Toward Silicon-Proven Detailed Routing for Analog and Mixed-Signal Circuits

Hao Chen
ECE Department, UT Austin
haoc@utexas.edu

Keren Zhu
ECE Department, UT Austin
keren.zhu@utexas.edu

Mingjie Liu
ECE Department, UT Austin
jay_liu@utexas.edu

Xiyuan Tang
ECE Department, UT Austin
xitang@utexas.edu

Nan Sun
ECE Department, UT Austin
nansun@mail.utexas.edu

David Z. Pan
ECE Department, UT Austin
dpan@ece.utexas.edu

ABSTRACT

Detailed routing is an intricate and tedious procedure in design automation and has become a crucial step for advanced node enablement. Compared with its advances in digital design, detailed routing for analog/mixed-signal (AMS) integrated circuits (ICs) is still heavily manual. In AMS designs, the sensitive net coupling issues and analog-specific constraints make detailed routing even more challenging. This work presents a novel and efficient detailed routing framework for automated AMS layout synthesis considering industrial design rules as well as analog-specific geometric and electrical constraints. Experimental results demonstrate the efficiency and effectiveness of our approach in optimizing circuit performance while satisfying the specified constraints. Post-layout simulations further prove that our detailed routing results can achieve sign-off quality.

1 INTRODUCTION

Routing is the most complicated process in the layout automation flow and has become one of the bottlenecks for advanced nodes enablement. Due to the intricate design rules and substantial solution space, conventional approaches typically divide routing into two stages: global routing and detailed routing. Global routing partitions the entire routing region into a set of bins and generates rough routing solutions to optimize overall timing, crosstalk, and alleviate congestion. Detailed routing completes the final geometric implementation while satisfying all the design rules. The detailed routing problem has been broadly studied for decades. Recent efforts show promising progress to diminish the gap between academic and real-world physical design needs [9, 10, 20]. Despite the revival of detailed routing research in recent years, limited work has demonstrated an end-to-end detailed routing scheme for analog/mixed-signal (AMS) circuits.

In addition to wirelength optimization, which is the primary target of conventional routing, an AMS router needs to comprehend more design aspects such as current balancing, voltage drop, and signal coupling. Among prior art on analog routing automation, the work [1, 5] conducts template-based routing based on well-optimized human design templates to handle design-specific concerns (e.g., layout retargeting) for high-quality layouts. The scalability of these template-based methodologies is limited by the complexity of input templates, which require tremendous design efforts. In [2], sensitivity analysis is applied during routing to identify net matching information and critical nets, and further guarantee the layout quality. The simulation-based approaches can be generalized to various performance metrics (e.g., speed, power dissipation, voltage swings) in the analog domain [18]. However, these approaches suffer from long simulation time and exhausted computational resources, thus not suitable for the implementation of large scale systems such as analog-to-digital converter (ADC) and phase-locked loop (PLL).

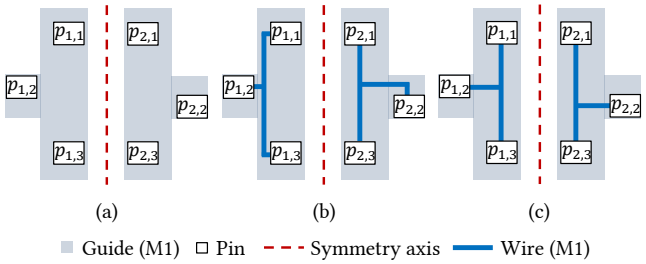


Figure 1: (a) A symmetric global routing result with asymmetric pin locations. (b) An undesirable detailed routing result. (c) A preferred detailed routing.

To tackle the AMS routing problem without jeopardizing the scalability in practices, constraint-based techniques are widely applied. Symmetry is an essential concept in analog layout synthesis [11]. A graph-based algorithm to handle the mirror symmetry constraint is demonstrated in [3]. In [12, 17, 21], maze-based routing methods considering mirror-symmetry constraints with asymmetric obstacles are proposed. Symmetry constraints can be further extended to different levels of geometric matching such as common-centroid, topology-matching, and length-matching constraints. The work [14] presents a length-matching routing method ensuring both minimum and maximum length restrictions are satisfied. The work [22] handles length-matching constraints for general routing topology. In [16], a maze routing algorithm supporting exact-matching constraints is shown. In [15], an integer linear programming formulation for analog routing is proposed to consider symmetry, common-centroid, topology-matching, and length-matching constraints simultaneously.

However, the specified heuristic constraints could fail to model the situation in detailed routing. As shown in Figure 1, the global routing results for two matched nets are symmetric, yet the exact pin locations are not. Thus, a fully symmetric detailed routing solution does not exist. Nevertheless, it is desirable to route the net with similar patterns for aesthetic concerns, which are crucial principles in analog layout [4, 8]. The aesthetic in analog routing is a relief for correctness to describe hard-to-model relations in analog layout and design expertise. Existing analog routing algorithms are still limited to consider such sophisticated scenarios. Moreover, the voltage drop problem can result in dramatic performance degradation in AMS circuits; thus, wider metal width and multiple-cut vias are required to carry analog current levels. To ensure the performance of the routed layouts, handling different wire widths

ICCAD '20, November 2–5, 2020, Virtual Event, USA

ACM ISBN 978-1-4503-8026-3/20/11…$15.00
https://doi.org/10.1145/3400302.3415660

and multiple-cut vias for nets while satisfying the specified constraints, aesthetic concerns, and rigid design rules is necessary.

In this paper, we propose a hierarchical AMS detailed routing framework capable of handling large AMS systems (e.g., ADC) with industrial design rules, analog-specific constraints, and aesthetic concerns consideration. Highlights of our work are summarized as follows.

- We propose a symmetry-constraint allocation algorithm to assign suitable net matching requirements according to the pin locations.
- We propose a pin access generation method that determines the valid access points with preferred access direction.
- We develop a novel pin clustering strategy to attain regular routing patterns, which can substantially improve the overall routing aesthetic.
- We design a robust hierarchical rip-up and re-routing scheme that is capable of industrial design rules and analog-specific constraints handling and alleviates the routing congestion.
- Experimental results demonstrate that our framework can complete routing in complicated AMS designs efficiently while considering specific constraints and achieve sign-off quality performance.

The remaining sections are organized as follows. Section 2 gives the preliminaries and formulates the analog detailed routing problem. Section 3 details our AMS detailed routing algorithm. Section 4 shows the experimental results, and Section 5 concludes the paper.

2 PRELIMINARIES

In this section, we introduce the routing grid graph, some real industrial design rules, and different forms of symmetry constraints. Then, we formulate the analog detailed routing problem.

