The ALIGN Automated Analog Layout Engine: Progress, Learnings, and Open Issues

Sachin S. Sapatnekar
Department of Electrical & Computer Engineering
University of Minnesota
Minneapolis, MN (USA)
sachin@umn.edu

ABSTRACT

The ALIGN (Analog Layout, Intelligently Generated from Netlists) project [1, 2] is a joint university-industry effort to push the envelope of automated analog layout through a systematic new approach, novel algorithms, and open-source software [3]. Analog automation research has been active for several decades, but has not found widespread acceptance due to its general inability to meet the needs of the design community. Therefore, unlike digital design, which has a rich history of automation and extensive deployment of design tools, analog design is largely unautomated.

ALIGN attempts to overcome several of the major issues associated with this lack of success. *First*, to mimic the human designer's ability to recognize sub-blocks and specify constraints, ALIGN has used machine learning (ML) based methods to assist in these tasks. *Second*, to overcome the limitation of past automation approaches, which are largely specific to a class of designs, ALIGN attempts to create a truly general layout engine by decomposing the layout automation process into a set of steps, with specific constraints that are specific to the family of circuits, which are divided into four classes: low-frequency components (e.g., analog-to-digital converters (ADCs), amplifiers, and filters); wireline components for high-speed links (e.g., equalizers, clock/data recovery circuits, and phase interpolators); RF/Wireless components (e.g., components of RF transmitters and receivers), and power delivery components (e.g., capacitor- and inductor-based DC–DC converters and low dropout (LDO) regulators). For each class of circuits, different sets of constraints are important, depending on their frequency, parasitic sensitivity, need for matching, etc., and ALIGN creates a unified methodological framework that can address each class. *Third*, in each step, ALIGN has generated new algorithms and approaches to help improve the performance of analog layout. *Fourth*, given that experienced analog designers desire greater visibility into the process and input into the way that design is carried out, ALIGN is built modularly, providing multiple entry points at which a designer may intervene in the process.

The ALIGN technique is inherently hierarchical, and functions in the same style as the human designer. It first identifies layout hierarchies in the netlist, then generates correct-by-construction layouts at the lowest level of hierarchy, and finally assembles blocks at each level of hierarchy during placement and routing. 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.

ISPD '23, March 26–29, 2023, Virtual Event, USA

ACM ISBN 978-1-4503-9978-4/23/03.
https://doi.org/10.1145/3569052.3578916

We overview the ALIGN flow, while highlighting novel techniques developed as part of the project. ALIGN proceeds as follows: The **auto-annotation** step takes a circuit netlist and identifies key building blocks in the design to create layout hierarchies. This task has traditionally been difficult because of the wide variety of layout topologies for each subcircuit (e.g., there are hundreds of topologies for an operational transconductance amplifier (OTA)). ALIGN addresses this problem through a mix of recognition methods [4–6] using conventional graph-based algorithms and machine-learning-based methods that can perform approximate graph isomorphism in the same manner as the human designer recognizes "approximately similar" schematic topologies as variants of the same circuit. Within and across these building blocks, geometric constraints, including symmetries along multiple axes, are inferred. The designer may augment or override the recognized constraints by specifying constraints using a constraint language defined within ALIGN.
The **design rule abstraction** step codifies the design rules from the process design kit (PDK) into a set spacing and length rules, using layer-specific grids that capture not only width and spacing, but also stopping points, using major grids and minor grids [2], appended with Boolean constraints as needed. The approach is particularly suitable capturing rules in gridded FinFET layouts with unidirectional routes and coloring rules, but has also been used for bulk PDKs. Translating a foundry PDK to a form that is usable within ALIGN is a manageable task, and requires a significant, but one-time effort that is well documented on the ALIGN repository.
Next, the **primitive layout** step generates parameterized layouts for primitives in the design. For example, a differential pair could be parameterized by the number of parallel transistors, or the number of stacked transistors, and can be built by using unit cells with a certain number of transistors/fins. About 20 primitive structures for commonly-encountered subcircuits are predefined within ALIGN, and user-defined primitives may also be inserted into the flow. For circuits that require matching, the ALIGN project has extensively worked on developing new algorithms for common-centroid layout of transistor arrays [7] and capacitor arrays [8–10], including the choice of the unit device [11]. The project has also investigated the utility of non-common-centroid (interdigitated/clustered) layouts, which can result in reduced parasitics [12].

