# The ALIGN Open-Source Analog Layout Generator: v1.0 and Beyond (Invited talk)

Tonmoy Dhar
University of Minnesota

Kishor Kunal
University of Minnesota

Yaguang Li
Texas A&M University

Yishuang Lin
Texas A&M University

Meghna Madhusudan
University of Minnesota

Jitesh Poojary
University of Minnesota

Arvind K. Sharma
University of Minnesota

Steven M. Burns
Intel Labs

Ramesh Harjani
University of Minnesota

Jiang Hu
Texas A&M University

Parijat Mukherjee
Intel Labs

Soner Yaldiz
Intel Labs

Sachin S. Sapatnekar
University of Minnesota

## CCS CONCEPTS

**• Hardware → Electronic design automation**; **Physical design (EDA)**; **Analog and mixed-signal circuit optimization**; **• Computing methodologies → Machine learning**; **Neural networks**.

## KEYWORDS

Analog layout automation, machine learning

**ACM Reference Format:**
Tonmoy Dhar, Kishor Kunal, Yaguang Li, Yishuang Lin, Meghna Madhusudan, Jitesh Poojary, Arvind K. Sharma, Steven M. Burns, Ramesh Harjani, Jiang Hu, Parijat Mukherjee, Soner Yaldiz, and Sachin S. Sapatnekar. 2020. The ALIGN Open-Source Analog Layout Generator: v1.0 and Beyond (Invited talk). In *IEEE/ACM International Conference on Computer-Aided Design (ICCAD '20), November 2–5, 2020, Virtual Event, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3400302.3415784

## 1 INTRODUCTION

Automating analog layout is a long-standing research problem, with a history that goes back several decades. While digital design is largely automated today, analog layout has been significantly more resistant: automation has not made much headway in industry settings. There are several reasons for this, including:

- The complexity of geometric constraints: Analog layouts require precise matching and symmetry, and layout engineers use clever techniques based on human intuition and expertise. To match this, algorithmic methods must cover a large search space, and this has been infeasible in the past.
- The absence of unifying performance metrics: unlike digital designs, which are characterized by power, performance, and area (PPA), the performance specifications of analog circuits are different for each class of circuits.
- The wide variety of circuit classes and topologies: there is no equivalent for the relatively compact standard cell libraries used for digital design, and even basic analog building blocks can be constructed in a large variety of ways.

Previous attempts at research in automating analog design have been successful in optimizing a set of fixed structures within a class, e.g., operational transconductance amplifiers (OTAs), low-noise amplifiers (LNAs), and voltage-controlled oscillators (VCOs), but a general methodology for layout generation, broadly applicable to a large class of analog circuits, has remained out of reach.

The ALIGN (Analog Layout, Intelligently Generated from Netlists) project attempts to remedy these limitations using a variety of strategies: (a) it recognizes geometric constraints and subcircuits through automated circuit recognition and annotation; (b) it translates performance specifications to layout constraints that can be used to limit layout parasitics; (c) it creates a unifying methodology that can be targeted to a wide range of analog circuits.

Specifically, the circuits targeted by ALIGN fall into four classes, each of which consists of different types of subcircuits and emphasizes different sets of constraint classes:

- Low-frequency components, e.g., analog-to-digital converters (ADCs), amplifiers, and filters.
- Wireline components, e.g., equalizers, clock/data recovery circuits, and phase interpolators.
- RF/Wireless components, e.g., components of RF transmitters and receivers.
- Power delivery components, e.g., capacitor- and inductor-based DC–DC converters and low dropout (LDO) regulators.

The ALIGN team brings together expertise from both academic and industry researchers to create open-source software for analog layout automation. ALIGN translates a SPICE-level netlist into a physical layout, with 24-hour turnaround and no human in the

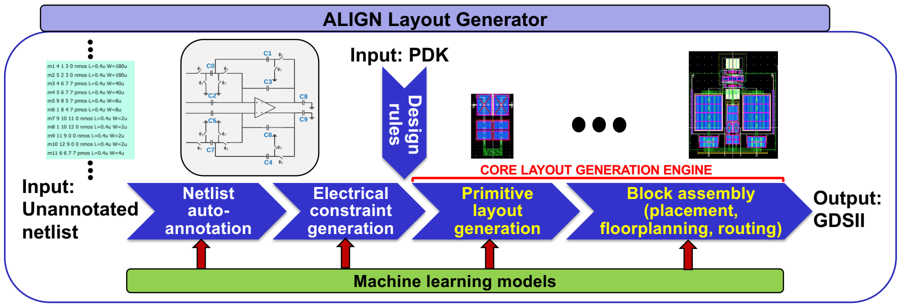
ALIGN Layout Generator
Input: PDK
Design rules
Input: Unannotated netlist
Netlist auto-annotation
Electrical constraint generation
CORE LAYOUT GENERATION ENGINE
Primitive layout generation
Block assembly (placement, floorplanning, routing)
Output: GDSII
Machine learning models

**Figure 1: Overview of the ALIGN flow.**

loop. The input consists of a netlist, specifications, and a process design kit (PDK), and the output is GDSII. The project started in 2018, and has undergone Alpha, Beta, and V1.0 releases [1] over this period. ALIGN has been used to create layouts of circuits in both bulk and FinFET technologies.

