [9], [10]. This paper will address the problem of handling common centroid constraint in placement.

As mentioned above, component mismatch have adverse effects on many analog circuits. One of the most important sources of mismatch is process gradient, like oxide thickness, threshold voltage, resistor-layer thickness, etc. These kinds of mismatch can be effectively suppressed by common centroid layout, which refers to a layout style in which a set of devices have a common center point. Devices will be split into a number of smaller ones and placed with the same center point. The devices can be arranged with a common centroid in one dimension as for a differential pair, or in two dimensions as for a capacitor or resistor array in data converters. Two-dimensional arrangement is desired especially when the number of devices is large, because of the floorplanning constraint and the close proximity requirement. Close proximity is desirable for better electrical properties such as parasitic matching and thermal gradients.

Common centroid constraint is not the same as 2-D symmetry constraint. To satisfy a 2-D symmetry constraint, blocks can only be placed along the horizontal and vertical axes of the symmetry group. The blocks will have a common centroid in such placements but there are still a lot of other kinds of feasible placements satisfying the common centroid constraint. Examples of these two types of constraints are shown in Figure 1. To the best of our knowledge, this is the first work in the literature to handle common centroid constraint for groups of blocks in placement.

We will present two approaches to solve the problem. In the first approach, the Corner Block List (CBL) [1] data structure is extended naturally to represent common centroid placements. This new representation, called *Center-based Corner Block List* (C-CBL), is complete and non-redundant in representing any mosaic packing of a common centroid group with pairs of blocks to be matched. Simulated annealing is used as the basic searching engine. In order to demonstrate the effectiveness of using C-CBL, we have compared this approach with an extension of a previous work [10] on symmetry constraint and the experimental results are promising in both run time and placement quality. To address the same problem with an additional constraint that devices are required to be placed uniformly to average out the parasitic errors, another grid-based approach is proposed and studied. In this second approach, the devices in a group are split into unit devices of the same dimensions. A set of feasible packings for each common centroid group will first be generated by placing these unit devices into a 2-D array satisfying the common centroid constraint. Experimental results show that both approaches are fast and promising, and have high scalability that even large data sets can be handled effectively.

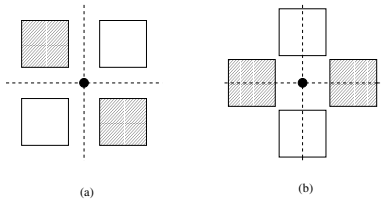

Fig. 1. 2-D symmetry (b) does not include placement (a) which also satisfies the common centroid constraint.

In the following, we will define the problem in section II, we will then discuss our two approaches in section III and IV. The experimental results will be reported in section V before the conclusion and discussion appear in the last section.

II. Problem Formulation

We are given a set of n blocks of areas A_i and aspect ratio bounds $[l_i, u_i]$ where $i = 1 \ldots n$, together with a set of m nets $N_1, N_2 \ldots N_m$. We are also given a set of p common centroid groups $G_1, G_2 \ldots G_p$. Each group G_i containing s_i devices is labeled by $\{g_{i1}, g_{i2} \ldots g_{is_i}\}$ where g_{ij} denotes the area of the j^{th} device in G_i and has an aspect ratio bound. The s_i devices, which can be further split into a set of smaller devices, are required to be placed in a common centroid geometry. Note that the shapes of those smaller devices produced after splitting are bounded by the corresponding aspect ratio range. Throughout this paper, we use n_i to denote the total number of devices in G_i after the splitting process, and each original device g_{ij} is called a *subgroup* of G_i. Our objective is to obtain a placement F of the circuit satisfying the common centroid constraint for all the common centroid groups, while minimizing a cost function $cost(F) = area(F) + \lambda \times wire(F)$ where $area(F)$ is

1-4244-1382-6/07/$25.00 ©2007 IEEE

the total area of F, $wire(F)$ is the total wire length of F measured by the half-perimeter estimation, and λ is a parameter specifying the relative importance between these two terms. Figure 2 shows a sample packing with two common centroid groups.

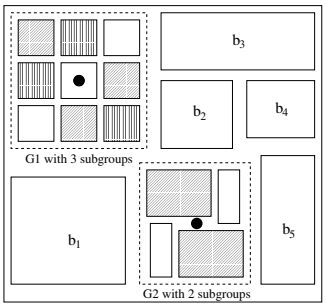

Fig. 2. A packing with two common centroid groups.

III. Methodology

Simulated annealing is used with the sequence pair representation [2] as the basic searching engine in our approach. In order to maintain the performance and stability of a circuit, it is desirable to restrict all the devices of a common centroid group to form a cluster in the final placement, without interleaving with other blocks not in the same group. Therefore, each common centroid group can be treated as a super-block in the sequence pair, whereas its internal structures will be handled specially in order to satisfy the common centroid constraint. During the annealing process, a set of random moves will be performed to perturb both the *global SP* that contains the super-blocks representing the common centroid groups and the other blocks of the circuits, and the internal structures of the common centroid groups. In the following, we will introduce the Center-based Corner Block List representation, which is a natural extension of CBL to represent a packing of block pairs with one common centroid.

