Abstract—In the layout design of analog integrated circuits, common centroid constraints can effectively reduce parasitic mismatch and thermal gradients, which may otherwise degrade circuit performance. In this paper, we first survey the frontline research of common-centroid-aware layout placement algorithms and implement a graph-based packing approach to handle common centroid constraints in the analog layout design. Then we develop a naïve center-based packing approach that features fast execution speed although not competitive in terms of area and dead space. By comparing both approaches above, we further propose a contour-based packing scheme that can achieve the best performance of area and dead space with slight increase in CPU time. Our experimental results demonstrate the effectiveness of the proposed scheme and validate a new direction in this area.

I. INTRODUCTION

Very large scale integration (VLSI) is an integrated circuit technique that can set millions of transistors into a single chip. VLSI started its life in the 1970s followed with the development of the complex semiconductor technology [1]. Because of the development of VLSI, great changes and convenience have been brought into our lives.

A VLSI design cycle consists of a series of design stages. They are system specification, architectural design, functional design, logic design, circuit design, physical design, fabrication and packaging [1]. In this paper, we focus on the placement problem in the physical design step. The VLSI physical design uses a geometric approach to express the representation of circuit. This kind of geometric representation is also named layout [1]. All the components of the circuit are converted into specific shapes on the layout. The goal of physical design is to find a reasonable layout of all the circuit components so that the chip can achieve a good performance.

Analog placement is a key step that must be considered in the physical design. It is time-consuming for gaining a reasonable placement, since several factors should be considered, such as area, shape and net length [1][2]. In analog placement, there always exists parasitic mismatch problem. Parasitic mismatch may reduce the power supply rejection ratio and result in high offset voltage. Such problems would greatly impact the performance of circuit [2].

In order to alleviate the influence from parasitic mismatch, common centroid constraints are used in placement [2]. In this way, several devices will be divided into some common centroid groups that are controlled under the common centroid constraints. The devices in each group share a common centroid point and are surrounded around this point. Figure 1 is a sample of placement with common centroid groups.

In this paper, we will discuss a common centroid analog placement which presented by Qiang Ma et al [2][6][10]. In their design, the Center-based Corner Block List (C-CBL) is used for representing common centroid placements. The C-CBL is extended from the Corner Block List (CBL), and is a complete and non-redundant representation for placement of common centroid devices [3]. For the placement algorithm, simulated annealing technique is used as the basic searching engine [2].

II. OVERVIEW OF ANALOG PLACEMENT

In general, a placement problem can be expressed as follows. To give a circuit which includes a set of blocks, each block has its area and net-list that indicates the interconnections with other blocks in the circuit, a designer should find one approach to reasonably place all the blocks so
that whole chip can be well routed and achieved a good performance [1].

A good placement can be determined by three factors, they are area of circuit, result of routing and circuit performance [1]. The high performance of a circuit can be achieved by minimizing the lengths of the critical paths, which is usually approximated by minimizing the length of the longest path [1]. However, in practice, some factors may affect each other. For instance, the minimization of layout area may cause the increasing of maximum wire length; conversely, the minimizing of maximum wire length also can change the area of circuit [1].

III. METHODOLOGY

In the placement, there are \( n \) ordinary blocks and \( p \) common centroid groups. Sequence pair (SP) is used to represent the whole circuit [4][5]. In the SP, each common centroid group is treated as a super block to avoid the interleaving with other ordinary block in the final placement [2]. C-CBLs are used to indicate block groups with common centroid constraint and simulated annealing are used for implementation of placement algorithm. In the following, we will introduce two packing approaches for C-CBL. The first one is introduced in [2], which first generates constrain graphs according to the information of C-CBL and then packs the modules based on such graphs. The other one is center-based; it simply places modules around the center point.

A. Graph-based Approach

For realizing the placement from a C-CBL, firstly we need to gain coordinates of all the blocks in a common centroid group. In [2], horizontal and vertical constrain graphs are generated depended on the information of C-CBL. The graphs are used for packing in later process.

According to these two graphs, the plane coordinate of each block and the width \( W \) and height \( H \) of whole packing can be obtained by using the bottom-left-compacted packing method [2]. Then, we will re-compute the coordinates of blocks so that to satisfy the common centroid constraint [2]. For the re-computation, we can seem one block in a common centroid pair as an anchor point, and then adjust the coordinate of the other block. For instance, if the block which we want to adjust is a UR corner block \( (b) \), the formulas for calculating the plane coordinate are shown as following, where \( a \) is corresponding LL corner block with \( b \) [2]:

\[ b_x = W - a_x - \text{width}(b), \quad b_y = H - a_y - \text{height}(b) \]

For example, there is a C-CBL, \( (C = c, S = (g1, g2), L = (1, 0), T = (0, 1)) \). There are five modules described in this C-CBL, which are shown in following figure:

![Figure 2. Common Centroid Modules](image)

First, we generate two constrains graphs, and then to compact place all modules according to such graphs. The graphs and initial packing result are shown in figure 3:

![Figure 3. Constrain Graphs and Initial Packing Result](image)

After that, a longest path algorithm is worked on both graphs to obtain the width and height of whole C-CBL packing. Finally, we re-compute the coordinates of center block and upper-right blocks. The final packing result is given in Figure 4.

