Constructive Common-Centroid Placement and Routing for Binary-Weighted Capacitor Arrays

Nibedita Karmokar, Arvind K. Sharma, Jitesh Poojary,
Meghna Madhusudan, Ramesh Harjani, and Sachin S. Sapatnekar
University of Minnesota, Minneapolis, MN, USA

***Abstract*—The accuracy and linearity of capacitive digital-to-analog converters (DACs) depend on precise capacitor ratios, but these ratios are perturbed by process variations and parasitics. This paper develops fast constructive procedures for common-centroid placement and routing for binary-weighted capacitors in charge-sharing DACs. Parasitics also degrade the switching speed of a capacitor array, particularly in FinFET nodes with severe wire/via resistances. To overcome this, the capacitor array is placed and routed to optimize switching speed, measured by the 3dB frequency. A balance between 3dB frequency and DAC INL/DNL is shown by trading off via counts with dispersion. The approach delivers high-quality results with low runtimes.**

I. Introduction

The accuracy and performance of circuits such as charge-scaling digital-to-analog converters (DACs) (Fig. 1) depend on binary-weighted capacitor ratios [1], [2], which may be perturbed by mismatch. We consider the problem of CC layout [3] of binary-weighted capacitor arrays for DACs, which reduces systematic mismatch, and take advantage of the problem structure to optimize DAC performance metrics. Several related works (e.g., [4], [5]) that address CC layout do not leverage the specific properties of DACs. For DAC structures, routing parasitics are critically important, but numerous prior CC placement methods [5]–[7] ignore their impact. Methods that do incorporate routing considerations [8]–[10] are based on computationally expensive stochastic search [1], [2], [8], [10]. In contrast, we develop a fast, constructive approach.

We focus on layout in FinFET nodes where per-unit wire/via resistances are high. Prior techniques address older bulk technology nodes and cannot easily be adapted to FinFET designs, e.g., routing detours in [11], [12] incur high resistance penalties. Analog design in FinFET nodes favors MOM capacitors with high capacitance density and low-resistance device layer connections, but few prior efforts [9], [10] consider CC layout for MOM capacitors; none address FinFET node issues.

The primary contributions of our work are as follows.
(1) We present a fast, constructive, place/route algorithm for CC layout of binary-weighted DAC capacitor arrays. The router is applied to a new *spiral* style, existing *chessboard* methods, and a new family of *block chessboard* placements.
(2) We target FinFET technologies, coping with via/wire resistances by building CC structures with few vias, and by implementing larger effective wire widths using parallel wires in low metal layers (as required under width quantization).
(3) We show that prior placement methods that improve dispersion (which reduces INL/DNL variance) result in unacceptable 3dB frequency. We introduce the *spiral* placement approach that shows large improvements in 3dB frequency over prior work with some cost to INL/DNL, and *block chessboard* approaches that trade off 3dB frequency with INL/DNL.

This work is supported in part by the DARPA IDEA program, as part of the ALIGN project, under SPAWAR Contract N660011824048.

II. Background

A. The impact of capacitance mismatch on a DAC

In a charge-scaling DAC (Fig. 1), the capacitor bottom plates are connected to buffers. The top plates are initially grounded by closing the Reset switch. After the Reset switch is opened, a digital code is applied to the bottom plate through the buffers, creating charge-shared voltage on the common node of the capacitor top plates. This is fed to an opamp to provide V_{OUT}.

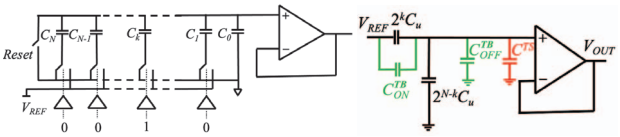

Fig. 1: Schematic and equivalent circuit of a charge-scaling DAC with the k^{th} bit set to 1 and all other bits set to 0.

Fig. 1 shows the equivalent circuit for code i. Let D_k be the k^{th} bit of the code; let C_T, $C_{ON}(i)$ and $C_{OFF}(i)$ be the total capacitance, and sum of capacitors whose bottom plates are connected to V_{REF} and ground, respectively. In a binary-weighted DAC, $C_k = n_k C_u$ where $n_k = 1$ for $k = 0$; $n_k = 2^{k-1}$ for $k \geq 1$. Noting that $C_0 = C_u$ is always grounded,

$$C_T(i) = C_u \cdot \left(1 + 1 + 2 + \cdots + 2^{N-1}\right) = 2^N C_u \quad (1)$$

$$C_{ON}(i) = \sum_{k=1}^{N} D_k(i) 2^{k-1} C_u \; ; \; C_{OFF}(i) = \sum_{k=1}^{N} \overline{D}_k(i) 2^{k-1} C_u$$

For perfect capacitor matching and an ideal opamp,

$$V_{OUT} = \frac{C_{ON}(i)}{C_T} V_{REF} = \sum_{k=1}^{N} D_k(i) 2^{(k-N-1)} V_{REF} \quad (2)$$

Shifts in the *top-plate capacitance* (Fig. 1) can alter V_{OUT}. Increasing C_u can reduce these effects, at the cost of increased power. Moreover, as C_u increases, so does the array area, with larger routing parasitics that lead to greater mismatch. *Bottom-plate parasitics* do not affect DAC linearity, but affect the load for V_{REF}, and impact power and switching frequency.

