Automating Analog Constraint Extraction: From Heuristics to Learning

(Invited Paper)

Keren Zhu, Hao Chen, Mingjie Liu and David Z. Pan
ECE Department, The University of Texas at Austin, Austin, TX, USA
{keren.zhu, haoc, jay_liu}@utexas.edu, dpan@ece.utexas.edu

2022 27th Asia and South Pacific Design Automation Conference (ASP-DAC) | 978-1-6654-2135-5/22/$31.00 ©2022 IEEE | DOI: 10.1109/ASP-DAC52403.2022.9712488

***Abstract*—Analog layout synthesis has recently received much attention to mitigate the increasing cost of manual layout efforts. To achieve the desired performance and design specifications, generating layout constraints is critical in fully automated netlist-to-GDSII analog layout flow. However, there is a big gap between automatic constraint extraction and constraint management in analog layout synthesis. This paper introduces the existing constraint types for analog layout synthesis and points out the recent research trends in automating analog constraint extraction. Specifically, the paper reviews the conventional graph heuristic methods such as graph similarity and the recent machine learning approach leveraging graph neural networks. It also discusses challenges and research opportunities.**

I. Introduction

The demand for analog and mixed-signal (AMS) integrated circuits (ICs) has increased, stimulated by many emerging applications, such as 5G networks, neuromorphic computing, and the Internet of Things. While shortening the turnaround time of AMS design is desired, AMS circuit design is still mainly a manual, time-consuming, and error-prone task. As a critical step in AMS circuit design, layout design in a typical flow has limited automation involved. Automating layout design for AMS circuits has become increasingly demanded in the modern IC industry.

The performance of modern AMS designs is vulnerable to parasitics, process variations, and layout-dependent effects because of the sensitivity of AMS circuit architectures. Manual layout design often applies special layout strategies to mitigate the sensitivity in performance to achieve robustness [1]. On the other hand, automatic analog place and route (PNR) algorithms usually formulate constrained optimization problems to mimic such manual layout practice. For example, automated PNR frameworks commonly consider geometrical matching constraints (e.g., symmetry, regularity) on modules and nets [2]. However, annotating detailed constraints itself takes effort and time. To resolve it, researchers have proposed techniques to automate analog constraint extraction (ACE).

In a typical automated AMS layout synthesis flow, constraints are manually annotated. They are treated as inputs to the downstream automated placement and routing frameworks, as shown in Figure 1 (a). On the other hand, ACE-enhanced flow replaces the manual constraint labeling step with an automated process, as illustrated in Figure 1 (b). An ACE engine takes unannotated circuit schematic design as inputs and generates placement and routing constraints. The extracted

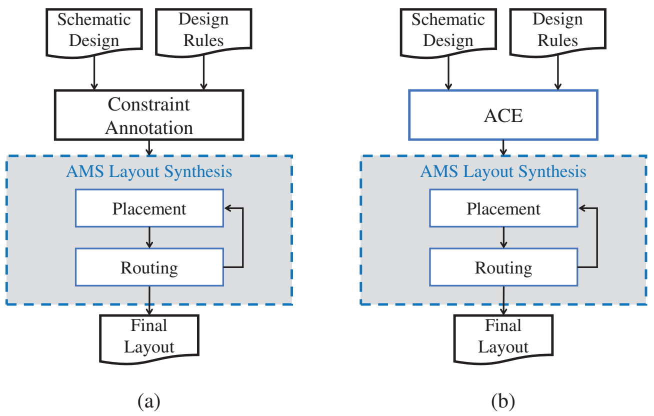


Fig. 1: PNR flow for AMS layout synthesis. (b) Conventional flow with manual constraint annotation. (b) Fully-automated flow with ACE.

constraints are then fed to the PNR engines and finally result in the layout. The whole process of the ACE-enhanced flow is fully-automated and does not require the involvement of humans. The development of ACE algorithms enables the AMS layout synthesis to be fully-automated and accelerate the design cycle.

