REVIEWS

Journal of Semiconductors
(2020) 41, 111407
doi: 10.1088/1674-4926/41/11/111407

Challenges and opportunities toward fully automated analog layout design

Hao Chen‡, Mingjie Liu‡, Xiyuan Tang‡, Keren Zhu‡, Nan Sun, and David Z. Pan†

Department of Electrical and Computer Engineering, The University of Texas at Austin, Austin 78712, USA

Abstract: Realizing the layouts of analog/mixed-signal (AMS) integrated circuits (ICs) is a complicated task due to the high design flexibility and sensitive circuit performance. Compared with the advancements of digital IC layout automation, analog IC layout design is still heavily manual, which leads to a more time-consuming and error-prone process. In recent years, significant progress has been made in automated analog layout design with emerging of several open-source frameworks. This paper firstly reviews the existing state-of-the art AMS layout synthesis frameworks with focus on the different approaches and their individual challenges. We then present recent research trends and opportunities in the field. Finally, we summaries the paper with open questions and future directions for fully-automating the analog IC layout.

Key words: VLSI; integrated circuit; ASIC; EDA; analog ic design; physical design

Citation: H Chen, M J Liu, X Y Tang, K R Zhu, N Sun, and David Z. Pan, Challenges and opportunities toward fully automated analog layout design[J]. *J. Semicond.*, 2020, 41(11), 111407. http://doi.org/10.1088/1674-4926/41/11/111407

1. Introduction

The demand for analog/mixed-signal (AMS) integrated circuits (ICs) has been increasing in various emerging applications, including the internet of things (IoT), 5G networks, advanced computing, and healthcare electronics. Therefore, a short turnaround time of analog IC design is desired. However, while the automation algorithms have been significantly improved over the past decades, implementing analog layout is still a manual, time-consuming, and error-prone task.

Despite the continuous efforts in analog layout automation, those achievements have not been well adopted in current industrial flows. The main reason is rooted in the characteristics of AMS circuits. AMS IC design is a complex task due to the high design flexibility. Compared to its digital counterpart, AMS design is considered complicated since it deals with a wide range of specific circuit classes, various device types, and requires a customized tuning for each circuit class. Besides, analog layouts are sensitive to signal couplings, layout-dependent effects, and process variations. The circuit performance could suffer from significant degradation with minor changes in the layout implementation. Furthermore, there lacks an effective way to model the layout effects on analog performance, which imposes significant challenges for automation tools.

While analog electronic design automation (EDA) tools are far behind their digital counterparts, the endeavor to automate analog layout design can be traced back to decades before. One of the early analog EDA efforts, ILAC[1], already has similar methodologies to some current AMS layout frameworks. ILAC contains two separate layout generators that can generate the layouts of different circuit submodules, and the two generators will interact with each other to produce the final layout. The first layout generator, STUCCO, is a procedure-based layout generator that stores pre-defined layout templates of some fundamental analog circuit building blocks and generates the layouts for different manufacturing nodes. The types of building blocks handled by STUCCO range from MOS transistors to current mirrors. The second layout generator, MOSAIC, is a general-purpose optimization-based layout generator that can take the STUCCO outputs as its inputs. MOSAIC will first place the blocks using the simulated annealing algorithm and then complete the circuit interconnections. The two analog layout generators in ILAC has laid the foundations of two mainstream analog layout automation approaches: procedure-based and optimization-based.

Promising advancements have been made in recent years. Strategies for designing analog circuits that are more synthesis-friendly have been proposed. Frameworks utilizing procedure-based layout techniques have been demonstrated. Also, optimization-based layout generation methods have been presented.

This paper aims to overview current AMS layout automation approaches and research trends and new visions on analog layout automation. Specifically, we will first review three primary AMS layout methodologies and related frameworks in Section 2. We then introduce several recent advances in academia with an emphasis on algorithms featuring machine learning (ML) and statistics in Section 3. We will finally present our visions on the open questions and trends in future EDA development for AMS layouts.

2. Current approaches and challenges

People have proposed different paradigms with various degrees of generality for different usage scenarios. However, introducing design-dependent strategies and prior knowledge in the automation flow often benefit the flow effective-

Hao Chen, Mingjie Liu, Xiyuan Tang, and Keren Zhu are in alphabetic order.
Correspondence to: David Z. Pan, dpan@ece.utexas.edu

Received 18 JULY 2020; Revised 22 SEPTEMBER 2020.

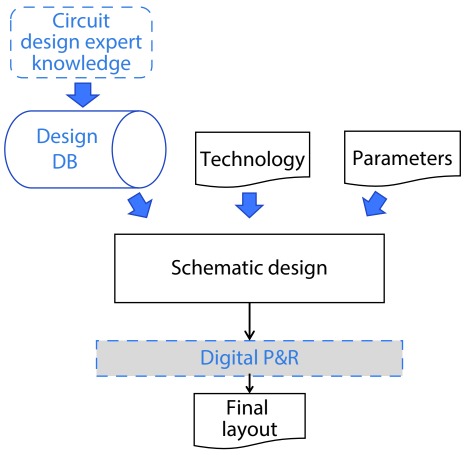


Fig. 1. (Color online) Synthesis-friendly analog circuits design flow.

ness in the cost of jeopardizing the generality of flow. In this section, we review three popular paradigms in the order of increasing generality.

(1) Synthesis-friendly AMS circuits: Design highly digital and modularized AMS circuits that are suitable for commercially available digital physical design tools.

(2) Procedure-based layout generation: Generate layouts based on the pre-designed templates in a procedural approach.

(3) Optimization-based layout synthesis: Formulate the layout generation as a constrained optimization problem and tends to model the layout quality as the objectives.

2.1. Synthesis-friendly analog circuits

In the past decade, enormous efforts have been devoted to improving the AMS circuits synthesizability. In the typical synthesis-friendly design flow, as shown in Fig. 1, a design database, including synthesizable analog blocks and architectures, is provided by the designer. By combining it with digital physical design methodologies, this flow aims to generate layouts using commercially available digital auto place and route (APR) tools, thus significantly improving design efficiency. This methodology leaves significant efforts in the design database generation, where highly-digital and modularized circuits are desirable.

One direction is to design analog circuits using only logic gates. This way, they directly fit the existing digital synthesis flow. Stochastic flash analog-to-digital converter (ADC) pioneered this direction[2]. Instead of combating the circuit offsets caused by random mismatch as in conventional AMS circuit designs, it utilizes this inherent offset as a distributed reference generator for a flash ADC. As a result, it only requires the custom-designed comparators besides digital standard cells. In Refs. [3, 4], the comparator is implemented by the logic gates. The fully digital architecture allows fast synthesis and provides sufficient programmability. Although it showcased the implementation of a fully-synthesized flash ADC, its complexity is quadrupled for each extra bit, restricting the suitability for only low to medium resolution (e.g., < 6 b). Besides, this fully digital approach is only suitable for a few specific circuit architectures.

