MAGICAL: An Open-Source Fully Automated Analog IC Layout System from Netlist to GDSII

Hao Chen, Mingjie Liu, Biying Xu, Keren Zhu, Xiyuan Tang, Shaolan Li, Yibo Lin, Nan Sun, and David Z. Pan
The University of Texas at Austin

Editor's notes:
This article presents MAGICAL, which is a fully automated analog IC layout system. MAGICAL takes a netlist and design rules as inputs, and it produces the final GDS layout in a fully automated fashion.
—Sherief Reda, Brown University
—Leon Stock, IBM
—Pierre-Emmanuel Gaillardon, University of Utah

THE EXPANDING MARKETS of emerging applications, including the Internet of Things (IoT), 5G networks, advanced computing, healthcare electronics, etc., create large demands for analog and mixed-signal (AMS) integrated circuits. This increasing demand calls for a shorter design cycle and time-to-market. When compared with the tremendous advancements in digital IC layout design automation tools, analog IC layout still remains a heavy manual, time-consuming, and error-prone task. This is due to its high design flexibility and sensitive impact on the circuit performance by even minor changes in the layout implementation.

Digital Object Identifier 10.1109/MDAT.2020.3024153
Date of publication: 14 September 2020; date of current version: 8 April 2021.

Traditional analog layout synthesis tools rely on various heuristic constraints to guide the layout generation process [1]. These heuristics are based on human layout techniques and enforced during the placement of devices and routing. Heuristic constraint-based methods face extreme difficulties in practical design flows, where handcrafted constraints are often design and technology dependent, lacking flexibility and generalization when meeting the detailed requirements of different scenarios. There is also the challenge of hard-encoding all such constraints in a legal procedure, especially when numerous contradictory constraints are present. Analytical approaches attempt to uncover the layout design tradeoffs either by deriving closed-form equations in evaluating the layout-dependent effects or sensitivity analysis simulations. With increased device scaling, analytical sensitivity estimates of parasitics and mismatch over performance are no longer accurate.

The most difficult thing above all is the limited availability of analog design tools. In contrast to the booming community of machine learning, where

Copublished by the IEEE CEDA, IEEE CASS, IEEE SSCS, and TTTC

popular frameworks and data sets are open-sourced, easily available, and heavily relied upon in research by the academic community, commonly used tools in analog design are largely proprietary. The lack of implemented frameworks, benchmark circuit data sets, and publicly available process design kits (PDKs), severely restrict the reproducibility of current research results and impede further improvement and extended research.

In this article, we present our work MAGICAL, a fully automated, end-to-end analog IC layout framework that generates a completed layout from a circuit netlist. Implemented modules include symmetry constraint generation, placement, and routing. These modules are implemented in C++ for optimal software performance, and off-the-shelf with user-friendly Python interface available. The source code[1] is released on GitHub with a number of sanitized benchmark circuits provided.[2] The layouts completed by MAGICAL are validated using industrial standard verification tools, demonstrating circuit performances close to those handcrafted by experienced designers.

Compared with prior on procedural layout generators, such as Berkeley analog generator (BAG) [2], MAGICAL reduces the cost of codifying specific constraints and detailed layout implementation such as circuit floorplan and routing topology. This is achieved with automated symmetry extraction leveraging pattern matching and graph similarity, and area and wirelength driven placement and routing optimization kernel. MAGICAL is also extensible to handle custom constraints. We also present several of our research and findings that build upon the MAGICAL framework, including applied machine learning techniques, custom constraints, specific design considerations, and leveraging post layout simulation results for performance modeling and optimizations in the "Extensions based on MAGICAL" section. Moreover, we hope to promote research and progress in the analog design automation community, where future researchers could embed new heuristics and algorithms leveraging our framework. Our tool also complements other existing design automation tools in the open-source community [2]–[4].

[1]https://github.com/magical-eda/MAGICAL
[2]https://github.com/magical-eda/MAGICAL-CIRCUITS

Magical framework

The overall flow of MAGICAL is shown in Figure 1. It takes an unannotated circuit netlist and design rules as inputs, and produces a complete GDSII layout as output fully automatically without human designers in the loop. The entire flow consists of four major modules, with each module being independent with a user-friendly Python interface. The design rules and the extracted layout constraints are honored throughout the entire back-end flow.