2.1 Routing Grid Graph

Routing tracks with preferred direction are standard in digital detailed routing. The state-of-the-art digital detailed routers [9, 10] perform routing on such structure and minimize the wrong-way and off-track wires. Analog circuit designers typically adopt the shape-based gridless routing methodology for connections, especially in bulk CMOS designs. However, the enormous solution space of gridless routing results in massive runtime overhead to an automated router. As a result, gridless approaches are not amenable to more complicated analog systems. In our framework, we introduce routing tracks without preferred direction on each metal layer to improve efficiency, scalability, and routability while maintaining the routing flexibility. The heavy-duty design-rule-checking (DRC) is also significantly relieved since the tracks are spaced as per the technology and constraints.

The routing tracks form a 3-D grid graph $G = (V, E)$ to model the entire routing region, where V is a set of vertices and $E \subseteq V \times V$ is a set of edges. The set V consists of *grid points*, which are the intersections points of the tracks. An edge $e = (u, v) \in E$ signifies the interconnection from vertex u to v. The edge between two neighboring vertices is either a horizontal (resp. vertical) wire segment for intra-layer connections, or a via for inter-layer connection. For local connections within a routing grid, gridless routing is utilized to handle design rules and sophisticated patterns.

2.2 Industrial Design Rules

To satisfy manufacturing requirements, design rules need to be well handled. We introduce some of the fundamental and typical design rules considered in detailed routing.

2.2.1 Parallel Run Spacing. According to the parallel run length of two wires, a minimum required spacing between them is specified regarding the maximum width of the two wires. Figure 2(a) illustrates the parallel run spacing constraint.

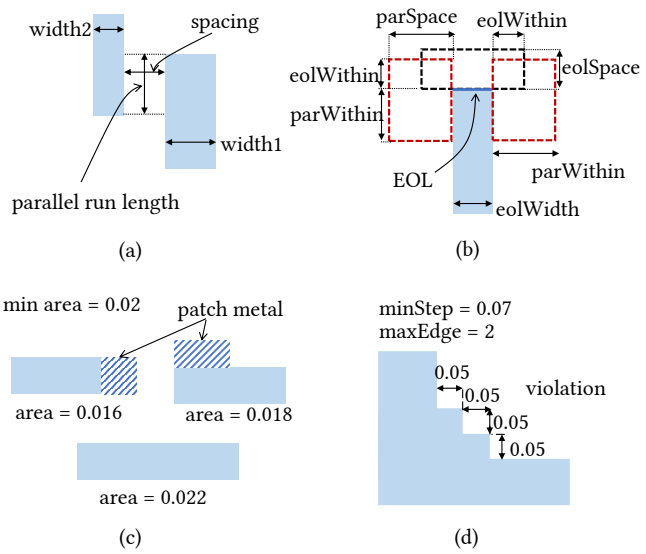


Figure 2: (a) Example of parallel run spacing. (b) Example of EOL spacing. (c) Example of the min area rule. (d) Example of a min-step max-edges violation.

2.2.2 End-of-line Spacing. The end-of-line (EOL) spacing specifies the minimum required spacing *eolSpace* anywhere less than the *eolWithin* distance beyond an EOL (i.e., a metal end with width smaller than *eolWidth*). The EOL spacing constraint is applied if any other edge parallel to EOL occurs in the *parallel edge windows* constructed by extending the corners of EOL sideward by *parSpace*, forward by *eolWithin*, and backward by *parWithin*. Figure 2(b) shows an example of the EOL spacing rule.

2.2.3 Min-area. As exemplified in Figure 2(c), the min-area rule specifies that the area of a metal polygon should be greater or equal to a specified threshold.

2.2.4 Min-step. To ensure the correctness of optical pattern correction (OPC) during mask creation in the manufacturing stage, the min-step rule is specified. The min-step rule states that no more than *maxEdge* consecutive edges with length less than *minStep* of a metal polygon is allowed. Figure 2(d) demonstrates an example of a min-step violation.

2.3 Analog-Specific Constraints

To improve the overall performance of an analog layout, additional constraints need to be specified.

2.3.1 Symmetry Constraints. In the previous analog routing approach, symmetry constraints are specified to route nets symmetrically regarding some joint axes. However, the straightforward mirroring technique [3, 15, 21] cannot describe different situations of matched nets. For example, the mirroring technique is not capable of routing two matched nets with pin locations on both sides of the symmetry axis since an overlap is inevitable when crossing the symmetry axis. In our framework, we extend the symmetry constraints into four variants: mirror-, cross-, self-, and partial-symmetry. Mirror-symmetry is applied when the pins of two matched nets located on each side of the symmetry axis. Cross-symmetry is employed when the pins of two matched nets appear on both sides of the symmetry axis. Self-symmetry specifies the matching requirement within a net. Partial-symmetry happens when the matched nets lack some symmetric pin pairs. To be specific, partial-symmetry constraints

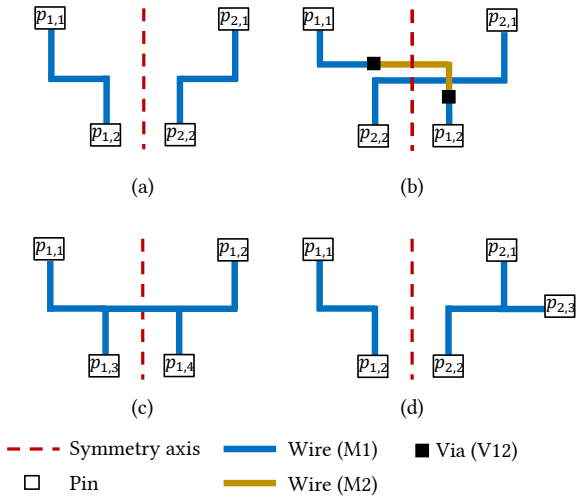


Figure 3: Examples of the four symmetry constraint variants. (a) Mirror-symmetry. (b) Cross-symmetry. (c) Self-symmetry. (d) Partial-symmetry.

are always bonded with other constraints, e.g., partial self-symmetry, partial cross-symmetry. Figure 3 illustrates the four variants of symmetry constraints, where $p_{i,j}$ denotes the j^{th} pin of the i^{th} net.

2.3.2 Electrical Constraints. IR-induced voltage drop can cause significant performance degradation in analog circuits. To maintain the performance, some critical nets are specified to have wider metal widths and multiple-cut vias for inter-layer connections. In our framework, we perform pre-layout simulation to extract the maximum tolerable parasitic resistance, which defines the minimum wire width and the via cut number, and specify them as constraints.

2.4 Problem Formulation

We formally define the analog detailed routing problem as follows.

Problem 1 (Analog Detailed Routing)**.** Given a set of nets $N = \{n_i | 1 \le i \le |N|\}$, a set of global routing guides $H = \{h_i | 1 \le i \le |H|\}$, and a set of placed devices $M = \{m_i | 1 \le i \le |M|\}$, generate a routing solution for each net $n_i \in N$ considering $h_i \in H$ such that n_i is connected without design rule violations and the specified constraints are satisfied.

