Common-Centroid Capacitor Placement Considering Systematic and Random Mismatches in Analog Integrated Circuits

Cheng-Wu Lin, Jai-Ming Lin, Yen-Chih Chiu, Chun-Po Huang, and Soon-Jyh Chang
Department of Electrical Engineering, National Cheng Kung University, Tainan, Taiwan, R.O.C.
lcw@sscas.ee.ncku.edu.tw; jmlin@ee.ncku.edu.tw; chew@livemail.tw; gppo@sscas.ee.ncku.edu.tw; soon@mail.ncku.edu.tw

ABSTRACT

One of the most important issues during the analog layout phase is to achieve accurate capacitance ratios. However, systematic and random mismatches will affect the accuracy of the capacitance ratios. A common-centroid placement is helpful to reduce the systematic mismatch, but it still needs the property of high dispersion to reduce the random mismatch [10]. To deal with this problem, we propose a simulated annealing [15] based approach to construct a common-centroid placement which exhibits the highest possible degree of dispersion. To facilitate this framework, we first propose the *pair-sequence* representation to represent a common-centroid placement. Then, we present three operations to perturb the representation, which can increase the degree of dispersion without breaking the common-centroid constraint in the resulting placement. Finally, to enhance the efficiency of our simulated annealing based approach, we propose three techniques to speed up our program. The experimental results show that our placements can simultaneously achieve smaller oxide-gradient-induced mismatch and larger overall correlation coefficients (i.e., higher degree of dispersion) than [10] in all test cases. Besides, our program can run much faster than [10] in larger benchmarks.

Categories and Subject Descriptors: B.7.2 [Integrated Circuits]: Design Aids – Layout, Placement and routing
General Terms: Algorithms, Design
Keywords: Analog placement, capacitor array

1. INTRODUCTION

The key performance of many analog integrated circuits (ICs) is related to the accuracy of capacitance ratios [3, 4, 10], such as analog-to-digital converters and switched-capacitor circuits [2]. Among these circuits, a successive-approximation-register (SAR) analog-to-digital converter (ADC) has attracted more attention recently due to its low power consumption (see Figure 1), and it is widely used in biomedical chips or portable/battery-powered instruments. One of the most important components in the SAR ADC is a capacitor array, which contains a set of capacitors C_1, …, C_{n+1} and these capacitors have to satisfy a predefined capacitance ratio (i.e., $C_1 = C_2$, $C_{i+1} = 2C_i$, $i = 2, …, n$). A capacitor array is considered **matched** if its capacitance ratio exactly meets the predefined value. The linearity of the SAR ADC is highly related to the matching of the capacitor array.

A well-matched capacitor array may become mismatched after IC manufacturing. The causes of mismatch in IC fabrication process can be divided into two categories: *systematic mismatch* and *random mismatch* [3, 4]. We illustrate each of them in the following:

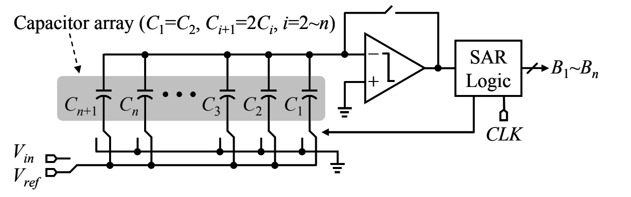
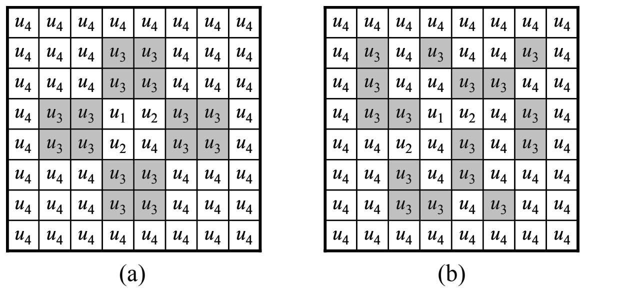


Figure 1. **Architecture of an *n*-bit SAR ADC. The linearity of the SAR ADC is highly related to the matching of the capacitor array.**



(a)

u_4	u_4	u_4	u_4	u_4	u_4	u_4	u_4
u_4	u_4	u_4	u_3	u_3	u_4	u_4	u_4
u_4	u_4	u_4	u_3	u_3	u_4	u_4	u_4
u_4	u_3	u_3	u_1	u_2	u_3	u_3	u_4
u_4	u_3	u_3	u_2	u_4	u_3	u_3	u_4
u_4	u_4	u_4	u_3	u_3	u_4	u_4	u_4
u_4	u_4	u_4	u_3	u_3	u_4	u_4	u_4
u_4	u_4	u_4	u_4	u_4	u_4	u_4	u_4

(b)

u_4	u_4	u_4	u_4	u_4	u_4	u_4	u_4
u_4	u_3	u_4	u_3	u_4	u_4	u_3	u_4
u_4	u_3	u_4	u_4	u_3	u_3	u_4	u_4
u_4	u_3	u_3	u_1	u_2	u_4	u_3	u_4
u_4	u_4	u_2	u_4	u_3	u_4	u_3	u_4
u_4	u_4	u_3	u_4	u_3	u_4	u_4	u_4
u_4	u_4	u_3	u_3	u_4	u_3	u_4	u_4
u_4	u_4	u_4	u_4	u_4	u_4	u_4	u_4

Figure 2. **Two different placements for a capacitor array. The unit capacitors denoted by u_3 are colored in gray for a clearer exhibition of different placement styles. (a) A common-centroid placement is good for reducing systematic mismatch. (b) The placement [10] exhibiting a higher degree of dispersion is better for reducing random mismatch.**

In the category of systematic mismatch, mechanisms have equal effects on each device. Thus, if two devices have identical layout size, they suffer the same percentage of difference from the mechanisms, which implies that the two devices still keep matched. Therefore, given a set of devices with different layout sizes, designers prefer to divide every device into several identical-layout-size sub-devices to achieve matching. However, the presence of process gradients also causes systematic mismatch [5]. If two devices have identical layout size but they are placed far away from each other in a layout, they would experience unequal magnitude of effects due to the process gradients and thus exhibit mismatch. Therefore, for a set of devices which require matching, they should be placed close to each other in a layout. Further, these devices should exhibit **symmetry** in the layout to average the effects induced by process gradients. In order to reduce systematic mismatch, designers usually adopt a common-centroid layout structure [1] to achieve these considerations.