PNR constraints for AMS circuits are closely related to the circuit architecture. For example, symmetry constraints are often needed for particular circuit substructures, such as symmetry pairs and current mirrors. Therefore, the ACE problem is closely related to the knowledge mining on netlist topology and, to some degree, requires understanding the circuit architecture. Conventionally, researchers build pattern libraries and apply heuristic methods to identify essential substructures from netlists. However, such a method often needs intensive efforts to develop the pattern library and has challenges generalizing the knowledge. Therefore, recent research has focused on principal techniques in extracting analog constraints. Machine learning (ML) especially plays a vital role in recent developments of ACE algorithms. Leveraging ML, researchers propose techniques to automatically learn analog constraints from existing circuits and apply learned knowledge to new designs. The purpose of this paper is to summarize the recent research progress in this direction and provide insights on the challenges and opportunities on ACE for future studies.

The rest of this paper is organized as follows. Section II introduces the background on the ACE problem and reviews the conventional approaches. Section III presents the recent ACE approaches with emphasis on the ongoing trend of phasing into ML-based methods. Section IV highlights the future research directions with an analysis of challenges and opportunities. Finally, Section V concludes this paper.

II. Background and Conventional Approaches

In this section, we first introduce the ACE problem in Section II-A. Then we review the conventional approaches II-B. We also introduce the preliminaries of graph neural networks (GNN) II-C

A. Constraint Extraction for AMS Circuits

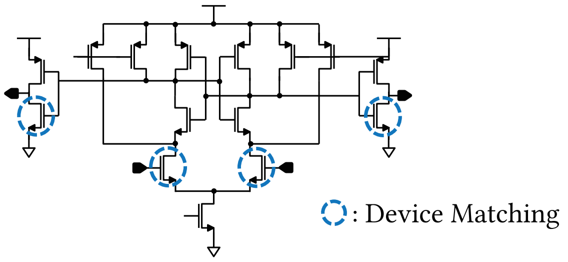
An ACE framework extracts constraints for the downstream PNR flow. Therefore, the types of constraints interested in the ACE problem closely depend on the targeting PNR formulations. The targeting constraints can range from symmetry and proximity to the current path and thermal constraints [3]. Some of those constraints are straightforwardly implied from the circuit netlists. For example, current paths can be directly inferred by tracing the paths of DC currents. On the other hand, many other constraints require a deeper understanding of the circuit architectures. Extracting this kind of constraint is more challenging and is the main focus of research in ACE.

The variants of matching constraints are the most targeting type in existing ACE research. Many analog circuits heavily rely on differential architectures to function. Therefore, matching between instances, sub-circuits and nets is critical to the circuit performance, as it avoids or reduces the mismatch on differential signals. Figure 2 shows examples of matching in AMS circuits. Pairs of differential devices often need to be matched (Figure 2 (a)). On the other hand, pairs of sub-circuits along critical differential signals are under the matching constraint (Figure 2 (b)). In the layout implementation, matching can be translated into several geometric constraints, such as symmetry, common centroid, and interdigitation, to reduce layout mismatch. Recent developments of ACE are mainly focusing on detecting matching constraints.

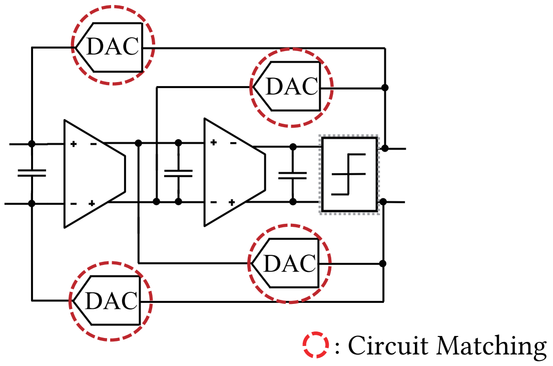
B. Conventional Approaches

Researchers have proposed simulation-based constraint generation. Sensitive analysis has been adopted to detect matching constraints [4]–[6]. Leveraging circuit simulation, the sensitive analysis identifies the critical components to the performance and generates constraints based on them. This simulation-involved approach is capable of automatically identifying high-quality constraints. However, as simulations are costly in runtime, such methodology is limited to block-level circuits and is hard to apply to larger scales. Most existing ACE algorithms are simulation-free.