A. Center-based Corner Block List

One way to solve the placement problem with common centroid constraint is to split each device into two smaller devices of the same dimension, and require all these pairs from the same group to be placed with a common centroid. This approach is good since the total number of devices after splitting will be small. Now, each common centroid group G_i contains s_i pairs of devices $(a_{i1}, b_{i1}), (a_{i2}, b_{i2}) \ldots (a_{is_i}, b_{is_i})$ with sizes $\frac{g_{i1}}{2}, \frac{g_{i2}}{2} \ldots \frac{g_{is_i}}{2}$. In order to represent a common centroid placement of all these s_i pairs of blocks, a new representation called Center-based Corner Block List (C-CBL) is used, which is a natural extension of CBL. In the following, we will first review CBL briefly.

1) Review of Corner Block List: Figure 3 shows a mosaic packing and its horizontal and vertical constraint graph (HCG and VCG). Mosaic floorplan, first introduced in [1], was defined as a packing without empty rooms or crossing junctions. In these constraint graphs, the nodes represent the vertical or horizontal segments in the packing while the edges represent the rooms for placing the modules. Additional nodes, labeled by W, E, S and N, are inserted to represent the west, east, south and north boundaries of the chip. A block is called a *corner block* if its corresponding edges in the HCG and VCG are pointing to the E and N node respectively. The *orientation* of a corner block is defined by the orientation of the T-junction at its lower left corner. If the T-junction is rotated by 90° counterclockwise, its orientation is vertical and is denoted by a bit zero; if the T-junction is rotated by 180° counterclockwise, it is horizontal and is denoted by a bit one.

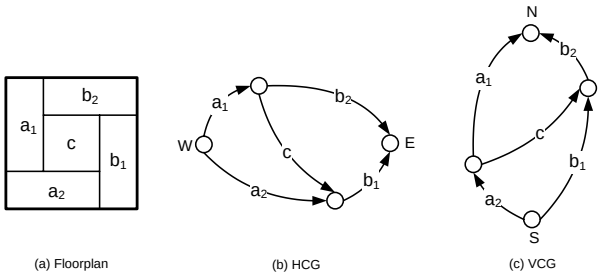

Fig. 3. A (a) mosaic packing with its (b) horizontal and (c) vertical constraint graphs.

Given a mosaic packing, its CBL representation can be obtained by *deleting* the corner blocks one after another. The corner block of a packing can be deleted by sliding the non-crossing segment the T-junction at its lower left corner to the right (for horizontal block) or to the top (for vertical block). In this way, a new packing with one less block will be formed and the deletion process can be repeated until there is no more blocks left. The CBL, a 3-tuple (S, L, T) where S is a sequence of block names, L is a list of orientations and T is a list of T-junction information, can be constructed in this recursive deletion process in a reversed order. Taking the packing in Figure 3 as an example, its CBL will be $(S = (a_2, a_1, c, b_1, b_2), L = (0, 1, 1, 0), T = (0, 0, 1, 1))$. The T-junction information is the number of T-junctions uncovered if the corner block is deleted from the packing. For example, in Figure 3, if the corner block b_2 is deleted, one T-junction will be uncovered, so the T-junction information of block b_2 is one. Notice that both L and T are one element less than S because the orientation and T-junction information for the last deleted block (the first element in S) do not matter. On the other hand, a mosaic packing can be obtained from a CBL by iteratively inserting back each block according to the S list and the orientation and T-junction information.

2) Center-based Corner Block List (C-CBL): The Center-based Corner Block List extends the basic CBL to represent the placement of a common centroid group. It is different from the original CBL that C-CBL works on *corner block pairs*. Given a mosaic packing, we can define the following:

Definition 1: **UR Corner Block** - A block is a UR (upper right) corner block if its corresponding edges in the HCG and VCG are pointing to the E and N nodes respectively. Its orientation is horizontal (vertical) if the T-junction at its lower left corner is rotated by 180° (90°) counterclockwise. A UR corner block can be deleted from the packing by sliding the non-crossing segment of its lower left T-junction to the right (top) if it is horizontal (vertical). When there is only one pair of blocks in the packing, the UR corner block is horizontal (vertical) if the other one is on its left (below it).

Definition 2: **LL Corner Block** - A block is an LL (lower left) corner block if its corresponding edges in the HCG and VCG are pointing from the W and S nodes respectively. Its orientation is horizontal (vertical) if the T-junction at its upper right corner is rotated by 0° (270°) counterclockwise. An LL corner block can be deleted from the packing by sliding the non-crossing segment of its upper right T-junction to the left (bottom) if it is horizontal (vertical). When there is only one pair of blocks in the packing, the LL corner block is horizontal (vertical) if the other one is on its right (top).