On the other hand, random mismatch is caused by statistical fluctuations in processing conditions or material properties. Since these fluctuations are random mechanisms, the sub-devices of each device should be distributed throughout a layout as uniformly as possible to reduce random mismatch, which means that the sub-devices should exhibit the highest possible degree of **dispersion** in a layout [10].

Figures 2(a) and 2(b) show two placements for a capacitor array according to different layout considerations. The capacitor array consists of four capacitors, and each capacitor is divided into several identical-layout-size unit capacitors, which are denoted by u_i, $i = 1, \ldots, 4$. Figure 2(a) shows a placement with a common-centroid structure. In this placement, the unit capacitors are placed symmetrically, and the centroid of each capacitor is exactly on or close to the center of the layout. Compared with the symmetric placement shown in Figure 2(a), Figure 2(b) shows a placement that exhibits higher random distribution (note that it is constructed based on [10]). Since Figure 2(a) has **symmetry** property, it is good for reducing systematic mismatch [6, 7], but is not sufficient to reduce random mismatch due to a lower degree of dispersion. On the contrary, the placement in Figure 2(b) exhibits a higher degree of **dispersion**, which is better for reducing random mismatch [10]. However, it lacks symmetry property to overcome the mismatch caused by process gradients. **Therefore, in this paper, we would like to propose an algorithm to find a common-centroid capacitor placement, which possesses the property of high dispersion, to reduce systematic and random mismatches simultaneously.**

1.1 Previous Work

To achieve better matching, the placement of a capacitor array is usually implemented by a common-centroid structure. Several works [6, 11-14] have studied common-centroid placement. Sayed and Dessouky [6] introduced an oxide gradient model to estimate the oxide-gradient-induced mismatch. Based on this model, they presented a deterministic procedure to construct a common-centroid placement. Later, three works used topological representations to tackle the placement problem with common-centroid constraint, such as C-CBL [11], B*-trees [12], and sequence-pairs [13]. Recently, Lin et al. [14] proposed a thermal-driven common-centroid placement algorithm. However, all these works did not consider random mismatch, and thus their common-centroid placements lacked the property of high dispersion.

To consider random mismatch, Luo et al. [8, 9] introduced a spatial correlation model for yield evaluation. They showed that a placement with higher correlation coefficients would have better matching. Moreover, they proposed a heuristic algorithm to obtain a placement with the highest, or near-highest, correlation coefficients for yield improvement [10]. Although their placement results exhibits a higher degree of dispersion, their placements are not common-centroid structure. Therefore, none of existing works has presented a method to construct a common-centroid placement with the property of high dispersion.

1.2 Our Contributions

Although [1] highlights that a common-centroid layout should exhibit the property of dispersion, there exists limited works discussing how to disperse sub-devices uniformly in a common-centroid placement. Thus, in this paper, we propose an algorithm to construct a placement which has common-centroid structure and exhibits the property of high dispersion so that this placement can reduce systematic and random mismatches simultaneously.

Based on the simulated annealing [15], we propose a method to deal with the problem of common-centroid capacitor placement considering systematic and random mismatches. First, we propose the *pair-sequence* representation to represent a common-centroid placement. Since the sub-devices in a pair can be automatically placed to the symmetric locations in a layout, the corresponding placement can easily satisfy the common-centroid constraint. Further, we present three operations to perturb a pair sequence, which can increase the degree of dispersion without breaking the common-centroid constraint in the resulting placement. Finally, to make our simulated annealing based approach more efficient, we propose three techniques to speed up our program. The experimental results show that our placements can simultaneously achieve smaller oxide-gradient-induced mismatch and larger overall correlation coefficients (i.e., higher degree of dispersion) than [10] in all test cases. Besides, our program can run much faster than [10] in larger benchmarks. Since [10] uses partially exhaustive search approach, they even cannot obtain result in the largest benchmark.

The remainder of this paper is organized as follows. Section 2 introduces two models for mismatch estimation. Section 3 formulates the common-centroid placement problem for capacitor arrays. Section 4 describes the transformation between a pair sequence and its corresponding placement. Section 5 shows how to initialize a pair sequence for a capacitor set. Section 6 presents the operations to perturb a pair sequence and the techniques to speed up our program. Section 7 reports the experimental results. Finally, Section 8 concludes this paper.

2. BACKGROUND

The oxide gradient model [6] can be used to estimate the oxide-gradient-induced ratio mismatch of a capacitor array. We will employ the model to compare the process-gradient-induced mismatches of different placements. Besides, the spatial correlation model [8, 9] will be applied to measure the degree of dispersion for a placement. Due to the limit of space, we do not detail the two models. In the next section, we will formulate the capacitor placement problem based on the two models.

