MAGICAL: Toward Fully Automated Analog IC Layout Leveraging Human and Machine Intelligence

(Invited Paper)

Biying Xu, Keren Zhu, Mingjie Liu, Yibo Lin, Shaolan Li, Xiyuan Tang, Nan Sun, and David Z. Pan
ECE Department, The University of Texas at Austin, Austin, Texas, USA
{biying, keren.zhu, jay_liu, yibolin, slliandy, xitang}@utexas.edu, nansun@mail.utexas.edu, dpan@ece.utexas.edu

***Abstract*—Despite tremendous advancement of digital IC design automation tools over the last few decades, analog IC layout is still heavily manual which is very tedious and error-prone. This paper will first review the history, challenges, and current status of analog IC layout automation. Then, we will present MAGICAL, a human-intelligence inspired, fully-automated analog IC layout system currently being developed under the DARPA IDEA program. It starts from an unannotated netlist, performs automatic layout constraint extraction and device generation, then performs placement and post-placement optimization, followed by routing to obtain the final GDSII layout. Various analytical, heuristic, and machine learning algorithms will be discussed. MAGICAL has obtained promising preliminary results. We will conclude the paper with further discussions on challenges and future directions for fully-automated analog IC layout.**

I. Introduction

The demand for analog integrated circuits (ICs) has been increasing in many emerging applications, including Internet of Things (IoT), 5G networks, advanced computing, healthcare electronics, etc., which necessitates a shorter design cycle of analog ICs. Despite the tremendous advancement of digital IC layout design automation tools, analog IC layout is still a heavily manual, time-consuming, and error-prone task, due to its high design flexibility and significant impact on the circuit performance.

Early endeavors of analog IC layout automation, including ILAC [1], KOAN/ANAGRAM II [2], LAYLA [3], applied simulated annealing to optimize the layout. More recent works [4]–[6] improved the analog layout constraint handling and the design space pruning. Algorithmic revolution on placement [7]–[13] and routing [14]–[17] further enhanced the scalability, efficiency, and design considerations specific to analog IC layouts. Nonetheless, existing analog layout tools usually require human designers to prepare very detailed constraints as inputs, which could be a tedious practice. Meanwhile, the poor accessibility of most tools prohibits testing and improvement from the circuit designers' side. The acceptance of the automated analog IC layout tools has thus been limited.

Recently, DARPA announced the IDEA program, with the mission to create a "no human in the loop" 24-hour turnaround circuit layout generator. The program is expected to spur the development of state-of-the-art IC design flows that are easier to use and more readily available. As part of the program efforts, we present our work MAGICAL, a fully automated analog IC layout system leveraging human and machine intelligence inherently. The main contributions are summarized as follows:

- MAGICAL is a fully automated, end-to-end analog IC layout system that generates a completed layout from a circuit netlist. The source code[1] is released on GitHub.
- Designer insights and expertise are strategically embedded into MAGICAL through pattern matching, heuristic, and deep learning techniques.
- The layouts completed by MAGICAL are validated using industrial standard verification tools, demonstrating circuit performances close to those handcrafted by experienced designers.

II. MAGICAL Design Flow

The overall flow of MAGICAL is shown in Fig. 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: automatically layout constraint extractor, parametric device generator, analytical analog placer with post-placement optimization, and analog router. The design rules and the extracted layout constraints are honored throughout the entire back-end flow. All the completed layout results are validated and evaluated by industrial standard tools. In the subsequent sections, the major tasks of the MAGICAL analog layout generation tool will be elaborated.

III. Parametric Device Generation

Before running the core layout flow, the device generation step first generates the GDSII layout of the devices and extracts their pins to facilitate the subsequent placement and routing stages. The inputs to device generation are instance parameters and process technology-dependent design rules. The generated GDSII layout are correct by construction based on the design rules.

MAGICAL supports different device types, including PMOS, NMOS, MOM capacitors, and poly resisters. Transistors can have additional attributes such as lvt (low threshold voltage), hvt (high threshold voltage) and na (native device).

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

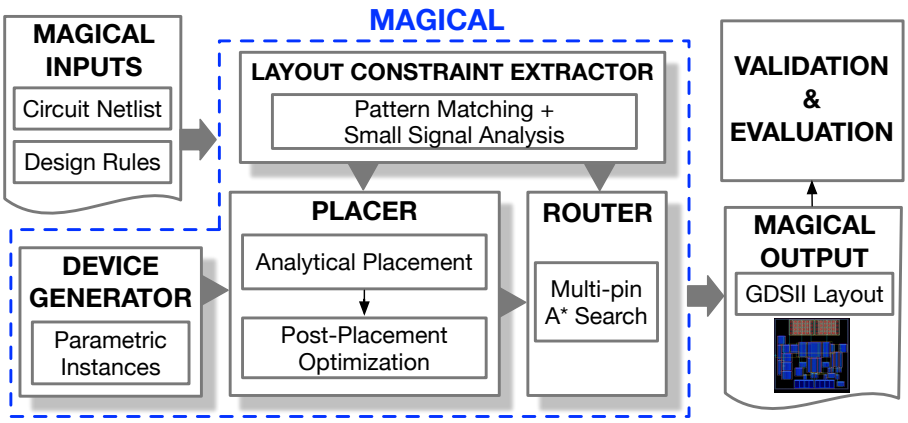
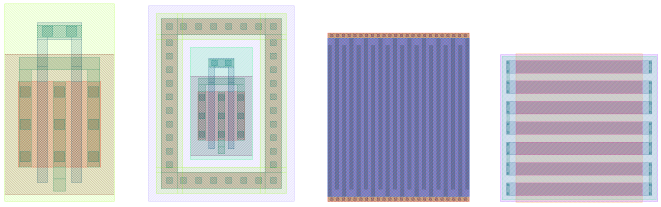


Fig. 1. Overall flow of MAGICAL analog layout system.