Definition 3: **Corner Block Pair** - The UR corner block and the LL corner block of a packing are called a corner block pair if and

only if they have the same orientation and have the same number of T-junctions uncovered after their deletions.

Given a mosaic packing of n pairs of blocks[1] satisfying the common centroid constraint, we can decompose the packing iteratively by removing it's UR and LL corner blocks. Each removed pair, representing devices a_{ij} and b_{ij}, will have the same orientation and T-junction information, i.e., they form a corner block pair. The C-CBL representation can thus be defined naturally according to this deletion process. In the following, we assume that there is always a single block located at the center of the placement, for the sake of unified representation. This center block will be a dummy one with zero width and height when there are only n pairs of blocks.

Definition 4: **Center-based Corner Block List (C-CBL)** - A C-CBL is a four tuple (C, S, L, T) where

- C is the name of the center block;
- S is a sequence of x block names;
- L is a list of x bits representing the orientations;
- T is a list of x integers representing the T-junction information.

(1) Obtaining C-CBL from a packing

Given a mosaic packing satisfying the common centroid constraint, we can obtain the corresponding C-CBL by recursively deleting the corner block pairs. For example, suppose that we are now given a common centroid placement containing five pairs of blocks $\{g_1(a_1, b_1), g_2(a_2, b_2), g_3(a_3, b_3), g_4(a_4, b_4), g_5(a_5, b_5)\}$. Figure 4 shows the process to obtain the C-CBL by iteratively performing corner block pair deletion. Note that the center block labeled as c_{dmy} denotes a dummy center. We can prove the following theorem:

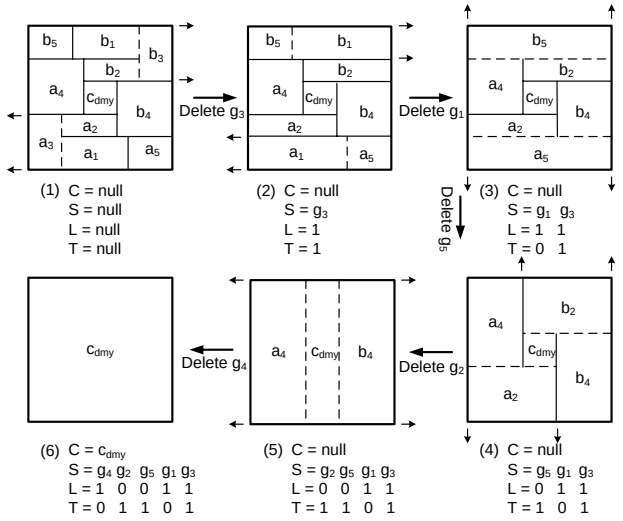

Fig. 4. Obtaining the C-CBL from a packing.

Theorem 1: A unique C-CBL can be obtained from a mosaic packing with n pairs of blocks (or n pairs plus one block) satisfying the common centroid constraint.

(2) Obtaining a packing from a C-CBL

A mosaic packing satisfying the common centroid constraint can be constructed from a C-CBL(C, S, L, T) by initializing the packing with the center block C and then inserting the corner block pairs one by one according to the lists' information. Figure 5 shows this packing procedure for the previous example, with the C-CBL given as $(C = c_{dmy}, S = (g_4, g_2, g_5, g_1, g_3), L = (1, 0, 0, 1, 1), T = (0, 1, 1, 0, 1))$. Similar to the CBL representation, an arbitrary 4-tuple may not correspond to a feasible packing since the number of available T-junctions may be less than the required number of T-junctions to be covered (according to the T list) when we try to insert the UR and LL corners blocks. However, this condition can be checked and maintained easily in the implementation. We can prove the following theorem:

[1]The C-CBL can also handle n pairs of blocks plus one single block.

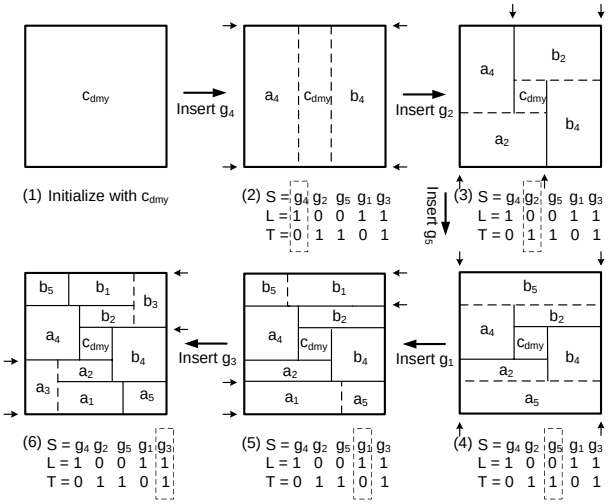

Fig. 5. Obtaining the packing from a C-CBL.

Theorem 2: A packing constructed from a C-CBL satisfies the common centroid constraint.