An alternative approach is to use varied heuristics to generate the constraints. The work [7] extracts the symmetry



(a)



(b)

Fig. 2: Examples of matching constraints for AMS circuits. (b) Device-level instance constraints. (b) System-level sub-circuit constraints.

constraints based on analyzing the mismatch in circuits. Kole et al. [8] identify the constraints by finding the symmetrical connection trees in a circuit graph. Researchers have proposed to identify the circuit building blocks and generate constraints based on them [9]–[11]. However, such heuristics take developing efforts and are challenging to generalize. Several studies propose matching sub-graphs in-circuit netlists with existing designs. [12]–[14]. Then this methodology can infer the constraints on new designs by mining the knowledge in labeled constraints from the knowledge database. These methods first convert the circuit netlists into graphs and then search the sub-graph in the design database. However, because identifying graph isomorphism exactly is unaffordable in runtime, the proposed techniques use heuristic algorithms to compare graphs. The knowledge mining idea bridge the human design expertise with automated AMS layout synthesis. However, it requires exact matching between new designs and patterns in the design knowledge database. Given the high degree of freedom in AMS circuit design, it is challenging to build a design pattern database covering the needs for new designs.

Both the heuristic constraint generation and knowledge mining methodologies face the challenge of generalization. Researchers have recently focused on resolving this issue by

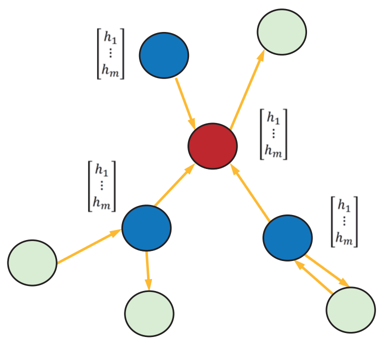


Fig. 3: A ConvGNN layer aggregates the neighboring nodes embedding.

leveraging approximation graph matching algorithms and alternative classification criteria. Especially, several ML-assisted techniques have advanced the research in the ACE problem.

C. *Graph Neural Network*

GNN has been a fast-growing topic in ML research. Unlike from many applications, such as images and text, where data can be presented in Euclidean space, graph data are more challenging to learning tasks. While there are many different types of GNN, convolutional graph neural network (ConvGNNs) has been a popular category of GNN architectures [15]. ConvGNNs have also demonstrated effectiveness in various CAD applications, such as parasitic prediction [16], layout decomposition [17], analog perforamce prediction [18] and logic synthesis [19].

ConvGNNs perform *convolution* on graph structure to obtain new node embedding. For the node *v*, a graph convolution operation aggregates the current embedding or feature of *v*'s neighboring nodes and perform transform the resulting vector, as illustrated in Figure 3. A typical graph convolution layer is shown in the Equation (1),

$$H^{l+1} = \sigma(\hat{D}^{-\frac{1}{2}}\hat{A}\hat{D}^{-\frac{1}{2}}H^{l}W^{l}), \tag{1}$$

where $H^{(l)}$ is the l^{th} layer output, $\sigma(\cdot)$ is the activation function, $\hat{A}$ is the sum of adjacent matrix A and identity matrix I, $\hat{D}$ is the diagonal matrix of $\hat{A}$ and W^l is the weight of l^{th} layer. After graph convolution layers, the node embedding can be used for downstream prediction tasks, such as graph classification and node-wise classification.