Researchers also explored to build analog standard cells to assist AMS circuit synthesis. In Ref. [5], a synthesized multi-stage noise-shaping (MASH) sigma–delta ($\Sigma\Delta$) modulator is reported. It is made possible by creating an analog library, including opamp, comparator, transmission gate, and unit capacitor. Each analog cell, including a custom-optimized layout, is treated as a digital standard cell by the APR engine. A synthesized current-steering DAC, is demonstrated in Ref. [6], where the current cells are individually designed and then APRed. These works demonstrate remarkable Verilog-to digital-gate based analog cells are developed to reduce manual efforts for each design. A synthesizable biquad filter reported in Ref. [7] achieves state-of-the-art linearity, where the amplifier is implemented by NAND, NOR, and INV logic gates. High programmability is provided over the gain, bandwidth, and input-referred noise to provide reconfigurability. Still, the usage of this specially designed analog cell is limited to specific architectures – layout synthesis capability. However, they still rely on the pre-optimized analog cells, which have to be developed separately for each specific design or technology node. Hence, they are difficult to generalize to other design specs or circuit types.

Digital-gate based analog cells are developed to reduce manual efforts for each design. A synthesizable biquad filter reported in Ref. [7] achieves state-of-the-art linearity, where the amplifier is implemented by NAND, NOR, and INV logic gates. High programmability is provided over the gain, bandwidth, and input-referred noise to provide reconfigurability. Still, the usage of this specially designed analog cell is limited to specific architectures.

SAR architecture draws attention in recent synthesizable ADC developments owing to the highly-digital nature. A highly automated SAR ADC design is demonstrated in Ref. [8]. Nevertheless, a dedicated custom-made capacitor DAC layout tool is required on top of the standard digital EDA tool. There is still a performance gap between the synthesized SAR ADC and the manually optimized one.

While the conventional voltage-domain analog circuit is challenged by the sub-par analog accuracy in standard cells and APR, time-domain analog signal processing shows promising synthesis friendliness. By representing signals using time-related variables, such as frequency and delay, it allows the circuit to be mostly digital by nature. As a result, time-domain AMS circuits are inherently suitable for synthesis.

A domino-logic ADC is reported in Ref. [9], where the input voltage controls a domino cell chain's delay. The input is quantized by counting the number of cells that the trigger signal propagates through. The highly modularized typology allows the fully automated synthesis of this type of ADC. However, the resolution is still limited to 6-bit because of the inherent nonlinearity of the delay chain.

Recent advancements of all-digital phase-locked loops (PLLs) benefit substantially from the digital-friendly nature of time-domain processing. Consequently, synthesized all-digital PLLs have shown performance comparable to state-of-the-arts[10, 11]. An all-digital low-dropout regulator (LDO) is implemented in Ref. [12], where the time-to-digital converter (TDC) is adopted for the digital supply voltage sensor, thus achieving fully synthesizability[12]. A fully-synthesized transmitter is reported in Ref. [13]. Thanks to the time-domain signal processing, it only requires digital standard cells and features a wide configuration range.

Inspired by the time-domain PLLs, fully synthesized VCO-based continuous-time $\Delta\Sigma$ modulators (CTDSMs) are repor-

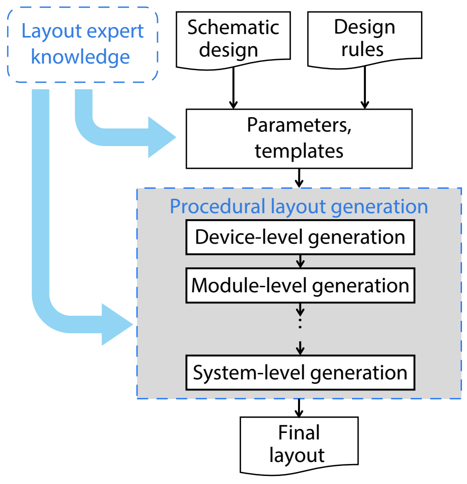


Fig. 2. (Color online) Procedural analog circuits layout design flow.

ted in Refs. [14, 15]. The analog input is converted into phase and frequency. Then, the quantization is performed by XOR gates instead of conventional voltage domain comparators. This architecture provides noise shaping capability and upconverts component mismatches, thus relaxing APR requirements. As a result, solely implemented by foundry digital standard cells and resistors, it demonstrates resolution beyond 11-ENOB with state-of-the-art energy efficiency, while maintaining a high automation level.

FASoc has carried it further[16, 17]. Many standard analog cells are created to assist the AMS synthesis, such as bootstrap switches, unit capacitor cells, and comparators. The analog cells leverage the latest advancements in AMS designs, where the highly digital implementations are adopted. For example, instead of the conventional Strong-Arm latch of Ref. [18], the comparator adopts the edge pursuit architecture where only NAND and INV gates are required[19]. The same philosophy also applies to target AMS circuit topology choices. A good example is the capacitance-to-digital converter design. FASoc adopts a delay chain based implementation of Ref. [20] that avoids the intensive analog blocks as required in the conventional designs. As a result, this tool can support many pre-defined AMS circuits with a wide range of programmability, including all-digital PLLs, power management, ADCs, and sensor interfaces.

As a final remark, although researchers have proposed encouraging solutions in the past decade, they still lack general applicability since detailed optimizations are required for each specific topology or performance metric.

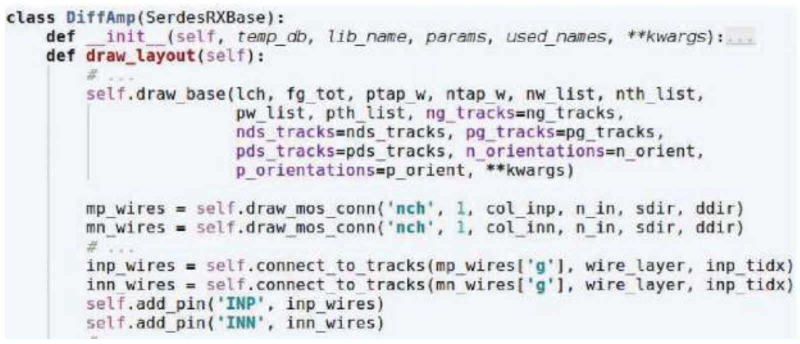
2.2. Procedural layout generation

As mentioned in Section 1, procedural layout generation is often used to generate modules[1, 21–24]. Procedural layout generators implement circuit layout by combining common primary devices and structures using module generators. These module generators usually generate layout structures according to pre-defined rules or template layout designs. A similar approach is adopted later for AMS circuits with a larger scale. A typical procedural layout generation flow, as shown in Fig. 2, embedding layout expert knowledge in the layout templates. The procedures are designed and programmed by layout experts.

Procedural layout generators date back to as early as the works of ILAC[1] and SLAM[21]. ILAC layout engine uses a procedural block layout generator, which supports a wide variety of circuit substructures such as current mirror, differential pair, interdigit resistance, capacitance. The layout generator assures good quality layout by "copying" proven "hand" layouts in a parameterizable way. Similarly, SLAM uses a set of parametric analog layout circuit primitives and proposes a method to detect the netlist's circuit primitives automatically. Stefanovic *et al.*[22] present a chart-based design environment that leverages a basic analog structure library for fast layout generation and parasitic estimation. Youssef *et al.*[23] developed a Python-based layout generation tool for primitive building blocks to assist designers in exploring electrical and physical trade-offs. Lopez *et al.*[25] further integrates the layout generator inside an optimization loop for parasitic-aware circuit sizing. Han *et al.*[26] developed a GUI-based template engine, significantly reducing the effort to generate DRC/LVS-clean layouts for high-speed serial link designs. Ding *et al.*[27] proposes a comprehensive framework, combining digital P&R flow with template-based libraries, to generate layouts for SAR ADCs.