3 HIERARCHICAL ROUTING FRAMEWORK

With an input hierarchical circuit netlist, as shown in Figure 4, we propose a bottom-up analog detailed routing approach to achieve high-quality solution and efficiency. The overall flow of our routing algorithm for each hierarchy is shown in Figure 5. In the flow, a symmetry constraint allocation technique based on the well-known graph matching algorithm [7] is applied if the symmetry constraints are not provided by the designer. Otherwise, our algorithm honors the specification by designer and takes it as input directly. After the pre-processing stage, the core of our routing framework includes three main phases: 1) pin access computation with preferred access direction, which models complicated pin shapes and reduces the complexity of the problem, 2) constraint-aware iterative routing, which connects all nets while considering design rules and constraints described in Section 2, and 3) post-processing, which refines the remaining design rule violations, to be detailed in the following. The notations used in this paper is summarized in Table 1.

3.1 Symmetry Constraint Allocation

The symmetry constraints input defined by circuit designers reflects their expertise, experience, and insights. For example, some sensitive

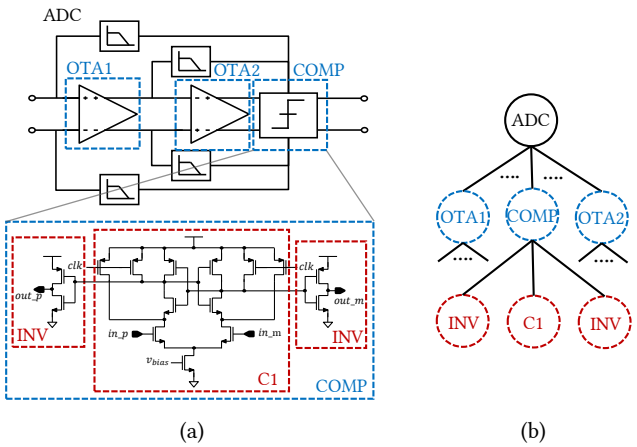


Figure 4: (a) Circuit netlist with hierarchical structure. (b) Routing hierarchy tree of (a).

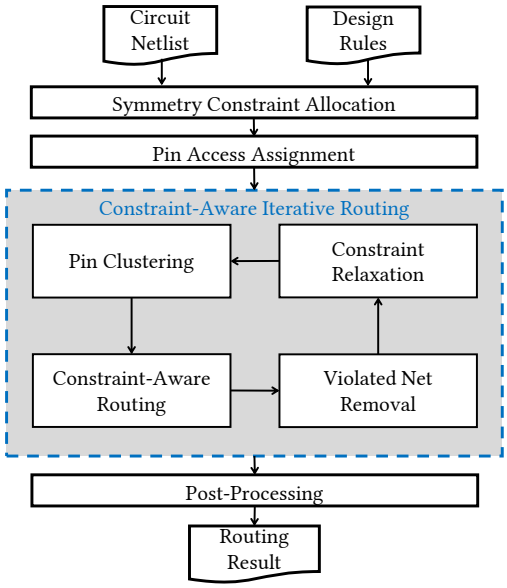
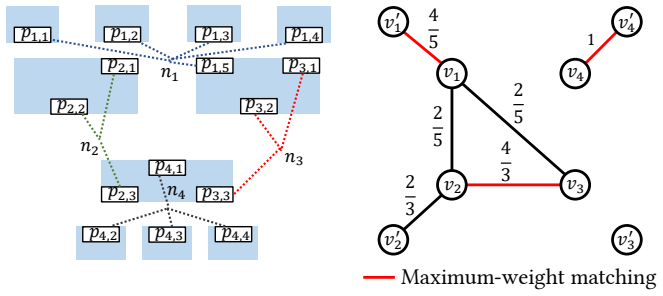


Figure 5: Computation flow of our AMS routing framework.

Table 1: Notations used in this paper.

Symbol	Description
N	The set of nets specified in the circuit netlist.
n_i	The i^{th} net in N, $1 \le i \le \|N\|$.
P	The set of all pins.
P_i	The set of pins in n_i.
$p_{i,k}$	The k^{th} pin in P_i, $1 \le k \le \|P_i\|$.
$x_{i,k}$	The center x-coordinate of $p_{i,k}$.
λ_i	The self-symmetry axis of n_i.
$\lambda_{i,j}$	The symmetry axis between n_i and n_j.
$a_{i,k}(x)$	A function with output 1, if $p_{i,k}$ has a corresponding symmetric pin in P_i with respect to x; 0, otherwise.
$b^j_{i,k}(x)$	A function with output 1, if $p_{i,k}$ has a corresponding symmetric pin in P_j with respect to x; 0, otherwise.

differential nets should have almost identical routing topology to avoid circuit performance degradation caused by the mismatch. Thus, our analog routing engine will honor the constraints provided by circuit designers. Though circuit designers have better common sense and more



(a) (b)

Figure 6: (a) A placement result with four nets. (b) The constructed matching graph for (a) and the weight of each edge is labelled aside.

Algorithm 1 *AllocateSymConstraints*(L, N)

Input: A placement layout L and a set of nets N.
Output: A set S_{sym} of symmetry constraints.

```
1:  Initialize an empty undirected graph G = (V, E);
2:  AddVertices(V);
3:  for each n_i ∈ N do
4:      for each n_j ∈ N do
5:          if n_i == n_j then
6:              ψ_i := ComputeEstSelfSymDeg(n_i);
7:              if ψ_i > 0 then
8:                  E := E ∪ {(v_i, v'_i, ψ_i)};
9:          else
10:             ψ_{i,j} := ComputeEstSymDeg(n_i, n_j);
11:             if ψ_{i,j} > 0 then
12:                 E := E ∪ {(v_i, v_j, 2ψ_{i,j})};
13: S_sym := MaxWeightedMathcing(G);
14: return S_sym;
```

insights in sensitivity analysis, they could suffer from handling complex geometrical implementation, such as specifying all the constraints among a massive amount of complicated interconnections. To alleviate the workload of designers, our analog routing engine is also able to identify and assign suitable symmetry constraints to matched nets concerning the geometrical locations of their pins.

After acquiring the placement layout and the routing netlist, we perform the symmetry constraint allocation procedure and determine appropriate constraints for matched nets. The procedure is sketched in Algorithm 1. We first construct an undirected graph $G = (V, E)$, where V is a set of vertices, and E is a set of weighted edges. The function *AddVertices* then initializes V by adding two separate vertices v_i and v'_i for each net n_i given in the circuit netlist. To build the weighted edge set E, we compute the weight of each edge according to the estimated degree of symmetry or self-symmetry of the nets. For simplicity, we only detail the computation of horizontal symmetry below, while vertical symmetry can be achieved in the same manner.