3. PROBLEM FORMULATION

Let $\mathcal{C} = \{C_1, C_2, \ldots, C_m\}$ denote a set of m capacitors, and each capacitor C_i consist of q_i unit capacitors ($1 \le i \le m$). The ratio of $C_1 : C_2 : \ldots : C_m$ is $q_1 : q_2 : \ldots : q_m$. Without loss of the generality, we assume $q_i \le q_j$ if $i < j$, where $1 \le i, j \le m$. The total number of unit capacitors is denoted by n (i.e., $n = q_1 + q_2 + \ldots + q_m$).

Given a capacitor array $\mathcal{C}$ and an $r{\times}s$ matrix $A_{r\times s}$, the capacitor placement problem is to assign each unit capacitor of $\mathcal{C}$ to a unique entry of the matrix, where $n = r{\times}s$. The objective is to minimize the **oxide-gradient-induced mismatch** and maximize the **overall correlation coefficients**, simultaneously.

4. PAIR-SEQUENCE REPRESENTATION

In this section, we first introduce a representation to represent the placement of elements in a matrix. Then, we show how to map each element in the representation to a unique entry in the matrix.

4.1 From a Matrix to its Pair Sequence

In this subsection, we first propose the *pair-sequence* $P_{r\times s}$ representation to denote the placement of n elements in an $r{\times}s$ matrix $A_{r\times s}$. The pair sequence $P_{r\times s} = [p_1, p_2, \ldots, p_m]$ is an array of m elements, and each element p_i in $P_{r\times s}$ denotes a pair of symmetric entries in $A_{r\times s}$ except the first element p_1, where $m = \lfloor(n+1)/2\rfloor$ and $n = r{\times}s$. If n is even, the first element p_1 still denotes a pair of entries in the matrix; however, it denotes only one entry in the matrix if n is odd. The subscript $r{\times}s$ of the representation indicates the dimension of the matrix mapped by the representation. Because the placement of elements may have various styles, it is necessary to distinguish those pair sequences having the same number of elements but mapping to matrices with different dimensions. For example, although the sizes of matrices $A_{2\times6}$, $A_{6\times2}$, $A_{3\times4}$, and $A_{4\times3}$ are identical, the dimensions of the four matrices are different. To reveal correct information, we append a subscript $r{\times}s$ to each pair sequence to denote the dimension of the corresponding matrix.

Now, we show how to encode the entries of a matrix $A_{r\times s}$ into the pair-sequence representation. Let a_{ij} denote an ith-row and jth-column entry in the matrix $A_{r\times s}$, where $1 \le i \le r$ and $1 \le j \le s$. The coordinate of a_{ij} is denoted by (x_i, y_j), where $x_i = i$ and $y_j = j$. Moreover, the matrix $A_{r\times s}$ has a unique center whose coordinate is denoted by (x_c, y_c), where $x_c = \lfloor(r+1)/2\rfloor$ and $y_c = \lfloor(s+1)/2\rfloor$. Let d denote the distance from a_{ij} to the center of $A_{r\times s}$, and it can be computed by the following equation:

$$d = \sqrt{(x_i - x_c)^2 + (y_j - y_c)^2} \qquad (1)$$

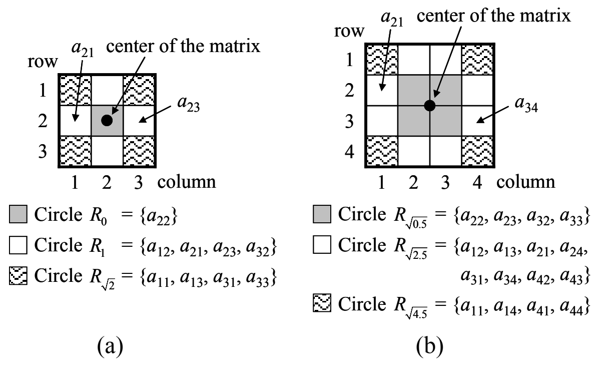
Those entries having the same distance d form a *circle*, which is denoted by R_d. We consider any two entries as a pair if they are in the same cycle and their locations are opposite with respect to the center of $A_{r\times s}$, which means the two entries are symmetric to each other. After all entries of the matrix are classified into different circles, the pairs of entries in a circle are picked out in serial counterclockwise from the twelve-o'clock direction. Then, we can construct a pair sequence by collecting these pairs from the inner circle to the outer circle. Note that the most inner circle of a matrix only contains one entry if the matrix comprises an odd number of entries. Therefore, the first pair in the corresponding pair sequence has only one element.

Figure 3(a) shows a matrix $A_{3\times3}$. According to the distance d of each entry, we can classify the entries into three circles, R_0, R_1, and $R_{\sqrt{2}}$. The circle R_0 consists of only one entry a_{22} because a_{22} is located at the center of the matrix (i.e., its distance d to the center is zero). The circles R_1 and $R_{\sqrt{2}}$ contain the entries $\{a_{12}, a_{21}, a_{23}, a_{32}\}$ and $\{a_{11}, a_{13}, a_{31}, a_{33}\}$ since the entry distances to the center are 1 and $\sqrt{2}$, respectively. Then, we start to pair all entries in each circle and then construct a pair sequence. Since the circle R_0 contains only one entry a_{22}, a_{22} is not paired. For the entries in circle R_1, we first select entry a_{12}, which is located at the twelve-o'clock direction, and then pick its opposite entry a_{32} in R_1. Since a_{12} and a_{32} are symmetric respect to the center of the matrix, we consider (a_{12}, a_{32}) as a pair. Following the counterclockwise order, we next pick entry a_{21} and find its opposite entry a_{23}, and consider (a_{21}, a_{23}) as another pair. Similarly, we pair the entries in circle $R_{\sqrt{2}}$ and then obtain the pairs (a_{11}, a_{33}) and (a_{31}, a_{13}). Note that the two entries of a pair have an order due to the definition of pairing procedure. After all entries are paired, a pair sequence $P_{3\times3}$ can be constructed with the ordered pairs from the inner circle to the outer circle. Thus, we derive the pair sequence $P_{3\times3} = [a_{22}, (a_{12}, a_{32}), (a_{21}, a_{23}), (a_{11}, a_{33}), (a_{31}, a_{13})]$. Figure 3(b) shows another matrix $A_{4\times4}$. Similarly, we first classify all entries into three circles, $R_{\sqrt{0.5}}$, $R_{\sqrt{2.5}}$, and $R_{\sqrt{4.5}}$, and then pair all entries in each circle. Thus, we derive the pair sequence $P_{4\times4} = [(a_{22}, a_{33}), (a_{32}, a_{23}), (a_{12}, a_{43}), (a_{21}, a_{34}), (a_{31}, a_{24}), (a_{42}, a_{13}), (a_{11}, a_{44}), (a_{41}, a_{14})]$.