The automatic parametric device generation considers the number of fingers for transistors, the number of segments for resistors, the metal layers for Metal-Oxide-Metal (MOM) capacitors, etc. Fig. 2 shows examples of the layouts of different types of devices generated by MAGICAL.

Pin information of the devices is extracted in this step. The pin shape with the minimal area is selected as the drain of a transistor to reduce drain parasitic capacitance. Lower metal layer shapes connected to the terminals of the devices are selected as the pins used for routing.



(a) (b) (c) (d)

Fig. 2. Examples of generated devices from MAGICAL. (a) NMOS, (b) PMOS with guard ring, (c) MOM capacitor, and (d) poly resistor.

IV. Analog Layout Constraint Extraction

The layout constraint extractor takes circuit netlist as input and generates constraints to guide the later stages. 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 and performance [18]. Mismatch of sensitive devices in the layouts often cause performance degradation to offset and common-mode rejection ratio (CMRR) [19]. 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. The automated placement and routing symmetry constraint extraction mainly consist of four stages listed below.

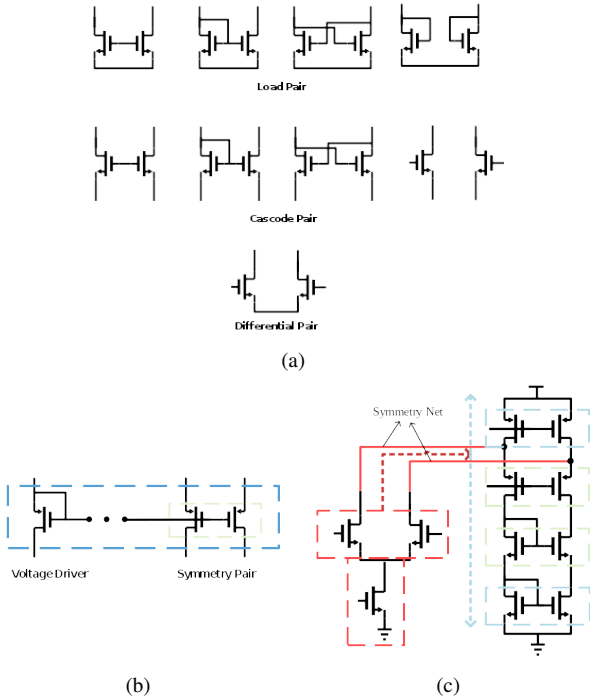


Fig. 3. Layout constraint extraction illustration. (a) Pattern library, (b) bias circuit symmetry detection, and (c) signal flow based traversal.

A. Graph Abstraction

Our constraint extraction is based on analysis of the graph abstracted from the circuit netlist. Firstly, the netlist file is parsed, and the circuit is abstracted into a graph representation. Since the pin connection information is crucial to the topological structure of analog circuits, we preserve pin information during the abstraction. Devices, pins, and nets are all represented as nodes. Pins are connected to the device they belong to. Nets connect to pins based on the circuit netlist

connections. Nets are never directly connected to devices in our graph representation since they alway connect to pins first.

B. Seed Pattern Detection

Transistor pairs that form certain structural patterns are detected as seed symmetric device pairs. These seed patterns are the starting or ending points for graph traversal in the next stage. Seed patterns are those in the pattern library where the source of the transistor pins are connected. Instead of using expensive graph isomorphism algorithms [20]–[23], we check the connection relationships and device attributes between the pairs of devices for matching. As an example, for recognizing differential pairs, we iterate all pairs of transistors connected to the same net through source pins and check if the gate pins are connected to different nets and the device attributes match. To better resolve constraint ambiguity, especially where digital circuits are extensively used in mixed-signal designs, we recognize differential pairs as seed patterns only if the connected source net is the virtual ground node (not power or ground nets).

C. Signal Flow Based Graph Traversal

We traverse the graph from the seed symmetric device pairs while recognizing new constraint patterns. This step is analogous to following the differential current in small-signal analysis [18]. Graph traversal not only expands the recognized constraints but also helps with reducing constraint ambiguity since the graph connections with other patterns guide the pattern matching process. Matching between passive device elements are also considered. For each pattern, we define current directions to guide the graph traversal process. As an example, if a cascode transistor pair pattern is reached through the source pins, the nets to proceed with graph traversal are the nets connected to the drain pins. Graph traversal ends when the two flow paths meet at the source connected transistors such as other seed patterns, or connected passive device elements. The visited symmetric transistor patterns from the same seed pattern during graph traversal form a symmetric group which would share the same symmetry axis in the placement. Nets connecting symmetric device pairs are recognized as symmetric nets.

D. Constraints Post-processing

In the last step, we post-process the symmetric groups and recognize additional symmetry constraints, including self-symmetric devices connected to virtual ground nodes and symmetric transistor pairs in the bias circuits. Virtual ground and certain clock nets are identified as self-symmetric nets. Bias circuit symmetry is detected by checking common-gate connected transistors and searching for diode-connected transistors. Additional symmetry constraints in the bias circuits are detected in this step. Since it may not be feasible to satisfy all the extracted constraints in the placement stage, to guarantee feasibility, we currently only allow a device have at most one symmetry constraint. More advanced matching-oriented constraints such as regularity and common-centroid would be supported in future versions of MAGICAL.