III. Recent developments

In this section, we review the recent studies on the ACE problem [20]–[23]. They are targeting the same type of constraint: the matching constraints. Their algorithms differ and range from graph matching to supervised graph learning. However, they are all attempting to resolve the generalization issue in the heuristic and knowledge mining-based ACE algorithms. Liu et al. [20] propose to use graph similarity as the criterion to judge whether two sub-circuits need to be matched. The statistics-inspired graph similarity enables a generalized way to compare two graphs without being limited to exact matching. Kunal et al. [21] adopt a similar idea and use graph edit distance (GED) as the graph similarity metric. Chen et al. [23] extend the graph similarity idea and propose a universal way to determine the device and circuit matching. An unsupervised graph learning scheme is proposed to embed the devices and sub-graphs into the vector domain. On the other hand, Gao et al. [22] apply ConvGNN to the knowledge mining methodology. Casting the ACE problem as a supervised classification problem, the work predicts the constraints on annotated netlists by inferring learned knowledge from the training dataset. In the rest of this section, we present these studies in detail.

A. *S^3DET: Subgraph Extraction and Graph Similarity*

The framework S^3DET [20] is proposed to extract the system-level circuit symmetry constraints. It assumes the criterion for detecting such matching is (1) the two sub-circuits must be similar in structure, and (2) the two circuits shall have similar neighbors. The first assumption is commonly used in conventional ACE frameworks, while the second is often not considered. In a typical AMS design, the matching circuit pairs are often symmetric along with critical signals. However, similar sub-circuits do not necessarily need symmetry constraints. For example, the four digital-analog converters (DACs) in Figure 1 (b) are divided into two symmetry groups instead of adding a constraint to every pair of the permutation. The example demonstrates, in addition to the similarity of internal structure, the similarity of neighbors also matters when labeling the constraints.

To consider the neighboring topology, S^3DET extracts the sub-circuits neighboring graph and then measures the graph similarity. It first translates the circuit netlist to a graph. When judging a pair of sub-circuits, it extracts the sub-graphs near the two sub-circuits, including the sub-circuits and the nearby structures. It compares the two sub-graphs and measures their topological similarity. If the graph similarity is larger than a threshold, S^3DET labels this pair as a symmetry pair.

Since the neighboring topology might be slightly different for a valid symmetry pair, the ideal graph similarity is preferred to be approximate and tolerable to a small mismatch. S^3DET uses a statistical method to provide such a graph similarity metric. It calculates the eigenvalues of the graph Laplacian and uses Kolmogorov-Smirnov (K-S) tests on the eigenvalues to score the graph similarity. The K-S statistic of two distribution functions $F_{1,n}(x)$ and $F_{2,m}(x)$ with sample size n and m is defined as.

$$D_n = \sup_x |F_{1,n}(x) - F_{2,m}(x)|. \tag{2}$$

It quantifies the difference between the two distributions and provides a tool to measure the difference of graphs. The proposed method is more computationally efficient than solving graph isomorphism. It also demonstrates higher accuracy against the heuristic-based ACE algorithm.

B. Accelerated Graph Similarity with Graph Neural Network

The work [21], on the other hand, leverages the graph neural network to assist the computation of graph similarity. Their framework traverses the netlists and identifies the candidate symmetry pairs. The symmetry pairs can be devices or sub-circuits. If a candidate pair are two devices, the proposed framework compares the parameters and labels as symmetry if their parameters are identical. On the other hand, if the candidate is a pair of sub-circuits, the two sub-circuits are translated into two graphs. The framework then compares their topology by evaluating their graph similarity. The graph is bipartite and consists of the net and instance nodes. The interconnection from the netlist is represented by the edges between instance nodes and net nodes. A three-bit label is given to each edge to distinguish transistors' different terminal types (drain, source, and gate). The graph similarity is measured as the graph edit distance (GED) between two graphs.

However, computing GED is NP-complete and often is not affordable in practice. The work proposes to leverage a ConvGNN to accelerate the process. It first uses an attention-based aggregation to obtain the graph embedding from ConvGNN node embedding outputs. Then the two graph embeddings are fed to another network to predict the GED between the two graphs. The network is trained in a supervised manner. This work proposes an alternative way to compute the graph similarity.