B. Modeling wire parasitics

For each metal layer, given the foundry-provided per-unit wire resistance r and per-unit wire capacitance c, the resistance [capacitance] of a wire segment of length l is $r \cdot l$ [$c \cdot l$]. For adjacent wires, if the per-unit coupling capacitance is $c_c(s)$ for a spacing of s, the coupling capacitance between two segments with overlap length $l_{overlap}$ is $c_c(s) \cdot l_{overlap}$.

C. Modeling variations in a capacitor array

To reduce systematic mismatch, capacitors are divided into identical-sized capacitor cells (called *unit cells*) that are placed in a gridded *common centroid matrix*.

1) Modeling systematic variation due to a linear gradient: We use a gradient model for systematic variation [4]. At the center of the CC array, t_0 denotes the spacing between MOM cap wires, and C_u is the unit capacitance. The oxide thickness at $(x_j,\ y_j)$ is $t_j = t_0 + \gamma(x_j \cos\theta + y_j \sin\theta)$. Here, γ and θ $(0 \leq \theta \leq 180^\circ)$ are the linear oxide gradient magnitude and angle at the origin, respectively. A unit capacitor at $(x_j,\ y_j)$ thus has value $C_u(t_0/t_j)$. Each capacitor C_k is shifted to

$$C_k^* = \textstyle\sum_j C_u \times (t_0/t_j) \tag{3}$$

2) Modeling random variations: A unit capacitor has zero-mean random variations with variance $\sigma_u^2 = A_f^2/(WH)$ [13], where A_f is similar to a Pelgrom mismatch coefficient [14], W and H are the width and height of the unit capacitor. The correlation coefficient for two unit capacitors A at (x_1, y_1) and B at (x_2, y_2) in the $(r \times s)$ CC matrix is [5]:

$$\rho_{AB} = (\rho_u)^{D(A,B)} \tag{4}$$

$$\text{where } D(A,B) = \left(\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}\right)/L_c \tag{5}$$

Here $0 < \rho_u < 1$ and L_c are process-specific parameters.

If $C_p = pC_u$ $[C_q = qC_u]$, with p $[q]$ unit capacitors, correlation coefficient $\rho_{pq} = Cov(p,q)/(\sigma_p\sigma_q)$ where

$$\sigma_p^2 = \sigma_u^2(p + 2S_p)\ ;\ \sigma_q^2 = \sigma_u^2(q + 2S_q)\ ;\ Cov(p,q) = \sigma_u^2 S_{pq} \tag{6}$$

$$S_p = \sum_{a=1}^{p-1}\sum_{b=a+1}^{p} \rho_{ab}\ ;\ S_q = \sum_{a=1}^{q-1}\sum_{b=a+1}^{q} \rho_{ab}\ ;\ S_{pq} = \sum_{a=1}^{p}\sum_{b=1}^{q} \rho_{ab}$$

III. Circuit-level metrics

A. Mismatch-induced nonlinearity error model (INL, DNL)

The *differential nonlinearity (DNL)* at input code i is:

$$DNL(i) = \left(V_{OUT}(i) - V_{OUT}(i-1) - V_{LSB}\right)/V_{LSB} \tag{7}$$

where $V_{LSB} = V_{REF}/2^N$. The *integral nonlinearity (INL)* is:

$$INL(0) = 0$$
$$INL(i) = \left(V_{OUT}(i) - V_{OUT}^{ideal}(i)\right)/V_{LSB} \tag{8}$$

Under nonidealities, we update $C_{ON}(i)$ and C_T in (2) as [7]:

$$V_{OUT} = V_{REF} \cdot \left(C_{ON}(i) + \Delta C_{ON}(i)\right)/(C_T + \Delta C_T) \tag{9}$$

where $\Delta C_{ON}(i)$ $[\Delta C_T]$ is the shift in $C_{ON}(i)$ $[C_T]$.

We model random mismatch using a 3σ model, as opposed to numerical yield integrals [7]. For an applied code i, depending on whether the k^{th} bit, D_k, is 0 or 1, capacitors may be connected to V_{REF} or ground, respectively. The values of $C_{ON}(i)$ and C_T in (1) are shifted due to nonidealities as:

$$\Delta C_{ON}(i) = \textstyle\sum_{k=1}^{N} D_k(i)\Delta C_k + C_{ON}^{TB} \tag{10}$$

$$\Delta C_T(i) = \textstyle\sum_{k=0}^{N} \Delta C_k + C_{ON}^{TB} + C_{OFF}^{TB} + C^{TS} \tag{11}$$

where C_{ON}^{TB}, C_{OFF}^{TB}, and C^{TS} represent the parasitics illustrated in Fig. 1; ΔC_k is the sum of the systematic variations, ΔC_k^{sys}, and statistical variations, ΔC_k^{sta}. Given C_k^* from (3),

$$\Delta C_k^{sys} = (C_k^* - n_k C_u) \tag{12}$$

The extent of statistical variations is given by the 3σ points of the statistical summations of C_k terms in (10) and (11). The variances of $\Delta C_{ON}(i)$ and $\Delta C_T(i)$ are shown below:

$$\sigma^2_{\Delta C_{ON}(i)} = \textstyle\sum_{j=1}^{N}\sum_{k=1}^{N} D_j(i)D_k(i)Cov(j,k) \tag{13}$$

$$\sigma^2_{\Delta C_T(i)} = \textstyle\sum_{j=1}^{N}\sum_{k=1}^{N} Cov(j,k) \tag{14}$$

where $Cov(i,j)$ is given by (6). Therefore,

$$\Delta C_{ON}(i) = \textstyle\sum_{k=1}^{N} D_k(i)(C_k^* - n_k C_u) + (3\sigma_{\Delta C_{ON}(i)} + C_{ON}^{TB})$$
$$\Delta C_T(i) = \textstyle\sum_{k=0}^{N}(C_k^* - n_k C_u) + (3\sigma_{\Delta C_T(i)} + C_{ON}^{TB} + C_{OFF}^{TB} + C^{TS})$$