Fig. 3 shows the library of some transistor patterns, detection of bias circuit symmetry, and an example of signal flow based graph traversal for constraint extraction.

V. Analog Placement

Given the placement constraints and devices generated in the previous steps, we develop an analog placement engine. The inputs to the placer include the circuit netlist, extracted constraints, generated devices, and process technology-dependent design rules. The outputs are the legalized GDSII layout of the placement result and pin information for routing. Besides symmetric group constraints, it also supports net criticality constraints if specified, by minimizing the weighted net lengths according to the its criticality. The placement engine consists of a core placement stage and a post-placement optimization stage.

A. Placement Algorithm

The core placer follows an analytical framework as in [24]. First, the global placement simultaneously optimizes multiple objectives in a non-linear objective function. Then, the legalization step generates a legal placement solution honoring the global placement result while satisfying the symmetry constraints and design rules. Finally, a linear programming (LP) based detailed placement is used to further optimize the wirelength.

1) Global Placement: Our global placement is similar to the non-linear global placement algorithms as in [13], [25], which simultaneously considers the following: (1) wirelength, (2) device overlapping, (3) placement boundary, and (4) symmetry constraints from the constraint extraction stage. To be specific, it minimizes the objective shown in Equation (1) using unconstrained non-linear conjugate gradient method.

$$Objective = f_{WL} + a \cdot f_{OL} + b \cdot f_{BND} + c \cdot f_{SYM}. \quad (1)$$

where:

- f_{WL} is the wirelength objective, which is defined as the total half-perimeter wirelength (HPWL).
- f_{OL} is the overlap penalty, which is modeled as an area overlap function similar to [26].
- f_{BND} is the penalty of violating the boundary constraints. To control the circuit area, white space ratio, and aspect ratio, a desirable placement bounding box for the design is derived.
- The last term f_{SYM} penalizes the violation of symmetry constraint, which requires: (1) each symmetric device pair within the same group to be symmetric with respect to the same axis; (2) the self-symmetric devices to be self-symmetric with respect to the same axis of the group.
- a, b, and c are the coefficients to realize the trade-off between different objective terms.

Log-sum-exponential (LSE) models [27] are used to smooth the max and min functions in the objective. Our non-linear optimization-based global placement runs iteratively, until all the penalties are below the specified thresholds, or the predefined maximum number of iterations is reached. By gradually

adjusting the coefficient values a, b, and c of different penalty functions in each iteration, we can get a global placement result encouraging symmetry and boundary constraints with short wirelength and small device overlapping.

2) Legalization and Detailed Placement: After global placement, we perform a legalization step to get a placement free from device overlapping, design rule violation, and symmetry constraint violation. During this step, we develop algorithms to construct the constraint graphs, and legalize the global placement result using LP-based compaction given the constructed constraint graphs.

Our constraint graph construction algorithm is based on the plane sweep algorithm presented in [28]. This algorithm encounters problems when the global placement result has overlaps between devices, which may over-constrain the legalization and result in a sub-optimal area. To get a more compact placement after legalization, we will remove the excessive constraint edges between each pair of overlapping devices by determining their relative positions greedily, i.e., to spread them in the direction that would induce less displacement. We will only keep the constraint edge corresponding to the chosen spreading direction, while other edges between them will be removed.

Nevertheless, there may be missing positional constraints in the constraint graphs obtained. A depth-first-search (DFS) based algorithm is developed in order to detect those missing positional constraint edges. Readers are referred to [24] for the detailed implementation. If both horizontal and vertical positional constraints are missing according to the DFS-based algorithm, we will add one edge to either the vertical or horizontal constraint graph greedily. To be specific, if the vertical spacing is larger than the horizontal spacing between the two devices in the global placement solution, we will add an edge to the vertical constraint graph. The horizontal constraint edges can be added similarly. We will also perform transitive reduction on both horizontal and vertical constraint graphs to remove the transitive edges.

After constructing the constraint graphs, we can get a legal compact placement solution using LP in accordance with the constraint graphs. There are two sets of constraints which are topology order (non-overlap) and symmetry constraints. The topology order constraints are from the constraint graphs obtained, while the symmetry constraints are from the layout constraint extraction stage.

Finally, we will perform LP-based detailed placement to further optimize the wirelength for the given legal placement. Symmetric group constraints and design rules are also honored during this step.

B. Post-Placement Optimization

After the device locations are determined in the placement stage, the post-placement step generates the well islands, and establishes the bulk and substrate connection pins for routing.

Our well island generation supports different approaches. The polygon-based approach follows WellGAN [29], a deep neural network-guided well generation framework. A trained generative adversarial network (GAN) model provides the guidance of the well island shapes, and a refinement step is performed to legalize the layout result given the GAN guidance. In general, this approach may generate more compact layout designs. However, legalizing the polygonal well islands can be challenging, and it may degrade the post-layout circuit performance due to layout-dependent effects if handled improperly, especially in advanced process technology nodes.