4.2 From a Pair Sequence to its Matrix

We have introduced how to derive a pair sequence from a matrix in the previous subsection. Now, we show how to obtain the corresponding placement from a pair sequence once the materials are arranged to the pair sequence.

Given a pair sequence $P_{r\times s}$, we first construct an $r\times s$ matrix $A_{r\times s}$ according to the subscript of $P_{r\times s}$. Since each element in a pair corresponds to a unique location in the matrix, we only need to place the materials located in a pair to the corresponding entries in the matrix in serial. For example, given a pair sequence $P_{3\times3}$ = [a, (b, c), (d, e), (f, g), (h, i)], we first construct a 3×3 matrix $A_{3\times3}$. Since each element in a pair associates with a unique location in the matrix, we first place the material "a" in the first pair to the corresponding entry a_{22} in $A_{3\times3}$, as shown in Figure 4(a). Then, the materials "b" and "c" in the second pair are placed at the entries a_{12} and a_{32}, and so on. Similarly, Figure 4(b) shows the placement of $P_{4\times4}$ = [(a, b), (c, d), (e, f), (g, h), (i, j), (k, l), (m, n), (o, p)].



a_{21}
center of the matrix
row
1
2
3
a_{23}
1 2 3 column
▨ Circle R_0 = $\{a_{22}\}$
☐ Circle R_1 = $\{a_{12}, a_{21}, a_{23}, a_{32}\}$
▨ Circle $R_{\sqrt{2}}$ = $\{a_{11}, a_{13}, a_{31}, a_{33}\}$
(a)
a_{21}
center of the matrix
row
1
2
3
4
a_{34}
1 2 3 4 column
▨ Circle $R_{\sqrt{0.5}}$ = $\{a_{22}, a_{23}, a_{32}, a_{33}\}$
☐ Circle $R_{\sqrt{2.5}}$ = $\{a_{12}, a_{13}, a_{21}, a_{24}, a_{31}, a_{34}, a_{42}, a_{43}\}$
▨ Circle $R_{\sqrt{4.5}}$ = $\{a_{11}, a_{14}, a_{41}, a_{44}\}$
(b)

Figure 3. (a) The circles of matrix $A_{3\times3}$, and (a_{21}, a_{23}) is a pair. (b) The circles of matrix $A_{4\times4}$, and (a_{21}, a_{34}) is a pair.

f	b	i
d	a	e
h	c	g

(a)

m	e	l	p
g	a	d	j
i	c	b	h
o	k	f	n

(b)

Figure 4. (a) Placement result of $P_{3\times3}$ = [a, (b, c), (d, e), (f, g), (h, i)]. (b) Placement result of $P_{4\times4}$ = [(a, b), (c, d), (e, f), (g, h), (i, j), (k, l), (m, n), (o, p)].

5. PLACEMENT INITIALIZATION

In this section, we would like to show how to obtain an initial matrix placement for a capacitor set, in which all capacitors are arranged in a common-centroid structure. In the first subsection, the procedure for pairing the unit capacitors in the capacitor set is given. After all unit capacitors are paired, we show how to arrange these unit-capacitor pairs to obtain a pair sequence, and thus a matrix placement can be derived. Finally, an algorithm for the whole procedure of pair-sequence initialization is given in the last subsection.

5.1 Unit-Capacitor Pairing

Unit-capacitor pairing is iteratively to pair two unit capacitors in a capacitor set until each unit capacitor belongs to one pair. The resulting pairs can be arranged to obtain a pair sequence. For each pair in the pair sequence, the associated two unit capacitors will be placed in symmetric locations with respect to the center of the matrix according to the definition in the last section. Therefore, any two unit capacitors belonging to the same capacitor is first paired. However, if a capacitor comprises an odd number of unit capacitors, one unit capacitor will be left. Thus, for each capacitor comprising an odd number of unit capacitors, we first pick out one unit capacitor, and the remaining unit capacitors can be paired.

Given a capacitor set $\mathcal{C} = \{C_1, C_2, \ldots, C_m\}$, we assume each capacitor C_i contains q_i unit capacitors. Let $C_i = \{u_i\}$ denote the set of unit capacitors belonging to C_i, where u_i denotes one unit capacitor in C_i and the size of C_i is q_i. Let k denote the number of capacitors, where each capacitor has only one unit capacitor, and these capacitors form a set, which is denoted by $\mathcal{C}_{unit}$. Let l denote the number of capacitors, where each capacitor comprises an odd number of unit capacitors and the number is greater than two (i.e., 3, 5, …), and the set of these capacitors is denoted by $\mathcal{C}_{odd}$. Therefore, the total number of capacitors that each capacitor comprises an odd number of unit capacitors is $k+l$. For each of these capacitors, if we pick one unit capacitor from it, the number of its remaining unit capacitors becomes even and they can be paired. Then, **for the picked unit capacitors (i.e., there exists *k*+*l* unit capacitors)**, we can pair them according to the following cases:

- **Case 1. *k* and *l* are both odd:** we pick one unit capacitor from *k* unit capacitors and from *l* unit capacitors, respectively, and consider the two selected unit capacitors as a pair. Thus, the remaining unit capacitors can form (k-1)/2 and (l-1)/2 pairs.
- **Case 2. *k* is odd and *l* is even:** we select one unit capacitor from *k* unit capacitors and treat it as a **single-unit pair** (i.e., a pair only containing one unit capacitor). Thus, the remaining capacitors would form (k-1)/2 and l/2 pairs.
- **Case 3. *k* is even and *l* is odd:** if $k \neq 0$, we pick one unit capacitor from *k* unit capacitors and consider it as a single-unit pair, and thus the remaining unit capacitors can be handled by Case 1; if $k = 0$, we select one unit capacitor from *l* unit capacitors and treat it as a single-unit pair, and the remaining unit capacitors can form (l-1)/2 pairs.
- **Case 4. *k* and *l* are both even:** the *k* unit capacitors form k/2 pairs, and the *l* unit capacitors form l/2 pairs.

After the *k*+*l* unit capacitors are paired, we can classify them into four types in the following:

- $S_{(unique,\,)}$**:** it consists of pairs that each pair has only one unit capacitor.
- $S_{(unit,\,unit)}$**:** the two unit capacitors come from different capacitors in $\mathcal{C}_{unit}$.
- $S_{(unit,\,odd)}$**:** one of the two unit capacitors belongs to a capacitor in $\mathcal{C}_{unit}$, and the other belongs to a capacitor in $\mathcal{C}_{odd}$.
- $S_{(odd,\,odd)}$**:** the two unit capacitors belong to different capacitors in $\mathcal{C}_{odd}$.

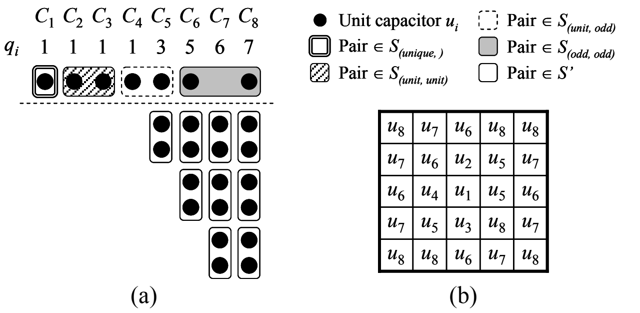
After above procedure, the number of the remaining unit capacitors in each capacitor becomes even, and thus we can pair them completely. These unit-capacitor pairs are classified into the following type:

- ***S'*:** the two unit capacitors in a pair belong to the same capacitor (comparing to the sets listed in the above).

Figure 5(a) shows a capacitor set $\{C_1, C_2, C_3, C_4, C_5, C_6, C_7, C_8\}$, and the ratio of these capacitors is 1 : 1 : 1 : 1 : 3 : 5 : 6 : 7. According to above definition, $\mathcal{C}_{unit} = \{C_1, C_2, C_3, C_4\}$ and $\mathcal{C}_{odd} = \{C_5, C_6, C_8\}$. We first pick one unit capacitor from each of these capacitors and try to pair them (i.e., u_1, u_2, u_3, u_4, u_5, u_6, and u_8 are selected). Since $k = 4$ and $l = 3$, it satisfies Case 3. Assume u_1 is selected from the first four unit capacitors u_1, u_2, u_3, and u_4, and it is considered as a single-unit pair (i.e., $(u_1,) \in S_{(unique,\,)}$). Then, the remaining three unit capacitors u_2, u_3, and u_4 and the last three unit capacitors u_5, u_6, and u_8 are handled according to Case 1. Thus, from each of the two groups, we select one unit capacitor from it, and consider the two selected unit capacitors as a pair (assume $(u_4, u_5) \in S_{(unit,\,odd)}$ is selected). Finally, the remaining capacitors u_2, u_3, u_6, and u_8 are paired, which form $(u_2, u_3) \in S_{(unit,\,unit)}$ and $(u_6, u_8) \in S_{(odd,\,odd)}$. After the above process, the remaining unit capacitors in each capacitor are paired, respectively, and they are classified into the type *S'*.

5.2 Pair Arrangement

After all unit capacitors in a capacitor set are paired, we can arrange these pairs to obtain a pair sequence, and then derive an



C_1 C_2 C_3 C_4 C_5 C_6 C_7 C_8
q_i 1 1 1 1 3 5 6 7
● Unit capacitor u_i
Pair $\in S_{(unique,\,)}$
Pair $\in S_{(unit,\,unit)}$
Pair $\in S_{(unit,\,odd)}$
Pair $\in S_{(odd,\,odd)}$
Pair $\in S'$
u_8 u_7 u_6 u_8 u_8
u_7 u_6 u_2 u_5 u_7
u_6 u_4 u_1 u_5 u_6
u_7 u_5 u_3 u_8 u_7
u_8 u_8 u_6 u_7 u_8
(a)
(b)

Figure 5. (a) An example of unit-capacitor pairing. (b) The corresponding placement in a matrix $A_{5\times5}$.

Algorithm 1: initializePairSequence(Capacitor array, Matrix size)

```
   /* All unit capacitors are paired and classified into five pair types; */
1: Unit-capacitor pairing for the capacitor array;
2: Allocate an empty pair sequence;
3: for highest-priority pair type to lowest-priority pair type do
4:     for all unit-capacitor pairs belonging to the pair type do
5:         Add the pair that has a unit capacitor $u_i$ with the smallest index $i$
           to the end of the pair sequence;
6:     end for
7: end for
8: Append the matrix size to the pair sequence;
```

initial matrix placement from it. To obtain a better matrix placement, we have to determine a sequence to place these pairs into a pair sequence.

