Analog Transistor Placement Optimization Considering Nonlinear Spatial Variations

Supriyo Maji[1], Sungyoung Lee[2], David Z. Pan[1]
[1]*ECE Department, The University of Texas at Austin, Austin, TX, USA*
[2]*EECS Department, Seoul National University, Seoul, South Korea*
supriyo.maji@austin, brianlsy@snu.ac.kr, dpan@ece.utexas.edu

***Abstract*— Analog circuit performance can degrade due to random and spatial variations. While random variations can be mitigated using larger-sized devices, such devices tend to have more spatial variations. To address this, a common technique involves employing symmetric layout like the common-centroid, which effectively reduces linear variations or first-order effect. However, achieving high performance in analog systems often necessitates mitigating nonlinear spatial variations, for which common-centroid layout is unsuitable. In response, this work introduces an efficient approach based on simulated annealing for transistor placement, with a particular focus on mitigating nonlinear spatial variations. Importantly, our proposed method can also handle important layout constraints, including routing complexity, layout-dependent effects, and diffusion-sharing within the optimization. Experimental results show the proposed method beats state-of-the-art in all important parameters while minimizing nonlinear spatial variations. Moreover, our approach gives users better control over optimization objectives than existing methods.**

I. INTRODUCTION

The performance of various analog circuits depends significantly on the precise matching of elements [1]. Matching of elements can be impacted by two types of variations: random and spatial. Random variations can be reduced by increasing the area of the device [2][3]. However, larger device may have more spatial variations as the distance between the devices increases [3]. Several factors, including process variations, thermal distribution or uneven mechanical stress from other circuit layers, can contribute to spatial variations [3]. The spatial variations can be modeled by Taylor series, where the lower order terms have greater impacts on the mismatch among devices [3][4][5][6]. Common-centroid (CC) is a special layout technique which has been shown to be beneficial in reducing first-order variations [6][7][8][9]. However, for high performance systems, layout must be optimized considering nonlinear or higher order effects [3][4][5][6][10][11].

Implementing any type of symmetricity in the layout can bring various challenges. Routing becomes more complicated with CC type layout, circuit area can increase if diffusion region is not shared, layout dependent effect can affect device threshold voltage, mobility and circuit performance [7][12][13]. Sharma et. al. [14][7] have proposed custom algorithm for symmetric transistor placement, which handles layout constraints. However, this method is limited to producing layouts of the CC type, making it inefficient in handling nonlinear variations. The method also lacks a well-defined objective in the problem formulation, hindering the ability to prioritize one constraint over another. Furthermore, the layout constraints are handled separately through post-processing steps after placement, which can lead to sub-optimal results.

Several prior studies have considered nonlinear spatial variations in device placement optimization. Lin et. al. [4] perform global placement of capacitors by adopting a conjugate gradient method and then legalizing the placement with an integer linear programming formulation. McAndrew [6] cancels the effect of nonlinear variations by swapping symmetrical devices. However, this approach can only manage two devices, and a dimension constraint requires the number of unit cells in each column to be a multiple of four. Vadipour [10] has proposed a ring-like structure to compensate for quadratic error. However, the techniques reported in [3][4][10][11] may not apply to general transistor circuits, and some of the approaches [4][5][6][10] dealing with device placement, focus on spatial variations and do not consider important layout constraints.

We present a simulated annealing based general transistor placement algorithm which can optimize nonlinear spatial variations while handling important layout constraints. Simulated annealing, a popular optimization technique [15], has found applications in both digital [16][17] and analog placement [3][18][19]. The general idea behind these approaches revolves around the random selection of sub-blocks, followed by swapping to enhance performance. However, these approaches have limitations – they are inherently customized to specific systems and are not well-suited for addressing the general transistor placement problem. This is particularly evident when considering the absence of critical layout constraints within their formulations.

We adopt a strategy similar to random selection and swap, yet our approach is more general. It can handle transistor placement of varied configurations, such as current mirror, differential input pair, load pair, so

979-8-3503-4859-0/DATE24/© 2024 EDAA

can find application across a wide spectrum of analog systems. Our approach starts with an initial placement of unit cells, subsequently fine-tuning it through a dynamic process involving random selection and swapping. At each iteration, the placement is evaluated with models, developed for this work, guiding the process. The major contributions are as follows.