B. Capacitor 3dB frequency

The switching response of a capacitor array is encapsulated in the 3dB frequency metric, which defines the maximum speed at which switches can close the bottom plates of the unit capacitors of the DAC. Thus, the 3dB bandwidth measures the maximum switching frequency of the clock, which represents how much data can be processed per unit time. Particularly for high-resolution DACs, the routing-induced parasitics may substantially affect the 3dB frequency. To our knowledge, no prior automated CC capacitor layout method has incorporated this consideration: as we will show in Section V, parasitics can severely impact this important performance metric.

The settling time, t_{settle}, of the DAC [15] must ensure that the capacitor is charged to a voltage that is ∼1/4 LSB from the final voltage, leaving room for other nonlinearities. Modeling the charge path as an RC circuit with time constant τ,

$$e^{-t_{settle}/\tau} = 2^{-N}/4 \text{ , i.e., } t_{settle} = \ln(2^{N+2})\tau \tag{15}$$

Since each cycle goes through a charge-discharge phase, the 3dB frequency, f_{3dB}, for a full cycle for a capacitor is

$$f_{3dB} = 1/(2(N+2)ln(2)\tau) \tag{16}$$

For each capacitor C_i, we measure τ by computing the Elmore delay of the RC mesh to the unit capacitors of C_i. Over all bits, the charging network with the maximum Elmore delay limits the frequency and is used as τ in (16). The Elmore delay for a mesh can be computed using standard techniques [16].

IV. Common Centroid Placement and Routing

Our constructive routing-friendly CC placement flow optimizes mismatch, interconnect wirelength, parasitic RCs, and 3dB frequency; next, a routing step optimizes DAC performance.

A. CC placement

1) Array size calculation: The CC array for an N-bit DAC consists of a grid of 2^N unit capacitors, each of which is typically built in a square aspect ratio. To minimize the impact of systematic variations, the aspect ratio of the rectangular CC array is made as close to a square as possible. The N capacitors, each consisting of $n_0, \cdots, n_{N-1}$ unit capacitors, can be placed in an $r \times s$ array, where

$$r = \left\lceil \sqrt{\textstyle\sum_{i=0}^{N} n_i} \right\rceil,\ s = \left\lceil \textstyle\sum_{i=0}^{N} n_i / r \right\rceil \tag{17}$$

For an N-bit DAC with binary-weighted arrays, the capacitor ratios are $[n_0 : n_1 : n_2 : \cdots : n_N] = [1 : 1 : 2 : \cdots : 2^{N-1}]$. Therefore, $\sum_{i=1}^{N} n_i = 2^N$. For odd N, D_C dummy capacitors are needed to complete the array, where $D_C = (r \times s) - 2^N$; for even N, $r = s = 2^{N/2}$, and no dummies are needed.

2) Placement tradeoffs between wire resistance and dispersion: Good matching under random variations is ensured through *dispersion*, which reflects the spread of the unit capacitances of C_0 through C_N in the CC array. An additional major consideration is to build routing-friendly placements that optimize interconnect parasitic effects. Previous efforts have not addressed the specific needs of FinFET technologies, with high wire resistance and higher via resistances. As FinFET technologies used reserved-direction routing, especially in lower metal layers that are used for MOM capacitors, every bend in a wire incurs a via resistance cost due to a layer change.

Reducing via count is critical for reducing interconnect resistance and improving 3dB frequency. An extension of high-dispersion chessboard placement [7] matches capacitive routing parasitics but neglects resistance: results show high via counts. We consider a range of new constructive placement solutions – **spiral** placement and **block chessboard (BC)** methods – to trade off interconnect parasitics with dispersion.

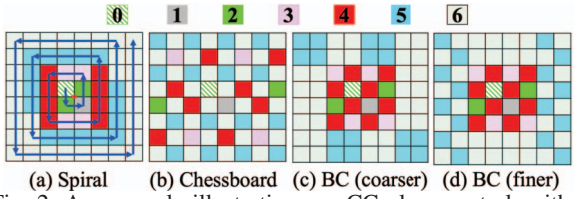

Fig. 2: An example illustrating our CC placement algorithm.

Spiral placement for optimized interconnect parasitics This solution minimizes the number of bends in the connections and is illustrated for a 6-bit DAC in Fig. 2(a). Since the number of unit capacitors in C_0 and C_1 is 1, an odd number, it is not possible to achieve a common-centroid placement. Instead, we place these as close to the common centroid as possible to limit the impact of process variations. Here, we place C_0 and C_1 diagonally opposite each other near the center. Next, we place all the capacitors of C_2, then C_3, and so on, in a spiral sequence from the center.

Whenever we place a unit capacitor at a location along a spiral, we also place another unit capacitor at its reflection to maintain the CC property. Considering the CC point as the origin (the red dot in the figure), if we place a unit capacitor in a square (d_1, d_2), it will be accompanied by another unit capacitor at location $(-d_1, -d_2)$. For example, when the spiral places a unit capacitor of C_2 at $(-1, -1)$ in the figure, we place another unit capacitor of C_2 at $(+1, +1)$. We place the unit capacitors of C_3 at the first empty location along the spiral, first at $(1, 2)$ and its reflection at $(-1, -2)$, and so on.