B. Realization of Common Centroid Placement from C-CBL

To obtain the coordinates of the blocks inside a common centroid group G_i from a C-CBL, we will first construct the HCG and VCG for G_i, as described in section III-A.1. Then a single source longest path algorithm will be performed on both graphs, with node W and S as the sources respectively. After this, we will obtain an (x, y) coordinates for each block, together with the width $w(G_i)$ and height $h(G_i)$ of the whole packing. However, we need to adjust the coordinates of the blocks since they may not satisfy the common centroid constraint after this bottom-left-compacted packing process. We will re-compute the coordinates of those UR corner blocks (assume that they are the b_{ij} blocks) as follows:

$$\begin{aligned} b_{ij}.x &= w(G_i) - a_{ij}.x - width(b_{ij}), \\ b_{ij}.y &= h(G_i) - a_{ij}.y - height(b_{ij}) \end{aligned}$$

In addition, the coordinates of the center block b_c (if it is not dummy) can be computed as $(w(G_i)/2 - w(b_c)/2, h(G_i)/2 - h(b_c)/2)$. Obviously, after this adjustment, the blocks in G_i will satisfy the common centroid constraints, while staying within the smallest possible region $w(G_i) \times h(G_i)$ without any overlapping. This procedure is performed whenever the C-CBL of any common centroid group G_i is perturbed, after which G_i will be regarded as a super-block with width $w(G_i)$ and height $h(G_i)$ in the global SP describing the whole circuit.

C. Global Sequence Pair

We use sequence pair to represent the whole circuit in the annealing process. In sequence pair, a packing is encoded by a pair of sequences, (α, β), both of which are combinations of module names. We use α_i to denote the module occupying position i in α, and use α_A^{-1} to denote the position of A in α. The relative position between

two modules specified by a SP is as follows. If $\alpha_A^{-1} < \alpha_B^{-1}$ and $\beta_A^{-1} < \beta_B^{-1}$, module A is to the left of module B; if $\alpha_A^{-1} < \alpha_B^{-1}$ and $\beta_B^{-1} < \beta_A^{-1}$, module A is above module B.

In the global SP, each common centroid group G_i is regarded as one entity and we will use an index $n+i$ to label this super-block in our implementation, where n is the total number of blocks given. The global SP consists of n ordinary blocks without any common centroid constraint and p super-blocks representing the common centroid groups, so the length of each sequence is $n+p$.

D. Simulated Annealing

Simulated annealing is used as our basic searching engine:

1) Set of Moves: There are seven kinds of moves employed to perturb a current candidate solution. The moves can be divided into two categories as described below.

(1) Global SP perturbations:

1) *Change aspect ratio* - The aspect ratio of a block b_i that is not in any common centroid group is changed.
2) *Swap two blocks in* α - Two randomly chosen blocks (including super-blocks) are swapped in the α sequence.
3) *Swap two blocks in both sequences* - Two randomly chosen blocks (including super-blocks) are swapped in both α and β.

(2) C-CBL perturbations:

1) *Change aspect ratio* - A pair of blocks in an arbitrary group are randomly selected and have their shapes changed.
2) *Swap two devices* - Two pairs are randomly chosen from an arbitrary group and their positions in the corresponding S list are swapped.
3) *Change orientation* - A randomly chosen bit in the L list of an arbitrary group is toggled.
4) *Change T-junction information* - The value at a randomly chosen position in the T list of an arbitrary common centroid group is changed. The new value must not exceed the number of current available T-junctions.

2) Cost Function: We use the cost function $cost(F) = area(F) + \lambda \times wire(F)$ to evaluate a packing F, where $area(F)$ is the area of F and $wire(F)$ is the total wire length estimated by the half perimeter method. The parameter λ specifies the relative importance between area and wire length.

3) Annealing Schedule: Before the annealing process, a random walk with 1000 moves will be performed to determine the value of the parameter λ in the cost function, where we try to balance the relative weighting between area and wire length. In our annealing engine, the temperature is set to 10^5 at the beginning and will drop at a rate of 0.95. At each temperature, $(n+\sum_{i=1}^{p} n_i)/2$ random moves are performed, where n is the number of blocks without constraints and $\sum_{i=1}^{p} n_i$ denotes the total number of blocks in all the common centroid groups after splitting. The annealing process stops when the temperature falls below 10^{-5}.

IV. An Alternative Approach with Grid

The C-CBL based approach can solve the problem effectively. However, in some cases, we may want to distribute all the devices in one group uniformly to average out the influence of the parasitic effects. Therefore, in the following, we will propose another approach to deal with this specific case, where we split all devices within a group into a set of small devices of the same dimension, called *basic unit block*[2], and place them with a common centroid. We assume that the areas of the devices in each group are integral ratios of each other. For example, for a group $\{2, 4, 8\}$, its three devices will have one, two and four basic unit blocks after splitting.

[2]The dimensions of the basic unit block will be fixed according to the aspect ratio range of the group.

