A Novel Approach for Automatic Common-Centroid Pattern Generation

Vadim Borisov, Kerstin Langner, and Jürgen Scheible
Reutlingen University, Robert Bosch Center for Power Electronics,
Reutlingen, Germany
{vadim.borisov, kerstin.langner, juergen.scheible}@reutlingen-university.de

Benjamin Prautsch
Fraunhofer IIS/EAS
Dresden, Germany
benjamin.prautsch@eas.iis.fraunhofer.de

Abstract—This paper introduces a novel placement methodology for a common-centroid (CC) pattern generator. It can be applied to various integrated circuit (IC) elements, such as transistors, capacitors, diodes, and resistors. The proposed method consists of a constructive algorithm which generates an initial, close to the optimum, solution, and an iterative algorithm which is used subsequently, if the output of constructive algorithm does not satisfy the desired criteria. The outcome of this work is an automatic CC placement algorithm for IC element arrays. Additionally, the paper presents a method for the CC arrangement evaluation. It allows for evaluating the quality of an array, and a comparison of different placement methods.

I. INTRODUCTION

Most integrated devices, such as resistors, capacitors, and transistors show extensive absolute deviations from their intended values of typically 20% or more caused by stochastic variations during the IC fabrication process. However, if two similar devices occupying the same piece of silicon are compared, it can be observed that they have nearly the same electrical parameters because they have experienced identical manufacturing conditions [1]. Therefore, the circuitry of analog ICs is frequently based on differential design techniques, which take advantage from this observation. Typical analog circuit examples are differential pairs and current mirrors (composed of similar transistors) or voltage dividers (composed of similar resistors or switched capacitors). Thus, the precision of analog ICs depends on the degree of device matching. That is, their performance can be optimized by reducing the parameter mismatch of the involved devices, which can be achieved through sophisticated layout techniques [2].

Before discussing these techniques, it has to be stated that beside the mentioned fabrication tolerances there are two additional major sources of mismatch: use conditions and fringe effects. The first appear in the use of a circuit, where parameter offsets can occur, e.g. caused by temperature gradients or by piezoresistive effects due to mechanical pressure gradients. The latter occur systematically at the physical boundaries of the devices which usually cause nonlinearities (e.g. arising from stray fields or stray currents) in the electric parameters dependence of the device dimensions. Fabrication tolerances, as well as use conditions, have in common that their impact on mismatch usually expands with increasing distance of the devices to be matched. Therefore, these effects are summarized as gradient mismatch.

In the following, the main layout techniques to minimize unwanted effects caused by systematic, and gradient mismatch in analog ICs are listed [3]. These techniques apply to all elements in a group of \( n \) devices which have to be matched. That is, they have to realize intended ratios of their electrical parameters with an optimal precision.

- Minimize distances between the elements (suppresses gradient mismatch)
- Split devices into same-sized sub-devices (suppresses fringe effects)
- Interdigitate sub-devices of different devices (improves suppression of gradient mismatch)
- Place elements in the same direction (synchronizes direction of current flow)
- Place elements symmetrical (additionally improves suppression of gradient mismatch)
- Add dummy elements around the group (suppresses fringe effects caused by different neighboring elements)
- Increase size of basic elements (minimizes fringe effects due to better area-boundary-ratio, but leads to increasing gradient effects [1])

Moreover, elements matching is a tremendous challenge for deep-submicron technologies, due to a small transistor size, a high number of fabrication steps, and subwavelength lithography effects.

A. Automatic Common-Centroid Pattern Generation

Nowadays, the field of integrated circuit design still does not have an accepted, general approach for automated IC device placement on CC structures [4]. This can be explained by the complexity of the problem, the diversity of EDA software, and the challenges of evaluating the arrangement of devices in the array.

Nevertheless, there have been many attempts to find a method for generic CC elements matching structures [4]–[8]. Numerous tools exist; however, it is not clear how they handle different inputs (e.g. incomplete arrays). Another reason for the lack of general solutions is the difficulty of integrating them into existing design flows; normally the CC pattern generators are technology dependent or software related.

Therefore, in most cases, device arrangement into the CC pattern is a hand-operated job based on existing pattern solutions [2] and design knowledge [9]. The manual work...
The paper is divided into four sections. In Section II, the proposed approach for sorting a set of elements into a CC pattern is presented. Section III presents the implementation and result of the study. Finally, Section IV contains a summary of the work.

II. THE PROPOSED APPROACH

In this section, the proposed approach for the CC pattern generation is presented. In order to achieve handling of various inputs, our method is applying two primary algorithms: a Constructive Algorithm (CA) and an Iterative Algorithm (IA).

Since the CA in general, requires much less computational effort, it is used for getting an initial arrangement of elements close to an optimum solution for the IA in the first stage.

In the second stage, the IA makes a decision using an evaluation method. If the result of the CA meets desirable criteria for CC matching, the proposed approach will output the final arrangement [1].

The following subsections discuss each algorithm separately.