Beyond C_2, this technique naturally aligns numerous unit cells of a capacitor to lie in the same row or column, the method reduces the number of vias (corresponding to wire "bends") required to connect them. This approach maintains adequate dispersion while also reducing the number of required turns (corresponding to vias) for the routing connections. Although the method is simple, it is different from previous methods: the nearest similar methods are [4], with a mix of rectangles and circles for placement, and [17] with interleaved rows, but it does not achieve good dispersion.

Chessboard placement for optimized dispersion [7] At the other extreme, [7] optimizes dispersion by interspersing unit capacitors in a chessboard pattern, as illustrated for a 6-bit DAC in Fig. 2(b). For a 6-bit DAC, the 32 unit capacitors of C_6 are first placed in an 8×8 array on the black squares of a "chessboard"; then the 16 unit capacitors of C_5 are placed; and so on. However, the routing resistance costs here are large.

Block chessboard (BC) approaches A block chessboard approach attempts to find the best of both worlds, by achieving the dispersion of the chessboard approach and the lower routing costs of the spiral approach. Examples of this approach for a 6-bit DAC are shown in Figs. 2(c) and 2(d). The inner core of this structure is a conventional chessboard layout for the capacitors with a smaller number of unit cells (here, C_0 through C_4): this provides good dispersion, and while it has a high number of bends/vias, its wire RCs are typically smaller than those of the larger capacitors C_5 and C_6, and do not constrain the 3dB frequency, which is determined by the worst-case time constant. The outer corridor here has a width of 2 cells. Since $n_6 : n_5 = 2 : 1$, we first lay out half the cells of C_6 in clusters and then perform chessboard routing, alternating the remaining cells of C_6 with C_5. Two layouts are shown for different granularities in the outer corridor.

Other BC structures may be built with the inner full-chessboard core of $C_0 - -C_k$, and an outer block structure for $C_{k+1} - -C_N$. MSB capacitors do not greatly affect DAC accuracy since their variation is averaged over more unit capacitors than LSB capacitors. MSB capacitors use fewer vias in BC than chessboard, resulting in higher 3dB frequency.

To create a block chessboard layout, starting from $i = k+1$, at each step we choose a block size for C_i and place the blocks in chessboard fashion. We increment i and repeat until $i = N$; if N is odd, we also add dummies in block chessboard fashion.

B. Routing

1) Routing patterns within the CC array: The following routing parasitics are seen for capacitor C_i in the CC array:
(1) The top-plate capacitance C_i^{TS} to ground (Fig. 1) participates in the evaluation of V_{out}. Parasitic C_i^{TS} leads to gain errors, affecting INL/DNL. Since all top plates are connected, these parasitics appear in parallel at an opamp input. This parasitic must be minimized using short routes.
(2) The top-plate-to-bottom-plate parasitic capacitance, C_i^{TB}, (Fig. 1) is in parallel with the capacitor C_i and effectively increases the value of C_i. Therefore, several techniques endeavor to make it proportional to C_i [11], [12].
(3) The bottom plate capacitance C_i^{BS} to ground is connected to the switches and driver that charge it to V_{REF}. By ensuring that the switches are on for a sufficiently long time (Section III-B), its impact on linearity metrics is minimal.

As in [8], we minimize C_i^{TB} with nonoverlapped routing that separates the wires that route the top-plate and bottom-plate. MOM capacitors often use 3 or more metal levels, and via-free connections are possible in the same direction even when metals are routed using a reserved layer direction.

2) Connected unit capacitor group formation: To connect all bottom plates of unit cpacitors of each C_i, we first create connected capacitor groups of neighboring unit capacitors of each C_i. We represent unit capacitors by nodes in graph G, with edges between nodes for these neighboring unit capacitors.

We apply a breadth first search (BFS) algorithm on G to find its connected components (*connected capacitor groups*). The bottom plates of neighboring unit capacitors in the BFS tree are connected using *branch wires*: each connection is immediately mirrored to the unit capacitor at the diagonally symmetric location in the CC placement, maintaining symmetric routing. The connected unit capacitor groups for a 6-bit CC array for a spiral placement is shown in Fig. 3(a). Section IV-B4 discusses how parallel connections for bottom-plate routing (shown here for C_6) improve performance. A similar method can be used for block chessboard placements. Chessboard placements have no bottom-plate connected capacitor groups.

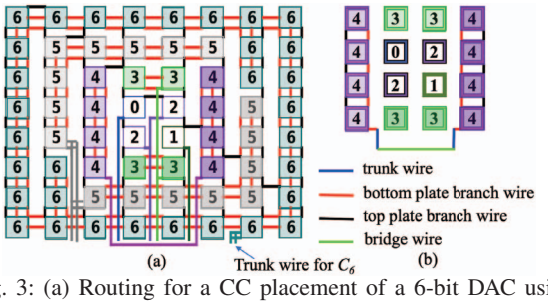

Fig. 3: (a) Routing for a CC placement of a 6-bit DAC using parallel wires. For connected capacitor groups (different shades of the same color), top-plate [bottom-plate] connections are shown in red [black]. (b) Routing topology for C_4.

3) Bottom-plate routing: Bottom-plate routing requires separate routes to connect the unit capacitor groups of each C_i. For a DAC, the bottom-plate terminals in the capacitive array are connected to switches and drivers that are clustered together outside the array. Since these are noisy digital structures, they are clustered together outside the array, away from sensitive analog elements. Without loss of generality, we assume that this cluster lies below the array, i.e., the terminals must go to the bottom of the array. Wires are routed between capacitor array columns in vertical *tracks*: since the number of wires is small, the spacing between columns for the tracks is negligible compared to the unit capacitor size.