Recent works on procedural layout generators have also demonstrated the capability of producing results verified by tape-out measurements. Wulff and Ytterdal[28] developed a Perl script based compiler that compiles a core SAR ADC into GDSII in a few seconds and reduces the effort necessary to port design to another technology. A proof-of-concept fabricated in 28-nm FOSOI achieves the state-of-the-art performance. Berkeley layout generator (BAG)[24] implements an interface to integrate all design flow steps into a single environment to aid the designer in developing truly parameterized and technology-independent circuit generators. Their extended work BAG2[29] further extended the original framework, while presenting tape-out verified designs of a time-interleaved SAR ADC and a SerDes transceiver frontend. Fig. 3 is an example of the layout script written by designers and its corresponding layout. Instead of manually drawing the layout, designers only need to codify high-level descriptive layout implementation decisions, where the layout generator engine handles the specific design rules. This process simplifies the codification of common tasks and foster design reuse, technology migration, and shortens time-to-market while remaining close to classical design flows for easy adoption. It also greatly eases the entire design steps from schematic circuit topology, transistor sizing, layout generation, and design verification, while the high-level layout descriptions are easily portable to various technology nodes. BAG also has several extended work[30, 31] that leverage the layout engine and machine learning algorithms, such as evolutionary algorithms and reinforcement learning, for automated transistor sizing that could be transferable across designs and technology nodes.

However, procedural layout generators still have a significant amount of manual labor involved in developing parametric cells and incorporating scripts to codify device placement and routing. Nevertheless, these approaches have demonstrated a significant reduction in manual labor in design migration to different technologies. There have also been several works in reducing the amount of manual overhead of procedural generators in layout migration[32–36]. Analog layout retargeting and technology migrations usually extract placement and routing features from existing manual layout and transfer the patterns and cope with new design rules in the target



```
class DiffAmp(SerdesRXBase):
    def __init__(self, temp_db, lib_name, params, used_names, **kwargs):...
    def draw_layout(self):
        # ...
        self.draw_base(lch, fg_tot, ptap_w, ntap_w, nw_list, nth_list,
                       pw_list, pth_list, ng_tracks=ng_tracks,
                       nds_tracks=nds_tracks, pg_tracks=pg_tracks,
                       pds_tracks=pds_tracks, n_orientations=n_orient,
                       p_orientations=p_orient, **kwargs)

        mp_wires = self.draw_mos_conn('nch', 1, col_inp, n_in, sdir, ddir)
        mn_wires = self.draw_mos_conn('nch', 1, col_inn, n_in, sdir, ddir)
        # ...
        inp_wires = self.connect_to_tracks(mp_wires['g'], wire_layer, inp_tidx)
        inn_wires = self.connect_to_tracks(mn_wires['g'], wire_layer, inp_tidx)
        self.add_pin('INP', inp_wires)
        self.add_pin('INN', inn_wires)
```

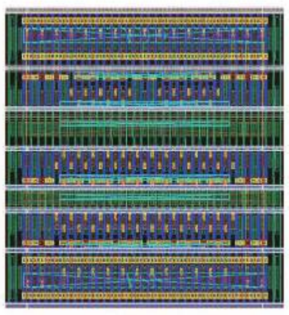


Fig. 3. (Color online) An example of the Python script and generated layout of BAG2[16].

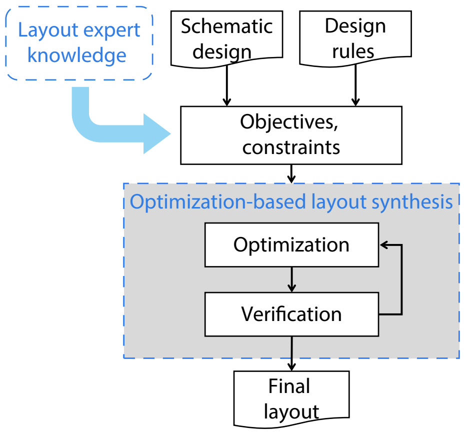


Fig. 4. (Color online) Optimization-based layout design flow.

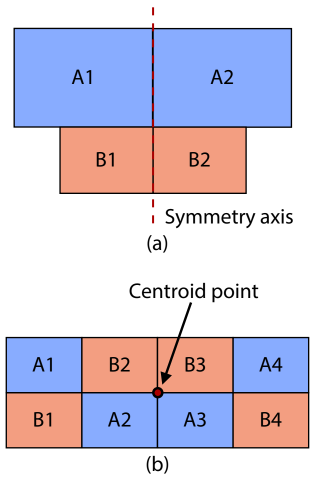


Fig. 5. (Color online) Examples of placement constraints. (a) Symmetry constraint. (b) Common-centroid constraint.

technology. These works could be seen as a hybrid approach between procedural and optimization based layout generators since they generally involve optimization for reduced area and wirelength, but still have limited capabilities in handling a wide variety of circuits.

2.3. Optimization-based layout synthesis

Another popular paradigm is to cast the layout generation into optimization problems. A typical flow for optimization-based layout synthesis, as shown in Fig. 4, formulates layout considerations as parameters or constraints defined by users. The layout synthesis is through an optimization engine without manual intervention. The high-quality layout is achieved by optimizing specific objectives, such as wirelength and area, under a set of constraints, such as symmetry between devices and design rules. This methodology is similar to digital EDA tools and often borrow ideas from them. It is top-down automation because the layout expert knowledge is applied in formulating the constraints and objectives[37].

Like digital EDA tools, optimization-based analog layout generation often separates the process into several stages for divide-and-conquer. A common practice is to have three stages, module generation, placement, and routing. The module generator generates the layout of building blocks in a parameterized manner. The placement stage then places the layout from the module generator. In the end, the routing stage connects the nets through metal wires and VIAs.

Module generator is an essential component in optimization-based flow as it provides the layouts of fundamental building blocks, e.g., transistors and resistors. Modern industry custom layout tools, such as Cadence Virtuoso, provide a parameterized device generator (PCells) to produce primary devices' layout. In academia, several frameworks are using customized device generators, such as ALIGN[38] and MAGICAL[39]. Procedural layout generation can be viewed as a more general module generator. Procedural analog layout generators, such as BAG[25], are also capable of generating parameterized device layout.

Placement takes the basic building block layouts as input and places them on the layout. Unlike the digital placement problem, the scale of AMS placement is often much smaller in the number of modules and more stringent requirements limited by layout-sensitive performance. A pivotal question to AMS placement problem formulation is how to correlate the placement optimization objectives with the layout quality. A well-adopted approach is to formulate the placement problem into a constrained optimization problem. Constraints are imposed to restrict the automated placement to established manual layout design patterns, and objectives encourage the solutions with a smaller area, shorter wirelength, etc. The most-widely-used type of such geometric constraint is the symmetry constraint. Symmetry constraint restricts the pairs of modules to be placed along with one or several symmetric axes. This formulation is mimicking similar techniques in manual layout designs. Symmetry constraint has been widely used in Refs. [1, 40–70]. Similarly, common-centroid is adopted in analog placement problem[50, 52, 53]. Fig. 5 shows examples of symmetry constraint and common-centroid constraint. There are other works focusing on regular array-like structure in layouts[54], monotonic current flow from VDD to VSS[62, 63, 68], thermal effects[48], system signal flow[71] and etc.