Framework methodology and software architecture

The MAGICAL system includes several individual submodules, i.e., layout constraint extractor, device generator, placer, and router. A top-level MAGICAL flow integrates the individual components and manage the physical synthesis of the

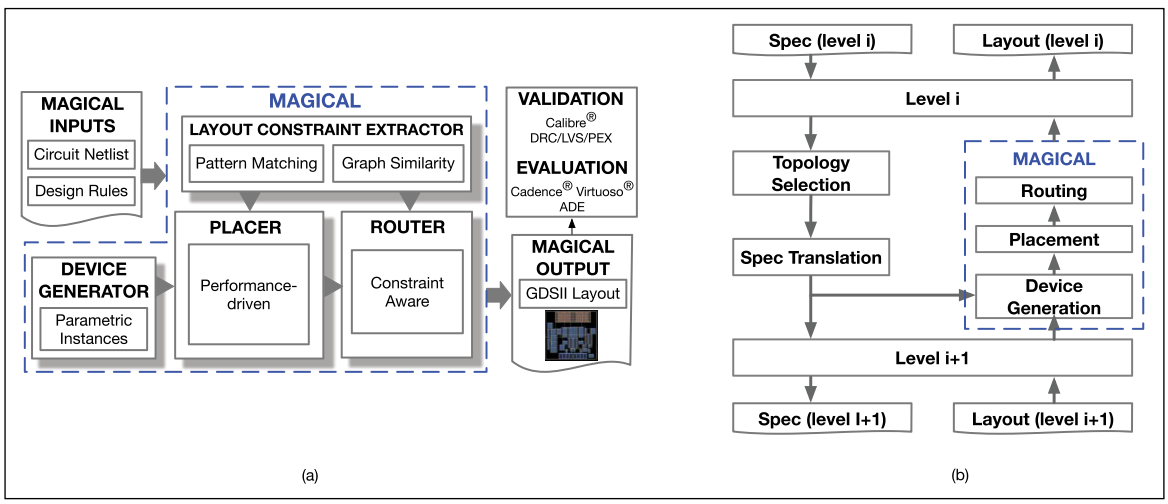

Figure 1. MAGICAL framework. (a) MAGICAL submodules. (b) MAGICAL hierarchical flow.

circuits. Figure 1 shows the architecture of the MAGICAL layout system. This architecture is rooted upon the principle of divide-and-conquer nature of the circuit design and motivated by the purposes of creating an extensible and flexible open-source environment.

The designs of complex analog systems, such as phase-locked loops or analog-to-digital converters (ADCs), are typically decomposed into smaller building blocks (e.g., comparators, filters, or amplifiers). Human designers adopt a top-down design methodology, where system-level performance is translated to lower-level building block specifications. The layout design process, on the other hand, is done bottom-up. The building block circuits are first implemented and the performance is optimized and verified. The system is then built with the building blocks. This divide-and-conquer practice avoids optimizing the whole system at once and decomposes it into smaller and more traceable subproblems. MAGICAL adopts this design methodology. The whole physical synthesis is decomposed into multilevel homogeneous subproblems. The top MAGICAL flow manages and schedules the subproblems, while the individual components build the layout in the bottom-up manner.

The separation of submodules also allow easy extension to the default MAGICAL flow. Conventionally, different components in a physical design flow is connected by scripts and exchangeable files, for example [4]. However, the complicated scripting potentially make interaction between different components difficult and hinder the flexibility on the flow. On the other hand, the popularity of machine learning algorithm also raises question that whether the conventional software methodology is suitable for the emerging framework. MAGICAL is developed for easy adaption of new components and changes on the flow. Although the main optimizing kernels are developed in C++ for better efficiency, each submodule has a Python interface. And the top level, MAGICAL flow uses Python interface to assign the subproblems. Such architecture is friendly to adoption of machine learning framework and makes interactions easy. In fact, there have been success on the extensions of default MAGICAL. The "Experimental results" section gives case studies of several MAGICAL extensions.

In the rest of this section, the default MAGICAL submodules are explained in the detail.