We use three types of wires for routing (Fig. 3(b)): *branch wires* are used to connect unit capacitors within capacitor groups, or unit capacitor groups to trunk wires, *trunk wires* connect disjoint connected capacitor groups along the vertical tracks, and *bridge wires* connect trunk wires at the periphery of the array. The routing method is outlined in Algorithm 1, and consists of three steps: channel selection (Step 1), track assignment (Step 2), and routing (Step 3).

Channel selection attempts to assign capacitor groups to channels so that they maximize track sharing. If two connected capacitor groups share a common vertical channel span, they can share a connection along a track. The outer loop over i iterates over all capacitors C_i. The next loop over j considers each unvisited capacitor group p for C_i, and looks for other capacitor groups q (in the loop over k) to share channels with. Line 14 checks whether p and q share a horizontal span: if not (e.g., for capacitor groups for C_4 in Fig. 3(a)), then they cannot share a channel; if they do (e.g., for the C_5 groups in the figure), the channels that can be shared are to the left/right of the columns that both p and q span (for C_5, the third to sixth column). Next, we choose the closest cells $u_p \in p$ and $u_q \in q$ (line 16), which will be connected to the trunk wire in the track: this choice minimizes the trunk wire length segment that connects p and q. In case of a tie, we choose the wire closest to the bottom of the layout, to minimize the connection length to the drivers at the bottom. In Fig. 3(a), we choose the lowest cell of the upper C_5 group in column 2 and the leftmost cell of the lower C_5 group in column 3; not that its symmetric opposite in columns 6 and 7 is eliminated by the tie-breaker because it incurs a higher routing length to the bottom.

We now commit to connecting p to other groups through u_p (line 18), and the remaining iterations attempt to find other capacitor groups that can be connected to u_p via the channel to its left or right, building lists P_l and P_r for such candidates (lines 20–24). Depending on whether P_l or P_r has more members, we choose to route u_p on its left or right (lines 30–32). Note that using u_p to connect to u_q is already a good choice: this step attempts to find even better choices that

Algorithm 1 Bottom-plate routing

1: **Input**: $C = [C_0, \cdots, C_N]$; CC placement; $I_C[i]$, a list of connected capacitor groups for each C_i.
2: **Output**: Routed layout for the CC placement
3: *// Step 1: Channel selection*
4: **for** $i = 1$ to len(C) **do** *//Over each capacitor*
5: **for** $j = 1$ to len($I_C[i]$) **do** *//Over each unvisited capacitor group of C_i*
6: $p = I_C[i][j]$
7: **if** p is not visited **then**
8: *//Find $U[i][j]$, set of capacitor groups with channel span intersecting p*
9: *//Find $D[i][j]$, indicating whether route is to the left or right of p*
10: $P_l, P_r \leftarrow \{\}$ *//Set of capacitor groups to the left, right of p*
11: $c[j] = -1$ *//Column number of p that is routed in the channel; initialization*
12: **for** $k = 1$ to len($I_C[i]$) **do** *//Over each unvisited group of C_i except p*
13: **if** ($q = I_C[i][k]$ is not visited) and ($p \neq q$) **then**
14: **if** (horizontal_span(p) $\cap$ horizontal_span(q) $\neq \{\}$) **then**
15: Choose the nearest unit cells $u_p \in p$, $u_q \in q$ for connection;
16: if tied, choose a unit cell pair closest to bottom of CC array.
17: **if** ($c[j] == -1$) **then**
18: $c[j] \leftarrow$ column number of cell u_p
19: **end if**
20: **if** u_q is in column ($c[j] - 1$) or $c[j]$ **then**
21: $P_l \leftarrow P_l \cup \{q\}$
22: **end if**
23: **if** u_q is in column $c[j]$ or ($c[j] + 1$) **then**
24: $P_r \leftarrow P_r \cup \{q\}$
25: **end if**
26: **end if**
27: **end if**
28: **end for**
29: **if** len(P_l) $>$ len(P_r) **then**
30: $D[i][j] \leftarrow$ left; $U[i][j] \leftarrow P_l$; Mark all $q \in P_l$ as visited
31: **else**
32: $D[i][j] \leftarrow$ right; $U[i][j] \leftarrow P_r$; Mark all $q \in P_r$ as visited
33: **end if**
34: **end if**
35: **end for**
36: **end for**
37: *// Step 2: Track selection for trunk wire routing recorded in array V*
38: Calculate #tracks for each channel as the number of C_is routed in the channel.
39: **for** $i = 1$ to len(C) **do** *//Over each capacitors*
40: **for** $j = 1$ to len($I_C[i]$) **do** *//Over each unvisited capacitor group of C_i*
41: **if** ($p = I_C[i][j]$) is not visited **then**
42: **if** $D[i][j] ==$ left **then** *//Assign track to left*
43: $V[i][j] \leftarrow$ Rightmost unused track t_k in column $c[j]$ to the left of p
44: **else**
45: $V[i][j] \leftarrow$ Leftmost unused track t_k in column $c[j]$ to the right of p
46: **end if**
47: **for** $k = 1$ to len($U[i][j]$) **do** *// For each $q \in U[i][j]$, use the same track*
48: $V[i][k] \leftarrow V[i][j]$; Mark $q = U[i][k]$ as visited
49: **end for**
50: **end if**
51: **end for**
52: **end for**
53: *// Step 3: Perform branch and bridge wire routing*
54: Create trunk wires using track assignment in V; Connect capacitor groups to trunks

share tracks. Each iteration guarantees a connection from p to the drivers, and therefore, each capacitor group is guaranteed to complete routing. For $|C|$ capacitors, the cost of this step, which dominates **algorithmic complexity**, is $O(|C|^2 \log(|C|))$. This is not a major bottleneck for typical values of $|C|$.