![Figure 4. Final Packing Result of Graph-based C-CBL Approach](image)

B. Center-based Approach

We also developed a simple approach which differs from the approach in [2]. This center-based approach does not need the constrain graphs. In packing process, the center block \( C \) is initially placed. Then other common centroid groups are placed around the block \( C \) according to their \( L \) and \( T \) information.

![Figure 5. Center-based Packing](image)

We use the previous example to explain the packing procedure. There is a C-CBL, \( (C = c, S = (g1, g2), L = (1, 0), T = (0, 1)) \). Figure 5 shows the steps for packing given C-CBL. At first, \( c \) is placed, then there is one pair of blocks, their orientations are 1 and \( T \) information is 0. Hence, we horizontally laid \( a1 \) and \( b1 \) around \( C \), and the north (south) bound of \( a1 \) (\( b1 \)) is at the same level of \( c \)'s north (south) bound. After that, \( a2 \) and \( b2 \) are placed. Their orientations are
0 and T information is 1. We vertically place them, and the adaptive y-coordinates are calculated according to the covered blocks. The west (east) bound of a2 (b2) is at the same level of covered block’s west (east) bound which is closer to west (east) boundary. The final result is shown in Figure 10(3). This approach runs very fast and the comparison with the traditional C-CBL approach will be discussed in Section V.

IV. DRAWBACKS AND IMPROVEMENT

For both C-CBL approaches which introduced in previous section, there is a common drawback. Since we treated each common centroid group as a super block, it makes some area consumption. Figure 9 is an example of graph-based C-CBL placement result, which contains 110 blocks with three common centroid groups. As it shown, there has some empty space around each common centroid group, and such space may be reused for placing some blocks.

Based on the graph-based C-CBL approach, we record the packing contour when blocks are being placed. Then, we compare current block with recorded contour to determine whether there is any room for reused. Dead space can be reused by applying such contour-based placement. Following is the pseudo code for contour processing:

```
1  Initialize hContour and vContour;
2  for each block in β sequence do
3    b = β[i];  // i = 1 to n;
4    create hTemp and vTemp;
5    compare b with each block recorded in hContour, if b covers any blocks, remove such blocks from hContour to hTemp;
6    compare b with each block recorded in vContour, if b covers any blocks, remove such blocks from vContour to vTemp;
7    if b is a superblock obtain its west and south contours, calculate the distance between b’s west/south contour and vTemp’s/hTemp’s contour so that determine the distance of horizontal/vertical movement;
8    else if b covers a superblock, do calculates the distance between b’s west/south bound and east/north contour of the covered superblock so that determine the distance of horizontal/vertical movement;
9    adjust coordinates of b;
10   push b into hContour and vContour;
11  End for
```

In the code, hContour and vContour are used to record the blocks’ information in horizontal and vertical contours. hTemp and vTemp are created to hold the contours which covered by current block. Then we will use some figures to describe the overview of such contour-based packing.

![Figure 6](image1.png)  (a) Super block 1 and (b) its real shape

Giving a sequence pair \((α, β) = (<4 3 5 1 2>, <5 3 4 1 2>)\), the dimensions for every block are: \(1(4\times10), 2(2\times6), 3(3\times3), 4(2\times3)\) and \(5(5\times4)\). Block 1 is a super block in the SP; its real shape is showed in figure 6(b).

![Figure 7](image2.png)  First three steps of contour-based packing

According to our fast-sp algorithm [5], we place blocks by the block order in sequence β. While one block is placing, we update the horizontal and vertical contours of current packing. The first three steps are showed in figure 8, the horizontal and vertical contours are represented by different dash line in the figure.

![Figure 8](image3.png)  (a) Packing before usable space calculation. (b) Packing after the calculation. (c) Packing result for SP \((α, β) = (<4 3 5 1 2>, <5 3 4 1 2>)\).

While placing the block 1, since it is a super block, we need to calculate whether there is usable space between its real contours and current SP packing contours. If there is usable space, all the blocks belongs to super block 1 will be moved to occupy such space. Then we update the SP packing contours and keep placing followed blocks. Figure 8 (c) shows the packing result for SP \((α, β) = (<4 3 5 1 2>, <5 3 4 1 2>)\).

V. EXPERIMENTAL RESULTS

All the algorithms are implemented in C++ [8][9]. Figure 9 was generated by GnuPlot [7]. It showed the packing results for graph-based and center-based C-CBL packing approaches. Figure 9(a) is the graph-based C-CBL packing results, as we can see, common centroid blocks are compacted around four boundary of packing. On the contrary in Figure 9(b), the common centroid blocks are compacted around the center.