A. The Constructive Algorithm

The basic idea of the CA can be described mathematically by four steps. The first step is:

\[ M_I = M_{in}^T \cdot P_{n-1} \]

(1)

where \( M_{in} \) is a square matrix with \( n \) rows, where each row represents a group of elements, and \( P_{n-1} \) is the permutation matrix corresponding to the inverse of the following permutation. If \( n \) is even \((n = 2k)\):

\[ \pi = \begin{pmatrix} 1 & 2 & 3 & \ldots & k & k+1 & k+2 & \ldots & n \\ 1 & 3 & 5 & \ldots & n-1 & 2 & 4 & \ldots & n \end{pmatrix} \]

(2)

If \( n \) is odd \((n = 2k - 1)\):

\[ \pi = \begin{pmatrix} 1 & 2 & 3 & \ldots & k & k+1 & k+2 & \ldots & n \\ 1 & 3 & 5 & \ldots & n-1 & 2 & 4 & \ldots & n-1 \end{pmatrix}(n-1) \]

(3)

For the next step, \( M_I \) is split into sub-matrices in the following way: let \( m_1, \ldots, m_n \) be the rows of \( M_I \), if \( n \) is even, each sub-matrix consists of two rows:

\[ M_{I,1} = \begin{pmatrix} m_1 \\ m_2 \end{pmatrix}, \ldots, M_{I,k} = \begin{pmatrix} m_{n-1} \\ m_n \end{pmatrix} \]

(4)

if \( n \) is odd, \( M_{I,k} \) contains only one row.

For the next step, the mirror function for matrices has to be defined; it allows to reflect each row in a matrix:

\[ \text{Mirror} : A(m, n) \rightarrow A(m, n), (a_{ij}) \mapsto (a_{i(n-j+1)}) \]

(5)

where \( A(m, n) \) is the set of matrices with \( m \) rows and \( n \) columns, with \( i = 1, \ldots, m \) and \( j = 1, \ldots, n \).

In the third step, each sub-matrix of \( M_I \) is divided into four same-sized sub-matrices as shown in the equation below:

\[ M_{I,i} = \begin{pmatrix} \alpha_1 & \beta_1 \\ \alpha_2 & \beta_2 \end{pmatrix} \]

(6)

After partitioning into sub-matrices the mirror function is applied to \( \alpha_1 \). Moreover, the bottom sub-matrices exchange positions and for \( \beta_2 \) the mirroring function is used. This leads to:

\[ M_{II,i} = \begin{pmatrix} \text{Mirror}(\alpha_1) & \beta_1 \\ \text{Mirror}(\beta_2) & \alpha_2 \end{pmatrix} \]

(7)

In the last step, all altered sub-matrices are combined into \( M_{CC} \) and it is multiplied with the permutation matrix from Eq. 1, but from the left side:

\[ M_{CC} = P_{n-1} \cdot M_{II} \]

(8)

where \( M_{CC} \) is a matrix with (nearly) common-centroid arrangement.

A similar algorithm based on the one presented above can be used for incomplete "matrices". The mathematical model cannot describe this; therefore and for the sake of illustration the algorithm is presented in a graphical way (Fig. 1).

The input array \( M_{in} \) is \( \{2A, 2B, 6C, 6D, 6E, 6F\} \) for the graphical example (Fig. 2 (a)). The first step of the algorithm is to initialize all the elements according to the groups to which they belong. The second step is to transpose the array, converting each column into a row as depicted in Fig. 2 (b). After that, the algorithm places odd and even columns separately into two groups, as depicted in Fig. 2 (c). In the next step, the procedure equivalent to Eq. 7 is applied for each two rows. The highlighted (bold lines) elements reverse and move to the left part of the array (Fig. 2 (d)), the untouched elements are combined in the right part of the array. For the last step (Fig. 2 (e)), the longest rows are moving to the middle of a new array, and after that, the remaining rows are added to the new array. Lastly, additional dummies can be added to this result.
in order to achieve a rectangular shape and same surrounding, which is desirable for a high-level element matching.

Since, the CA does not always provide an optimum solution for a broad input range, the IA algorithm is used.

B. The Evaluation Method

The proposed method allows to use multiple evaluation methods, one of them is to calculate group and array centers, and then calculate an offset vector. The proposed evaluation method is divided into three major steps. The first step is the calculation of the center of the array and the individual centers of each group of sub-devices. In Fig. 3 (a) the center of group A (a square symbol) is \((2, 3)\), group B shares the same center with the array’s center (a circle symbol) in \((2, 2)\), and the center of group C (a triangle symbol) is \((2, 1.5)\).

In the second step, the evaluation method determines the offset vectors between the center of the array and centers of each group of devices. For the arrangement in Fig. 3 (a) the offset vectors are \(\vec{v}_A = (0, 1)\), \(\vec{v}_B = (0, 0)\) and \(\vec{v}_A = (0, -0.5)\).