In Step 2 (line 37), we first use the information above to compute the number of tracks required. For each channel, this is simply the number of C_is that choose to use the channel. We now assign connections to tracks for each C_i in lines 39–52, sequentially assigning connections for C_i to the closest available track. The channel widths are very small in practice *(even using parallel routes)*, and DAC performance is insensitive to track assignment within the channel. Finally, once the trunk wires are assigned, Step 3 (line 54) creates the actual routes and connects them to the unit capacitor groups. Fig. 3(b) shows how the short trunk wires for the two groups of C_4 reach the bottom of the layout and are connected by bridge wires.

4) Parallel wire routing: To reduce resistance and improve 3dB frequency in FinFET nodes under discrete wire widths, we use multiple parallel wires for critical bits. This also allows multiple parallel vias as wire direction changes. With p parallel wires, wire and via resistance reduce by $p\times$ and $p^2\times$, respectively, but wire capacitance increases by $p\times$: all RC parasitic changes are considered in our results. During capacitor group formation, parallel wires are added between two adjacent unit capacitors, and parallel routes are used on trunk/bridge wires. Parallel wires for critical bit C_6 is shown in Fig. 3(a) (C_6 uses no bridge wires and a short trunk wire with p^2 vias).

5) Top-plate routing: The objective of top-plate routing is to minimize C^{TS}. We create a graph G such that each vertex $v \in G$ is a unit capacitor for any C_i (since all C_i top plates must be connected). Each unit capacitor is connected to its north, south, east, and west neighbor (if they exist), with an edge weight corresponding to the horizontal or vertical spacing, as applicable. In our case, since the vertical space between unit capacitors is less than the horizontal spacing for channels, the minimum spanning tree (MST) can be built by simply connecting all unit capacitors in each column using branch wires, and then connecting the unit capacitors in adjacent columns using a branch wire. The use of this MST minimizes, shown in Fig. 3(a), the parasitic capacitance, C_i^{TS}.

V. Results and Discussion

Our approach is implemented in Python and evaluated on a commercial 12nm technology for N-bit binary-weighted DAC arrays with capacitors ratios of $1:1:2:4:\cdots:2^{N-1}$, with N ranging from 6 to 10. We evaluate four techniques: the placement in [1]; the chessboard placement [7]; spiral placement ("S") and the block chessboard ("BC") [Section IV-A]. Several BC structures are considered, as shown in Fig. 4 and the **best BC result** is reported. All even/odd bit DACs use the same BC structure with a full chessboard for the inner core.

Our routing approach is applied to the S and BC methods. Since [7] only proposes a placement without routing, we use our router on their placement (Section IV-B).

For systematic variations, the wire spacing t_0 is based on a wire pitch of 64nm. The per-unit models for resistance, capacitance to ground, and coupling-capacitance are taken from a commercial 12nm process. The systematic variation parameters were set to $\gamma = 10$ppm, $\rho_u = 0.9$, $L_c = 1$mm [1], [8], and we use $A_f^2 = 0.85\% \times 1$fF [13]. A unit capacitance value of 5fF is used for all cases. The MOM capacitors are built in three metal layers, with the bottom-plate and top-plate terminals available in metal1 and metal2, respectively.

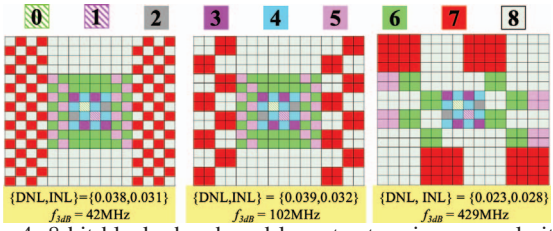

Fig. 4: 8-bit block chessboard layouts at various granularities.

Table I shows various parameters associated with the RC parasitics for routing. The capacitances, ΣC^{TS} (total top-plate-to-substrate), ΣC^{wire} (total wiring capacitance), and ΣC^{BB} (total bottom-plate to bottom-plate), represent the parasitics shown in Fig. 1; C^{TB} is negligible due to nonoverlapped routing (Section IV-B). The next set of metrics – ΣN_V, the total number of vias and ΣL the total wire length – are correlated with the total resistance. Since f_{3dB} only depends on resistances on the critical bit with the largest RC delay, the last column shows the total via resistance, R_V, and the total wire+via resistance, R_{total} for the critical bit.

Here, S has low resistive parasitics; BC has moderate parasitics, much lower than [1], [7]. Both S and BC use our parallel routing method: when parallel routing is used on the MSB, the second-most MSB, then the third-most MSB, etc., may become critical, and parallel routing is used there too. If parallel wires need channel resources, the spacing between columns is increased appropriately. For any number of bits for S, the only vias are at the input connection. Unit capacitors use nearest-neighbor connections using the same metal layer with no vias.

The C^{TS} values for S and BC are better than those of [1]; we apply the same solution to our routing for [7] (although their subsequent work [12] leads to higher top-plate wire lengths, i.e., higher C^{TS}). For other metrics (C^{wire}, C^{BB}, N_V, L and R), the spiral approach provides the best solution and the chessboard method [7] the worst, and the block chessboard method provides an intermediate solution.