C. Supervised Graph Learning

Different from leveraging graph similarity, Gao et al. [22] propose to infer device-level symmetry via supervised graph learning. They label the symmetry constraints in a dataset and train a ConvGNN model to classify whether two nodes are symmetrical or not. When inference on unseen circuits, they translate the netlist into a graph and apply the trained model. The outputs from the classifier are treated as extracted symmetry constraints.

When constructing the graph, each device is decomposed into pins and devices. For example, a PMOS transistor is represented as one device node and three-pin nodes: the gate, the drain, and the source. The net between pins is constructed as a clique of edges. Each node contains a vector of features: (1) two-dimensional 0-1 vector denoting whether the node is a pin or device, (2) thirteen-dimensional one-hot vector denoting the device type, and (3) a real number describing the distance from this node to the ground. A GraphSage [24] model is utilized for graph neural network. The loss function is binary cross-entropy. Experimental results demonstrate the proposed method outperforms the heuristic symmetry detection method [10] in predicting accuracy.

The ConvGNN layers aggregate the neighboring information into the node embedding and make the feature transformation operation learnable. Since the symmetry constraints are primarily determined by similarity, the ConvGNN in some sense learns a way to determine node-wise similarity. However, supervised learning enables the ConvGNN to learn how to discriminate the false alarms utilizing the ground truths in the training set. For example, supervised learning can possibly learn it through data if two devices with identical surrounding structures do not need symmetry constraints. On the other hand, it is more challenging to be discriminated against with pre-defined similarity criteria. The proposed supervised learning methodology, therefore, enables more flexibility in the ACE problems.

D. Graph Similarity Considering Sizing With Unsupervised Graph Learning

The work [23] leverages ConvGNN and proposes a universal method for detecting both the device-level and sub-circuit-level symmetry constraints. As illustrated in [20], the criterion for symmetry is the similarity between the device/sub-circuit pairs and their neighbors. The "similarity," on the other hand, is twofold. First, the similar indicates the circuit structures are similar. Second, the similarity is also measured by the device sizing. In S^3DET [20], the proposed method extracts the neighbors and compares two sub-circuit graphs by their topological structures. However, this method cannot be extended to device-level constraint detection and does not consider the internal device sizing in the sub-circuit. On the other hand, the work [21] compares the sizing but misses the neighboring topological structures on the device level. When extracting sub-circuit-level constraints, the proposed method only measures the distance between graph structures without the sizing information.

Chen et al. [23] a new methodology to bridge the gap between the device-level and sub-circuit-level ACE problem. As introduced in Sec. II-C, ConvGNNs aggregate neighboring node features with the convolution operation. After stacking multiple graph convolutional layers, each node embedding contains the information from its neighboring sub-graph. In some sense, this process is similar to the sub-graph extraction scheme proposed in S^3DET. In the meantime, the graph convolution operations also distinguish different graph topology and node features. The proposed framework in [23] utilizes this mechanism. For a circuit-under-test, the proposed framework first convert the netlist into a directed graph, Each node in the graph contains the device sizing information as initial node features. Then the graph is fed into a trained gated graph sequence Neural Network [25] as shown in Equation 3.

$$h_v^{(k)} = GRU\big(h_v^{(k-1)}, \sum_{u \in \mathcal{N}_{in}(v)} W_{e_{uv}} h_u^{(k-1)}\big), \tag{3}$$

where $h_v^{(k)}$ symbolizes the feature vector of vertex v at the k^{th} layer of the GNNs, $GRU(\cdot,\cdot)$ is the computation function of a gated recurrent unit, $\mathcal{N}_{in}(v)$ signifies the in-neighbors of v, and $W_{e_{uv}}$ is the linear transformation matrix corresponding to the edge e_{uv}. When detecting symmetry constraint device-level pairs, the framework takes the two resulting node embedding vectors and computes their cosine similarity. The proposed method first uses mean aggregation on the device node embedding vectors for sub-circuit-level ACE to obtain the circuit embedding. Then the same cosine-similarity-based symmetry criterion can be applied to sub-circuit comparison.