The philosophy of ALIGN is to compositionally synthesize the layout by first identifying layout hierarchies in the netlist, then generating correct-by-construction layouts at the lowest level of hierarchy, and finally assembling blocks at each level of hierarchy during placement and routing. In doing so, ALIGN mimics the human designer, who identifies known blocks, lays them out, and then builds the overall layout. The hierarchy goes from the lowest level of an individual transistor or passive device, to larger structures ("primitives") that are a collection of a regular connection of these devices (e.g., differential pairs, current mirrors, resistor arrays, capacitor arrays), up to the level of sub-blocks (e.g., OTAs, LNAs, VCOs), and then to higher levels that recursively assemble groups of sub-blocks.

The ALIGN flow (Fig. 1) consists of the following steps:
*Auto-annotation and constraint generation*: A key step in ALIGN is to identify these hierarchies to recognize the building blocks of the design using techniques such as those described in [2, 3], which employ a mix of graph-based and machine-learning-based methods. Within and across these building blocks, geometric constraints, including symmetries along multiple axes, are inferred. Electrical constraints are generated by translating performance specifications to layout constraints: techniques developed to date include those in [4, 5], and further enhancements are currently under investigation.
*Design rule abstraction and parameterized primitive layout generation*: ALIGN defines a systematic method for translating a complex design rule manual into a simplified grid representation, appended with Boolean constraints as needed. The approach has been shown to be capable of capturing rules in both FinFET and bulk PDKs.

Based on these rules, a primitive generator builds the layout for any primitive in a parameterized manner: for example, a differential pair could be parameterized by the number of parallel transistors. Setting up primitive generation requires the definition of a few tens of primitive structures that cover a wide range of circuits. A user may also define a primitive that is not provided in the ALIGN library. Current efforts include the use of the primitive cell generation to a wider set of structures, incorporating constraints such as cell height, and exploring methods for circuit design using common-centroid and interdigitated layouts.
*Block-level assembly*: This step uses the constituent hierarchical structure of the design and performs placement and routing to assemble blocks in the design, while meeting geometric and electrical constraints specified earlier. Ongoing efforts include methods for automated reliable power distribution, and enhanced methods for global and detailed routing.

To enable the application of ALIGN in practical settings, the flow creates a separation between open-source code and proprietary data. PDK models are translated into an abstraction that is used by the layout generators. For parts of the flow driven by ML models, we provide infrastructure for training models on user data.

In addition to full automation with no human in the loop, as described above, we have tested multiple entry points into the flow: user-defined annotation and constraint detection; user-defined place-and-route constraints; and user-codified place-and-route directives that provide a user with much more stringent control over the structure of the layout. These alternatives provide a user with varying levels of control over the eventual result, and allow for a range of user expertise.

A more detailed view of ALIGN is provided in an upcoming article [6]. The project is in active development, with V2.0 expected in 2021 and V3.0 in 2022. We have already successfully demonstrated high-performance layouts for a variety of circuits from each of the four classes, and continue to develop new methods that enhance the performance and applicability of ALIGN.

## ACKNOWLEDGMENTS

This work was supported in part by DARPA IDEA program under SPAWAR contract N660011824048.

## REFERENCES

[1] "ALIGN: Analog layout, intelligently generated from netlists," Software repository, accessed August 24, 2020. https://github.com/ALIGN-analoglayout/ALIGN-public.
[2] K. Kunal, *et al.*, "GANA: Graph convolutional network based automated netlist annotation for analog circuits," in *Proc. DATE*, 2020.
[3] K. Kunal, *et al.*, "A general approach for identifying hierarchical symmetry constraints for analog circuit layout," in *Proc. ICCAD*, 2020.
[4] Y. Li, *et al.*, "Exploring a machine learning approach to performance driven analog IC placement," in *Proc. ISVLSI*, 2020.
[5] Y. Li, *et al.*, "A customized graph neural network model for guiding analog IC placement," in *Proc. ICCAD*, 2020.
[6] T. Dhar, *et al.*, "ALIGN: A system for automating analog layout," *IEEE Design & Test (to appear)*, 2020. Available at *arXiv:2008.10682*.