Before we give our procedure, we first analyze the property of different kind of capacitors. If a capacitor belongs to $\mathcal{C}_{unit}$, it is better to be placed near to the center of a matrix such that its centroid can be close to the matrix's center. If a capacitor belongs to $\mathcal{C}_{odd}$, one of its unit capacitors would form a single unit capacitor and others form several unit-capacitor pairs. Since each unit-capacitor pair can be automatically placed to the symmetric locations in the matrix, the capacitor's centroid can be located close to the center of the matrix if the single unit capacitor is also placed near to the matrix's center. Finally, if a capacitor comprises an even number of unit capacitors, its unit capacitors can be paired completely. Therefore, its centroid must be exactly at the center of the matrix.

Through the above analysis, some unit capacitors should be placed closer to the matrix's center than others in order to get a common-centroid placement. If a pair can be arranged near to the head of a pair sequence, the placement of the associated unit capacitors would be close to the matrix's center; otherwise, its placement is far from the matrix's center. Therefore, we give the priorities of different pair types in the following:

- $S_{(unique,\,)} > S_{(unit,\,unit)} > S_{(unit,\,odd)} > S_{(odd,\,odd)} > S'$

$S_{(unique,\,)} > S_{(unit,\,unit)}$ means $S_{(unique,\,)}$ has a higher priority than $S_{(unit,\,unit)}$, and so on. A pair belonging to a pair type with higher priority will be arranged more near to the head of a pair sequence.

5.3 Algorithm for Pair-Sequence Initialization

In this subsection, we give the complete procedure for initializing a pair sequence (please see Algorithm 1). We first pair unit capacitors and divide them into different types (see Line 1). Then, we serially place all pairs into a pair sequence according to the priorities of the pair types. Since the pairs with the same pair type have identical priority, we arrange them into a pair sequence according to a deterministic order (see Lines 3-7).

Given the capacitor array shown in Figure 5(a) and a matrix $A_{5\times5}$ for placement, Algorithm 1 will generate an initial pair sequence $P_{5\times5} = [u_1, (u_2, u_3), (u_4, u_5), (u_6, u_8), (u_5, u_5), (u_6, u_6), (u_6, u_6), (u_7, u_7), (u_7, u_7), (u_7, u_7), (u_8, u_8), (u_8, u_8), (u_8, u_8)]$. According to the pair sequence, we can derive an initial matrix placement near a common-centroid structure, as shown in Figure 5(b).

6. THE PLACEMENT ALGORITHM

After an initial matrix placement has been generated, we can further apply the simulated annealing (**SA**) [15] to obtain better results. The SA algorithm repeatedly perturbs the pair sequence until a predefined termination condition is satisfied. Its objective function is to minimize oxide-gradient-induced mismatch and maximize overall correlation coefficients simultaneously. In the following subsections, we first introduce the operations to perturb our representation, and then show our objective function. Finally, several techniques used to speed up our program are given in the last subsection.

6.1 Perturbation

The property of symmetry in a pair sequence should be maintained during perturbation. To avoid breaking this property, we propose three operations in the following:

- **Op1:** it chooses one pair p_i which belongs to $S_{(unit,\ odd)}$ or $S_{(odd,\ odd)}$, and then reverse the order of its unit capacitors.
- **Op2:** it chooses two pairs p_i and p_j from any pair types except S', and then exchange one unit capacitor u_k in p_i with another unit capacitor u_l in p_j.
- **Op3:** it chooses two pairs p_i and p_j from any pair types except $S_{(unique,\ )}$, and then exchange the order of p_i and p_j in the pair sequence.

Since the contents of two pairs are exchanged in Op2, the types of the two pairs may be changed after applying the operation.

6.2 Objective Function

There exist two major objectives in our program: one is to minimum oxide-gradient-induced mismatch, and the other is maximum overall correlation coefficients. Let M denote the value of oxide-gradient-induced mismatch, and L denote the value of overall correlation coefficients. During SA, we will maintain the average values for oxide-gradient-induced mismatch and overall correlation coefficients, which are denoted by M_{avg} and L_{avg}, respectively. Thus, an objective function Φ is given as follows:

$$\Phi = \alpha \times \frac{M_{avg} - M}{M_{avg}} + (1-\alpha) \times \frac{L - L_{avg}}{L_{avg}} \qquad (2)$$

where α is a user-specified parameter, $0 \le \alpha \le 1$. The goal of our algorithm is to find a placement with **maximum Φ**.

6.3 Speedup Techniques in Our Program

Since the SA is a time consuming process, we also propose several techniques to speed up our program in the following:

- **Bucket data structure:** to facilitate the operations, we use the bucket list data structure to record necessary information. Figure 6 shows the data structure used in our algorithm. There exists a bucket array H_B, where each entry represents a pair type, and another array H_P, which stores a pair sequence. For each pair associated with a pair type, it is connected to the entry of the type in H_B by a doubly-linked list. Besides, there also exists a point referring to the pair from the pair sequence H_B. **With this data structure, we can easily extract a pair with a specified type during perturbation.** Once the type of a pair is changed after perturbation, we also have to modify its information in the corresponding doubly-linked lists.

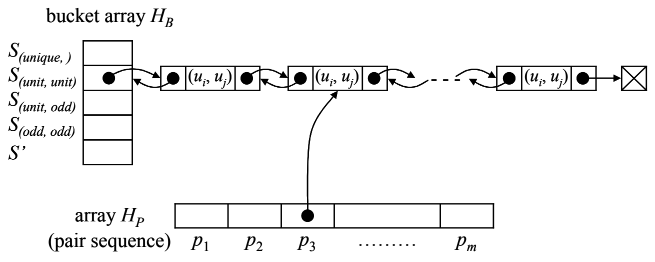


Figure 6. The bucket data structure used in our algorithm.