In other words, the proposed method provides a universal ACE scheme for both the device-level and sub-circuit-level tasks.

The ConvGNN has several trainable parameters in each layer. These parameters determine how the node embedding is transformed. The target is to filter the similar nodes/sub-circuits from the graph, and the ideal network parameters ought to make the resulting node embedding vectors dissimilar to irrelevant nodes. The work uses an unsupervised training scheme to achieve this target. The learning framework takes the neighbor nodes as positive examples, as they have similar neighboring structures, and train the network so that their resulting node embedding vectors are similar. On the other hand, the learning framework also takes negative samples and maximizes their dissimilarity to help filter out the irrelevant pairs in the inference stage. The resulting loss function can be represented as follows in Equation 4.

$$\begin{aligned}\mathcal{L}(z_v) = &- \sum_{u \in \mathcal{N}_{in}(v)} \log(\sigma(z_u^\intercal z_v)) \\ &- \sum_{i=1}^{B} \mathbb{E}_{\tilde{u} \sim Neg(v)} \log(1 - \sigma(z_{\tilde{u}}^\intercal z_v)),\end{aligned} \tag{4}$$

where z_v is the output embedding of a node v, $\sigma(x) = 1/(1+e^{-x})$ is the logistic sigmoid function, $Neg(v)$ is the negative sampling distribution with respect to v, and B is the total number of negative samples. The negative samples are randomly selected from the graph.

This work transfers ConvGNN into an automatically learned heuristic for computing the similarity between graph nodes. The experiment results demonstrate it improves accuracy and computational efficiency compared to the proposed graph similarity scheme in S^3DET.

IV. Challenges and Opportunities

The existing ACE methods have sophisticatedly studied the symmetry constraint detection problem. Especially with dedicated similarity-based criteria, the state-of-the-art framework has demonstrated high accuracy in the task. However, there are challenges and opportunities in extending the success in symmetry detection into generalized ACE problems. This section discusses several unsolved open questions towards a universal paradigm for extracting AMS PNR constraints leveraging ML techniques.

Learning Data for Supervised Training: The possible solution to extracting other constraint types is following the supervised learning scheme similar to the work [22]. By providing the ML model with examples of those constraints, the ML model can potentially learn how to detect them from the netlist. However, there are several challenges to this methodology. First, annotating data take manual effort. The training data are likely to consist of netlists and labeled constraints on the instances. On the other hand, supervised learning usually requires a large dataset. How to collect such a dataset is an unanswered question in the field. Second, the AMS PNR constraints are not always very defined. For example, different from symmetry routing, which is generally appreciated, many manual routing strategies are design-dependent and ambiguous [26]. Different designers and different technologies can have a significant impact on how constraints shall be labeled. In the absence of universal ground truth, learning the analog layout constraints through supervised training is challenging. Third, different circuit architectures also have different layout considerations. AMS circuit designs are strongly tied to their functionalities. The layout effects on circuit performance are therefore design-dependent. It is questionable to argue that a common AMS constraint dataset is suitable for each circuit type inference.

One possible solution to some of these challenges is to leverage few-shot learning. Few-shot learning is a fast-growing ML research area that attempts to achieve good accuracy with a small amount of training data [27]. For example, the ML model can be trained on a large general constraint dataset and then be fine-tuned on a specific small dataset to achieve higher accuracy for a particular constraint type or circuit architecture. A similar approach has been seen in analog performance prediction [28].

Limitation on Graph Neural Network: While GNN has demonstrated success in the field, there are still several challenges for the GNN algorithm applying to circuits. First, the capability of GNN on learning the circuit structures is not well unveiled. Researchers have noticed that the mechanism of some GNN variants is similar to filters on the graph spectral domain [29]. A study also proposes a GNN variant for graph structural learning, demonstrating equivalent representation power as the Weisfeiler-Lehman graph isomorphism test [30]. As the circuit's functionality is strongly determined by its connection topology, it is an unanswered question whether the exiting GNN models can capture the knowledge from circuit structures.