Table II shows circuit-level metrics: Area of the routed CC array; $|DNL|$ / $|INL|$ the maximum absolute $DNL(i)$ (Eq. 7) / $INL(i)$ (Eq. 8); f_{3dB}, the 3dB frequency (Eq. (16)). Like [1], [7], we evaluate DNL/INL under capacitor nonidealities, assuming an ideal opamp. Area is lowest for the spiral method due to low routing overhead and comparable for other methods (except 7-bit and 9-bit solutions for [7], which double the unit capacitors). The table shows the INL/DNL vs. f_{3dB} tradeoff (particularly for >8 bits): *S has the best f_{3dB} but the worst INL/DNL; [7] is the opposite; BC is a good compromise.* All INL/DNL values are below 0.5LSB and are acceptable.

Figs. 5(a) and (b) show a Python-generated view of the placement and routing for an 8-bit DAC in a commercial 12nm process, using the approach in [7] and our spiral placement method, respectively. While the former requires five vertical tracks in the vertical channels, the spiral approach, even with

TABLE I: CC array: Electrical metrics ($C_u = 5$fF)

# bits	$\sum C^{TS}$ (fF)				$\sum C^{wire}$ (fF)				$\sum C^{BB}$ (fF)				$(\sum N_V, \sum L)$ (μm)				(R_V, R_{total}) (KΩ) for critical bit			
	[1]	[7]	S	BC	[1]	[7]	S	BC	[1]	[7]	S	BC	[1]	[7]	S	BC	[1]	[7]	S	BC
6	0.02	0.03	0.03	0.03	1.8	2.8	0.9	1.4	13.4	6.5	0.5	1.4	(42, 149)	(81, 229)	(43, 77)	(78, 120)	(0.3, 1.2)	(1.1, 2.6)	(0.002, 0.03)	(0.03, 0.26)
7	–	0.09	0.05	0.06	–	12.6	1.9	2.0	–	28.9	1.5	1.5	–	(295, 1862)	(46, 167)	(82, 171)	–	(4.1, 10.0)	(0.002, 0.05)	(0.03, 0.30)
8	0.07	0.09	0.09	0.09	4.8	12.7	3.0	4.0	21.7	29.8	1.7	2.0	(92, 393)	(295, 1884)	(75, 256)	(86, 335)	(1.0, 3.1)	(4.1, 10.0)	(0.002, 0.06)	(0.03, 0.51)
9	0.14	0.36	0.17	0.17	8.5	59.6	5.4	5.5	61.0	242.7	3.4	7.6	(143, 703)	(1126, 9076)	(78, 453)	(92, 463)	(1.2, 4.2)	(15.8, 39.7)	(0.002, 0.10)	(0.03, 0.57)
10	–	0.36	0.32	0.33	–	59.9	9.7	12.6	–	242.7	5.1	21.5	–	(1126, 9126)	(107, 816)	(177, 1050)	–	(15.8, 39.7)	(0.002, 0.16)	(0.03, 1.03)

Notes: (1) [7] doubles the number of unit capacitors for odd bits ⇒ {7-bit, 8-bit}, {9-bit, 10-bit} results are similar. (2) 7-bit, 9-bit DACs not reported in [1].

TABLE II: CC array: Performance metrics ($C_u = 5$fF)

# bits	Area (μm^2)				$\{\|DNL\|, \|INL\|\}$ (LSB)				f_{3dB} (MHz)			
	[1]	[7]	S	BC	[1]	[7]	S	BC	[1]	[7]	S	BC
6	200	205	200	204	{0.00,0.01}	{0.00,0.01}	{0.01,0.01}	{0.01,0.01}	929	434	39613	8651
7	–	819	427	459	–	{0.01,0.01}	{0.02,0.02}	{0.01,0.01}	–	25	10862	6639
8	803	819	806	819	{0.03,0.05}	{0.01,0.02}	{0.06,0.03}	{0.02,0.03}	75	23	3962	908
9	1655	3521	1669	1643	{0.08,0.11}	{0.02,0.04}	{0.06,0.07}	{0.04,0.07}	25	1.3	1072	714
10	–	3521	3235	3296	–	{0.05,0.09}	{0.25,0.16}	{0.11,0.11}	–	1.2	286	91

TABLE III: Runtimes for the proposed CC layout algorithms

#bits	6-bit	7-bit	8-bit	9-bit	10-bit
Spiral	0.02s	0.04s	0.12s	0.35s	1.11s
BC	0.03s	0.05s	0.19s	0.38s	2.25s

parallel routes, requires two routing tracks, resulting in lower C^{BB}, as documented by the total C^{BB} number in Table I. The total routing wirelength is significantly higher for the placement of [7], leading to higher C^{wire} parasitics, as shown in the same table. Both factors degrade the 3dB frequency of [7]. This effect is worse as the number of bits in the DAC increases.

In FinFET nodes, N_V adversely affects performance due to high via resistance. The spiral method uses the fewest vias of all methods, and the chessboard method [7] uses the most.

Fig 6(a) shows the impact of using parallel routes, which reduce interconnect resistance, on the 3dB frequency for spiral placement. We show the frequency improvement factor, i.e., the ratio of the 3dB frequency using k wires vs. using one wire. The increase in parasitic capacitance due to parallel wires is minimal and is dominated by the capacitance in the array, but the wire resistance reduction is significant. As k increases, we see diminishing returns. Similar trends are seen for the block chessboard scheme. When two parallel wires are used, the frequency improvement factor exceeds 2: for this resistance-dominated case, the connection from the trunk wire to a branch

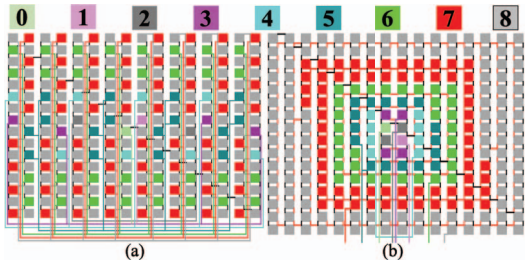

Fig. 5: CC layout in (a) [7] (b) spiral approach. High wirelength for [7] is inevitable: cells are spread for high dispersion.