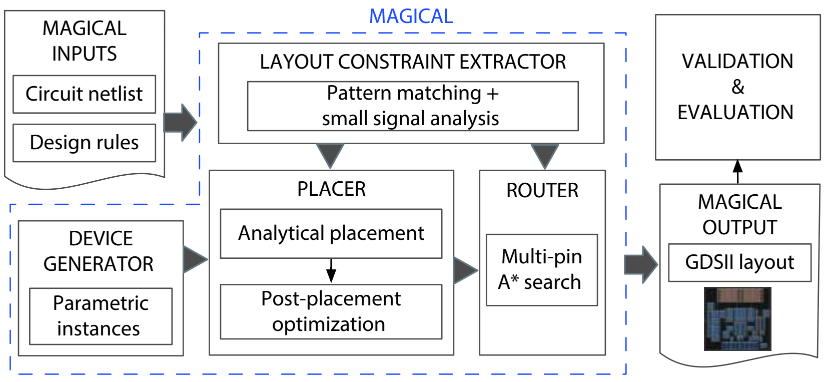


Fig. 6. The overall flow of MAGICAL[39].

Unlike the digital flow, some existing analog placement framework considers soft modules, for example, merging the transistors[72].

A recent trend in literature for analog placement is to explore methods for performance-driven optimization. In the work[71], the authors use the regularity of the system signal flow as one of the optimization objectives. Li *et al.*[73], on the other hand, proposes to build a model for estimating the parasitics in placement and optimizes the estimated parasitics for performance. Recent advances in machine learning also encourage the research to optimize performance directly through neural network-based performance prediction models[74]. More details about the performance prediction are presented in Section 3.1.

Routing takes the placed layout and connects the nets with metal wires and VIAs. Analog routing usually is similar to digital routing with additional considerations on symmetry constraints[75]. Some other considerations include avoiding routing over active region[76], optimizing power routing[77] and etc. Recently, Chen *et al.*[78] propose to not only consider the given symmetric constraints but also optimize for the total symmetry across the design. They present a new algorithm to match the pins in the layouts and route the nets in symmetry. Chen *et al.* also study the challenge for tape-ready analog detailed routing. Equipped with sophisticated design-rule handling schemes, the proposed detailed router can reach "tape-ready" quality in terms of the design rule violation clean and post-layout simulation.

Recently, there are rising interests in fully automated or "no-human-in-the-loop" physical design flow[38, 39, 79]. The aforementioned optimization-based frameworks usually regard the constraints; for example, symmetry pairs, are user-defined inputs. However, labeling the constraints itself is time-consuming and tedious. Optimization-based AMS layout generation frameworks also sometimes embed the fully automated constraint generation in the flow. Although the automatic constraint generation algorithm has not been mature and comprehensive yet, it has been adopted in some current frameworks. Fig. 6 shows the overall flow of the MAGICAL framework. Starting from unannotated schematic netlists, it extracts the symmetry constraints and generates primitive devices. Then it places the devices and routes the interconnection. In Section 3.3, a review on current constraint generation algorithms is presented.

In the existing optimization-based AMS layout automation framework, several challenges are remaining unresolved.

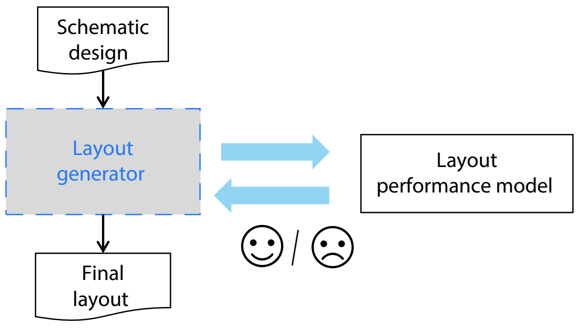


Fig. 7. (Color online) Layout generator with layout performance prediction.

First, there yet lacks a principal method in formulating the optimization problems. The existing geometric constraints are distilled from manual layout heuristics, which are design-dependent and technology-dependent[80]. Although some constraints, such as symmetry, are usually regarded as universal design guidelines, many manual layout behaviors are questionable to be suitable for implying as a hard geometric constraint. Second, it takes efforts to label the correct constraints by hand. Although there existing works for detecting symmetry constraints automatically[81], the scalability and generality of these algorithms is concerned. Furthermore, few studies are working on more dedicated geometric constraints. Third, there lacks a direct approach to optimize circuit performance. Existing work on optimization-based analog place and route is still relying on the objectives such as wirelength and area. However, such optimization objectives are questionable as to whether the circuit performance is well-reflected. Finally, it is not easy to compare different frameworks. Layout quality is often evaluated through post-layout simulation. However, as the post-layout simulation requires fully functional layouts, the extensive efforts on fundamental building blocks such as design rule handling and module generator make individual research focus on the abstract level problem instead of real circuit performance.

3. Recent research trends

In this section, we introduce several recent efforts and present our view on the research trend in this area.

3.1. Layout performance prediction

A significant challenge in automatic AMS layout synthesis is the lack of an effective method to model the layout effects on circuit performance. Fig. 7 shows a potential flow of

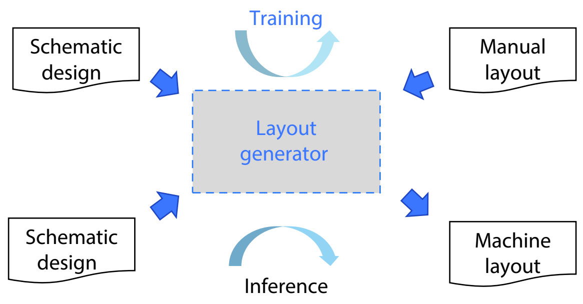


Fig. 8. (Color online) A flow for learning knowledge from manual layouts.

performance prediction-assisted automated layout generation flow. A performance modeling can give feedback to automatic layout generation and guide the tool to generate high-performance layouts.

The authors of the work[82] intend to tackle this challenge with the computer vision method. A new problem of predicting post-layout performance is formulated. The authors generate many different placements for the same circuit, and the placement is routed with an automatic routing engine. Post-layout simulations are performed on the routed layouts. The placement results and the corresponding simulation results are used for training data for CNN ML models. The ML model can then predict the circuit performance from placement and bridge the modeling gap between the placement and post-layout simulation.

In Ref. [82], a preliminary attempt to model the circuit performance from layouts is presented. The idea is further extended in the work[74], where a graph neural network incorporating the connectivity information from the circuit netlist is utilized to improve the accuracy. Li *et al.*[74] also proposes to use the performance prediction as optimization objectives in a simulated-annealing-based analog placer. However, it is limited to a small range of circuit types, and the existing techniques to integrate the performance modeling in automatic optimization is still immature. Our vision is that the ML-based performance modeling will help develop AMS layout synthesis by introducing a more direct objective in optimization-based place and route.