In face of the challenges posed by the polygon-based well generation approach, MAGICAL also supports generating rectangular well islands for each individual devices with wells. This approach can overcome the legalization difficulty and performance effects of WellGAN, at the cost of slightly increased total area and routing complexity. In fact, for high-performance analog designs, circuit performance is often considered more important than the total area. Moreover, the number of nets in an analog circuit is usually relatively few compared with the layout area. Hence, routability is generally not a critical issue for analog circuit layouts. From this perspectives, generating individual wells for certain devices could also be a desirable practice.

After well generation, MAGICAL will automatically establish the well island and substrate connection points. Both guard rings and contacts are supported and can be inserted free from design rule violation by construction. After that, we will legalize and spread out the well islands with the consideration of design rules. Finally, the placement result in GDSII format as well as the pins of the devices, well islands, and the substrate will be output for routing.

VI. Analog Routing

After the placement and post-placement optimization stage, an end-to-end router connects all the pins with metal wires in aware of the analog circuit-specific constraints and design consideration.

In addition to connectivity and design rules, analog routing problem is also imposed with symmetric net constraints for matching [30]. In MAGICAL, the routing engine takes inputs from the layout constraint generator and honors the symmetric and self-symmetric constraints for nets. Out analog routing consists of two stages, global and detailed routing, similar to the conventional digital routing flow. In the global routing stage, the design is divided into rough grids, and the topology of routing is generated. The purpose of global routing is to efficiently find a global routing solution without extensively spending computational resources in the detailed design rule and congestion resolving. While in the detailed routing stage, the physical geometry of metal wires are implemented following the global routing guide, symmetric constraints, and design rules.

A. Global Routing

A sequential symmetry-aware grid-based A* search routing engine is employed in global routing stage.

Before performing the routing, several steps are executed on the input placement solution and the process technology

information. The placement is firstly divided into grid cells unified in size. By default, MAGICAL decides on the width and height of each grid cell based on track width on M1 layer. To be specific, the global routing will attempt to divide the layout into 3D grid so that the width and height of each grid cell are four-track widths. However, as computation efficiency is low when the number of grid cells is relatively large, MAGICAL limits the amount of grid cells to be below $200 \times 200 \times N_{layers}$, where N_{layers} denotes the number of layers. After generating the routing grid, routing capacity is calculated on 2D grid edges based on the actual grid cell width and track width.

Besides the global routing grid, the input pins are pre-processed before routing. Unlike cell-based digital physical design flow, customized analog circuit devices do not have an identical convention of pins such as layer and shape. Pin shapes could be polygons varied in size and shape, depending on the device type and parameters. In MAGICAL routing engine, the pins are decomposed into searching points in the path search scheme. The polygon is firstly separated into rectangles. The intersection points between the center lines of the resulting rectangles and the tracks of the metal layer above are identified as the search points. If no search point is identified based on the strategy above, the center points of the rectangles are considered as the search points in the A* search algorithm.

After processing the input placement, every multi-pin net is split into two-pin nets based on minimum spanning tree with HPWL as the edge costs. The two-pin net splitting is matched for symmetric net pairs so that the symmetry is maintained. Then symmetric net pairs and self-symmetric nets are routed on the 3D global routing grid with exact symmetry along the symmetric axis. A rip-up and reroute scheme is applied when failing to achieve a feasible solution in the early iterations.

B. Detailed Routing

After determining the rough routing topology in the global routing stage, detailed routing engine completes the routing and assigns metal wire geometries to each net. Similar to the global routing, a sequential A* search kernel is applied. However, instead of routing abstract nets on the rough global routing grid, detailed router explicitly handles design rules in physical layouts.

Since the input pins and previously generated search points are not necessarily aligned with the routing tracks, the detailed analog routing engine does not restrict the wire segments and VIAs to be exactly aligned with routing tracks as its digital counterpart. Instead, they only need to align to the manufacturing grid specified in the process technology design rules. The detailed router will attempt to search for the path for the nets on multiple layers with default search step of track width. The design rules of metal wires and VIAs are checked during the A* search. When routing the symmetric net pairs or self-symmetric nets, both the two sides are routed at the same time. Therefore the resulting routing solution is feasible for both sides of the symmetric axis.

After the detailed routing stage, a complete placed and routed layout is generated fully automatically, and a GDSII format layout file is exported.

Due to the complexity of analog routing, other analog circuit-specific routing considerations will also be incorporated into MAGICAL to improve the post-layout circuit performance. The preliminary results in GeniusRoute [31] demonstrate the effectiveness of our neural network guided analog routing techniques.

VII. 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 3.4GHz Intel(R) CPU and 32GB memory. The layout results are validated using Calibre DRC/LVS/PEX, and evaluated using Cadence Virtuoso ADE simulation environment.

TABLE I
Results of the comparator circuit (COMP).

Metrics	Output Delay (ps)	Input-referred Noise (μVrms)	Power (μW)	Input-referred Offset (mV)
Manual	150	380	16.8	0.15
MAGICAL	152	334	18.7	0.50

TABLE II
Results of the miller-compensated OTA (OTA1).

Metrics	Gain (dB)	UGB (MHz)	PM (degree)	Noise (μVrms)	CMRR (dB)	Offset (mV)
Manual	37.7	110.0	67.8	219.0	103.0	0.20
MAGICAL	38.0	107.5	62.3	221.5	92.5	0.48

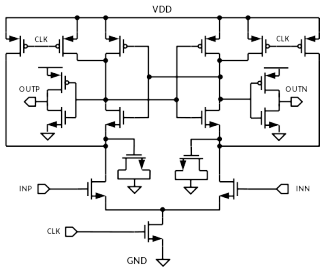
TABLE III
Results of the feed-forward compensated OTA (OTA2).