For n_i equal to n_j, we compute the estimated degree of self-symmetry ψ_i and the correlated symmetry axis λ_i by the following equation,

$$\begin{aligned} \psi_i &= f_i(\lambda_i), \\ \lambda_i &= \underset{1\le k,\, k'\le |P_i|}{\operatorname{argmax}} f_i\left(\frac{x_{i,k}+x_{i,k'}}{2}\right), \\ f_i(x) &= \frac{\sum_k a_{i,k}(x)}{|P_i|}. \end{aligned} \tag{1}$$

An edge (v_i, v'_i, ψ_i) is added to E if and only if $\psi_i > 0$. Otherwise, we compute the estimated degree of symmetry $\psi_{i,j}$ and the correlated symmetry axis $\lambda_{i,j}$ as follows.

$$\begin{aligned} \psi_{i,j} &= f_{i,j}(\lambda_{i,j}), \\ \lambda_{i,j} &= \underset{1\le k\le |P_i|,\, 1\le k'\le |P_j|}{\operatorname{argmax}} f_{i,j}\left(\frac{x_{i,k}+x_{j,k'}}{2}\right), \\ f_{i,j}(x) &= \frac{\max\left(\sum_k b^j_{i,k}(x), \sum_{k'} b^i_{j,k'}(x)\right)}{\max(|P_i|, |P_j|)}. \end{aligned} \tag{2}$$

Similarly, we add an edge $(v_i, v_j, 2\psi_{i,j})$ to E if and only if $\psi_{i,j} > 0$.

Example 1. For net n_1 in Figure 6(a), we have $\psi_1 = \frac{4}{5}$ and $\lambda_1 = \frac{x_{1,1}+x_{1,4}}{2} = \frac{x_{1,2}+x_{1,3}}{2}$ since $p_{1,1}$ and $p_{1,2}$ are symmetric to $p_{1,4}$ and $p_{1,3}$ with respect to λ_1. For nets n_2 and n_3, we have $\psi_{2,3} = \frac{4}{3}$ and $\lambda_{2,3} = \frac{x_{2,2}+x_{3,2}}{2} = \frac{x_{2,3}+x_{3,3}}{2}$ since $p_{2,2}$ and $p_{2,3}$ are symmetric to $p_{3,2}$ and $p_{3,3}$ with respect to $\lambda_{2,3}$.

The complete edge set is built when all the combinations of net pairs are examined, as shown in Lines 3-12 in Algorithm 1. After the graph construction process, we perform Edmond's blossom algorithm [7] on the graph to find the maximum-weight matching. Then, we translate the matching result to specify the symmetry constraints for nets.

The complexity of the symmetry constraint allocation algorithm is derived as follows. The calculation of the weight of edges is dominated by the iterative for-loops. Hence, the complexity is in $O(|P|^2)$. For the maximum-weight matching, which is the dominating factor of the entire process, we have $O(|V|^2|E|)$ for the Edmond's blossom algorithm theoretically to get an optimal matching, where $|V| = 2|N|$ and $|E| \le (|N|^2 + |N|)/2$. In practice, the graph G is sparse since many subsets in N are *mutually exclusive*, and a large number of nets are *self-disassociate*. The circuit hierarchy further splits the entire netlist into partitions. Therefore, the algorithm executes efficiently.

Definition 1. A set of nets is *mutually exclusive* if and only if the estimated degree of symmetry between every two nets is 0.

Definition 2. A net is *self-disassociate* if and only if the estimated degree of self-symmetry is 0.

Example 2. Figure 6(a) describes a scenario of matching. We can observe that $\{n_1, n_4\}$, $\{n_3, n_4\}$ and $\{n_2, n_4\}$ are mutually exclusive, and n_3 is a self-disassociate net. Figure 6(b) shows the final matching result for symmetry constraint allocation.

3.2 Pin Access Assignment

The pin access problem is one of the most challenging subroutines in detailed routing. To reduce the complexity of the analog detailed routing problem and further improve the routability, we propose a pin access assignment method considering preferred access direction to model intricate pin shapes.

For each pin, we assign the covering same-layer grid points in the routing grid graph as its access points. Then, we determine a preferred direction set for each access point considering the active region of the corresponding device. The preferred direction set provides information for our detailed router to generate desirable access patterns without violating design rules.

To collect the preferred directions of an access point, a set of *pattern candidates* is first generated as follows. Given a routing grid graph $G = (V, E)$ and an access point on a grid point $u \in V$. If exists an edge $e = (u, v) \in E$ for some grid points v such that v does not lie on the same pin with u, a pattern candidate is created according to e. To be specific, if u and v lie in the same layer, we construct a wire along the edge e; otherwise, a via is generated as the candidate. The pattern is then put

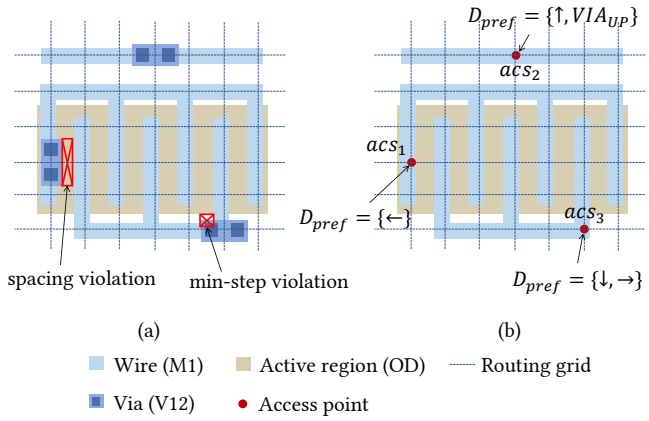


Figure 7: Examples of pin access points with preferred directions.

into the candidate list if not causing any design rule violation. Next, we calculate the overlapping area between each candidate and the active region and select the candidate with the smallest overlapping area. After that, we add the direction of the pattern to the preferred direction set.

Example 3. Figure 7 shows an example of the access point generation technique. In Figure 7(a), we can see that the candidate via of access point acs_1 violates the design rules in layer M1. Thus, the preferred direction set D_{pref} of acs_1 is $\{\leftarrow\}$, as shown in the snapshot in Figure 7(b).

For those pins not covering a same-layer grid point, we adopted the concept in [10] to find an access point in other layers, without setting the preferred access direction. However, in practical analog designs, most of the pins have at least one same-layer access point since the scale of an analog device is typically more substantial. The cross-layer access points occur when some small digital cells are used in a mixed-signal design.

3.3 Constraint-Aware Iterative Routing

Before we detail our iterative routing algorithm, we define the following terminologies.