- We propose a simulated annealing based general approach for analog transistor placement considering nonlinear spatial variations.
- Compared to the state of the art [14][7], where layout constraints are handled separately through post-processing steps, our approach considers the constraints within the optimization.
- Experimental results show the proposed approach beats the state-of-the-art in important parameters while minimizing nonlinear spatial variations.
- The proposed approach provides users with better control over the objectives than the state-of-the-art.

The rest of the paper is organized as follows. Section II formulates the problem, Section III describes the proposed approach, Section IV presents experimental results, and Section V concludes the paper.

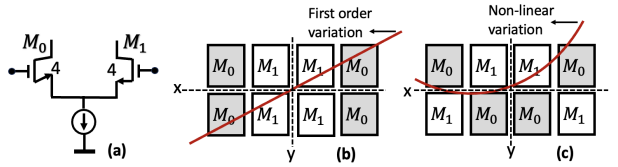

Fig. 1: (a) Differential input pair. (b) A layout considering only first-order effect. (c) A better layout considering nonlinear effect.

II. Transistor Placement Problem Formulation

Spatial variations can cause mismatch issues in analog circuits. Spatial variations have linear and nonlinear components. In Fig. 1(b)-(c), two different layouts of a differential pair (Fig. 1(a)) are shown. While the layout in Fig. 1(b) is effective when only considering the first-order effect, it is not the best layout for minimizing nonlinear variations. To reduce nonlinear variations, as a general rule, devices must be placed with the highest possible dispersion [6], as shown in Fig. 1(c). It is important to note that both these layouts are CC type. While it is possible to generate a CC layout that performs well for nonlinear variations involving a few units and devices, in general, as the number of units or devices increases, a non-CC layout consistently outperforms a CC layout. However, a layout that is designed to perform well for spatial variations may not align with critical layout constraints such as diffusion break, routing complexity, and layout-dependent effects. Therefore, our transistor placement formulation problem encompasses all these constraints to achieve a comprehensive and balanced optimization approach.

Based on the mismatch in spatial variations function $MV(\cdot)$, routing complexity function $RC(\cdot)$, mismatch in length of diffusion function $MILD(\cdot)$, and diffusion break function $DB(\cdot)$, the optimization objective is defined as follows.

$$\min\ \eta_{MV}MV(f_p) + \eta_{RC}RC(f_p) + \eta_{MILD}MILD(f_p) \quad (1)$$

$$\text{such that, } DB(f_p) == 0 \quad (2)$$

Here, f_p is a feasible layout solution and η_c, $c \in \{MV, RC, MILD\}$ denotes the coefficients. A hard constraint is put on diffusion sharing as this parameter has the most impact on the quality of the layout, directly or indirectly. A layout that does not share diffusion region can lead to worse routing length and may degrade spatial variations due to increased layout area. Consequently, our proposed method can ensure diffusion-break-free layout which is not possible with other methods [14][20][8]. Importantly, [14] is more focused on generating CC layout, constraints are handled separately through post-processing steps. We handle layout constraints directly within the optimization framework, this can lead to better results. Moreover, our formulation allows users control over the objectives through coefficients η_c, which is not possible by other methods due to a fixed-type formulation that lacks a well-defined objective.

III. The Proposed Approach to Transistor Placement

We first discuss the details on the funtions in Eqs. (1) and (2) before presenting the optimization algorithm in Section III-E to solve them.

A. Mismatch in Spatial Variations

Thermal-induced and technology-related spatial variations can be approximated using a Taylor series, which takes the following form [3][4][5][6].

$$p_n(x,y) = \sum_{i=1}^{\infty} G_i(x,y) + C \quad (3)$$

$$G_i(x,y) = \sum_{j=0}^{i} g_{j,i-j}x^j y^{i-j} \quad (4)$$

Here, $(x,\ y)$ is the location of a unit cell and C is a constant independent of x and y. $G_i(x,y)$ is the i^{th} order variations component and $p_n(x,y)$ is the sum of such components. Our objective is to minimize the difference (or mismatch) in spatial variations for all the devices. We propose the following model.