Parametric device generation

Before running the core layout flow, the device generation step first generates the layout of the devices and extracts their pins to facilitate the subsequent placement and routing stages.

The generated GDSII layout is correct by construction based on the design rules. MAGICAL currently supports numerous different device types, including pMOS, nMOS, metal–oxide–metal (MOM) capacitors, and poly resisters. The automatic parametric device generation considers the number of fingers for transistors, the number of segments for resistors, the metal layers for MOM capacitors, etc.

The MAGICAL framework is also extensible for custom-designed devices, digital standard cells, or even subcircuits such as capacitor or resistor arrays.

Analog layout constraint extraction

The layout constraint extractor takes a circuit netlist as input and generates constraints to guide the later stages. Analog designs frequently use differential topologies to reject common-mode noise and enhance circuit robustness and performance [5]. Thus, correctly identifying symmetry constraints between sensitive devices are crucial for ensuring the quality of placement and routing.

The constraint extraction reads in the input netlist and generates constraints for placement and routing based on the circuit connections. A significant challenge for constraint extraction is in generating high-quality constraints and resolving constraint ambiguity. Since the characteristics of symmetry among devices and between building blocks are vastly different, we use different methods to generate symmetry constraints.

Device symmetry

Only the symmetry constraints between devices need to be considered for building blocks. We adopt a method similar to the works of Eick et al. [6]. The building block circuit is abstracted into a graph. A pattern library of the commonly used differential topology of transistors is predefined. We use graph isomorphic algorithms to detect matching patterns on the building block circuits with the pattern library. The circuit graph is then traversed from the matched patterns to recognize new symmetry constraints of passive devices, self-symmetry devices, and symmetry routing constraints.

System symmetry

System symmetry differs from device symmetry because on the system level, template libraries are difficult to generate, and graph isomorphic algorithms are expensive. Since the same building block could be referenced multiple times in system design, we propose to extract graphs that include the neighboring circuit topology of the building blocks to resolve the ambiguity. The extracted graphs are then compared using an efficient graph similarity metric leveraging spectral graph analysis [7]. Self-symmetry constraints and symmetry net constraints could also be extracted similar to the approach in device symmetry.

Analog placement

Given the placement constraints and devices generated in the previous steps, we develop an analog placement engine. The placer places each device or building block in the layout satisfying the given constraints while optimizing for the wirelength and layout area.

The placement engine follows an analytical framework as in [8]. First, the global placement simultaneously optimizes multiple objectives in a nonlinear objective function to generate a rough legal placement. Then, the legalization step uses linear programming (LP) algorithm to leaglize the global placement results honoring input constraints and design rules. Finally, another LP-based detailed placement is used to optimize the wirelength further.

Analog routing

To determine the wire connections between all the placed devices while satisfying the design considerations for better circuit performance, a constraint-aware analog routing algorithm is applied.

In addition to connectivity and design rules, an analog routing problem is usually imposed with symmetric net constraints, which are specified to ensure matched nets routed symmetrically on some axes. In MAGICAL, the routing engine takes the constraints specification as an input from the layout constraint generator and honors the symmetric and self-symmetric requirement for matched nets.

Our routing framework divides the routing problem into two stages, global routing and detailed routing, similar to the standard digital routing flow. To generated a global routing solution, a sequential symmetry-aware grid-based A* search routing engine is employed. The circuit is cut into unified grids whose width and height are decided based on track width on the first metal layer. More specifically, the global routing engine divides the layout into a 3-D graph with grids as the vertices and the connection between neighboring grids as the edges. The capacity of each edge is calculated based on free space modeling and the actual congestion inside the grids. The symmetry-aware global routing algorithm using mirroring techniques then generates the solution for each net.

Given the global routing results as guidance, the detailed routing engine completes the final routing and assigns metal wire geometries. In contrast to digital circuits, analog designs usually have various metal widths and multiple via cuts for different nets, along with a number of special specifications such as symmetric constraints. To solve the detailed routing problem, the symmetric-aware A* path searching algorithm is performed while satisfying design rules and specific requirements for each net (e.g., wire width). After the detailed routing stage, the final GDSII format layout file is exported.