Definition 3. A *symmetric net* is a net specified with a mirror- or cross-symmetry constraint, with or without a further specified partial-symmetry constraint.

Definition 4. A *self-symmetric net* is a net specified with a self-symmetry constraint or a partial self-symmetry constraint.

Definition 5. A *symmetric pair* consists of two symmetric nets sharing the same symmetry axis.

To connect all the nets considering design rules and constraints, we perform constraint-aware iterative routing. The negotiation-based rip-up and re-route methodology [13] is adopted to alleviate congestion. To obtain a better net routing order, we propose a hybrid priority function

$$PR_i = \alpha \cdot HPWL_i + \beta \cdot |P_i| + \gamma \cdot d_i + \delta \cdot z_i \tag{3}$$

for a maximum-priority queue to maintain all the unrouted nets, where PR_i represents the cost of a net n_i in the queue, $HPWL_i$ is the half-perimeter wirelength of n_i according to its pins, d_i is the estimated degree of symmetry (resp. self-symmetry) obtained by Equation (2) (resp. Equation (1)) if n_i is a symmetry (resp. self-symmetry) net, z_i denotes the contiguous routing failures, and α, β, γ, δ are weighted constants. In our framework, α, β, γ, and δ are set to 0.1, 2, 100, and 50, respectively.

Nets with more pins are more difficult to route. Considering the number of pins in the cost function gives those nets higher priorities. In our implementation, γ is set to a higher value since it is also much challenging to route nets with symmetry constraints. With the hybrid cost function,

Algorithm 2 *ConstraintAwareRoute*(n_i)

Input: A net n_i with a set of pin clusters C_i.
Output: The routing result R_i of n_i.

```
1: Union the obstacles on both sides of the symmetry axis;
2: R_i := ∅, P_rest := P_i;
3: for each c_k ∈ C_i do
4:     if c_k not routed then
5:         r_k := Route(c_k);
6:         if r_k is legal then
7:             R_i := R_i ∪ {r_k, Mirror(r_k)};
8:             P_rest := P_rest \ c_k;
9:             P_rest := P_rest \ SymCluster(c_k);
10: r_rest := Route(P_rest);
11: R_i := R_i ∪ {r_rest};
12: return R_i;
```

we can increase the routing success rate and decrease the number of total iteration. The turnaround time of the entire process is thus reduced significantly.

After initializing the routing order, the core algorithm of the routing scheme includes four main stages: 1) pin clustering, 2) constraint-aware routing, 3) violated net removal, and 4) constraint relaxation.

3.3.1 Pin Clustering. To handle all the symmetry constraints mentioned in Section 2.3.1, a novel pin clustering technique is proposed. We brief the clustering process for three different types of nets: symmetric nets, self-symmetric nets, and normal nets, as follows.

For symmetric nets, we split the pins of a symmetric pair (n_i, n_j) into two disjoint subsets according to their symmetry axis, respectively. For instance, $P_i^l, P_i^r \subseteq P_i$ are the two disjoint subsets for n_i, where P_i^l comprises the pins on the left-hand side of the symmetry axis, and P_i^r consists of the pins on the right-hand side. We detect the maximum fully symmetric parts between (P_i^l, P_j^r), (P_i^r, P_j^l), and set them as *symmetry clusters*. Note that each symmetry cluster in a symmetric net has a corresponding symmetry cluster that belongs to the other net.

For a self-symmetric net n_i, the subsets P_i^l and P_i^r are constructed by the aforementioned method. If there is a pin with its center locating on the symmetry axis, it is added to both P_i^l and P_i^r. We then identify the maximum fully symmetric part among the pins concerning its symmetry axis and build the *self-symmetry clusters*.

For normal nets, since no symmetry constraint is attached, the pin clustering step is skipped.

Example 4. In Figure 8(a), a partial cross-symmetric net pair (n_1, n_2) is shown. Observe that the disjoint pin subsets of n_1 are $P_1^l = \{p_{1,1}, p_{1,2}\}$ and $P_1^r = \{p_{1,3}, p_{1,4}, p_{1,5}\}$; while n_2 has $P_2^l = \{p_{2,3}, p_{2,4}\}$ and $P_2^r = \{p_{2,1}, p_{2,2}\}$. Since $p_{1,1}, p_{1,2}$ are symmetric to $p_{2,1}, p_{2,2}$, the pin clusters c_1 and c_2 are constructed for n_1 and n_2, respectively. The clusters c_3 and c_4 are obtained by the same method.

3.3.2 Constraint-Aware Routing. Before the routing procedure, the required minimum wire width and minimum cut number for inter-layer connection for each net is defined through circuit schematic simulation. However, the electrical constraints set by the schematic simulation are bottom lines, and different circuit placements may lead to significant routing diversities. Long signal wires in complex AMS system without enough width will degrade the circuit performance due to voltage drop. As a result, we estimate the potential routing wirelength by the HPWL model. Then, the electrical constraints are adjusted appropriately by increasing the required wire width and via cuts of some nets.

In [3, 21], a mirroring technique by taking the union of obstacles on both sides of the symmetry axis is described. However, the mirroring

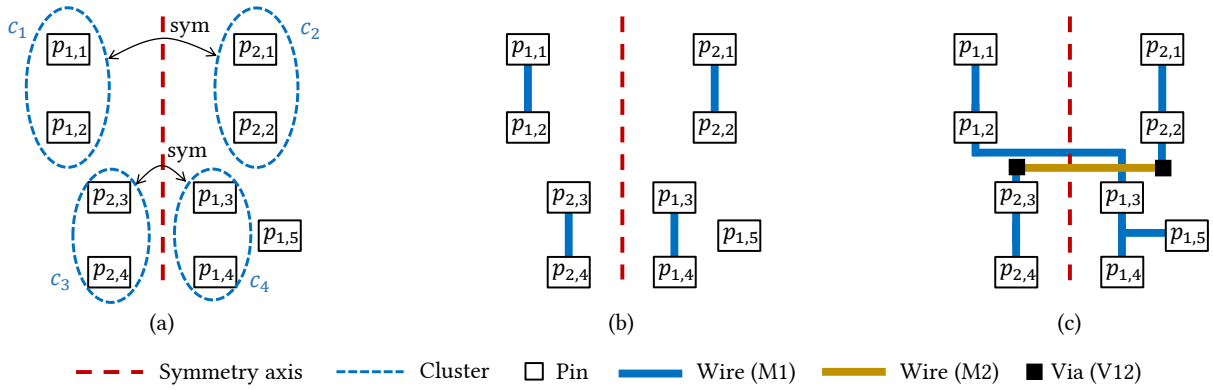


Figure 8: An example of our routing procedure with the pin clustering technique.

technique is limited to the mirror-symmetry constraint. We extend and combine the concept of mirroring with our proposed pin cluster structure to route the cross-symmetry and partial-symmetry nets. Algorithm 2 shows our constraint-aware routing approach.