Metrics	Gain (dB)	UGB (MHz)	PM (degree)	CMRR (dB)	Offset (mV)
Manual	35.3	2200	77.6	126.1	$<$0.01
MAGICAL	35.3	2200	77.9	88.9	0.161

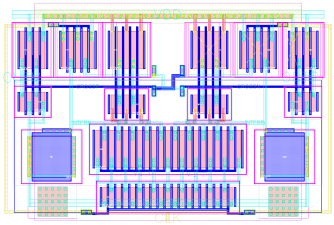
TABLE IV
Results of the inverter-based OTA (OTA3).

Metrics	Gain (dB)	UGB (MHz)	PM (degree)	CMRR (dB)	Offset (mV)
Manual	69	1300	58	94.5	0.016
MAGICAL	69	1130	56.5	110	0.001

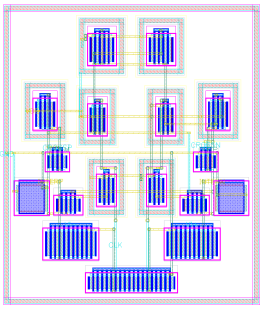
The circuit performances of the layout results generated by MAGICAL are compared against tape-out quality manual layouts by experienced analog IC designers, under the same test bench suites. The post-layout simulation results for 3 benchmark circuits, a comparator (COMP), a 2-stage miller-compensated operational transconductance amplifier (OTA1), a 2-stage feed-forward compensated OTA (OTA2), and an inverter-based OTA (OTA3), are shown in Table I, II, III, and IV, respectively, where UGB means unity-gain bandwidth, PM refers to phase margin, and CMRR is common-mode rejection ratio. The results demonstrate that MAGICAL can automatically generate validated layouts from unannotated



(a) circuit schematic

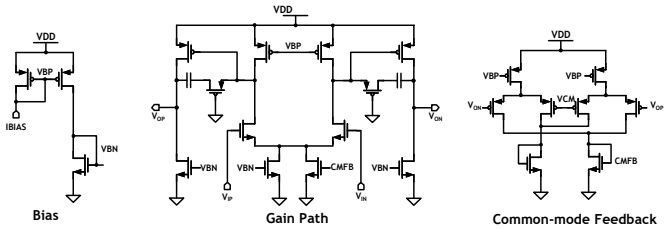


(b) manual layout

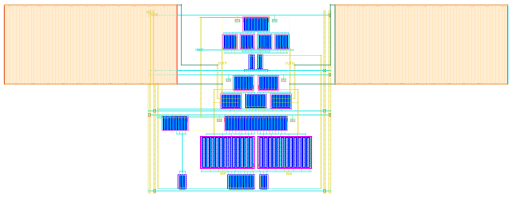


(c) layout from MAGICAL

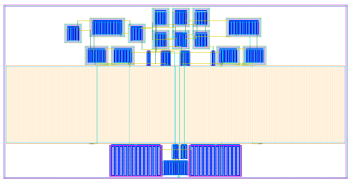
Fig. 4. Comparator circuit schematic, manual layout, and layout result from MAGICAL.



(a) circuit schematic



(b) manual layout



(c) layout from MAGICAL

Fig. 5. Miller-compensated OTA circuit schematic, manual layout, and layout result from MAGICAL.

circuit netlist (supporting both Spectre and HSPICE format), and the post-layout performances are close to the manual designs by experienced designers. 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. Figures 4, 5, 6, and 7 show the circuit schematics, the handcrafted layouts, and the layout results by MAGICAL of the benchmark circuits. Note that some device parameters of the manual layout may not necessary match the schematic after adjustment by layout designers, e.g., number of fingers for transistors, while MAGICAL generates the device layouts corresponding to the circuit netlist. Therefore, the devices in the manual layout may appear to be different from those generated by MAGICAL.

TABLE V
RUNTIME OF MAGICAL.

Circuit	COMP	OTA1	OTA2	OTA3
Runtime (s)	3.91	10.51	6.03	26.6

Table V shows the runtime for the full MAGICAL flow. As shown in the table, it can generate the complete routed layout in seconds, while manual layouts usually take hours. The fast turn-around time could significantly shorten the design cycle and facilitate the design closure.

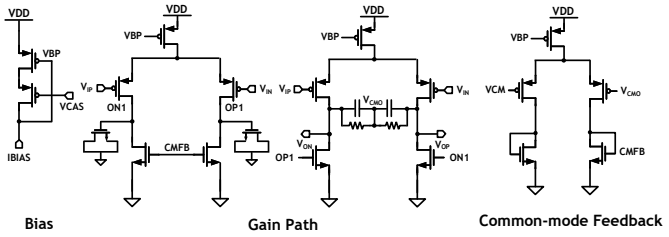
VIII. FUTURE DIRECTIONS

Future directions of MAGICAL include the followings.

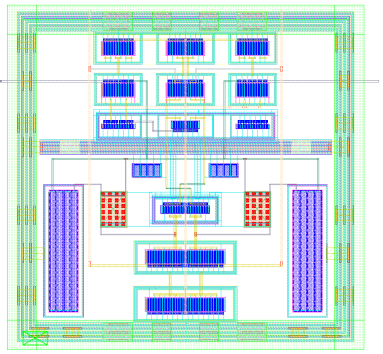
1) Interaction with Open-Sourced EDA Ecosystem: Encouraged by the DARPA IDEA/POSH program, a number of open-source projects have emerged recently [32]–[34]. Being part of the open hardware/EDA ecosystem, the future development of the MAGICAL will both benefit from and contribute to the community.