3.2. Leveraging human knowledge from existing layouts

Another trend of applying ML to AMS layout is to leverage ML to learn the manual layout techniques from existing layouts. Fig. 8 shows a potential flow for automatically learning from manual layouts. Using human layouts are golden examples, a machine learning model can learn the strategies in designing layouts (the training phase) and use the learned model for new designs (the inference phase).

Generating wells and inverting contacts are often customized-designed in manual layouts. On the other hand, existing AMS layout synthesis frameworks often integrate the well generation in module generation. For example, MAGICAL generates separate NWELL contacts for each PMOS device[39]. The paradigm influences this digital place and route flow strategy, where wells are designed in standard cells, and contacts are inserted to standardized row-based placement methodology. However, as analog circuit performance is sensitive to layout and individual NWELL contacts strategy introduces overhead on area, a human-like well generation is desirable. WellGAN[83] proposes to generate the wells by mimicking the manual layout behaviors automatically. It formulates the well generation problem into a computer vision task and learns how wells are designed in manual layouts with supervised learning. The manual layouts are used as training data, and the wells drawn in those layouts are extracted to be the ground truth for the ML model. A generative adversarial network (GAN) predicts where human engineers draw the wells from the placement layouts. After training, the GAN model can generate images of wells based on placement. The generated well images are further legalized and inserted into the layout.

A similar idea is used in analog routing. Current analog routing in optimization-based AMS layout synthesis is usually a constrained optimization problem. Manual layout techniques are imposed as constraints, and analog routers usually optimize for the wirelength and area. However, it raises whether this constraint-driven methodology is suitable for a more extensive range of circuits. GeniusRoute[80] proposes an alternative. Similar to WellGAN, GeniusRoute uses a generative ML model to learn manual layout techniques. It predicts where a human designer from the placement likely routes the nets. The predicted routing region is then fed to an automatic routing engine as routing guidance. A variational autoencoder (VAE) is used similar to the GAN model in WellGAN. Fig. 9 shows the flow of GeniusRoute. The routing patterns are extracted from existing manual layouts and used to train a VAE model in the training phase. In the inference phase, the trained VAE is used to predict the routing region with unrouted placement.

In both cases, supervised generative learning is automatically used to learn the manual layout techniques from existing designs. To some degree, it relieves the efforts to design heuristics in automatic AMS place and route algorithms and provides a new approach to learn human knowledge from existing layouts.

3.3. Automatic constraint generation

AMS CAD flows rely on designers to provide layout constraints that are honored during layout implementation. These constraints are often design-dependent and related to performance-sensitive layout effects. Common constraint examples are device proximity and dummy insertions during device placement and increased wire spacing for reduced net coupling during routing. The analog circuits' performance is often closely related to these detailed layout implementations, and layout dependant effects could significantly degrade circuit performance and even change circuit functionality. Thus by enforcing these constraints during layout implementation, the designers hope to mitigate parasitic and layout dependent effects on the final post-layout circuit performance.

Symmetry constraints are one of the most essential and widely adopted constraints applied during analog layout synthesis. Analog designs frequently use differential topologies to reject common-mode noise and enhance circuit robustness. Mismatch in the devices of these topologies would significantly degrade the circuit performance. In layout designs, these devices need to be placed and routed symmetrically to enforce matching. Although most layout heuristic con-

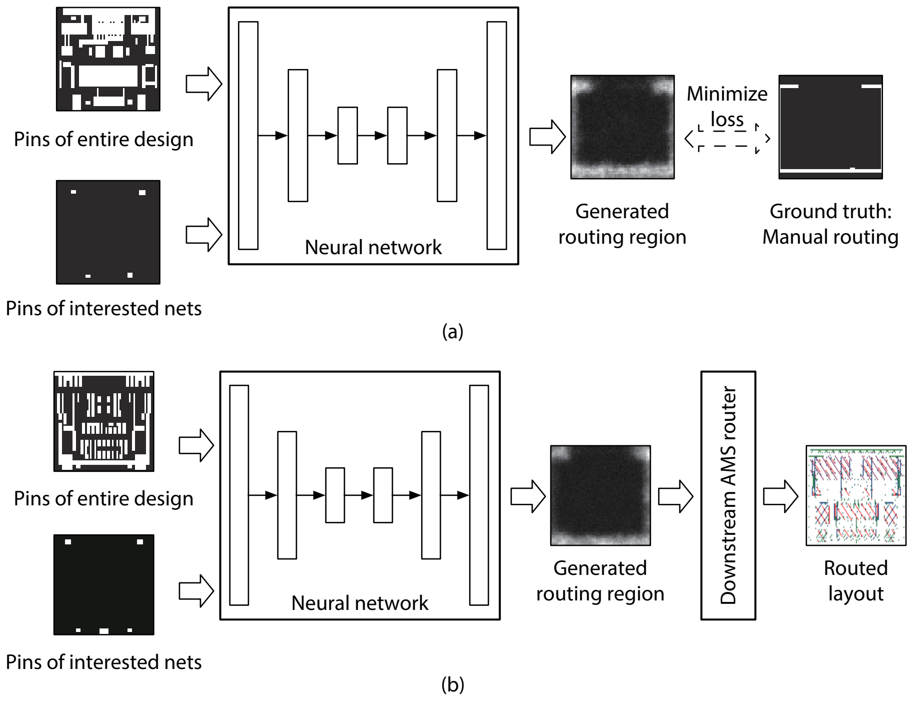


Fig. 9. GeniusRoute framwork. (a) Training phase. (b) Inference phase[80].

straints are design-dependent, several attempts to automatically extract symmetry constraints from the design netlist.

Existing works for symmetry constraint detection can be categorized into two major types: 1) circuit analysis and 2) graph matching. Charbon *et al.*[84] use sensitivity analysis to identify symmetry and matching constraints by circuit simulations. This approach is useful in extracting high-quality and critical matching constraints directly related to performance. Another type of algorithm uses graph matching. This algorithm represents circuits as different types of graphs and uses various techniques to convert symmetry detection into graph matching. The works of Refs. [85, 86] convert analog circuits into bipartite graphs with signal flow analysis and use graph automorphism to detect graph symmetry. Wu *et al.*[87] further extend the graph representation by embedding pin connection information into edge weights and generate constraints based on subgraph isomorphism with a template circuit library. Eick *et al.*[88] directly generate constraints by pattern matching and group constraints hierarchically based on structural signal flow graphs.

Recent works have improved prior approaches and scaling to larger system-level designs. Liu *et al.*[81] develops a fast graph similarity-based approach to detect system symmetry constraint between circuit building blocks. This idea is further extended to use graph neural network with supervised learning techniques[89]. The network is trained to predict the graph edit distance as a metric for measuring the graph similarity. Kunal *et al.*[90] proposes a graph learning-based approach to first classify circuit blocks and further identify circuit primitives, significantly improving prior approaches' scalability to extensive designs.