Extensions based on magical

In this section, we present several of our research that builds on top of the default MAGICAL framework. We hope to demonstrate the extensibility of the framework and the increased efficiency in the development of novel algorithms and methods by leveraging MAGICAL.

Machine learning guided well generation

Generating wells and inserting contacts are required in layout synthesis. The default MAGICAL framework generates separate NWELL contacts for each individual pMOS devices. This provides superior device isolation and reduction in well proximity effect at the increased overhead of area.

However, individual well contacts for transistor is seldom adopted in manual layout strategy. Sharing the same body connection leads to more compact layout. *WellGAN* [9] provides an alternative to the individual well contact. It formulates the well generation problem into a computer vision task and supervisedly learns how human designers draw the well using generative adversarial network (GAN). After the training, the GAN model can generate images of well based on the placement results and a legalization routine can draw the well and insert the contacts based on the model prediction.

Machine learning guided analog routing

In the default MAGICAL, the routing is targeting wirelength and geometric constraints. However, the router can be easily changed with a machine learning guided algorithm.

Zhu et al. [10] proposed *GeniusRoute* is developed upon the MAGICAL router. It attempts to learn the manual analog routing strategy by leveraging a variational autoencoder (VAE) model. Similar to *WellGAN*, *GeniusRoute* formulates the learning into an image generation task. The VAE models learn where human designers are likely to route the nets in analog circuits and generate prediction on unseen new circuits. The router is modified to adapt the routing prediction.

Analog placement quality prediction

In the default MAGICAL framework, the analog analytical placement engine only optimizes the wire-length and area. While the wire-length might be a natural surrogate for performance and highly correlated with the power and performance of digital circuits, analog layout performance rarely has strong relevance to the total wire-length. Thus, to satisfy post layout performance requirements and achieve design closure, a feedback loop from performance simulation to the design flow is needed in the development of practical layout synthesis tools.

To reduce the design exploration runtime and limit the number of performance simulations, the work of Liu et al. [11] proposes to predict of the layout quality early in the layout design flow. To overcome the difficulty of obtaining high-quality human layout training data, MAGICAL was used to generate multiple layout solutions for the same circuits automatically. An effective placement feature extraction method with 3-D convolution neural network was developed for effective placement quality prediction. The number of training data needed to obtain satisfactory classification results was significantly reduced by leveraging transfer learning.

Efficient layout synthesis with Bayesian optimization

The works of Liu et al. [12] extended the default MAGICAL framework considering custom constraints and design-specific considerations. It leverages post layout simulations in driving the layout implementation process for building block circuits. By formulating the performance optimization as a multiobjective black-box optimization problem, it closes the design loop and guarantees post layout performance through iterative simulations and a data-efficient Bayesian optimization algorithm.

Since system-level transistor simulation is unaffordable, Liu et al. [12] optimized the system-level layout by extending the original MAGICAL framework to include custom constraints and design specific considerations. Specific constraints and considerations include net criticality, routing sequence, net spacing assignments, and regularized signal flow paths. The layout for a complete ADC system with regularized signal flow paths were generated, achieving close to schematic simulation results.