$$\frac{1}{N}\sum_{l=1}^{N-1}\sum_{m=l+1}^{N}\left(\frac{\sum_{u=1}^{n_l} p_u(x,y)}{n_l} - \frac{\sum_{u=1}^{n_m} p_u(x,y)}{n_m}\right) \quad (5)$$

$$p_u(x,y) = g_{1,0}x + g_{0,1}y + g_{2,0}x^2 + g_{1,1}xy + g_{0,2}y^2 \quad (6)$$

Here, N is the number of devices and n_l, n_m is the number of units for the device l and m, respectively. $p_u(x,y)$ is the sum of first and second-order spatial

variations with coefficients $g_{1,0}, g_{0,1}, g_{2,0}, g_{1,1}, g_{0,2}$ being modeled as multivariate Gaussian random variables. By iterating mismatch calculation following Eq. (5), M times, M number of mismatch values are obtained and the standard deviation of these values represents the magnitude of difference in spatial variations over all devices, referred to as $MV(\cdot)$.

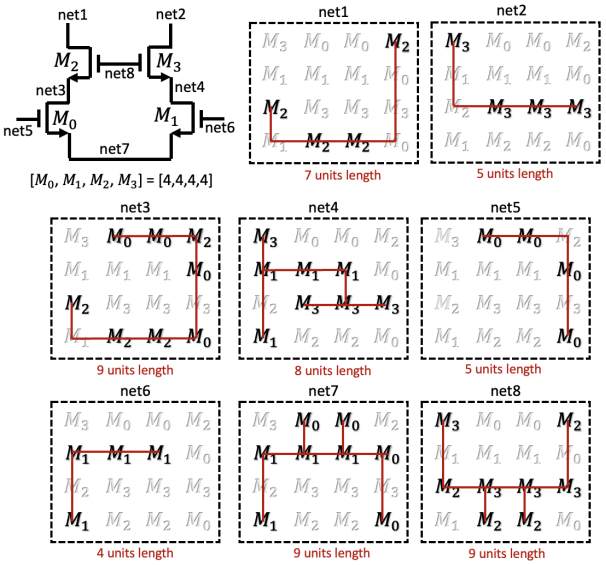

Fig. 2: RMST for calculating routing complexity of a differential input pair cascode (DIPC).

B. Routing Complexity

The routing complexity is affected by the position of the unit cells in the layout, making it essential to take routing considerations into account during the placement optimization. Sharma et al. [14] address routing complexity, but the approach has a limitation in that it handles routing after placement step, which can lead to sub-optimal results. In our approach, routing model is integrated within the placement step.

We use rectilinear minimum spanning tree (RMST) to calculate routing complexity separately for every net in the netlist and sum the values. For illustration purpose, we use the circuit in Fig. 2, where each device has 4 units. First, we construct an undirected graph in which a unit cell connected to the net is represented by a node, and rectilinear edges are used to connect every pair of nodes on the net. For example, net1 has unit cells from $M2$, so a total of 4 nodes and 6 edges, net3 has unit cells from $M2$ and $M0$, so a total of 8 nodes and 28 edges. We determine the RMST that connects all the units on the net with the fewest possible edges. For example, the routing complexity of net1 and net3 are 7 and 9 units, respectively. Total routing complexity is determined by summing the edge weights of the spanning trees for all the nets, which in this case is 56. A pseudocode is presented below. The time complexity of our algorithm is $O(n^2)$ with n being the number of units.

Algorithm 1 Routing Complexity Model

```
getRoutingComplexity(pattern, netlist)
route_complexity = 0
for each net in netlist do
    for each unit connected to net do
        G.node ← unit
    for every pair of node (u,v) in G do
        G.edge ← (u,v)
        find location (loc) of u and v in pattern
        G.weight ← |loc.u.x - loc.v.x| + |loc.u.y - loc.v.y|
    find G_RMST in G
    route_complexity = route_complexity + sum(G_RMST.weight)
return route_complexity
end getRoutingComplexity
```

C. Mismatch in Length of Diffusion

Among several layout dependent effects (LDEs), the most critical is length of diffusion (LOD), which captures the variations in the threshold voltage (V_{th}) of a transistor due to stress effect [12][13][14].