![Figure 9](image4.png)  (a) Graph-based C-CBL  (b) Center-based C-CBL

110 Blocks with 3 common centroid groups
Table I shows the comparison results between graph-based C-CBL approach placement and center-based one. In the table, names of data set are shown in first column and second column refers to the number of blocks in each data set. In CC Groups column, it indicates the number of common centroid groups, and the amount of devices for each group is described in the amount column. The amount of devices for each group is described in parenthesis. The rest of three columns respectively indicate the packing area, dead space and run time.

As we can see, the graph-based approach is able to gain less packing area and dead space. However, the run time of center-based approach is greater than graph-based one.

**TABLE I. PACKING RESULTS OF GRAPH-BASED AND CENTER-BASED C-CBL APPROACHES**

<table>
<thead>
<tr>
<th>Data set#</th>
<th>Blk #</th>
<th>CC Groups</th>
<th>Area (e+06)</th>
<th>Dead space (%)</th>
<th>Run Time (s)</th>
<th>Area (e+07)</th>
<th>Dead space (%)</th>
<th>Run Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>b1</td>
<td>9</td>
<td>1 (5)</td>
<td>56.4</td>
<td>17.5</td>
<td>2.3</td>
<td>55.2</td>
<td>15.71</td>
<td>0.2</td>
</tr>
<tr>
<td>b2</td>
<td>33</td>
<td>3 (5, 4, 6)</td>
<td>1.3</td>
<td>17.2</td>
<td>31.8</td>
<td>1.6</td>
<td>20.01</td>
<td>2.6</td>
</tr>
<tr>
<td>b3</td>
<td>49</td>
<td>4 (4, 5, 9, 8)</td>
<td>41.2</td>
<td>14.0</td>
<td>66.7</td>
<td>43.4</td>
<td>18.12</td>
<td>7.1</td>
</tr>
<tr>
<td>b4</td>
<td>65</td>
<td>3 (11, 11, 15)</td>
<td>36.6</td>
<td>14.2</td>
<td>68.6</td>
<td>39.0</td>
<td>16.81</td>
<td>12.3</td>
</tr>
<tr>
<td>b5</td>
<td>110</td>
<td>3 (17, 20, 9)</td>
<td>55.1</td>
<td>17.5</td>
<td>146.7</td>
<td>55.3</td>
<td>17.71</td>
<td>36.3</td>
</tr>
</tbody>
</table>

Figure 10 shows the packing results for contour-based and graph-based C-CBL packing approaches. As we can see in figure 10(a), the dead space of one common centroid group is successfully reused.

As we can see in Table II, the graph-based C-CBL and contour-based C-CBL have similar processing time. However, the result showed that packing area and dead space of contour-based C-CBL are less than original C-CBL. The dead space percentage for graph-based C-CBL is 16.13% on average, such percentage for contour-based C-CBL is 9.72% on average.

**TABLE II. PACKING RESULTS OF GRAPH-BASED AND CONTOUR-BASED C-CBL APPROACH**

<table>
<thead>
<tr>
<th>Data set</th>
<th>Blk#</th>
<th>CC Groups</th>
<th>Graph-based C-CBL</th>
<th>Contour-based C-CBL</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Area (e+06)</td>
<td>Dead space (%)</td>
</tr>
<tr>
<td>b1</td>
<td>9</td>
<td>1 (5)</td>
<td>56.4</td>
<td>17.5</td>
</tr>
<tr>
<td>b2</td>
<td>33</td>
<td>3 (5, 4, 6)</td>
<td>1.3</td>
<td>17.2</td>
</tr>
<tr>
<td>b3</td>
<td>49</td>
<td>4 (4, 5, 9, 8)</td>
<td>41.2</td>
<td>14.0</td>
</tr>
<tr>
<td>b4</td>
<td>65</td>
<td>3 (11, 11, 15)</td>
<td>36.6</td>
<td>14.2</td>
</tr>
<tr>
<td>b5</td>
<td>110</td>
<td>3 (17, 20, 9)</td>
<td>55.1</td>
<td>17.5</td>
</tr>
</tbody>
</table>

VI. CONCLUSIONS

In this paper, we study the common centroid constrained analog placement. We implemented the C-CBL approach which introduced in [2]. Meanwhile, we also found a naïve C-CBL approach, and made some improvement based on the original C-CBL to reduce the packing area.

By comparing three C-CBL approaches, center-based C-CBL placement has the fastest processing speed, but it consumes more packing area and results in large dead space as its simple packing rule. Contour-based C-CBL placement makes a good improvement based on traditional C-CBL approach and the packing result can be optimized by reusing the dead space.

From what has been discussed above, we truly believe that if there is a way which can combine the advantages of center-based and contour-based approaches, we would obtain sound results in processing time and packing area.

REFERENCES