3.4. AMS CAD with in-loop-simulation

Incorporating simulation in AMS CAD is not a new idea. In early work[84], schematic simulations are used in the sensitivity analysis to detect the symmetry constraints. Recently,

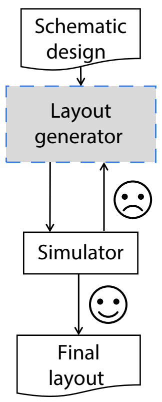


Fig. 10. A flow with in-loop simulation.

post-layout simulations are used in automatic AMS CAD flows. Fig. 10 shows a typical flow of incorporating simulations in the loop. After an automatic layout generator produces a layout, simulation is used to decide whether the post-layout performance meets the design specifications. Then the layout generator can further optimize or change the layout until it passes the verification of simulations.

The work[91] proposes to integrate simulations in an automatic place and route for AMS circuits. For hierarchical design, it iteratively generates the layouts for low-level analog building blocks using MAGICAL and executes post-layout simulations to obtain circuit performance. Then bayesian optimization is utilized to update the layout generation engine parameters until a layout with satisfying performance is obtained. Similar in-loop performance verification is also proposed in the work[28], where an automatic layout generation flow is explicitly designed for SAR ADCs. This methodology can ensure the building blocks performance and also benefits the performance of the high-level mixed-signal system. However, as the system-level simulation is time-consuming, there yet lacks a similarly effective way to ensure the system-level layout quality.

The recent development of AMS layout synthesis frameworks, such as ALIGN, BAG, and MAGICAL, provides the essential to integrate layout knowledge to research problems.

4. Open questions in analog layout automation

Amid recent fast development of automatic AMS layout synthesis, several open questions and challenges are yet to be solved. This section presents our view on future directions on research in automating AMS layout design.

4.1. Open-source efforts on comprehensive layout synthesis flow

It takes considerable efforts to compose a complete layout synthesis flow from fundamental devices to detailed routing[38, 92, 93]. This significant development overhead discourages the research in performance-driven AMS layout synthesis. Fortunately, recent advances in open-source AMS layout generation tools relieve such issues. ALIGN, BAG and MAGICAL have provided three different approaches, all with end-to-end netlist-to-GDS layout generation flow. These three frameworks choose Python as a top-level programming language that is flexible and suitable for fast prototyping and software development. We expect an increasing amount of research to leverage the open-source tools and focus on performance-driven layout generation verified by post-layout simulations.

However, the open-source environment is not yet mature. For BAG, layout design needs manual efforts, and the open-source layout design templates are relatively limited in number. On the other hand, both ALIGN and MAGICAL have supported selected technologies in the current public version. A major challenge is on the development overhead of transferring tools to other technologies. For example, the design rule and device layouts are different in different technologies.

An active open-source community also enables collaborations among different projects. In the current open-source AMS layout generator tools, some efforts are repetitive and overlapping with each other. For example, all of the ALIGN, BAG, and MAGICAL have implemented the device generators. A possible approach is to standardize the interface and file formats between the tools. For example, similar to digital, physical design flow, the open-source AMS layout synthesis tools can use standardized exchangeable file formats to enable the integrated usage of different tools. We believe it is the future trend of the open-source community for AMS layout synthesis.

4.2. Sign-off-quality layout generation

For optimization-based approaches, an open question is on how to obtain sign-off quality layouts. In other words, how to fill in the current gap between optimization to real circuit performance.

A potential direction is to use the ML method to model the layout effects on circuit performance. Preliminary research is surveyed in Section 3.1. However, this work has not answered the question of how to optimize the performance directly. Even with accurate performance modeling, the optimization method to consider the ML model is remaining unanswered.

Another direction is to integrate more in-loop simulations. As introduced in Section 3.4, post-layout simulations have already been successfully used in automatic layout generation of analog building blocks. However, due to the unacceptable simulation time costs, this approach is difficult to be applied to larger system-level designs. Further research in this direction might answer the open question of ensuring the layout quality from automatic tools.

On the other hand, with both the advances in placement and routing algorithm, optimization-based tools have demonstrated the capability to generate layouts for mixed-signal ADCs with tape-out-ready quality[71, 78]. We expect future research in the tools to prove their capability in silicon and reduce the gap between automatic flows and layout quality guarantees.

4.3. Understanding the designs

AMS designs differ in functionalities. Furthermore, understanding the circuit design is crucial in fully automatic layout generation. For example, an operational amplifier and a comparator may have different ideal layout strategies for different considerations. Furthermore, extracting suitable constraints in optimization-based analog place and route from schematic also needs knowledge from circuit design. Currently, state-of-the-art constraint generations are focusing on symmetry[81, 90]. However, there are many more different constraints used in automatic place and route, as introduces in Section 2.3. Many of them may need a deeper understanding of schematic designs. For example, maintaining a regular system signal flow in ADC mixed-signal system is also crucial beyond symmetry[71, 91]. On the other hand, research has observed that different types of nets often need different routing strategies[80]. There are considerations often in manual layout design that are often difficult to formulate as simple geometric constraints such as symmetry. Furthermore, how to understand the specific functionality of a circuit is still an unanswered question. For example, for the C-DAC modules in a SAR ADC design, the placement strategy should consider matching the capacitor array. Though there exist specified placement techniques for capacitor array[94], it currently needs to be designated explicitly by human engineers.

5. Conclusion

This paper presents an overview of the current frameworks of automatic layout generation for analog and mixed-signal circuits. Recent advances and trends in state-of-the-art research are summarized. We review the significant challenges of AMS layout generation and survey the latest advancements in open-source frameworks with ML that addresses those open questions.

Acknowledgements

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

References

[1] Rijmenants J, Litsios J B, Schwarz T R, et al. ILAC: An automated layout tool for analog CMOS circuits. IEEE J Solid-State Circuits, 1989, 24(2), 417

[2] Weaver S, Hershberg B, Knierim D, et al. A 6b stochastic flash analog-to-digital converter without calibration or reference ladder.

Proc ASSCC, 2008, 373

[3] Weaver S, Hershberg B, Moon U K. Digitally synthesized stochastic flash ADC using only standard digital cells. IEEE Trans Circuits Syst I, 2013, 61(1), 84

[4] Fahmy A, Liu J, Kim T, et al. An all-digital scalable and reconfigurable wide-input range stochastic ADC using only standard cells. IEEE TCAS II, 2015, 62(8), 731

[5] Waters A, Moon U K. A fully automated verilog-to-layout synthesized ADC demonstrating 56 dB-SNDR with 2 MHz-BW. Proc ASSCC, 2015, 1

[6] Ansari E, Wentzloff D D. A 5 mW 250 MS/s 12-bit synthesized digital to analog converter. Proceedings of the IEEE 2014 Custom Integrated Circuits Conference, 2014, 1

[7] Liu J, Fahmy A, Kim T, et al. A fully synthesized 0.4 V 77 dB SFDR reprogrammable SRMC filter using digital standard cells. Proc CICC, 2015, 1

[8] Seo M J, Roh Y J, Chang D J, et al. A reusable code-based SAR ADC design with CDAC compiler and synthesizable analog building blocks. IEEE Trans Circuits Syst II, 2018, 65(12), 1904

[9] Weaver S, Hershberg B, Maghari N, et al. Domino-logic-based ADC for digital synthesis. IEEE Trans Circuits Syst II, 2011, 58(11), 744