$$\Delta V_{th} \propto \frac{1}{LOD} = \sum_{i=1}^{n} \left(\frac{1}{SA_i + 0.5L_g} + \frac{1}{SB_i + 0.5L_g}\right) \quad (7)$$

Here, n is the number of unit cells, L_g is the gate length of the unit. SA_i and SB_i are the distances from poly-gate of the unit cell i to the diffusion/active edge on either side of the layout. Two unit cells with different SA_i, SB_i values will lead to the mismatch in threshold voltage shift (ΔV_{th}). To reduce variations induced by LOD, we aim to minimize the difference (i.e. mismatch) of the mean values of $\frac{1}{LOD}$ over all devices, referred to as $MILD(\cdot)$. We propose the following model for $MILD(\cdot)$.

$$MILD(\cdot) = \sum_{k=1}^{N-1} \sum_{l=k+1}^{N} \left| \frac{(\frac{1}{LOD})_k}{n_k} - \frac{(\frac{1}{LOD})_l}{n_l} \right| \quad (8)$$

N is the number of devices and n_i is the number of unit cells of the device i. Hence, there are $\sum_{i=1}^{N} n_i$ number of unit cells in total. $(\frac{1}{LOD})_i$ is the $\frac{1}{LOD}$ value of the device i. For the comparison between different placement, it is not essential to calculate the exact value of $\frac{1}{LOD}$, rather the following simplification of Eq. (7) is sufficient.

$$(\frac{1}{LOD})_i = \sum_{u=1}^{n_i} \left(\frac{1}{x_u} + \frac{1}{w + 1 - x_u}\right) \quad (9)$$

Here, x_u is the column of unit cell u and w is the number of unit cells in each row, and these values can be easily obtained with changing placements. For example, if the unit cell u is placed at the lowest left corner of 4×6 placement grid, x_u is 1 and w is 6.

D. Diffusion Break

The diffusion region can be shared between two units of the same device and between units from different devices if their drain/source regions are connected. This practice can significantly reduce layout area, thereby improving routing length and spatial variations. In

cases where sharing is not possible, there is diffusion break. Our objective is to have zero diffusion break.

Below, we present pseudocode for counting the number of diffusion breaks. The algorithm iterates through each unit in the row following the layout pattern. In each iteration, we maintain two units: 'curr' and 'next,' each comprising three fields: 'name,' 'left' terminal, and 'right' terminal. The process begins by checking whether starting from a drain ('D') as the leftmost terminal would result in a diffusion-break-free solution. Following the netlist, if two consecutive units are found to be not connected, the process is repeated with the source ('S') as the leftmost terminal. If we still reach two consecutive units having no connection, it indicates the presence of a diffusion break. To resolve this, a dummy unit is inserted between the units. Now, the left terminal of the unit, next to the dummy, becomes the leftmost terminal. The process continues until the end of the row is reached. The process is repeated for every row, and the diffusion break numbers $n\_db(i)$, $i \in \{1, 2, ..., row\}$ are summed up to determine the total number of diffusion breaks. The time complexity of our algorithm is $O(n)$, where n is the number of units.

Algorithm 2 Diffusion Break Model

```
getDiffusionBreak(pattern_{row×column}, netlist)
for i = 1 to row do
    n_db(i) = 0 // number of diffusion break in row i
    curr.left = 'D'; curr.right = 'S'; startSourceDone = false
    j = 0; start_s = 0; start_d = 0
    while j++ <= column-2 do
        curr.name = pattern(i,j); next.name = pattern(i,j+1)
        if curr.name == next.name then
            next.left = curr.right
            if next.left == 'D' then next.right = 'S'
            else next.right = 'D'
            curr = next; continue
        else if curr connected to 'S' of next in netlist then
            next.left = 'S'; next.right == 'D'; curr = next; continue
        else if curr connected to 'D' of next in netlist then
            next.left = 'D'; next.right == 'S'; curr = next; continue
        else if startSourceDone == false then
            startSourceDone = true; curr.left = 'S'; curr.right = 'D'
            start_d = j; j = min(start_s, start_d)
        else
            startSourceDone = false; n_db(i)++; curr.left = 'D'; curr.right = 'S'
            start_s = j; j = max(start_s, start_d); start_s = j; start_d = start_s
return sum(n_db)
end getDiffusionBreak
```