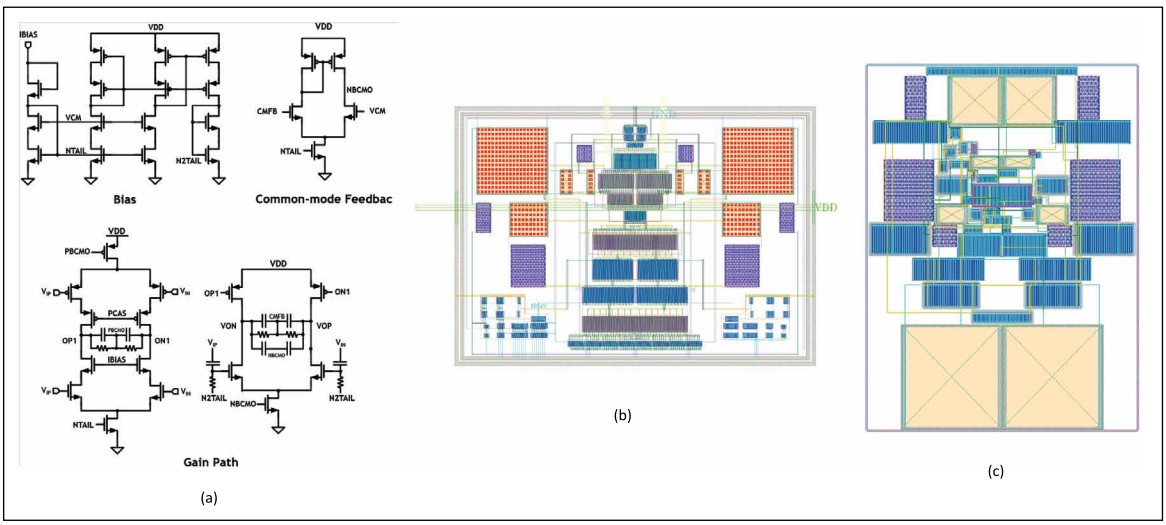

Figure 2. OTA results. (a) Circuit schematic. (b) Manual layout. (c) MAGICAL layout.

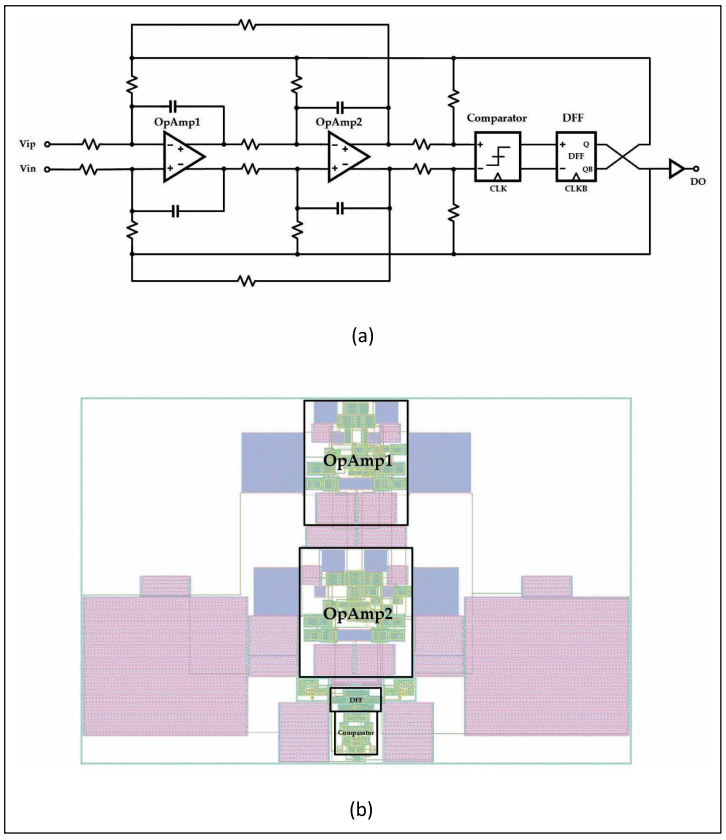

Figure 3. CTDSM results. (a) System architecture. (b) MAGICAL layout.

Table 1. OTA simulation results.

Metrics	Gain (dB)	UGB (MHz)	PM (degree)	CMRR (dB)	Offset (mV)
Schematic	69.5	2192	73.5	-	-
Manual	69.8	1567	65.8	97.2	0.012
MAGICAL	69.7	1496	63.2	92.1	0.027

Table 2. CTDSM simulation results.

	Schematic	MAGICAL
Supply (V)	1.2	
F_s (MHz)	320	
BW (MHz)	5	
SNDR (dB)	67.8	65.9
SFDR (dB)	84.7	80.5
Power (mW)	0.84	0.86
Area (um^2)	–	9450

Experimental results

The MAGICAL flow is implemented in Python and C/C++, and the experiments are performed on a Linux server with an 8-core 3.4-GHz Intel CPU and 32-GB memory. All designs are in TSMC 40-nm technology. The layout results are validated using Calibre DRC/LVS/PEX, and evaluated using Cadence Virtuoso ADE simulation environment.