Second, most existing GNN variants are not position-aware. ConvGNN performs convolution operations on neighboring nodes. Therefore, the learned knowledge has a strong locality and sometimes loss the capture of the global view. For example, conventional ConvGNN can hardly distinguish locally isomorphic sub-structures [31]. In Figure 4, the nodes v_1 and v_2 are on the two branches in the graph. They have the same neighboring structures, and a conventional 2-layer ConvGNN cannot distinguish them since their 2-hop neighbors are identical. As a result, the final node embedding of v_1 and v_2 are unavoidable identical. However, it is problematic when applying to many applications that require capturing the position information.

Interaction with Layout Synthesis Tools: Interaction between ACE and automated AMS layout synthesis tools is also an opportunity in the field. After producing the initial analog constraint set, the ACE framework can probably get feedback from the PNR flow to fine-tune the constraints. It is precious when the detection scheme is not perfect. There has been related work integrating simulation in the PNR flow to ensure the performance is satisfying [32]. Enabling a synergistic framework to integrate the ACE and layout synthesis tool is an opportunity in the field.

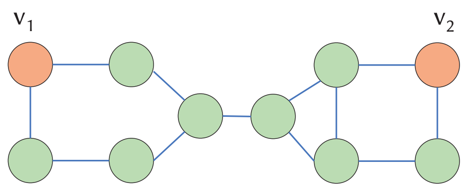


Fig. 4: An example of a locally isomorphic graph. Nodes $v1$ and v_2 are non-distinguishable for a conventional 2-layer ConvGNN.

V. Conclusion

The paper presents an overview of automatic constraint extraction for analog and mixed-signal circuit layout synthesis. We review the conventional approaches and survey the latest advancements leveraging the machine learning techniques that address those open questions. We also present our perspectives on the challenges and opportunities in the field.

Acknowledgement

This work is supported in part by NSF under grants 1704758 and 2112665, and by DARPA under the IDEA program.