E. Optimizing Placement

Simulated annealing (SA) is a probabilistic optimization technique based on a global search metaheuristic [15]. SA is particularly useful when the search space is discrete. While SA typically performs well for small to medium size problems, it may have problems for large size problems, potentially getting stuck in local minima. Additionally, due to its iterative nature, SA can be time-consuming when applied to large problems. The motivation behind using SA is that the search space in our problem is not large enough that SA would be considered inefficient, and it is also not small enough that a brute force method would work. Note that there are $\frac{(\sum_{i=1}^{N} n_i)!}{\prod_{i=1}^{N} (n_i!)}$ possible placement solutions, where n_i is the number of units for the device i and N is the number of devices.

Let $f : \mathcal{S} \rightarrow \mathbb{R}$ be an objective function defined in the solution space $\mathcal{S}$, and $\mathcal{N}(s)$ be the neighbor function for $s \in \mathcal{S}$. SA starts with an initial solution $s \in \mathcal{S}$. At each iteration, neighboring solution $s^{'} \in \mathcal{N}(s)$ of the current solution s is selected through a predefined perturbation strategy. The metropolis criterion determines the acceptance probability for the system's transition from the current solution s to a candidate counterpart $s^{'}$ according to the following.

$$P(s, s^{'}, t_k) = \begin{cases} 1, & \text{if } f(s^{'}) \leq f(s) \\ exp(\frac{-(f(s^{'}) - f(s))}{t_k}), & \text{if } f(s^{'}) > f(s) \end{cases} \quad (10)$$

where $f(\cdot)$ is the objective function defined in Eq. (1) and t_k represents the temperature value at the k^{th} iteration. The metropolis criterion guides the system toward a state of lower cost.

1) Initial Solution: The initial step in our proposed algorithm involves determining the number of rows and columns based on the width and height of the unit cell and the total number of units from the netlist. It is understood that a square-like pattern offers the best matching performance [14][11]; therefore, we strive to achieve such a pattern, even if it necessitates the inclusion of dummy devices. The initial pattern is then generated with a sequential placement of device (with units of the same device placed together) with two consecutive devices sharing either drain and source, or source and source, or source and drain in the netlist. For example, for the circuit in Fig. 2, we can place units of $M2$, $M0$, $M1$, and $M3$ in the four rows, from top to bottom, respectively. This sequential arrangement ensures there is no diffusion break to begin with. We have observed that this technique of initial pattern generation yields better final results than starting from a completely random placement. In cases where we begin from a completely random placement, the algorithm may fail to converge to a diffusion-break-free solution.

2) Random Perturbation: A neighboring placement solution is produced by some predefined perturbation strategy. We follow the below rules for the perturbation.

- Swap only if the pair is from different devices.
- Swap only if the new solution does not degrade diffusion break.
- Swap if there is an improvement in the objective.
- Swap a worse solution with some probability defined in Eq. (10).

We measure the quality of a swap using models developed in prior sections. When we encounter a candidate that improves the current best objective (Eq. (1)) without degrading diffusion break, best solution is updated. We update the current solution if a candidate improves the current objective without degrading diffusion break. We may also update the current solution

with the candidate solution even if it deteriorates the objective. The likelihood of that happening decreases as the temperature decreases. Note that a solution is never accepted if it degrades diffusion break. Fig. 3(a) shows the execution of the algorithm, how best, current, and candidate objectives change over the course of the iterations as temperature changes. Below, we present a pseudocode of the placement optimization algorithm.

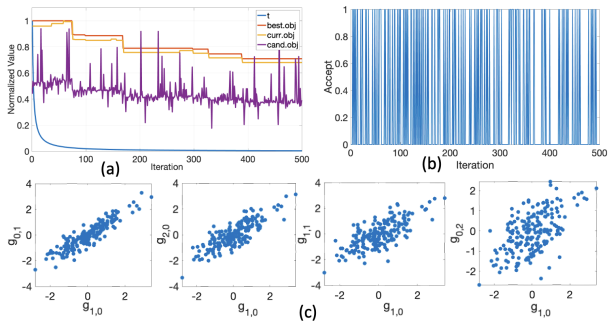

Fig. 3: (a) Placement algorithm run. (b) As temperature decreases, the likelihood of accepting a worse new solution reduces. (c) Spatial variations data generated using multivariate statistics.