The post-layout simulation results for two benchmark circuits, a two-stage operational transconductance amplifier (OTA), and a continuous time ΔΣ modulator (CTDSM), are shown in Figures 2 and 3, respectively. The circuit performances of the OTA layout results generated by MAGICAL are compared against tape-out quality manual layouts by experienced analog IC designers, under the same test bench suites. The simulation results are shown in Table 1, where UGB stands for the unity gain bandwidth, PM denotes the phase margin, and CMRR denotes the common-mode rejection ratio. The simulation results for the CTDSM are shown in Table 2, where F_s denotes the sampling frequency, BW is the bandwidth, SNDR denotes the signal-to-noise and distortion ratio, and SFDR denotes the spurious-free dynamic range. The results demonstrate that MAGICAL can automatically generate validated layouts from unannotated circuit netlist (both Spectre and HSPICE format), and the post-layout performances are close to the schematic designs. Some performance metrics, including input-referred offset and CMRR, could be further improved by extensively considering layout dependent effects, minimizing coupling to sensitive nets, etc.

Future directions

Being part of the open-source hardware/EDA ecosystem, the future development of the MAGICAL will both benefit from and contribute to the community. Although the existing components in different open-source EDA tools may have different algorithms and methodologies, there are some overlapping between their functionality. Both analog and digital layout automation flows share many common infrastructural components with MAGICAL. MAGICAL can learn from the recent emerging open-source EDA tools.

Besides the EDA tools, open-sourcing AMS circuit designs is another driving force for analog layout automation. On the one hand, lacking of training data has been a major challenge in machine learning-based EDA algorithm. On the other hand, the lack of a unified test circuit benchmark suite makes it difficult to evaluate and compare different analog EDA tools. Open-source designs will not only make it possible for the EDA tools to have common

evaluation metrics, but also provide training data for machine learning-based EDA algorithms.

While MAGICAL has demonstrated satisfactory results, it currently only minimizes post-layout circuit performance degradation implicitly by considering the analog layout constraints. Although direct optimization methods have been applied and demonstrated to be effective, the overhead of repetitive simulations is still expensive and impractical, especially for system-level designs. In the future research and development, MAGICAL will investigate into the performance-aware techniques, especially machine learning algorithms, throughout its entire flow.

Preliminary simulation results have also demonstrated the potential of generating entire system-level designs with MAGICAL. However, MAGICAL still needs to improve its current placement and routing algorithms for better design rule handling. Furthermore, there is still large room for improvement, especially for system-level designs, including circuit reliability, clock coupling mitigation, IR drop aware routing, and integration with digital flows. Generating tape-out quality layout designs proven with silicon chip measurements will be the future goal of MAGICAL.

In this article, we presented MAGICAL, an open-source fully automated end-to-end analog IC layout system from circuit netlists to GDSII layouts. Human and machine intelligence are strategically incorporated into MAGICAL by pattern matching and deep learning techniques. The circuit performances of the layouts completed by MAGICAL are close to those handcrafted by experienced designers, while the design cycle is shortened substantially. ■

Acknowledgments

Hao Chen, Mingjie Liu, Biying Xu, and Keren Zhu contributed equally to this work. This work was supported in part by the National Science Foundation (NSF) under Grant 1704758 and in part by the DARPA IDEA program.

References

[1] M. P.-H. Lin, Y.-W. Chang, and C.-M. Hung, "Recent research development and new challenges in analog layout synthesis," in *Proc. 21st Asia South Pacific Design Autom. Conf. (ASP-DAC)*, Jan. 2016, pp. 617–622.

[2] J. Crossley et al., "BAG: A designer-oriented integrated framework for the development of AMS circuit generators," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, Nov. 2013, pp. 74–81.

[3] K. Kunal et al., "ALIGN: Open-source analog layout automation from the ground up," in *Proc. 56th Annu. Design Autom. Conf.*, Jun. 2019, pp. 1–4.

[4] T. Ajayi et al., "Toward an open-source digital flow: First learnings from the OpenROAD project," in *Proc. 56th Annu. Design Autom. Conf.*, Jun. 2019, pp. 1–4.

[5] B. Razavi, *Design of Analog CMOS Integrated Circuits*, 1st ed. New York, NY, USA: McGraw-Hill, 2001.