For each unrouted cluster, an obstacle-aware pathfinding using the A* search algorithm is applied. To honor the global routing result, penalties are added to a search-node if it is outside the routing guide. Once a route is found, it is mirrored to the other side to give a symmetric solution. After routing the pins inside the clusters, mirror-symmetric and self-symmetric nets can be completely routed. However, cross-symmetric and partial-symmetric nets are still not fully connected. An additional pathfinding step is then performed to connect the remaining pins and the clusters.

For the symmetric pairs and partial self-symmetric nets that do not exist a perfectly symmetric routing solution, our routing algorithm can still generate highly regular routing patterns through the pin cluster structure. This routing methodology further raises the degree of symmetry of the final routing results, and thus achieve better layout aesthetic. Figure 8 gives an example of the routing flow. In Figure 8(b), symmetric routing patterns are generated for pin clusters. The final routing result is then obtained in Figure 8(c).

3.3.3 Violated Net Removal. In the rip-up phase, we apply design-rule-checking to verify the routing result and remove the nets that cause design rule violations. To avoid obtaining the same routing result and alleviate congestion, a history cost is updated to the history map before the violated wires are removed. By maintaining the history map for negotiation-based rip-up and re-route, a detour will occur in the next iteration due to the higher routing cost.

3.3.4 Constraint Relaxation. To ensure all the nets can be routed without design rule violations, we check if there are some hard-to-route nets and relax the specified constraints. If the number of times that a net fails to obtain a violation-free routing solution exceeds a defined threshold, the symmetry constraint that specified on the net is then waived, in order to improve the routability.

3.4 Post-Processing

In this step, we check the remaining design rule violations in the routing solution and perform a layer by layer refinement. For each set of connected metals, we transform them into a single polygon and iterate through its edges clockwise to validate the min-step rule. According to the min-step rule described in Section 2.2, no more than *maxEdge* consecutive edges with length less than *minStep* is allowed. We handle this issue more aggressively; any two adjacent edges with a length smaller than *minStep* is treated as a violation. We identify either the two adjacent

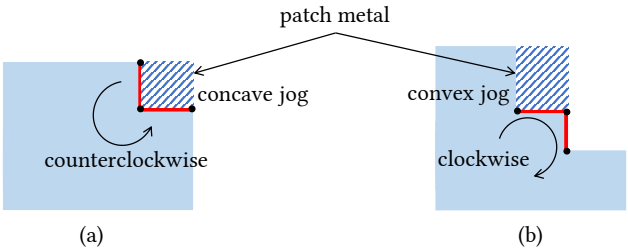


Figure 9: (a) Example of a concave jog. (b) Example of a convex jog.

edges form a *concave jog* or a *convex jog*, by computing the orientation of the three ordered points of the two edges. Since we iterate the edges of a polygon clockwise, a clockwise orientation of the three ordered points indicates a convex jog; otherwise, a concave jog. To resolve the violation, we add patch metal to eliminate excessive edges. Figure 9 shows an example of the metal patching method for the min-step rule violation fixing.

4 EXPERIMENTAL RESULTS

The proposed analog detailed routing framework is implemented in C++ with the Boost C++ libraries [19] and the LEMON graph library [6]. All experiments are conducted on a Linux workstation with an Intel i9 3.3GHz CPU with 128GB memory. To assess the scalability of our proposed framework, we conduct experiments on five analog designs with different scales, including three building block level designs and two complete mixed-signal systems. For the block level circuits, we have one comparator and two operational transconductance amplifiers (OTAs), denoted as COMP, OTA1, and OTA2. For the mixed-signal systems, we have two continuous-time $\Delta\Sigma$ modulators (CTDSMs), denoted ADC1, and ADC2. Note that a CTDSM system consists of OTAs, comparators, feedback DACs, and digital cells. All benchmark circuits and placement results are designed by experienced analog circuit designers under TSMC 40nm process. The global routing guides for each benchmark are generated by our analog global router. Table 2 lists the benchmarks statistics.

Table 2: Benchmark circuits information.

Benchmark	#Devices	#Nets	#Pins	Die Size
COMP	16	12	52	15.0×17.8 μm^2
OTA1	25	18	78	97.8×32.5 μm^2
OTA2	34	26	107	39.7×40.2 μm^2
ADC1	206	127	419	139.9×121.5 μm^2
ADC2	153	109	345	128.2×262.0 μm^2

Table 3: Comparison of total wirelength (WL(nm)), via number (VIA), total degree of symmetry (d_{SYM}), total design rule violations (DRV), and runtime(s).

Benchmark	[23]					DR_{wl}					DR_{sym}				
	WL	VIA	d_{SYM}	DRV	Runtime	WL	VIA	d_{SYM}	DRV	Runtime	WL	VIA	d_{SYM}	DRV	Runtime
COMP	145665	90	0.37	83	1.34	143400	19	0.88	0	0.15	138400	19	0.95	0	0.11
OTA1	520640	167	0.31	170	36.30	369400	38	0.63	0	1.86	386800	38	0.88	0	1.71
OTA2	546875	191	0.19	130	15.18	443400	74	0.27	0	0.37	523400	79	0.70	0	0.30
ADC1	2898835	498	0.37	550	39.65	2625000	171	0.49	0	5.95	2686600	175	0.62	0	2.70
ADC2	N/A	N/A	N/A	N/A	N/A	3188800	180	0.47	0	23.02	3327600	184	0.69	0	18.82
Norm.	1.13	3.60	0.40	-	24.75	0.95	0.98	0.70	-	1.42	1.00	1.00	1.00	-	1.00

4.1 Routing Metrics Evaluation

To evaluate the routing solution, we compare our router with the analog detailed router of [23] in terms of wirelength, via number, degree of symmetry, design rule violations, and runtime. The degree of symmetry d_{SYM} of a routing result reflects the layout aesthetic. It is defined as the total wirelength of symmetry wires over total wirelength, as shown in the following equation,

$$d_{SYM} = \frac{\sum_{i=1}^{|N|} \sum_{j=1}^{|Seg(n_i)|} g_i(s_{i,j}) \cdot len(s_{i,j})}{\sum_{i=1}^{|N|} \sum_{j=1}^{|Seg(n_i)|} len(s_{i,j})}, \quad (4)$$

where $Seg(n_i)$ is the set of routed segments of net n_i, $s_{i,j}$ is the j^{th} routing segment of n_i, $len(s_{i,j})$ is the length of segment $s_{i,j}$, and $g_i(s_{i,j})$ is a binary output function with output 1 if exists a symmetry segment of $s_{i,j}$ with respect to the symmetry axis of n_i.