Algorithm 3 Optimize Placement

Input : *netlist, init_pattern, n_iteration, temp,* η_c, $c \in \{MV, RC, MILD\}$
Output : *best_pattern*

best.obj = f(init_pattern, netlist, η_c*); best.DB = DB(init_pattern, netlist)*
curr = best
for *i=1 to n_iteration* ***do***
  Generate random number, *x and y*
  cand = curr
  tmp.pattern = cand.pattern
  cand.pattern(x.row, x.col) = curr.pattern(y.row, y.col)
  curr.pattern = tmp.pattern
  cand.obj = f(cand_pattern, netlist, η_c*); cand.DB = DB(cand_pattern, netlist)*
  if *cand.obj < cand.obj* $\wedge$ *cand.DB* $\leq$ *cand.DB* ***then*** *best = cand*
  t_i *= temp/(i+1)*
  diff.obj = cand.obj - curr.obj
  diff.DB = cand.DB - curr.DB
  metropolis = exp(-diff.obj/t_i*)*
  Generate random number, *z*
  if *(diff.obj < cand.obj* $\wedge$ *diff.DB* $\leq$ *0)* $\vee$ *(z < metropolis* $\wedge$ *cand.DB* $\leq$ *best.DB)* ***then***
    curr = cand
return *best_pattern*

IV. Experimental Results

We have implemented the proposed algorithm in Python, referred to as SA here. We contacted the authors of the paper [14] and discussed the implementation of their algorithm, which we refer to as SOTA (State-of-the-Art). Some of the results presented for the SOTA approach are based on this discussion and others are taken directly from [14]. Since it is difficult to obtain the exact values of the coefficients in Eq. (6) [3], and the second order terms can be as significant as the first order terms [6], we define the five random variables in Eq. (6), with a mu of [0 0 0 0 0], and a positive semidefinite covariance matrix of [0.9 0.8 0.7 0.6 0.5; 0.8 0.9 0.8 0.7 0.6; 0.7 0.8 0.9 0.8 0.7; 0.6 0.7 0.8 0.9 0.8; 0.5 0.6 0.7 0.8 0.9], following the multivariate statistics. Correlation of these variables is shown in Fig. 3(c). η_c for $c \in \{MV, RC, MILD\}$, and $temp$ in the simulated annealing algorithm are set to 1, and 100, respectively. In all the tests, our algorithm converges within 1000 iterations ($n\_iteration$). All experiments are conducted on a Linux environment with an Intel Core 3.3GHz CPU with 128GB memory.

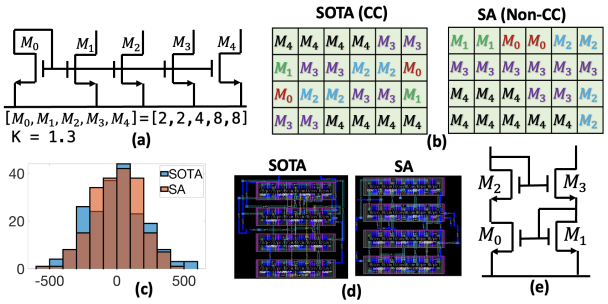

Fig. 4: (a) Current mirror (CM) circuit. (b) Pattern produced by SOTA and SA algorithms. (c) Histogram plot of the mismatch. (d) Layout of the patterns using standard industry tool. SOTA and SA layouts have 88 and 67 units route length, respectively. (e) Differential load pair cascode (DLPC).