[10] Park Y, Wentzloff D D. An all-digital PLL synthesized from a digital standard cell library in 65 nm CMOS. 2011 IEEE Custom Integrated Circuits Conference (CICC), 2011, 1

[11] Deng W, Yang D, Narayanan A T, et al. A 0.048 mm^2 3 mW synthesizable fractional-N PLL with a soft injectionlocking technique. ISSCC Digest of Technical Papers, 2015, 1

[12] Bang S, Lim W, Augustine C, et al. A fully synthesizable distributed and scalable all-digital LDO in 10 nm CMOS. 2020 IEEE International Solid-State Circuits Conference (ISSCC), 2020, 380

[13] Park Y, Wentzloff D D. An all-digital 12 pJ/pulse IR-UWB transmitter synthesized from a standard cell library. IEEE J Solid-State Circuits, 2011, 46(5), 1147

[14] Li S, Xu B, Pan D Z, et al. A 60-fJ/step 11-ENOB VCO-based CTDSM synthesized from digital standard cell library. Proc CICC, 2019, 1

[15] Xu B, Li S, Sun N, et al. A scaling compatible, synthesis friendly VCO-based delta-sigma ADC design and synthesis methodology. Proc DAC, 2017, 1

[16] IDEA-FASoC project

[17] Ajayi T, Cherivirala Y, Kwon K, et al. Fully autonomous mixed signal SoC design & layout generation platform. 2020 Hot Chips: A Symposium on High Performance Chips, 2020

[18] Kobayashi T, Nogami K, Shirotori T, et al. A current-mode latch sense amplifier and a static power saving input buffer for low-power architecture. Symposium on VLSI Circuits Digest of Technical Papers, 1992, 28

[19] Shim M, Jeong S, Myers P D, et al. Edge-pursuit comparator: An energy-scalable oscillator collapse-based comparator with application in a 74.1 dB SNDR and 20 kS/s 15 b SAR ADC. IEEE J Solid-State Circuits, 2017, 52(4), 1077s

[20] Jung W, Jeong S, Oh S, et al. A 0.7 pF-to-10nF fully digital capacitance-to-digital converter using iterative delay-chain discharge. Proc ISSCC, 2015, 1

[21] Chen D J, Lee J, Sheu B J. Slam: A smart analog module layout generator for mixed analog-digital VLSI design. Proc ICCD, 1989, 24

[22] Stefanovic D, Kayal M, Pastre M, et al. Procedural analog design (pad) tool. Proc ISQED, 2003, 313

[23] Youssef S, Javid F, Dupuis D, et al. A python-based layout-aware analog design methodology for nanometric technologies. IEEE International Design and Test Workshop (IDT), 2011, 62

[24] Crossley J, Puggelli A, Le H, et al. BAG: A designer-oriented integrated framework for the development of AMS circuit generators. Proc ICCAD, 2013, 74

[25] Castro-Lopez R, Guerra O, Roca E, et al. An integrated layout-synthesis approach for analog ICs. IEEE TCAD, 2008, 27(7), 1179

[26] Han S, Jeong S, Kim C, et al. GUI-Enhanced layout generation of ffe sst txs for fast high-speed serial link design. Proc DAC, 2020

[27] Ding M, Harpe P, Chen G, et al. A hybrid design automation tool for sar adcs in IoT. IEEE TVLSI, 2018, 26(12), 2853

[28] Wulff C, Ytterdal T. A compiled 9-bit 20-MS/s 3.5-fJ/conv. step SAR ADC in 28-nm FDSOI for bluetooth low energy receivers. IEEE J Solid-State Circuits, 2017, 52(7), 1915

[29] Chang E, Han J, Bae W, et al. BAG2: A process-portable framework for generator-based ams circuit design. Proc CICC, 2018, 1

[30] Hakhamaneshi K, Werblun N, Abbeel P, et al. Bagnet: Berkeley analog generator with layout optimizer boosted with deep neural networks. Proc ICCAD, 2019, 1

[31] Settaluri K, Haj-Ali A, Huang Q, et al. Autockt: Deep reinforcement learning of analog circuit designs. Proc DATE, 2020, 490

[32] Hammouda S, Said H, Dessouky M, et al. Chameleon art: a non-optimization based analog design migration framework. Proc DAC, 2006, 885

[33] Pan P, Chin C, Chen H, et al. A fast prototyping framework for analog layout migration with planar preservation. IEEE TCAD, 2015, 34(9), 1373

[34] Dong X, Zhang L. Process-variation-aware rule-based optical proximity correction for analog layout migration. IEEE TCAD, 2017, 36(8), 1395

[35] Jangkrajarng N, Zhang L, Bhattacharya S, et al. Template-based parasitic-aware optimization and retargeting of analog and rf integrated circuit layouts. Proc ICCAD, 2006, 342

[36] Zhang L, Jangkrajarng N, Bhattacharya S, et al. Parasitic-aware optimization and retargeting of analog layouts: A symbolic-template approach. IEEE TCAD, 2008, 27(5), 791

[37] Rutenbar R A. Analog circuit and layout synthesis revisited. Proc ISPD, 2015, 83

[38] Kunal K, Madhusudan M, Sharma A K, et al. ALIGN: Open-source analog layout automation from the ground up. Proc DAC, 2019, 1

[39] Xu B, Zhu K, Liu M, et al. MAGICAL: Toward fully automated analog ic layout leveraging human and machine intelligence. Proc ICCAD, 2019, 1

[40] Cohn J M, Garrod D J, Rutenbar R A, et al. KOAN/ANAGRAM II: New tools for device-level analog placement and routing. IEEE J Solid-State Circuits, 1991, 26(3), 330

[41] Balasa F, Lampaert K. Module placement for analog layout using the sequence-pair representation. Proc DAC, 1999, 274

[42] Pang Y, Balasa F, Lampaert K, et al. Block placement with symmetry constraints based on the O-tree non-slicing representation. Proc DAC, 2000, 464

[43] Balasa F, Maruvada S C, Krishnamoorthy K. Efficient solution space exploration based on segment trees in analog placement with symmetry constraints. Proc ICCAD, 2002, 497

[44] Balasa F, Maruvada S C, Krishnamoorthy K. Using red-black interval trees in device-level analog placement with symmetry constraints. Proc ASPDAC, 2003, 777

[45] Lin J M, Wu G M, Chang Y W, et al. Placement with symmetry constraints for analog layout design using TCG-S. Proc ASPDAC, 2005, 1135

[46] Balasa F, Maruvada S C, Krishnamoorthy K. On the exploration of the solution space in analog placement with symmetry constraints. IEEE TCAD, 2006, 23(2), 177

[47] Long D, Hong X, Dong S. Signal-path driven partition and placement for analog circuit. Proc ASPDAC, 2006

[48] Liu J, Dong S, Ma Y, et al. Thermal-driven symmetry constraint for analog layout with CBL representation. Proc ASPDAC, 2007, 191

[49] Lin P H, Lin S C. Analog placement based on hierarchical module clustering. Proc DAC, 2008, 50

[50] Strasser M, Eick M, Gräb H, et al. Deterministic analog circuit placement using hierarchically bounded enumeration and enhanced shape functions. Proc ICCAD, 2008, 306