As satisfying the symmetry constraints achieves better layout regularity in the cost of increasing total wirelength, it is desirable to explore the trade-off between the degree of symmetry and wirelength. As a result, we implement our framework with two different parameter sets, denoted as DR_{wl} and DR_{sym}, respectively. DR_{wl} executes more rip-up and re-route iterations to optimize wirelength, while DR_{sym} has a more rigorous criteria for constraint relaxation to maximize the degree of symmetry. All binaries are executed on the same Linux workstation for a fair comparison, and the symmetry constraints are generated by Algorithm 1. After generating the routed layouts, Calibre nmDRC and nmLVS are applied for design-rule-checking and layout-versus-schematic verification.

Table 3 shows the comparisons between the routing results. The detailed router of [23] fails to complete the routing in ADC2 due to some local congestion. Comparing the solutions of [23] and DR_{sym}, the detailed router of [23] results in 13% longer total wirelength and 3.6× more vias than DR_{sym}. Furthermore, DR_{sym} generates DRC-clean results with 2.5× higher degree of symmetry with the runtime speedup of above 24×. On the other hand, comparing the results of [23] and DR_{wl}, we can see that [23] has 19% longer total wirelength and 3.67× more vias than ours. DR_{wl} also achieves DRC-clean results with 1.75× higher degree of symmetry with above 17× runtime speedup.

Compared with DR_{sym}, DR_{wl} achieves an average of 5% reduction on wirelength and 2% reduction on via number in the cost of 30% reduction in the degree of symmetry. As wirelength and the degree of symmetry of a routed layout may relate to different performance metrics in analog designs, DR_{wl} and DR_{sym} can be suitable for various categories of analog circuits. Figure 10 shows the ADC2 layout generated by DR_{sym}.

4.2 Post-Layout Simulations

Post-layout simulations are performed to verify the quality of the routed layouts. To construct a more accurate simulation model, Calibre PEX is used to extract parasitic resistance, parasitic capacitance, and coupling capacitance (R+C+CC). R+C+CC extraction is essential for evaluating the quality of layouts generated by an automated router to check if the electrical constraints are well-handled. After layout extraction, we use Cadence Spectre to evaluate the extracted netlist.

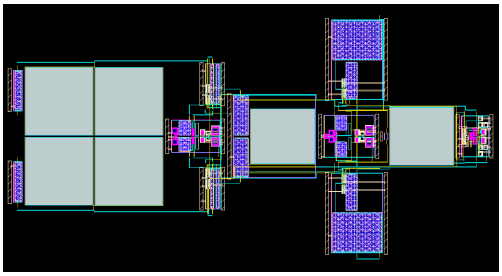


Figure 10: Layout of ADC2 routed by our router.

Table 4: Simulation results of schematic, [23], and our AMS detailed routing framework.

Benchmark		Schematic	[23]	DR_{wl}	DR_{sym}
COMP	Delay (ps)	69.5	177.3	126.0	125.0
	Offset (μV)	-	2804.1	462.3	827.0
	Noise (μV_{rms})	384.1	410.2	376.2	350.0
	Power (μW)	13.7	20.6	20.8	20.7
OTA1	DC Gain (dB)	45.9	N/A	45.8	45.8
	Bandwidth (MHz)	110.5	N/A	99.0	99.0
	Phase Margin (°)	64.7	N/A	64.6	64.6
	Offset (μV)	-	N/A	186.5	167.2
	Noise (μV_{rms})	222.0	N/A	231.9	231.8
	Power (μW)	776.9	N/A	760.3	761.0
OTA2	DC Gain (dB)	54.0	52.9	54.1	54.1
	Bandwidth (MHz)	605.2	444.8	478.8	477.0
	Phase Margin (°)	64.1	75.2	75.2	76.1
	Offset (μV)	-	893.3	154.2	145.7
	Noise (μV_{rms})	12070.1	9711.5	9818.2	9822.1
	Power (μW)	428.7	424.3	433.3	439.7
ADC1	SNDR (dB)	65.3	63.1	64.4	64.6
	SFDR (dB)	79.5	77.2	78.6	78.9
	Power (mW)	837.1	806.6	868.5	857.9
ADC2	SNDR (dB)	69.6	N/A	66.9	66.1
	SFDR (dB)	80.0	N/A	80.9	79.8
	Power (μW)	695.2	N/A	759.4	759.0

Table 4 shows the simulation results. Note that the post-layout simulation can still be performed even if the results of [23] obtain design rule violations. For COMP, DR_{wl} (resp. DR_{sym}) achieves 29% reduction in output delay (Delay), 85% (resp. 71%) reduction in input-referred offset (Offset), and better input-referred noise (Noise) with similar power consumption (Power), compared with the results of [23]. In this particular

Table 5: Process corners simulation results of ADC2 layout generated by DR_{sym}.

Corner	SNDR (dB)	SFDR (dB)	Power (uW)
TT-N	66.1	79.8	759.0
TT-C	67.4	80.8	747.8
TT-H	64.3	78.3	774.4
FF-C	71.9	83.5	812.2
FF-H	65.4	82.7	854.4
SS-C	62.4	77.8	679.8
SS-H	62.1	76.8	711.8

N: Normal (27°C), C: Cold (-20°C), H: Hot (100°C)

benchmark, though the layout generated by DR_{sym} obtains the highest degree of symmetry, it suffers from the clock coupling issue as described in [23], which results in a slightly larger input-referred offset. For the OTA1 layout generated by [23], the common-mode feedback loop is dysfunctional due to the excess routing parasitic resistance. Overall, our router obtains routing solutions with superb performance. For OTA2, our work outperforms [23] in unity-gain bandwidth, phase margin, input-referred offset, and noise. Also, the power consumption of our work is closer to the schematic. Since the input-referred offset of an OTA is more related to the overall degree of symmetry of a layout, DR_{sym} achieves the best offset performance in OTA1 and OTA2.

For complicated AMS systems, ADC1 and ADC2, our work shows outstanding performance in both signal-to-noise and distortion ratio (SNDR) and spurious-free dynamic range (SFDR). Since the feedback loops in CTDSM provide tolerance to the system offsets, the performance degradation caused by the unbalanced layout is not reflected in SNDR/SFDR. As a result, DR_{wl} and DR_{sym} reach comparable performance in ADC1 and ADC2.

To guarantee the layout quality and cover more comprehensive sign-off checks, we also perform simulations over process corners (i.e., TT, FF, SS) and Monte-Carlo (MC) sampling. Table 5 shows the process corners simulation results of the ADC2 layout generated by DR_{sym}, observe that the performance variation of the ADC2 layout on different process corners is within a stable range. Figure 11 shows the distribution of performance metrics of ADC2 by MC sampling over 20 points. The simulation results show that our layouts withstand sign-off checks. In fact, the ADC2 chip implemented using TSMC 40nm technology has been sent to TSMC for tape-out where the routing is 100% automatically generated by this work.