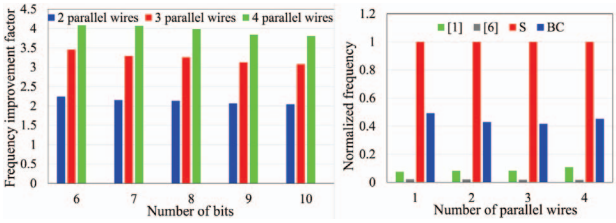

Fig. 6: 8-bit CC array with parallel wires: 3dB frequency improvements for (a) the spiral method. (b) all methods.

wire creates a 2×2 mesh with four vias. The gain lies between $2\times$ (wire-dominated case) and $4\times$ (via-dominated case). With more wires, wire capacitance becomes noticeable, leading to lower improvement. Fig. 6(b) shows the impact of parallel wires for all methods, normalized to the 3dB frequency for S. Both BC and [1] improve, but have much lower baseline frequencies than S; chessboard [7] is bottlenecked by high via counts.

CPU times for both the spiral method and for each block chessboard are similar and are reported in Table III. Because the method is constructive, it is much faster than stochastic optimization, while providing excellent quality of result.

VI. Conclusion

We present a routing-friendly constructive CC placement and routing method to optimize mismatch and maximize performance. Spiral and block chessboard placements are presented, and it is shown that trading off wire parasitics with dispersion provides a balance between the 3dB frequency and INL/DNL.

References

[1] M. P.-H. Lin, *et al.*, "Common-Centroid Capacitor Layout Generation Considering Device Matching and Parasitic Minimization," *IEEE TCAD*, vol. 32, pp. 991–1002, 2013.

[2] W.-H. Hsiao, *et al.*, "Automatic Common-Centroid Layout Generation for Binary-Weighted Capacitors in Charge-Scaling DAC," in *Proc. SMACD*, pp. 173–176, 2012.

[3] N. Karmokar, *et al.*, "Common-centroid layout for active and passive devices: A review and the road ahead," in *Proc. ASP-DAC*, 2022.

[4] D. Sayed and M. Dessouky, "Automatic Generation of Common-Centroid Capacitor Arrays with Arbitrary Capacitor Ratio," in *Proc. DATE*, pp. 576–580, 2002.

[5] P.-W. Luo, *et al.*, "Impact of Capacitance Correlation on Yield Enhancement of Mixed-Signal/Analog Integrated Circuits," *IEEE TCAD*, vol. 27, pp. 2097–2101, 2008.

[6] C.-C. Huang, *et al.*, "PACES: A Partition-Centering-Based Symmetry Placement for Binary-Weighted Unit Capacitor Arrays," *IEEE TCAD*, vol. 36, pp. 134–145, 2016.

[7] F. Burcea, *et al.*, "A New Chessboard Placement and Sizing Method for Capacitors in a Charge-Scaling DAC by Worst-Case Analysis of Nonlinearity," *IEEE TCAD*, vol. 35, pp. 1397–1410, 2015.

[8] M. P.-H. Lin, *et al.*, "Parasitic-Aware Common-Centroid Binary-Weighted Capacitor Layout Generation Integrating Placement, Routing, and Unit Capacitor Sizing," *IEEE TCAD*, vol. 36, pp. 1274–1286, 2017.

[9] N.-C. Chen, *et al.*, "High-Density MOM Capacitor Array with Novel Mortise-Tenon Structure for Low-Power SAR ADC," in *Proc. DATE*, pp. 1757–1762, 2017.

[10] P.-Y. Chou, *et al.*, "Matched-Routing Common-Centroid 3-D MOM Capacitors for Low-Power Data Converters," *IEEE TVLSI*, vol. 25, pp. 2234–2247, 2017.

[11] K.-H. Ho, *et al.*, "Coupling-Aware Length-Ratio-Matching Routing for Capacitor Arrays in Analog Integrated Circuits," *IEEE TCAD*, vol. 34, pp. 161–172, 2014.

[12] Y. X. Ding, *et al.*, "PASTEL: Parasitic Matching-Driven Placement and Routing of Capacitor Arrays With Generalized Ratios in Charge-Redistribution SAR-ADCs," *IEEE TCAD*, vol. 39, pp. 1372–1385, 2019.

[13] V. Tripathi and B. Murmann, "Mismatch Characterization of Small Metal Fringe Capacitors," *IEEE TCAS-I*, vol. 61, pp. 2236–2242, 2014.

[14] M. J. Pelgrom and A. C. Duinmaijer, "Matching Properties of MOS Transistors," in *Proc. ESSCIRC*, pp. 327–330, 1988.

[15] F. Maloberti, *Data Converters*. New York, NY: Springer, 2007.

[16] S. S. Sapatnekar, *Timing*. Boston, MA: Kluwer, 2004.

[17] G. Chen, *et al.*, "Routability of Twisted Common-Centroid Capacitor Array Under Signal Coupling Constraints," in *Proc. MWSCAS*, 2016.