A. Candidate Placements of Each Group

In this second approach, the common centroid groups are also handled separately. We will first independently generate a collection of feasible placement solutions for each group G_i, followed by a pruning procedure to eliminate the redundant solutions dominated by others. Then, each common centroid group will be treated as a super-block in the global SP, with a set of discrete aspect ratios available.

Since the n_i blocks in G_i are of the same shape and size, we can naturally pack them in a regular $N_c \times N_r$ grid geometry, where N_c is the number of columns and N_r is the number of rows in the grid. The grid is empty at the beginning and the basic unit blocks are then assigned symmetrically (in both the x and y-direction) into the vacant positions. For each group G_i, we will generate at most n_i different feasible packings called *j-column-compacted solutions* for $j = 1, 2 \ldots n_i$. A j-column-compacted solution is defined as:

Definition 5: **j-Column-Compacted Solution** - A common centroid placement of G_i is a j-column-compacted solution (denoted by S_{ij}) if it is a grid-based placement with j columns to accommodate all the basic unit blocks, while minimizing the number of rows used.

Here, we use column number to count the solutions. Note that this is the same as using row number, since we will have a move in the annealing process to rotate one whole common centroid group. Figure 6 shows all the j-column-compacted solutions of a common centroid group consisting of two subgroups, each of which contains two basic unit blocks. Before the annealing process, all the feasible j-column-compacted packing solutions for $1 \le j \le n_i$ will be generated for each group G_i. Details of this solution generation procedure will be discussed in the following sections.

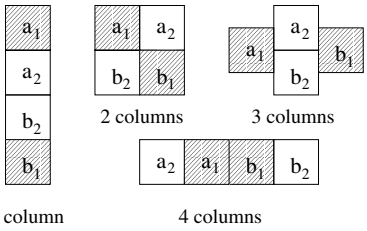

Fig. 6. All j-column-compacted solutions of a common centroid group consisting of four basic unit blocks.

1) Matching of Basic Unit Blocks in Subgroups: To ensure that all subgroups of G_i share a common centroid, the basic unit blocks in each subgroup will form a number of pairs (or a number of pairs plus a single block or a triple). Notice that if all these pairs, triples or single block have the same centroid, the whole group will share a common centroid $C(G_i)$. Therefore, we need to pair up the basic unit blocks in each subgroup first. There are three cases to consider: (1) A subgroup contains an even number of basic unit blocks:

We will just arbitrarily pair up all the basic unit blocks in the subgroup, since they are all equivalent. For example, assume that the j^{th} subgroup of G_i contains $2t$ blocks labeled as $b_1, b_2, \ldots b_{2t}$, we will pair them up and require them to be symmetrical in both the x and y direction as follows:

$$sym\_x(b_{2l-1}) = sym\_y(b_{2l-1}) = b_{2l},$$
$$sym\_x(b_{2l}) = sym\_y(b_{2l}) = b_{2l-1}.$$

where $sym\_x(b)$ ($sym\_y(b)$) denotes the block that is symmetrical to b in the x-direction (y-direction). The above formulation indicates

that b_{2l-1} and b_{2l} are placed symmetrically to each other with respect to the common centroid $C(G_i)$ in both the x and y-direction, where $l = 1, 2 \ldots t$.

(2) A subgroup contains only a single basic unit block:

We put this single block b_s at the center of the group placement, so b_s should be symmetrical with itself in both the x and y-direction:

$$sym\_x(b_s) = b_s, sym\_y(b_s) = b_s.$$

It is impossible to handle the situation that more than one subgroups contain a single basic unit block, since these blocks will overlap with each other due to the common centroid requirement. However, this is sometimes the case in realistic analog circuits. We will deal with this situation in a reasonable way of approximation that those single blocks will be placed adjacent to each other at the center position.

(3) For a subgroup containing an odd number (≥ 3) of basic unit blocks, there are two options to handle this case:

- Option 1: Select one block b in this subgroup as a self symmetry block, and pair up the remaining blocks.
- Option 2: Three blocks (labeled as b_x, b_y and b_d) are selected and placed in symmetrical relative positions as shown in Figure 7:

$$sym\_x(b_x) = b_x, sym\_y(b_x) = b_d,$$
$$sym\_x(b_y) = b_d, sym\_y(b_y) = b_y,$$
$$sym\_x(b_d) = b_y, sym\_y(b_d) = b_x.$$

The remaining blocks in the subgroups (must be in even number) will be paired up arbitrarily. We will choose between these two options in different situations and details will be explained in later sections.

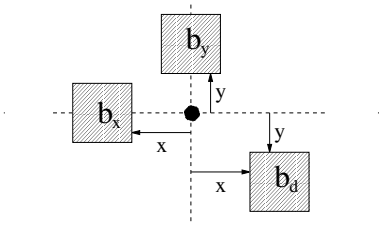

Fig. 7. Arrangement of a triple with a common centroid.