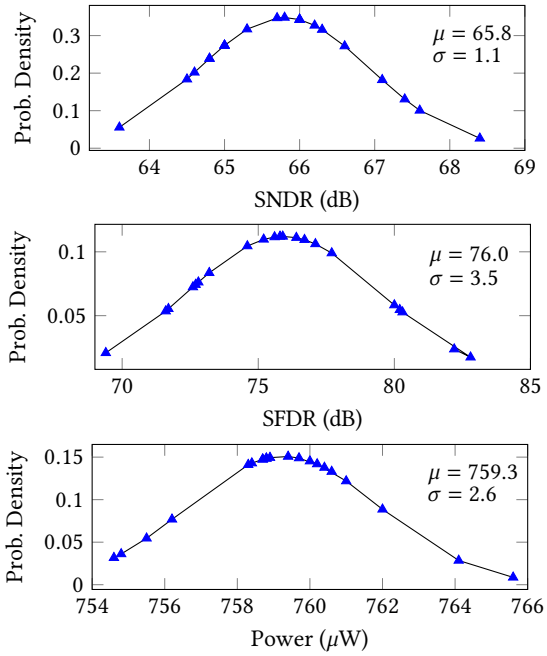


Figure 11: Monte-Carlo sampling results over 20 points of the ADC2 layout generated by DR_{sym}.

5 CONCLUSION

This work has presented an end-to-end hierarchical analog/mixed-signal detailed routing framework. A graph-based symmetry constraint allocation algorithm has been proposed to identify symmetry constraints accurately. A pin access assignment method has been shown to reduce the routing complexity. A symmetry-driven rip-up and re-route scheme with pin clustering has been proposed to consider real-world design rules and analog-specific constraints. Experimental results have demonstrated the efficiency and effectiveness of the proposed analog routing framework in producing sign-off quality routing solutions.

ACKNOWLEDGEMENT

This work is supported in part by the NSF under Grant No. 1704758, and the DARPA IDEA program.

REFERENCES

[1] E. Chang, J. Han, W. Bae, Z. Wang, N. Narevsky, B. NikoliC, and E. Alon. 2018. BAG2: A process-portable framework for generator-based AMS circuit design. In *Proc. CICC*. 1–8.
[2] U. Choudhury and A. Sangiovanni-Vincentelli. 1990. Constraint generation for routing analog circuits. In *Proc. DAC*. 561–566.
[3] U. Choudhury and A. Sangiovanni-Vincentelli. 1993. Constraint-based channel routing for analog and mixed analog/digital circuits. *IEEE TCAD* 12, 4 (1993), 497–510.
[4] J. M. Cohn. 2000. *Analog Device-Level Layout Automation* (1 ed.). Kluwer Academic Publishers, USA.
[5] J. Crossley, A. Puggelli, H. . Le, B. Yang, R. Nancollas, K. Jung, L. Kong, N. Narevsky, Y. Lu, N. Sutardja, E. J. An, A. L. Sangiovanni-Vincentelli, and E. Alon. 2013. BAG: A designer-oriented integrated framework for the development of AMS circuit generators. In *Proc. ICCAD*. 74–81.
[6] B. Dezso, A. Jüttner, and P. Kovács. 2011. LEMON - An open source C++ graph template library. *Electronic Notes in Theoretical Computer Science* 264 (2011), 23–45.
[7] J. Edmonds. 1965. Paths, Trees, and Flowers. *Canadian Journal of Mathematics* 17 (1965), 449–467.
[8] A. Hastings and R. A. Hastings. 2001. *The Art of Analog Layout* (1 ed.). Prentice Hall, USA.
[9] A. B. Kahng, L. Wang, and B. Xu. 2018. TritonRoute: An Initial Detailed Router for Advanced VLSI Technologies. In *Proc. ICCAD*. 1–8.
[10] H. Li, G. Chen, B. Jiang, J. Chen, and E. F. Y. Young. 2019. Dr. CU 2.0: A Scalable Detailed Routing Framework with Correct-by-Construction Design Rule Satisfaction. In *Proc. ICCAD*. 1–7.
[11] M. Liu, W. Li, K. Zhu, B. Xu, Y. Lin, L. Shen, X. Tang, N. Sun, and D. Z. Pan. 2020. S^3DET: Detecting System Symmetry Constraints for Analog Circuits with Graph Similarity. In *Proc. ASPDAC*. 193–198.
[12] E. Malavasi, E. Charbon, E. Felt, and A. Sangiovanni-Vincentelli. 1996. Automation of IC layout with analog constraints. *IEEE TCAD* 15, 8 (1996), 923–942.
[13] L. McMurchie and C. Ebeling. 1995. PathFinder: A Negotiation-Based Performance-Driven Router for FPGAs. In *Proc. FPGA*. 111–117.
[14] M. Mustafa Ozdal and M. D. F. Wong. 2006. A Length-Matching Routing Algorithm for High-Performance Printed Circuit Boards. *IEEE TCAD* 25, 12 (2006), 2784–2794.
[15] H. Ou, H. C. Chien, and Y. Chang. 2014. Nonuniform Multilevel Analog Routing With Matching Constraints. *IEEE TCAD* 33, 12 (2014), 1942–1954.
[16] M. M. Ozdal and R. F. Hentschke. 2012. Maze routing algorithms with exact matching constraints for analog and mixed signal designs. In *Proc. ICCAD*. 130–136.
[17] P. Pan, H. Chen, Y. Cheng, J. Liu, and W. Hu. 2012. Configurable analog routing methodology via technology and design constraint unification. In *Proc. ICCAD*. 620–626.
[18] B. Razavi. 2001. *Design of Analog CMOS Integrated Circuits* (1 ed.). McGraw-Hill, Inc., USA.
[19] B. Schaeling. 2014. *The Boost C++ Libraries* (2 ed.). XML Press, USA.
[20] F. Sun, H. Chen, C. Chen, C. Hsu, and Y. Chang. 2018. A Multithreaded Initial Detailed Routing Algorithm Considering Global Routing Guides. In *Proc. ICCAD*. 1–7.
[21] L. Xiao, E. F. Y. Young, X. He, and K. P. Pun. 2010. Practical placement and routing techniques for analog circuit designs. In *Proc. ICCAD*. 675–679.
[22] T. Yan and M. D. F. Wong. 2008. BSG-Route: A length-matching router for general topology. In *Proc. ICCAD*. 499–505.
[23] K. Zhu, M. Liu, Y. Lin, B. Xu, S. Li, X. Tang, N. Sun, and D. Z. Pan. 2019. GeniusRoute: A New Analog Routing Paradigm Using Generative Neural Network Guidance. In *Proc. ICCAD*. 1–8.