MGICAL can learn from the recent emerging open-source EDA tools. Both AMS and digital layout automation flows share many common infrastructural components with MAGICAL, e.g., OpenROAD [33] which aims to build a completely open-source digital physical design flow and ALIGN [34] which is another AMS layout generator. Although the existing components in different open-source EDA tools may have different algorithms and methodologies, there are some overlapping between their functionality. For example, ALIGN, RAIL [35], and MAGICAL all contain modules for generating the device layouts. Similarly, ALIGN, TritonRoute [36], DR. CU [37], and MAGICAL all provide different implementations of detailed router. It would be beneficial for the community to leverage some well-established efforts from each other and focus on the key differentiation. Besides the physical design framework, there are also open-source utilities that could be valuable for applying in MAGICAL. For instance, Cpp-taskflow [38] can potentially be applied in MAGICAL for improving efficiency by parallelizing the algorithm, and circuit sanitizer [39] would make sharing designs easier among the community by avoiding disclosing process design kit (PDK) related information.

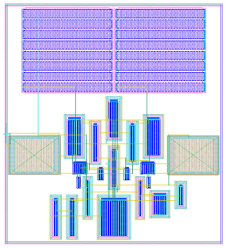
Besides the EDA tools, open-sourcing AMS circuit designs is another driving force for AMS layout automation. On one hand, lacking of training data has been a major challenge in



(a) circuit schematic



(b) manual layout



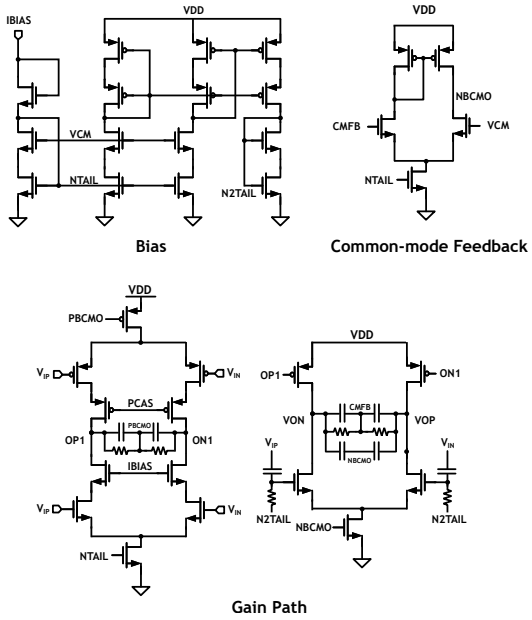
(c) layout from MAGICAL

Fig. 6. Feed-foward compensated OTA circuit schematic, manual layout, and layout result from MAGICAL.

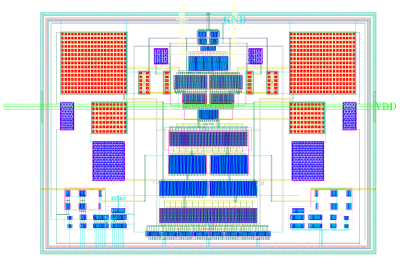
machine learning-based EDA algorithm, e.g., [29], [31]. On the other hand, the lack of a unified test circuit benchmark suite makes it difficult to evaluate and compare different AMS EDA tools. Recently, both MAGICAL and the other DARPA IDEA/POSH teams have released a number of PDK information stripped state-of-the-art AMS designs, including state-of-the-art SAR ADCs [40], ADCs [41], [42], VCO-based ADCs [43], [44], and temperature sensors [45], [46], etc. Open-source designs not only make it possible for the EDA tools to have common evaluation metrics, but also provide training data for machine learning-basd EDA algorithms.

2) Performance-aware Layout Constraint Generation, Place and Route (P&R) Flow: While MAGICAL has demonstrated satisfactory results, it currently only minimizes post-layout circuit performance degradation implicitly by considering the analog layout constraints. Approaches for directly optimizing circuit performance still need to be integrated into the current flow. In the future research and development, MAGICAL will investigate into the performance-aware techniques, especially machine learning algorithms, throughout its entire flow. As an example, in addition to the component-level symmetry constraint generation, MAGICAL will try to extract net criticality constraints and system-level symmetry from the netlists. On the other hand, with the recent advancement in neural network-guided physical design algorithms including [31], MAGICAL will further leverage human and machine intelligence in the P&R flow.

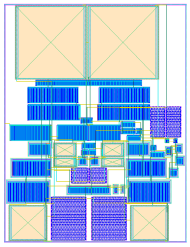
3) System-level Hierarchical Layout Optimization: Modern mixed-signal circuits, such as data converters, are hierarchical



(a) circuit schematic



(b) manual layout



(c) layout from MAGICAL

Fig. 7. Inverter-based OTA circuit schematic, manual layout, and layout result from MAGICAL.

designs with both analog and digital parts. Currently MAGICAL is focusing on the component-level analog circuit blocks such as OTAs and comparators. Although those circuits are usually among the most representative and crucial parts of the system, system-level hierarchical layout flow is still essential for the target of fully-automating AMS layout generation. To optimize the layout in the system level, floorplanning and system interconnection synthesis algorithms will also be the future development directions of MAGICAL.