2) Assignment into Grids: After the pairing up process, all the basic unit blocks in G_i will fall into three categories: singles, pairs and triples (singles and triples can only be formed from odd number subgroups). We will then construct all the j-column-compacted solutions S_{ij} for G_i, where $1 \leq j \leq n_i$. A procedure called Grid Position Assignment will be performed to generate S_{ij}, in which the singles, pairs and triples of G_i will be assigned into a grid of j columns in such a way that they share a common centroid, while trying to minimize the number of rows. The basic unit blocks in G_i will be assigned into the grid following an order of (1) singles, (2) triples and (3) pairs. Notice that the number of columns of a grid is bounded below according to the following theorem:

Theorem 3: We need a grid of size at least $N \times N$ to accommodate a group with a total of t triples or $t-1$ triples plus one single in a common centroid grid-based layout where $N = t$ when t is odd, and $N = t + 1$ otherwise. Besides, all the triples and single can be placed in this minimum size grid with a common centroid.

Figure 8 illustrates an example of this position assignment step. In this example, we are going to generate a 3-column-compacted packing solution for a common centroid group consisting of three subgroups, g_1, g_2 and g_3, which contain one (a_1), three (a_2, b_2, c_2), and two (a_3, b_3) basic unit blocks respectively. After pairing up, there will be one single (a_1), one triple (a_2, b_2, c_2) and one pair (a_3, b_3). We will begin with an empty 3×3 grid (the minimum number of columns is three according to Theorem 3, and there is no 1-column-compacted or 2-column compacted solutions). The single a_1 is given the highest priority, thus a_1 will be first considered and assigned at the center point of the grid. The triple is taken into consideration next, and the three blocks in g_2 will be assigned into the grid with relative positions as described in Figure 7. Finally, the two blocks in g_3 will be inserted into the two symmetrical vacant positions left. If, unfortunately, there are no such symmetrical positions available, we will increase the row number of the grid until such positions appear. According to Theorem 3, all the triples and single can be assigned into the grid, and the remaining pairs can always be placed symmetrically by increasing the number of rows (note that the number of columns is unchanged). After performing this position assignment step, we can easily obtain the absolute coordinates of each basic unit block in the group. In this step, we do not differentiate between the singles, pairs and triples from different subgroups. This will be taken care of in the moves of the annealing process.

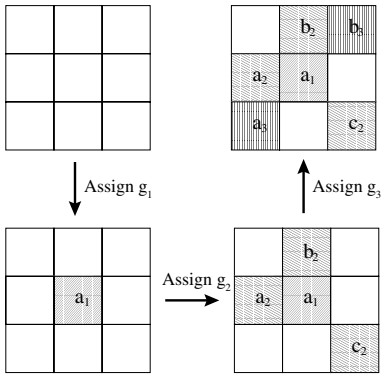

Fig. 8. An example of assigning basic unit blocks into a grid.

3) Solution Pruning: After all the feasible j-column-compacted solutions for each group G_i have been generated, we will delete those redundant ones (dominated by at least one other solution in both the width and height). A procedure *Solution_Pruning()* will be performed to eliminate the redundant solutions, leaving only with those potentially optimal ones. For example, in Figure 6, the 3-column-compacted solution is redundant because it is dominated by the 2-column-compacted solution which also uses two rows but one column less, and it will be deleted by the pruning procedure.

4) Summary: We can now summarize the steps to generate the candidate placement solutions for each common centroid group. Note that this *All_Solution_Gen()* procedure is called only once at the beginning and will not contribute much to the total run time.

Pseudocode *All_Solution_Gen()*

// This procedure is to generate all the feasible j-column-compacted
// packing solutions, where $1 \leq j \leq n_i$, for each common
// centroid group G_i, and store them in a look up table.

1. *For $i = 1$ to p do // p is the number of groups*
2. *Pair up the basic unit blocks in G_i;*
3. *(min_column, min_row) = get_min_grid_size();*
4. *For $j = min\_column$ to n_i do*
5. *S_{ij} = grid_position_assignment(G_i, j);*
6. *Calculate the exact coordinates of each block in S_{ij};*
7. *Calculate the width and height of S_{ij};*
8. *Store S_{ij} to the look-up table;*
9. *Solution_Pruning(G_i);*

B. Simulated Annealing

This grid approach is similar to the C-CBL approach that a global SP is used to represent the whole circuit. In the annealing process, the cost function, the annealing schedule and the random moves to perturb the global SP are exactly the same as that in the C-CBL approach, but we need to re-define the moves that perturb a common centroid group as follows:

1) *Change packing solution of a group* - A common centroid group G_i is randomly selected and a new packing solution is chosen from its look-up table. This move will also include changing the mapping of the singles, pairs and triples with different subgroups in G_i.
2) *Rotate a group* - In this move, a common centroid group is randomly selected and rotated.