To compare the proposed approach with SOTA, we use five configurations of the current mirror structure, as reported in [14], referred to as CM:1-5 in Table I. Our algorithm outperforms SOTA. Both algorithms produce layout free of diffusion break (DB). We further validate quality of the layout using standard industry tool. We place the devices following the pattern generated by the algorithms and auto-route the circuit. The total route length ('Router' in Table I) is estimated using a script provided by industry representative. Our algorithm consistently performs better than SOTA, thereby validating correctness of the routing model. It is important to note that a shorter route length should result in reduced parasitics and IR drop, an important requirement in analog design. However, standard industry tools do not factor in spatial variations, as foundry models lack sensitivity to the distance between devices. While these tools do consider LOD effect, quantifying its isolated impact is challenging due to the presence of other layout effects and random variations. LDEs are considered in post-layout simulation in [14], nevertheless, the improvement in circuit offset performance mainly comes from the dummies, placed around the CC structure. In Fig. 4(b), we present the layout pattern of the circuit in Fig. 4(a). Fig. 4(c) presents histogram plot of the mismatch in spatial variations. The actual layout of this example is presented in Fig. 4(d).

We have created six more tests in Table II, two from each of the three different configurations, CM (Fig. 4(a)), DIPC (Fig. 2), and DLPC (Fig. 4(e)). The proposed algorithm outperforms SOTA. Importantly, the

Test	Algo.	*MV*	*MILD*	*RC*		*DB*
				Model	Router	
CM:1	SOTA	208	0.58	75	88	0
[2,2,4,8,8], K=1.3	SA	174	0.44	65	67	0
CM:2	SOTA	381	0.46	55	61	0
[2,2,4,10], K=2	SA	280	0.40	47	54	0
CM:3	SOTA	77	0.36	46	53	0
[2,2,4,8], K=1.3	SA	18	0.31	44	50	0
CM:4	SOTA	425	0.26	76	82	0
[4,4,8,8], K=1.3	SA	31	0.13	64	69	0
CM:5	SOTA	738	0.33	108	113	0
[4,4,4,10,10], K=2	SA	251	0.03	86	91	0

TABLE I: Comparison with [14] for five current mirror (CM) configurations reported in [14].

proposed algorithm produces layout with no diffusion break while SOTA has diffusion break. Note that the 'Router' and model values differ significantly in SOTA due to dummies being added to the layout to enable diffusion sharing while keeping the CC structure intact. All the patterns produced by our algorithm in Tables I and II are non-CC type. Our algorithm is fast, and takes less than 7 mins for the largest test.

Test	Algo.	*MV*	*MILD*	*RC*		*DB*
				Model	Router	
CM:1	SOTA	359	0.67	56	96	2
[2,2,2,2,10], K=2	SA	233	0.40	46	57	0
CM:2	SOTA	447	0.67	58	98	2
[2,2,2,6,6], K=2	SA	213	0.37	48	56	0
DIPC:1	SOTA	319	0.01	127	182	4
[6,6,10,10], K=2	SA	98	0.11	90	91	0
DIPC:2	SOTA	199	0.07	158	191	4
[10,10,10,10], K=2	SA	111	0	118	107	0
DLPC:1	SOTA	157	0.28	47	64	0
[2,2,6,6], K=2	SA	46	0.35	40	62	0
DLPC:2	SOTA	290	0.18	65	116	3
[6,6,6,6], K=1.3	SA	47	0.12	60	70	0

TABLE II: Comparison with [14] for six tests.

V. Conclusions

We have proposed an efficient algorithm for analog transistor placement optimization focusing on nonlinear spatial variations while addressing important layout constraints within the optimization. Our approach has a well-defined objective and can handle varied configurations. Considering the constraint imposed by centroid matching, which limits the optimization search space, opting for a non-CC layout becomes especially advantageous when dealing with nonlinear effect.

VI. Acknowledgement

This work was supported in part by NSF under grants CCF-1704758 and CCF-2112665, SRC under task 3160.007, and an equipment donation from Nvidia.

References

[1] A. Hastings, *The Art of Analog Layout*. Prentice Hall, 2001.

[2] M. Pelgrom, A. Duinmaijer, and A. Welbers, "Matching properties of MOS transistors," *IEEE Journal of Solid-State Circuits*, vol. 24, no. 5, pp. 1433 – 1439, 1989.

[3] T.-C. Yu, S.-Y. Fang, C.-C. Chen, Y. Sun, and P. Chen, "Device Array Layout Synthesis With Nonlinear Gradient Compensation for a High-Accuracy Current-Steering DAC," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, no. 4, pp. 717–728, 2018.