[6] M. Eick et al., "Comprehensive generation of hierarchical placement rules for analog integrated circuits," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 30, no. 2, pp. 180–193, Feb. 2011.

[7] M. Liu et al., "S3DET: Detecting system symmetry constraints for analog circuits with graph similarity," in *Proc. 25th Asia South Pacific Design Autom. Conf. (ASP-DAC)*, Jan. 2020, pp. 193–198.

[8] B. Xu et al., "MAGICAL: Toward fully automated analog IC layout leveraging human and machine intelligence: Invited paper," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, Nov. 2019, pp. 1–8.

[9] B. Xu et al., "WellGAN: Generative-adversarial-network-guided well generation for analog/mixed-signal circuit layout," in *Proc. 56th Annu. Design Autom. Conf.*, Jun. 2019, pp. 1–6.

[10] K. Zhu et al., "GeniusRoute: A new analog routing paradigm using generative neural network guidance," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, Nov. 2019, pp. 1–8.

[11] M. Liu et al., "Towards decrypting the art of analog layout: Placement quality prediction via transfer learning," in *Proc. Design Autom. Test Eur. Conf. Exhibition (DATE)*, Mar. 2020, pp. 496–501.

[12] M. Liu et al., "Closing the design loop: Bayesian optimization assisted hierarchical analog layout synthesis," in *Proc. DAC*, Jul. 2020, pp. 1–6.

Hao Chen is currently pursuing a PhD with the Department of Electrical and Computer Engineering, The University of Texas at Austin, Austin, TX. His research interests include physical design, CAD for analog/mixed signal designs, logic synthesis, and emerging technology. Chen has a BS in electrical engineering from the National Taiwan University, Taipei, Taiwan (2019).

Mingjie Liu is currently pursuing a PhD with the Department of Electrical and Computer Engineering, The University of Texas at Austin, Austin, TX. His research interests include applied machine learning

for design automation and layout design automation for analog and mixed-signal integrated circuits.

Biying Xu is currently a Lead Software Engineer with Cadence Design Systems, San Jose, CA. Her research interests include physical design automation for digital, analog, and mixed-signal integrated circuits. Xu has a PhD from the Department of Electrical and Computer Engineering, The University of Texas at Austin, Austin, TX (2019).

Keren Zhu is currently pursuing a PhD with the Department of Electrical and Computer Engineering, The University of Texas at Austin, Austin, TX. His research focuses on VLSI CAD. Zhu has a BS in electrical engineering with the highest distinction from the University of Wisconsin-Madison, Madison, WI.

Xiyuan Tang is currently a Postdoctoral Researcher with The University of Texas at Austin, Austin, TX. His research interests include digitally assisted data converters, low-power mixed-signal circuits, and analog data processing. Tang has a PhD in electrical engineering from The University of Texas at Austin (2019).

Shaolan Li is currently an Assistant Professor with Georgia Tech, Atlanta, GA. His research interests include analog and mixed-signal IC design, solid-state medical imaging platform, and artificial intelligence hardware. Li has a PhD from The University of Texas at Austin, Austin, TX (2018).

Yibo Lin is an Assistant Professor with the Computer Science Department, Peking University, Beijing, China. His research interests include physical design, machine learning, and GPU acceleration. Lin has a PhD from the Electrical and Computer Engineering Department, The University of Texas at Austin, Austin, TX (2018).

Nan Sun is currently a Professor with the Department of Electronic Engineering, Tsinghua University, Beijing, China. His current research interests include analog, mixed-signal, and RF integrated circuit design, and analog circuit design automation.

David Z. Pan is currently a Silicon Laboratories Endowed Chair of electrical engineering with the Department of Electrical and Computer Engineering, The University of Texas at Austin, Austin, TX. His research interests include cross-layer nanometer IC design for manufacturability, reliability, security, machine learning and hardware acceleration, design/CAD for analog/mixed signal designs and emerging technologies.

■ Direct questions and comments about this article to David Z. Pan, Department of Electrical and Computer Engineering, The University of Texas at Austin, Austin, TX 78712 USA; dpan@ece.utexas.edu.