V. Experimental Results

Since no previous works have studied this common centroid placement problem and no existing benchmarks are available, we have constructed our own benchmarks from some realistic analog circuits[3], on which we have tested our algorithms. In order to demonstrate the effectiveness of our methods, we will compare them with an extension of a previous work [10] on 1-D symmetry to solve this common centroid problem. The extension can be done in a straight forward way of adding dummy vertices and edges with variable weights to both the vertical and horizontal constraint graphs. Table I is a comparison between this extended method and our C-CBL approach. Result shows that our C-CBL approach can perform much better. In Table I, the column *Block No.* refers to the number of blocks without common centroid constraints, and the column *CC Groups* shows the information of the common centroid groups, e.g., for data set *c3*, there are two common centroid groups with eight and ten devices respectively (note that each of them will be split into two). The deadspace percentage is 8.29% on average.

We have also performed another set of experiments in which the devices are split into a number of basic unit blocks. Table II shows the results with our grid approach on this data set. The column *CC Groups* indicates the information of the common centroid groups, e.g., for data set $c5\_int$, there are two common centroid groups both of which have seven devices with the same area ratios of 1:2:4:8:16:32:64. The results show that our algorithm is fast and has high scalability. Even large data sets can be handled in a reasonable amount of time. The deadspace percentage is 7.24% on average. The algorithms, including the extended version of [10], are implemented in the C programming language and all the experiments are performed on a Sun Blade 2500 with a 1.6 GHz CPU and 2 GB RAM.

Figure 9-13 are several resultant packings generated by our approaches. Note that in each common centroid group, the blocks belonging to the same device are labeled with the same number. The three packings in Figure 9, 10 and 11 are generated by the C-CBL approach. Figure 9 is a packing with two common centroid groups. Notice that one group has a center block while the other one does not. Figure 10 is a packing of a test case containing one common centroid group and one 1-D symmetry group. The 1-D symmetry group is handled using the approach in [10]. We can see that the C-CBL method can be combined with other approaches smoothly to handle mixed types of placement constraints. Figure 11 is a resultant packing of the test case $c8$, which is a large data set consisting of 300 ordinary blocks and six common centroid groups. A placement of a digital analog converter circuit (DAC) containing a common centroid group with nine devices with area ratios {1, 1, 2, 4, 8, 16, 32, 64, 128} is shown in Figure 12. Another test case containing a common centroid group of 11 devices with area ratios $\{1, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$ is generated to test the performance of the grid based approach in handling subgroups with odd numbers of basic unit blocks. We can see in Figure 13 that the odd subgroups are properly arranged to satisfy the common centroid requirement.

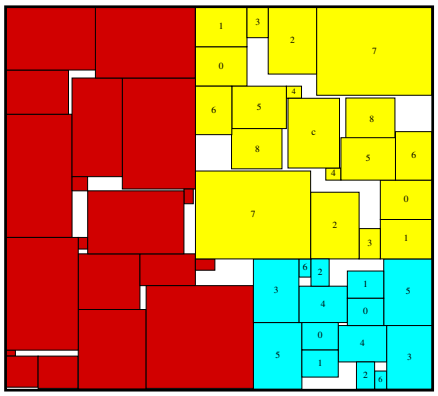

Fig. 9. Resultant packing with two common centroid groups.

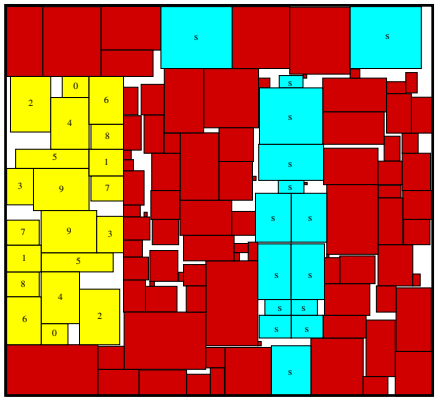

Fig. 10. Resultant packing with one common centroid group and one 1-D symmetry group.

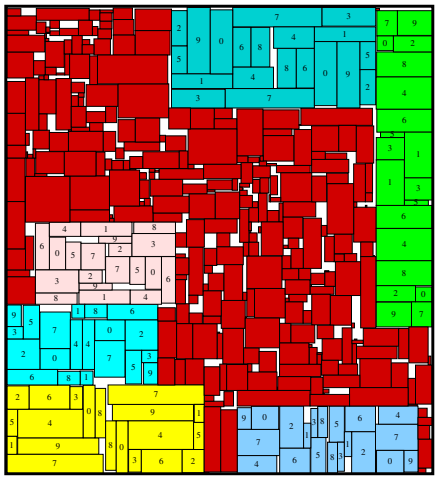

Fig. 11. Resultant packing with 300 blocks and six common centroid groups.

VI. Conclusion

In this paper, we study the placement problem for analog circuits with common centroid constraints. We defined the problem and

[3]We will make these benchmarks publicly available on the www.