[4] J.-M. L. C.-W. Lin, C.-L. Lee and S.-J. Chang, "Analytical-based approach for capacitor placement with gradient error compensation and device correlation enhancement in analog integrated circuits," in *International Conference On Computer-Aided Design*, 2012, p. 635–642.

[5] X. Dai, C. He, H. Xing, D. Chen, and R. Geiger, "An N^{th} Order Central Symmetrical Layout Pattern for Nonlinear Gradients Cancellation," in *International Symposium on Circuits and Systems*, 2005.

[6] C. C. McAndrew, "Layout Symmetries: Quantification and Application to Cancel Nonlinear Process Gradients," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 36, no. 1, pp. 1–14, 2017.

[7] A. K. Sharma, M. Madhusudan, S. M. Burns, P. Mukherjee, S. Yaldiz, R. Harjani, and S. S. Sapatnekar, "Common-Centroid Layouts for Analog Circuits: Advantages and Limitations," in *Design, Automation and Test in Europe Conference*, 2021.

[8] P.-H. Wu, M. P.-H. Lin, X. Li, and T.-Y. Ho, "Parasitic-Aware Common-Centroid FinFET Placement and Routing for Current-Ratio Matching," *ACM Transactions on Design Automation of Electronic Systems*, vol. 21, no. 3, pp. 1–22, 2016.

[9] P.-Y. Chou, N.-C. Chen, M. P.-H. Lin, and H. Graeb, "Matched-routing common-centroid 3-D MOM capacitors for low-power data converters," *IEEE Transactions on Very Large Scale Integration Systems*, vol. 25, no. 8, pp. 2234 – 2247, 2017.

[10] M. Vadipour, "Gradient Error Cancellation and Quadratic Error Reduction in Unary and Binary D/A Converters," *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 50, no. 12, pp. 1002–1007, 2003.

[11] F. Burcea, H. Habal, and H. E. Graeb, "A new chessboard placement and sizing method for capacitors in a charge-scaling DAC by worst-case analysis of nonlinearity," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 35, no. 9, p. 1397–1410, 2015.

[12] K.-W. Su, Y.-M. Sheu, C.-K. Lin, S.-J. Yang, W.-J. Liang, X. Xi, C.-S. Chiang, J.-K. Her, Y.-T. Chia, C. Diaz, and C. Hu, "A scaleable model for STI mechanical stress effect on layout dependence of MOS electrical characteristics," in *Custom Integrated Circuits Conference*, 2003.

[13] P. Drennan, M. L. Kniffin, and D. R. Locascio, "Implications of proximity effects for analog design," in *Custom Integrated Circuits Conference*, 2006.

[14] A. K. Sharma, M. Madhusudan, S. M. Burns, P. Mukherjee, R. Harjani, and S. S. Sapatnekar, "Performance-Aware Common-Centroid Placement and Routing of Transistor Arrays in Analog Circuits," in *International Conference On Computer-Aided Design*, 2021.

[15] C. D. G. S. Kirkpatrick and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, no. 4598, p. 671–680, 1983.

[16] T.-C. Chen and Y.-W. Chang, "Modern Floorplanning Based on Fast Simulated Annealing," in *International Symposium on Physical Design*, 2005.

[17] D. Vashisht, H. Rampal, H. Liao, Y. Lu, D. Shanbhag, E. Fallon, and L. B. Kara, "Placement in Integrated Circuits using Cyclic Reinforcement Learning and Simulated Annealing," in *Neural Information Processing Systems*, 2020.

[18] C.-W. Lin, J.-M. Lin, Y.-C. Chiu, C.-P. Huang, and S.-J. Chang, "Mismatch-aware common-centroid placement for arbitrary-ratio capacitor arrays considering dummy capacitors," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 31, no. 12, p. 1789–1802, 2012.

[19] Q. Ma, L. Xiao, Y.-C. Tam, and E. F. Y. Young, "Simultaneous Handling of Symmetry, Common Centroid, and General Placement Constraints," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 30, no. 1, pp. 85–95, 2011.

[20] D. Long, X. Hong, and S. Dong, "Optimal two-dimension common centroid layout generation for MOS transistors unit-circuit," in *International Symposium on Circuits and Systems*, 2005.