IX. Conclusion

In this paper, we present 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. Future work includes further improvements on circuit performance-aware and system-level layout optimization.

ACKNOWLEDGEMENT

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

REFERENCES

[1] J. Rijmenants, J. B. Litsios, T. R. Schwarz, and M. G. Degrauwe, "Ilac: An automated layout tool for analog cmos circuits," *IEEE Journal Solid-State Circuits*, vol. 24, no. 2, pp. 417–425, 1989.

[2] J. M. Cohn, D. J. Garrod, R. A. Rutenbar, and L. R. Carley, "Koan/anagram ii: New tools for device-level analog placement and routing," *IEEE Journal Solid-State Circuits*, vol. 26, no. 3, pp. 330–342, 1991.

[3] K. Lampaert, G. Gielen, and W. M. Sansen, "A performance-driven placement tool for analog integrated circuits," *IEEE Journal Solid-State Circuits*, vol. 30, no. 7, pp. 773–780, 1995.

[4] N. Lourenço, M. Vianello, J. Guilherme, and N. Horta, "Laygen-automatic layout generation of analog ics from hierarchical template descriptions," in *2006 Ph. D. Research in Microelectronics and Electronics*. IEEE, 2006, pp. 213–216.

[5] R. Martins, N. Lourenco, and N. Horta, "Laygen ii-automatic layout generation of analog integrated circuits," *IEEE TCAD*, vol. 32, no. 11, pp. 1641–1654, 2013.

[6] K. Lampaert, G. Gielen, and W. M. Sansen, *Analog layout generation for performance and manufacturability*. Springer Science & Business Media, 2013, vol. 501.

[7] M. Strasser, M. Eick, H. Gräb, U. Schlichtmann, and F. M. Johannes, "Deterministic analog circuit placement using hierarchically bounded enumeration and enhanced shape functions," in *Proc. ICCAD*, 2008, pp. 306–313.

[8] P.-H. Lin, Y.-W. Chang, and S.-C. Lin, "Analog placement based on symmetry-island formulation," *IEEE TCAD*, vol. 28, no. 6, pp. 791–804, 2009.

[9] Q. Ma, L. Xiao, Y.-C. Tam, and E. F. Young, "Simultaneous handling of symmetry, common centroid, and general placement constraints," *IEEE TCAD*, vol. 30, no. 1, pp. 85–95, 2011.

[10] M. P.-H. Lin, Y.-T. He, V.-H. Hsiao, R.-G. Chang, and S.-Y. Lee, "Common-centroid capacitor layout generation considering device matching and parasitic minimization," *IEEE TCAD*, vol. 32, no. 7, pp. 991–1002, 2013.

[11] B. Xu, S. Li, X. Xu, N. Sun, and D. Z. Pan, "Hierarchical and analytical placement techniques for high-performance analog circuits," in *Proc. ISPD*, 2017, pp. 55–62.

[12] B. Xu, B. Basaran, M. Su, and D. Z. Pan, "Analog placement constraint extraction and exploration with the application to layout retargeting," in *Proc. ISPD*, 2018, pp. 98–105.

[13] H.-C. Ou, K.-H. Tseng, J.-Y. Liu, I.-P. Wu, and Y.-W. Chang, "Layout-dependent effects-aware analytical analog placement," *IEEE TCAD*, vol. 35, no. 8, pp. 1243–1254, 2016.

[14] M. M. Ozdal and R. F. Hentschke, "An algorithmic study of exact route matching for integrated circuits," *IEEE TCAD*, vol. 30, no. 12, pp. 1842–1855, 2011.

[15] ——, "Algorithms for maze routing with exact matching constraints," *IEEE TCAD*, vol. 33, no. 1, pp. 101–112, 2014.

[16] R. Martins, N. Loureno, A. Canelas, and N. Horta, "Electromigration-aware and ir-drop avoidance routing in analog multiport terminal structures," in *Proc. DATE*, March 2014, pp. 1–6.

[17] Q. Gao, Y. Shen, Y. Cai, and H. Yao, "Analog circuit shielding routing algorithm based on net classification," in *Proc. ISLPED*, Aug 2010, pp. 123–128.

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

[19] H. Ou, K. Tseng, J. Liu, I. Wu, and Y. Chang, "Layout-dependent effects-aware analytical analog placement," *IEEE TCAD*, vol. 35, no. 8, pp. 1243–1254, Aug 2016.

[20] M. Eick, M. Strasser, K. Lu, U. Schlichtmann, and H. E. Graeb, "Comprehensive generation of hierarchical placement rules for analog integrated circuits," *IEEE TCAD*, vol. 30, no. 2, pp. 180–193, Feb 2011.

[21] T. Massier, H. Graeb, and U. Schlichtmann, "The sizing rules method for cmos and bipolar analog integrated circuit synthesis," *IEEE TCAD*, vol. 27, no. 12, pp. 2209–2222, Dec 2008.

[22] Qinsheng Hao, Sheqin Dong, Song Chen, Xianlong Hong, Yi Su, and Zhiyi Qu, "Constraints generation for analog circuits layout," in *2004 International Conference on Communications, Circuits and Systems*, vol. 2, June 2004, pp. 1339–1343 Vol.2.

[23] P. Wu, M. P. Lin, and T. Ho, "Analog layout synthesis with knowledge mining," in *European Conference on Circuit Theory and Design (ECCTD)*, Aug 2015, pp. 1–4.