References

[1] A. Hastings, *The Art of Analog Layout*. Prentice, 2005.
[2] H. Chen, M. Liu, X. Tang, K. Zhu, N. Sun, and D. Z. Pan, "Challenges and opportunities toward fully automated analog layout design," *Journal of Semiconductors*, vol. 41, no. 20070021, p. 111407, 2020.
[3] M. P.-H. Lin, Y.-W. Chang, and C.-M. Hung, "Recent research development and new challenges in analog layout synthesis," in *Proc. ASPDAC*, 2016.
[4] U. Choudhury and A. Sangiovanni-Vincentelli, "Constraint generation for routing analog circuits," in *Proc. DAC*, 1990.
[5] E. Charbon, E. Malavasi, and A. Sangiovanni-Vincentelli, "Generalized constraint generation for analog circuit design," in *Proc. ICCAD*, 1993.
[6] E. Malavasi, E. Charbon, E. Felt, and A. Sangiovanni-Vincentelli, "Automation of ic layout with analog constraints," *IEEE TCAD*, vol. 15, no. 8, pp. 923–942, 1996.
[7] J. Liu, S. Dong, X. Hong, Y. Wang, O. He, and S. Goto, "Symmetry constraint based on mismatch analysis for analog layout in soi technology," in *Proc. ASPDAC*, 2008.
[8] M. Kole, J. Smit, and O. Herrmann, "Modeling symmetry in analog electronic circuits," in *Proc. ISCAS*, 1994.
[9] 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, 2011.
[10] B. Xu, K. Zhu, M. Liu, Y. Lin, S. Li, X. Tang, N. Sun, and D. Z. Pan, "MAGICAL: Toward fully automated analog IC layout leveraging human and machine intelligence," in *Proc. ICCAD*, 2019.
[11] Z. Zhou, S. Dong, X. Hong, Q. Hao, and S. Chen, "Analog constraints extraction based on the signal flow analysis," in *Proc. ASICON*, 2005.
[12] S. Bhattacharya, N. Jangkrajarng, R. Hartono, and C.-J. Shi, "Hierarchical extraction and verification of symmetry constraints for analog layout automation," in *Proc. ASPDAC*, 2004.
[13] P.-H. Wu, M. P.-H. Lin, T.-C. Chen, C.-F. Yeh, X. Li, and T.-Y. Ho, "A novel analog physical synthesis methodology integrating existent design expertise," *IEEE TCAD*, vol. 34, no. 2, pp. 199–212, 2015.
[14] P.-H. Wu, M. P.-H. Lin, and T.-Y. Ho, "Analog layout synthesis with knowledge mining," in *European Conference on Circuit Theory and Design (ECCTD)*, 2015.
[15] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, "A comprehensive survey on graph neural networks," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 1, pp. 4–24, 2021.
[16] H. Ren, G. F. Kokai, W. J. Turner, and T.-S. Ku, "Paragraph: Layout parasitics and device parameter prediction using graph neural networks," in *Proc. DAC*, 2020.
[17] W. Li, J. Xia, Y. Ma, J. Li, Y. Lin, and B. Yu, "Adaptive layout decomposition with graph embedding neural networks," in *Proc. DAC*, 2020.
[18] Y. Li, Y. Lin, M. Madhusudan, A. Sharma, W. Xu, S. Sapatnekar, R. Harjani, and J. Hu, "A customized graph neural network model for guiding analog ic placement," in *Proc. ICCAD*, 2020.
[19] K. Zhu, M. Liu, H. Chen, Z. Zhao, and D. Z. Pan, "Exploring logic optimizations with reinforcement learning and graph convolutional network," in *Proceedings of the ACM/IEEE Workshop on Machine Learning for CAD*, 2020.
[20] M. Liu, W. Li, K. Zhu, B. Xu, Y. Lin, L. Shen, X. Tang, N. Sun, and D. Z. Pan, "S^{3}DET: Detecting system symmetry constraints for analog circuits with graph similarity," in *Proc. ASPDAC*, 2020.
[21] K. Kunal, P. Poojary, T. Dhar, M. Madhusudan, R. Harjani, and S. Sapatnekar, "A general approach for identifying hierarchical symmetry constraints for analog circuit layout," in *Proc. ICCAD*, 2020.
[22] X. Gao, C. Deng, M. Liu, Z. Zhang, D. Z. Pan, and Y. Lin, "Layout symmetry annotation for analog circuits with graph neural networks," in *Proc. ASPDAC*, 2021.
[23] H. Chen, K. Zhu, M. Liu, X. Tang, N. Sun, and D. Z. Pan, "Universal symmetry constraint extraction for analog and mixed-signal circuits with graph neural networks," in *Proc. DAC*, 2021.
[24] W. L. Hamilton, R. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Proc. NIPS*, 2017.
[25] Y. Li, R. Zemel, M. Brockschmidt, and D. Tarlow, "Gated graph sequence neural networks," in *Proc. ICLR*, 2016, pp. 1–20.
[26] 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.
[27] Y. Wang, Q. Yao, J. T. Kwok, and L. M. Ni, "Generalizing from a few examples: A survey on few-shot learning," *ACM Comput. Surv.*, vol. 53, no. 3, Jun. 2020.
[28] M. Liu, K. Zhu, J. Gu, L. Shen, X. Tang, N. Sun, and D. Z. Pan, "Towards decrypting the art of analog layout: Placement quality prediction via transfer learning," in *Proc. DATE*, 2020.
[29] F. Wu, A. Souza, T. Zhang, C. Fifty, T. Yu, and K. Weinberger, "Simplifying graph convolutional networks," in *Proc. ICML*, 2019.
[30] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?" in *Proc. ICLR*, 2019.
[31] J. You, R. Ying, and J. Leskovec, "Position-aware graph neural networks," in *Proc. ICML*, 2019.
[32] M. Liu, K. Zhu, X. Tang, B. Xu, W. Shi, N. Sun, and D. Z. Pan, "Closing the design loop: Bayesian optimization assisted hierarchical analog layout synthesis," in *Proc. DAC*, 2020.