[51] Lin P H, Chang Y W, Lin S C. Analog placement based on sym-

metry-island formulation. IEEE TCAD, 2009, 28(6), 791
[52] Xiao L, Young E F Y. Analog placement with common centroid and 1-D symmetry constraints. Proc ASPDAC, 2009, 353
[53] Lin P H, Zhang H, Wong M D F, et al. Thermal-driven analog placement considering device matching. Proc DAC, 2009, 593
[54] Nakatake S, Kawakita M, Ito T, et al. Regularity-oriented analog placement with diffusion sharing and well island generation. Proc ASPDAC, 2010, 305
[55] Lin C W, Lin J M, Huang C P, et al. Performance-driven analog placement considering boundary constraint. Proc DAC, 2010, 292
[56] Ma Q, Xiao L, Tam Y C, et al. Simultaneous handling of symmetry, common centroid, and general placement constraints. IEEE TCAD, 2011, 30, 85
[57] Tsao H F, Chou P Y, Huang S L, et al. A corner stitching compliant b*-tree representation and its applications to analog placement. Proc ICCAD, 2011, 507
[58] Chou P Y, Ou H C, Chang Y W. Heterogeneous b*-trees for analog placement with symmetry and regularity considerations. Proc ICCAD, 2011, 512
[59] Lin C W, Lu C C, Lin J M, et al. Routability-driven placement algorithm for analog integrated circuits. Proc ISPD, 2012, 71
[60] Wu P H, Lin M P H, Chen Y R, et al. Performance-driven analog placement considering monotonic current paths. Proc ICCAD, 2012
[61] Chien H C C, Ou H C, Chen T C, et al. Double patterning lithography-aware analog placement. Proc DAC, 2013
[62] Ou H C, Chien H C C, Chang Y W. Simultaneous analog placement and routing with current flow and current density considerations. Proc DAC, 2013
[63] Wu P H, Lin M P H, Chen T C, et al. Exploring feasibilities of symmetry islands and monotonic current paths in slicing trees for analog placement. IEEE TCAD, 2014, 33(6), 879
[64] Wu I P, Ou H C, Chang Y W. QB-trees: Towards an optimal topological representation and its applications to analog layout designs. Proc DAC, 2016
[65] Ou H C, Tseng K H, Liu J Y, et al. Layout-dependent effects-aware analytical analog placement. IEEE TCAD, 2016, 35(8), 1243
[66] Xu B, Li S, Xu X, et al. Hierarchical and analytical placement techniques for high-performance analog circuits. Proc ISPD, 2017
[67] Lu Y S, Chang Y H, Chang Y W. WB-trees: A meshed tree representation for FinFET analog layout designs. Proc DAC, 2018
[68] Patyal A, Pan P C, A A K, et al. Analog placement with current flow and symmetry constraints using PCP-SP. Proc DAC, 2018
[69] Xu B, Basaran B, Su M, et al. Analog placement constraint extraction and exploration with the application to layout retargeting. Proc ISPD, 2018
[70] Xu B, Li S, Pui C W, et al. Device layer-aware analytical placement for analog circuits. Proc ISPD, 2019
[71] Zhu K, Chen H, Liu M, et al. Effective analog/mixed-signal circuit placement considering system signal flow. Proc ICCAD, 2020
[72] Lampaert K, Gielen G, Sansen W M. A performance-driven placement tool for analog integrated circuits. IEEE J Solid-State Circuits, 1995, 30(7), 773
[73] Li Y, Lin Y, Madhusudan M, et al. Exploring a machine learning approach to performance driven analog IC placement. 2020 IEEE Computer Society Annual Symposium on VLSI (ISVLSI), 2020, 24
[74] Li Y, Lin Y, Madhusudan M, et al. A customized graph neural network model for guiding analog IC placement. ICCAD 2020: IEEE/ACM International Conference on Computer-Aided Design, 2020
[75] Choudhury U, Sangiovanni-Vincentelli A. Constraint-based channel routing for analog and mixed analog/digital circuits. IEEE TCAD, 1993, 12(4), 497
[76] Xiao L, Young E F Y, He X, et al. Practical placement and routing techniques for analog circuit designs. Proc ICCAD, 2010, 675
[77] Lin J W, Ho T Y, Jiang I H R. Reliability-driven power/ground routing for analog ICs. ACM TODAES, 2019, 17, 1
[78] Chen H, Zhu K, Liu M, et al. Toward silicon-proven detailed routing for analog and mixed-signal circuits. Proc ICCAD, 2020
[79] Ajayi T, Chhabria V A, Fogaça M, et al. Toward an open-source digital flow: First learnings from the openroad project. Proc DAC, 2019, 76, 1
[80] Zhu K, Liu M, Lin Y, et al. GeniusRoute: A new analog routing paradigm using generative neural network guidance. Proc ICCAD, 2019, 1
[81] Liu M, Li W, Zhu K, et al. S3DET: Detecting system symmetry constraints for analog circuits with graph similarity. Proc ASPDAC, 2020, 193
[82] Liu M, Zhu K, Gu J, et al. Towards decrypting the art of analog layout: Placement quality prediction via transfer learning. Proc DATE, 2020
[83] Xu B, Lin Y, Tang X, et al. WellGAN: Generative-adversarial-network-guided well generation for analog/mixedsignal circuit layout. Proc DAC, 2019, 1
[84] Charbon E, Malavasi E, Sangiovanni-Vincentelli A. Generalized constraint generation for analog circuit design. Proc ICCAD, 1993, 408
[85] Hao Q, Dong S, Chen S, et al. Constraints generation for analog circuits layout. International Conference on Communications, Circuits and Systems, 2004, 2, 1339
[86] Zhou Z, Dong S, Hong X, et al. Analog constraints extraction based on the signal flow analysis. International Conference on ASIC, 2005, 2, 825
[87] Wu P, Lin M P, Ho T. Analog layout synthesis with knowledge mining. 2015 European Conference on Circuit Theory and Design (ECCTD), 2015, 1
[88] Eick M, Strasser M, Lu K, et al. Comprehensive generation of hierarchical placement rules for analog integrated circuits. IEEE TCAD, 2011, 30(2), 180
[89] Kunal K, Poojary P, Dhar T, et al. A general approach for identifying hierarchical symmetry constraints for analog circuit layout. Proc ICCAD, 2020
[90] Kunal K, Dhar T, Madhusudan M, et al. GANA: Graph convolutional network based automated netlist annotation for analog circuits. Proc DATE, 2020
[91] Liu M, Zhu K, Tang X, et al. Closing the design loop: Bayesian optimization assisted hierarchical analog layout synthesis. Proc DAC, 2020
[92] Chen H, Liu M, Xu B, et al. Magical: An open-source fully automated analog IC layout system from netlist to GDSII. IEEE Design & Test, 2020
[93] Ajayi T, Chhabria V A, Fogaça M, et al. Toward an open-source digital flow: First learnings from the openroad project. Proc DAC, 2019
[94] Lin C W, Lee C L, Lin J M, et al. Analytical-based approach for capacitor placement with gradient error compensation and device correlation enhancement in analog integrated circuits. Proc ICCAD, 2012