[24] B. Xu, S. Li, C.-W. Pui, D. Liu, L. Shen, Y. Lin, N. Sun, and D. Z. Pan, "Device layer-aware analytical placement for analog circuits," in *Proc. ISPD*, 2019, pp. 19–26.

[25] T.-C. Chen, Z.-W. Jiang, T.-C. Hsu, H.-C. Chen, and Y.-W. Chang, "Ntuplace3: An analytical placer for large-scale mixed-size designs with preplaced blocks and density constraints," *IEEE TCAD*, vol. 27, no. 7, pp. 1228–1240, 2008.

[26] S. Kuwabara, Y. Kohira, and Y. Takashima, "An effective overlap removable objective for analytical placement," *IEICE transactions on fundamentals of electronics, communications and computer sciences*, vol. 96, no. 6, pp. 1348–1356, 2013.

[27] W. Naylor, "Non-linear optimization system and method for wire length and delay optimization for an automatic electric circuit placer," *US Patent No. 6301693*, 2001.

[28] J. Doenhardt and T. Lengauer, "Algorithmic aspects of one-dimensional layout compaction," *IEEE TCAD*, vol. 6, no. 5, pp. 863–878, 1987.

[29] B. Xu, Y. Lin, X. Tang, S. Li, L. Shen, N. Sun, and D. Z. Pan, "Wellgan: Generative-adversarial-network-guided well generation for analog/mixed-signal circuit layout," in *Proc. DAC*, 2019, p. 66.

[30] L. Xiao, E. F. Y. Young, X. He, and K. P. Pun, "Practical placement and routing techniques for analog circuit designs," in *Proc. ICCAD*, Nov 2010, pp. 675–679.

[31] K. Zhu, M. Liu, Y. Lin, B. Xu, S. Li, X. Tang, N. Sun, and D. Z. Pan, "Geniusroute: A new analog routing paradigm using generative neural network guidance," in *Proc. ICCAD*, 2019.

[32] T.-W. Huang, C.-X. Lin, G. Guo, and M. D. F. Wong, "Essential building blocks for creating an open-source eda project," in *Proc. DAC*, 2019.

[33] T. Ajayi, V. A. Chhabria, M. Fogaça, S. Hashemi, A. Hosny, A. B. Kahng, M. Kim, J. Lee, U. Mallappa, M. Neseem, G. Pradipta, S. Reda, M. Saligane, S. S. Sapatnekar, C. Sechen, M. Shalan, W. Swartz, L. Wang, Z. Wang, M. Woo, and B. Xu, "Toward an open-source digital flow: First learnings from the openroad project," in *Proc. DAC*, 2019, pp. 76:1–76:4.

[34] K. Kunal, M. Madhusudan, A. K. Sharma, W. Xu, S. M. Burns, R. Harjani, J. Hu, D. A. Kirkpatrick, and S. S. Sapatnekar, "Align: Open-source analog layout automation from the ground up," in *Proc. DAC*, 2019, pp. 77:1–77:4.

[35] "RAIL12," https://github.com/uwidea/rail12, accessed: 2019-8-1.

[36] A. B. Kahng, L. Wang, and B. Xu, "Tritonroute: An initial detailed router for advanced vlsi technologies," in *Proc. ICCAD*, 2018, pp. 1–8.

[37] G. Chen, C.-W. Pui, H. Li, J. Chen, B. Jiang, and E. F. Y. Young, "Detailed routing by sparse grid graph and minimum-area-captured path search," in *Proc. ASPDAC*, 2019, pp. 754–760.

[38] T.-W. Huang, C.-X. Lin, G. Guo, and M. D. F. Wong, "Cpp-taskflow: Fast task-based parallel programming using modern c++," in *International Parallel and Distributed Processing Symposium (IPDPS)*, 2019.

[39] "Sanitizer," https://github.com/USCPOSH/Sanitizer, accessed: 2019-8-1.

[40] X. Tang, L. Chen, J. Song, and N. Sun, "A 1.5fj/conv-step 10b 100ks/s sar adc with gain-boosted dynamic comparator," in *Proc. ASSCC*, Nov 2017, pp. 229–232.

[41] J. Liu, S. Li, W. Guo, G. Wen, and N. Sun, "A 0.029mm2 17-fj/conv.-step ct $\delta\sigma$ adc with 2nd-order noise-shaping sar quantizer," in *Proc. VLSI*, June 2018, pp. 201–202.

[42] "MAGICAL-CIRCUITS," https://github.com/magical-eda/MAGICAL-CIRCUITS, accessed: 2019-8-1.

[43] M. Hassanpourghadi, P. K. Sharma, and M. S. Chen, "A 6-b, 800-ms/s, 3.62-mw nyquist rate ac-coupled vco-based adc in 65-nm cmos," *IEEE TCAS I*, vol. 64, no. 6, pp. 1354–1367, June 2017.

[44] "AMPSE," https://github.com/USCPOSH/AMPSE, accessed: 2019-8-1.

[45] A. Wang, C. Chen, and C. R. Shi, "A 9-bit resistor-based all-digital temperature sensor with a sar-quantization embedded differential low-pass filter in 65nm cmos consuming 57pj with a 2.5 μs conversion time," in *Proc. CICC*, 2019.

[46] "UW-IDEA_AnalogTestCases," https://github.com/uwidea/UW-IDEA_AnalogTestCases, accessed: 2019-8-1.