TABLE I

EXPERIMENTAL RESULTS WITH C-CBL APPROACH

Data Set	Block No.	CC Groups		Net No.	C-CBL				[10]			
		No.	Device No.		Area	Dead Space (%)	HPWL	Run Time (s)	Area	Dead Space (%)	HPWL	Run Time (s)
c1	30	1	8	36	512.9	4.24	303.7	4.51	517.4	5.07	312.5	96.30
c2	50	1	10	62	821.8	6.85	534.2	14.31	824.3	7.13	568.2	307.2
c3	70	2	8,10	104	1236	6.56	589.3	43.73	1274	9.38	633.7	914.5
c4	100	3	8,9,10	138	1820	7.89	631.8	129.9	1875	10.6	713.9	3914
c5	120	4	10,10,10,10	187	2654	8.71	802.2	254.5	-	-	-	-
c6	150	5	10,10,10,10,10	242	3201	8.93	1021	508.1	-	-	-	-
c7	200	5	10,10,10,10,10	379	4020	10.8	1912	1046	-	-	-	-
c8	300	6	10,10,10,10,10,10	622	4848	12.4	2348	3396	-	-	-	-
Average					-	8.29	-	-	-	-	-	-

TABLE II

EXPERIMENTAL RESULTS WITH GRID APPROACH

Data Set	Block No.	CC Groups		Net No.	Area	Dead Space (%)	HPWL	Run Time (s)
		No.	Area Ratios of each Group					
c1_int	30	1	{1,1,2,3,4,5,6,7,8,9,10}	42	581.6	6.81	392.4	7.13
c2_int	50	1	{1,1,2,4,8,16,32,64,128}	85	1251	3.06	1036	23.7
c3_int	70	1	{1,3,5,7,9,11,13,15,17,19}	105	1119	5.71	855.1	42.4
c4_int	100	1	{1,1,2,4,8,16,32,64,128}	160	2063	5.04	1629	150.3
c5_int	120	2	{1,2,4,8,16,32,64},{1,2,4,8,16,32,64}	212	2493	6.11	1873	262.1
c6_int	150	2	{1,3,5,7,9,11,13},{1,1,2,4,8,16,32,64,128}	265	3377	9.25	2274	473.8
c7_int	200	2	{1,1,2,4,8,16,32,64},{1,1,2,4,8,16,32,64,128}	412	4682	10.3	3022	984.6
c8_int	300	2	{1,1,2,4,8,16,32,64},{1,1,2,4,8,16,32,64,128}	580	6057	11.6	3973	3026.2
Average					-	7.24	-	-

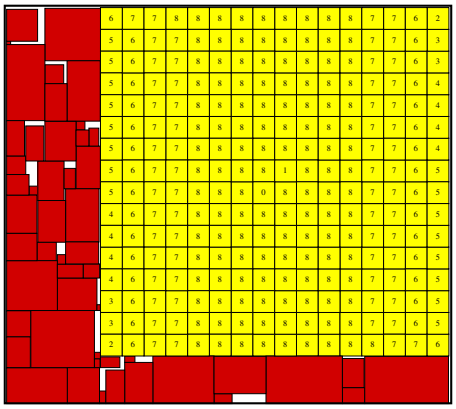

Fig. 12. Resultant packing of a DAC circuit ($c2\_int$) with a large common centroid group.

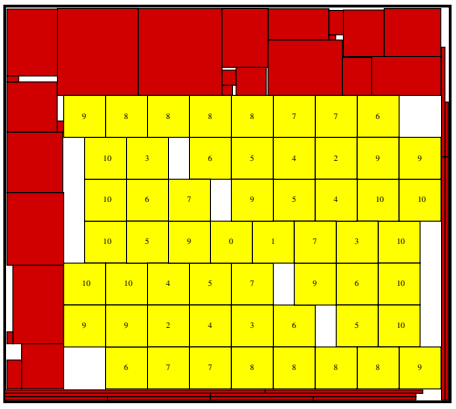

Fig. 13. Resultant packing of a test case ($c1\_int$) with a common centroid group {1, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10}.

devised a new representation, called Center-based Corner Block List, for common centroid placement with pairs of blocks to be matched. Based on this representation, we proposed a simulated annealing based method to solve the common centroid placement problem. Experimental results have shown that placements satisfying the constraints can be generated efficiently with an average dead space of 8.29%. To deal with the special cases that devices are required to be distributed uniformly, we proposed another approach that produces regular grid-based placements for the common centroid groups and the experimental results are also very promising. Furthermore, since both proposed approaches use global Sequence Pair, they can actually be combined to solve this common centroid placement problem, with each common centroid group handled by either of the two approaches, according to the specific characteristics of each group.

REFERENCES

[1] X. Hong, G. Huang, Y. Cai, J. Gu, S. Dong, C.-K. Cheng, and J. Gu. Corner Block List: An Effective and Efficient Topological Representation of Non-Slicing Floorplan. *Proceedings of the International Conference on Computer-Aided Design*, 2000.

[2] 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.

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

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

[5] 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.

[6] 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.

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

[8] 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.

[9] 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.

[10] Y.-C. Tam, Evangeline F.-Y. Young, and Chris C.-N. Chu. Analog Placement with Symmetry and Other Placement Constraints. *Proceedings of the International Conference on Computer-Aided Design*, 2006.