For the reason of simplicity the 1-norms of the offset vectors are summed up to the Total Mismatch Offset Coefficient (TMOC) of the array. As an example of TMOC for the array in Fig. 3 (a) is:

\[
TMOC = \|\vec{v}_A\|_1 + \|\vec{v}_B\|_1 + \|\vec{v}_C\|_1 = 1 + 0 + 0.5 = 1.5
\]

Hence, the TMOC for the perfect CC arrangement must be 0.

C. The Iterative Algorithm

If the output data from the CA does not meet the desirable criteria (e.g. TMOC is more than 1), based on knowledge from the evaluation step, the exploration method (Fig. 1) changes the order of the elements in the array in order to convert it to a perfect CC placement. In case the perfect CC arrangement can not be found, the IA returns the arrangement with the lowest possible TMOC. Due to a huge computation time to traverse all possible combination for each input array:

\[
O \left( d! \prod_{i=1}^{n} k_i! \right)
\]

where \(d\) is a total number of devices in an array, and the \(k_i\) is a number of sub-devices in a group \(n\), a genetic algorithm was used, with the swap mutation technique [11].

III. IMPLEMENTATION & RESULT

The proposed algorithm was tested within the IIP Framework [10] which is an environment for the development and execution of flexible and highly technology-independent analog generators. The object-oriented structure of this environment is used to make placement algorithms such as the proposed CC pattern calculation accessible for each IIP generator. This way, the placement algorithm can be maintained at a central location rather than for each generator separately. In particular, the CC algorithm was tested in parameterizable IIPs for the user-defined and automatic generation of schematics and layouts of both, transistor arrays and capacitor arrays.

The generators mentioned can create a variety of topologies such as current mirrors, differential pairs, or capacitor arrays connected in either series or parallel. With aid of the pattern calculation algorithms such as the proposed one highly matching layouts can efficiently be implemented for all of those flexible IIP generators.

The input parameters for the CC algorithm include the following:
Table I

<table>
<thead>
<tr>
<th>Test</th>
<th>Input</th>
<th>Output</th>
<th>CPU Time†</th>
<th>Number of Matrices</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>{2A, 2B}</td>
<td>A1 B1 B2 A2</td>
<td>16.4 (\mu)s</td>
<td>1,000</td>
</tr>
<tr>
<td>2</td>
<td>{4A, 4B}</td>
<td>B1 A1 A2 B1 A2 A1 A3 B3 A4 B4</td>
<td>31 (\mu)s</td>
<td>3,000</td>
</tr>
<tr>
<td>3</td>
<td>{2A, 2B, 2C, 2E}</td>
<td>A1 B1 C1 E1 D0* E2 C2 B2 A2</td>
<td>100 ms</td>
<td>7,000</td>
</tr>
<tr>
<td>4</td>
<td>{1A, 3B, 3C}</td>
<td>B1 B2 C1 D0* A1 D0* C2 C3 B3</td>
<td>103 ms</td>
<td>10,000</td>
</tr>
<tr>
<td>5</td>
<td>{2A, 4B, 8C, 16E}</td>
<td>D0* E1 C1 B1 E2 C3 E4 E5 B2 C2 D6 D0* E7 C8 C1 A4 C8 E8 D0* D0* E9 C5 A2 C6 E10 D0* E11 C7 B3 E12 D0* E14 E15 B4 C8 E16 D0*</td>
<td>56.6 (\mu)s</td>
<td>15,000</td>
</tr>
</tbody>
</table>

* Dummy elements.
† A 2.4 GHz dual-core Intel Core i5-4258U processor was used.

- Number of devices, which have to match the CC pattern;
- Number of columns or rows;
- Number of dummy rings.

Table I shows the various outputs for the proposed array with computation time. It can be noticed that tests 3 and 4 have a significantly higher computation time compared to tests 1, 2 or 5. This can be explained because the IA was used, since the output from the CA was not satisfying the chosen criteria (e.g. TMOC has to be less than 1).

Fig. 4 displays results of the distribution of TMOC for about 7000 randomly created different input matrices with up to 10 rows and columns, after constructive (blue) and iterative (green) algorithms. It can be seen that the IA largely decreases the total mismatch offset vector after the constructive algorithm. The standard deviation for the output from CA is 4.08 and for the output from IA is 1.98 respectively.

The described algorithm is implemented in about 200 lines of python code for a commercial IC design framework.

IV. Conclusion

In this paper, the common-centroid placement of devices was considered. This has resulted in a general approach for sorting same-sized devices into the perfect common-centroid pattern, if possible. Even if such an arrangement is not possible, the proposed method can find a solution that minimizes the error. Additionally, the evaluation method for common-centroid arrangements was introduced and used to verify the outcome of the CC placement generator. Due to its systematic and simplistic nature, the proposed algorithm can be quickly implemented in any EDA software.

Acknowledgment

Vadim Borisov would like to acknowledge for fruitful discussion and support Uwe Eichler and Torsten Reich.

References