- **Redundancy elimination:** our program neglects redundant operations during the SA. For example, for two capacitors belonging to $\mathcal{C}_{unit}$, the resulting placement will be geometrically identical to the previous one if their positions are exchanged in a placement. Therefore, it is considered as a redundant operation.
- **Operation combination:** we can enhance the convergence of the SA by applying an operation that can lead to a large change. Hence, we combine Op1 and Op3 to get a new perturbation. The operation is to choose two pairs p_i and p_j which belong to any pair types except $S_{(unique,\ )}$, and then perform one of the following cases:
  - p_i and p_j execute Op3; then p_i executes Op1.
  - p_i and p_j execute Op3; then p_j executes Op1.
  - p_i and p_j execute Op3; then both p_i and p_j execute Op1.

  Note such kind of operations might deteriorate the solution quality while a solution is going to converge. Therefore, we give a higher probability to apply these operations at high temperature, but much lower probability at low temperature.

7. EXPERIMENTAL RESULTS

Our capacitor placement algorithm was implemented in C++ and run on a 2.5 GHz Intel Core2 Quad PC. We performed experiments on six benchmarks. The first three capacitor arrays, which are named SCF_1, SCF_2, and SCF_3, are sourced from [6] and [10]. The others are based on the capacitor arrays used in real SAR ADCs designed by us. Since their resolutions are 8-bit, 9-bit, and 10-bit, respectively, we name them SAR_8bit, SAR_9bit, and SAR_10bit, respectively. For fair comparisons, the experiment setups are the same as those in [6] and [10]. The oxide thickness is 40 nm, oxide gradient is 10 ppm, the correlation coefficient of the unit capacitor is 0.9, and the geometry of unit capacitors is depicted in Figure 7(a).

In order to compare our method with related works, we implemented the heuristic algorithm [10], and the results are shown in Table 1. Columns 2, 3, and 4 in Table 1 show the information of a capacitor set, which includes number of capacitors, capacitance ratio, and total number of unit capacitors, respectively. Column 5 shows the dimension of a matrix for placement. For each approach, we show the value of oxide-gradient-induced mismatch (denoted by M), the value of overall correlation coefficients (denoted by L), and the running time, respectively. The experimental results show that our placement results can simultaneously achieve smaller oxide-gradient-induced mismatch and larger overall correlation coefficients than [10] in all cases. Note [10] cannot obtain result in the largest benchmark SAR_10bit. Although our approach is slower than [10]

Table 1. Comparisons of oxide-gradient-induced mismatches, overall correlation coefficients, and running time for the heuristic algorithm [10] and our work. (*M*: oxide-gradient-induced mismatch; *L*: overall correlation coefficients)

Array Name	# of Cap.	Capacitance Ratio	# of Unit Cap.	Matrix Size	Heuristic Algorithm [10]			Our Work		
					Max. *M* (%)	*L*	Time (s)	Max. *M* (%)	*L*	Time (s)
SCF_1	5	2 : 6 : 7 : 7 : 8	30	6 × 5	0.138	9.651	1	0.109	9.688	3
SCF_2	5	1 : 2 : 2 : 10 : 17	32	8 × 4	0.679	9.318	1	0.601	9.343	4
SCF_3	4	1 : 2 : 16 : 45	64	8 × 8	0.650	5.567	2*	0.546	5.571	8
SAR_8bit	9	1 : 1 : 2 : 4 : 8 : 16 : 32 : 64 : 128	256	16 × 16	0.800	32.074	602	0.720	32.089	188
SAR_9bit	10	1 : 1 : 2 : 4 : 8 : 16 : 32 : 64 : 128 : 256	512	32 × 16	1.077	38.072	20503	0.979	38.306	591
SAR_10bit	11	1 : 1 : 2 : 4 : 8 : 16 : 32 : 64 : 128 : 256 : 512	1024	32 × 32	-	-	-**	1.155	45.400	3706

*This running time reported in [10] is 46.9 s, but the same algorithm implemented by us takes 2 s. It may be caused from the difference in programming and running platform.
**No result is reported because the running time is too long.

in the smaller benchmarks, we can run much fast than [10] in the larger benchmarks. Since [10] partially exhaustively searches possible combinations and computes the correlation coefficients for each combination to obtain the best one, their computational time increases significantly while the number of the available entries increases. On the contrary, we use SA to enhance our results and propose several techniques to speed up SA. That's why we can get better results and run much faster than [10].

Figure 7(b) shows our placement result for SCF_3. Compared with the placement result shown in Figure 2(b) (note that it is constructed based on [10]), our placement has comparable overall correlation coefficients (i.e., similarly high degree of dispersion), but the symmetry property is better (i.e., common-centroid structure). Figure 8 compares three different placements for SCF_3. As shown in the figure, since the placement based on [10] is not common-centroid structure, it has largest oxide-gradient-induced mismatch. Although both placements in Figure 2(a) and ours are common-centroid structure, we still get smaller oxide-gradient-induced mismatch because the common-centroid placement is enhanced by our SA based approach.

8. CONCLUSIONS

We have presented an SA based approach to implement a common-centroid placement with the property of high dispersion in order to reduce systematic and random mismatches. Besides, we have proposed a pair-sequence representation to represent a common-centroid placement, and presented the associated perturbations to increase the degree of dispersion without breaking the common-centroid constraint in the resulting placement. The experimental results have shown that our common-centroid placement approach is effective to reduce capacitor mismatch.

9. ACKNOWLEDGEMENTS

This work was supported in part by National Science Council of Taiwan under Grant No's NSC-99-2220-E-006-019 and NSC-98-2221-E-006-156-MY3.

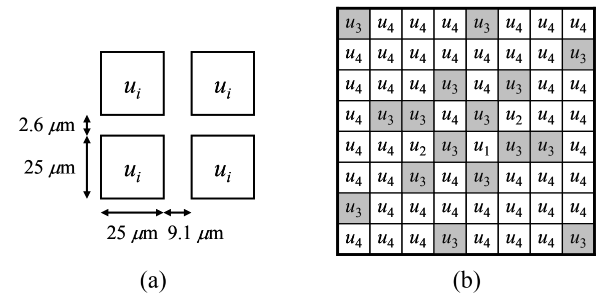
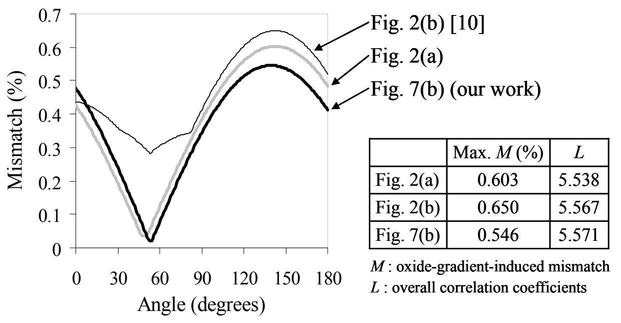

u_i
u_i
u_i
u_i
2.6 μm
25 μm
25 μm 9.1 μm
(a)
u_3 u_4 u_4 u_4 u_3 u_4 u_4 u_4
u_4 u_4 u_4 u_4 u_4 u_4 u_4 u_3
u_4 u_4 u_4 u_3 u_4 u_3 u_4 u_4
u_4 u_3 u_3 u_4 u_3 u_2 u_4 u_4
u_4 u_4 u_2 u_3 u_1 u_3 u_3 u_4
u_4 u_4 u_3 u_4 u_3 u_4 u_4 u_4
u_3 u_4 u_4 u_4 u_4 u_4 u_4 u_4
u_4 u_4 u_4 u_3 u_4 u_4 u_4 u_3
(b)

Figure 7. (a) Experiment setup for the geometry of unit capacitors. (b) Our placement result for SCF_3.



	Max. *M* (%)	*L*
Fig. 2(a)	0.603	5.538
Fig. 2(b)	0.650	5.567
Fig. 7(b)	0.546	5.571

Figure 8. The oxide-gradient-induced mismatches and overall correlation coefficients of three different placements for SCF_3.

10. REFERENCES

[1] A. Hastings, *The Art of Analog Layout*, 2nd Ed., Prentice Hall, 2006.
[2] D. A. Johns and K. Martin, *Analog Integrated Circuit Design*, Wiley, 1997.
[3] M. J. McNutt, S. LeMarquis, and J. L. Dunkley, "Systematic capacitance matching errors and corrective layout procedures," *IEEE JSSC*, vol. 29, no. 5, pp. 611-616, May. 1994.
[4] J.-B. Shyu, G. C. Temes, and F. Krummenacher, "Random error effects in matched MOS capacitors and current sources," *IEEE JSSC*, vol. 19, no. 6, pp. 948-956, Dec. 1984.
[5] E. Felt, A. Narayan, and A. Sangiovanni-Vincentelli, "Measurement and modeling of MOS transistor current mismatch in analog IC's," *Proc. ICCAD*, 1994, pp. 272-277.
[6] D. Sayed and M. Dessouky, "Automatic generation of common-centroid capacitor arrays with arbitrary capacitor ratio," *Proc. DATE*, 2002, pp. 576-580.
[7] D. Khalil, M. Dessouky, V. Bourguet, M.-M. Louerat, A. Cathelin, and H. Ragai, "Evaluation of capacitor ratios in automated accurate common-centroid capacitor arrays," *Proc. ISQED*, 2005, pp. 143-147.
[8] P.-W. Luo, J.-E Chen, C.-L. Wey, L.-C. Cheng, J.-J. Chen, and W.-C. Wu, "Impact of capacitance correlation on yield enhancement of mixed-signal/analog integrated circuits," *IEEE TCAD*, vol. 27, no. 11, pp. 2097-2101, Nov. 2008.
[9] J.-E Chen, P.-W. Luo, and C.-L. Wey, "Yield evaluation of analog placement with arbitrary capacitor ratio," *Proc. ISQED*, 2009, pp. 179-184.
[10] J.-E Chen, P.-W. Luo, and C.-L. Wey, "Placement optimization for yield improvement of switched-capacitor analog integrated circuits," *IEEE TCAD*, vol. 29, no. 2, pp. 313-318, Feb. 2010.
[11] Q. Ma, E. F. Y. Young, and K. P. Pun, "Analog placement with common centroid constraints," *Proc. ICCAD*, 2007, pp. 579-585.
[12] M. Strasser, M. Eick, H. Graeb, U. Schlichtmann, and F. M. Johannes, "Deterministic analog circuit placement using hierarchically bounded enumeration and enhanced shape functions," *Proc. ICCAD*, 2008, pp. 306-313.
[13] L. Xiao and E. F. Y. Young, "Analog placement with common centroid and 1-D symmetry sonstraints," *Proc. ASPDAC*, 2009, pp. 353-360.
[14] P.-H. Lin, H. Zhang, M. D. F. Wong, and Y.-W. Chang, "Thermal-driven analog placement considering device matching," *Proc. DAC*, 2009, pp. 593-598.
[15] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, no. 4598, pp. 671-680, May 1983.