The **block-level assembly** step follows the primitive generation step and successively moves up the design hierarchy defined in the annotation step (or by the designer through the constraint language) to construct the layout – both placement and routing – at each level of hierarchy, respecting all geometric constraints. The layout is augmented with well taps, which can be optimally chosen [13, 14]. Several placement engines are available to the user: enumerative placement when the number of blocks is small; integer linear programming (ILP) based placement for an intermediate number of blocks; and separate simulated annealing based and analytic placement engines [15] for large designs. Within placement, ALIGN has extensively investigated techniques for meeting electrical constraints: by using the concept of charge flow to determine current path directions to determine net criticalities [16], and by employing ML methods that apply wire length limit constraints [17], or that can use an ML predictor to determine whether a layout will meet constraints [18, 19]. Methods for smart wire sizing using ML-based methods have also been investigated [20, 21].

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. The user may incorporate prebuilt blocks such as Pcells by "black-boxing" them and exposing only the terminals, which are then aligned to the gridded paradigm used in ALIGN.

ALIGN has been used to create layouts of circuits in both bulk and FinFET technologies. It has been used in numerous industry and academic settings on designs reported in [22–25], including successful tapeouts that have validated the methodology.

Several problems require further intensive effort. First, the task of efficiently converting electrical constraints to layout constraints is an open problem. A typical designer use case involves setting these constraints manually in ALIGN. Second, while ALIGN has made major strides in comprehending designer intent by automatically generating hierarchies, this area merits more investigation. Third, the ALIGN effort is possibly one of the first to systematically explore the issue of whether common-centroid, interdigitated, or clustered layouts are optimal within a specific context; many designers blindly use common-centroid layout even when it is not needed, paying a performance overhead [12]. Further research could lead to systematic set of best practices and diagnoses that guide a design towards optimal layout. Finally, increased use and development may enable greater interoperability of ALIGN with other flows.

CCS CONCEPTS

• **Hardware → Physical design (EDA)**; **Software tools for EDA**; **Analog and mixed-signal circuits**; **Analog and mixed-signal circuit optimization**.

KEYWORDS

Analog circuits, layout, machine learning, design automation

ACM Reference Format:

Sachin S. Sapatnekar. 2023. The ALIGN Automated Analog Layout Engine: Progress, Learnings, and Open Issues . In *Proceedings of the 2023 International Symposium on Physical Design (ISPD '23), March 26–29, 2023, Virtual Event, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3569052.3578916

ACKNOWLEDGMENTS

This work is supported in part by the DARPA IDEA program under SPAWAR Contract N660011824048, by the Semiconductor Research Corporation (SRC), and by the National Science Foundation under Award 2212345.

ALIGN is a joint effort between the University of Minnesota†, Intel Labs‡, and Texas A&M University§. Contributors include Steve Burns‡, Tonmoy Dhar†, Ramesh Harjani†, Jiang Hu§, Nibedita Karmokar†, Desmond Kirkpatrick‡, Kishor Kunal†, Yaguang Li§, Yishuang Lin§, Meghna Madhusudan†, Parijat Mukherjee‡, Jitesh Poojary, Arvind K. Sharma†, Ramprasath S.†, Wenbin Xu§, and Soner Yaldiz‡.

REFERENCES

[1] K. Kunal, *et al.*, "ALIGN: Open-source analog layout automation from the ground up," in *Proc. DAC*, pp. 77–80, 2019.

[2] T. Dhar, *et al.*, "ALIGN: A system for automating analog layout," *IEEE Des. Test*, 2020. Available at *arXiv:2008.10682*.

[3] "ALIGN: Analog layout, intelligently generated from netlists," Software repository, accessed January 29, 2023. https://github.com/ALIGN-analoglayout/ALIGN-public.

[4] K. Kunal, *et al.*, "GANA: Graph convolutional network based automated netlist annotation for analog circuits," in *Proc. DATE*, 2020.

[5] K. Kunal, *et al.*, "A general approach for identifying hierarchical symmetry constraints for analog circuit layout," in *Proc. ICCAD*, 2020.

[6] K. Kunal, *et al.*, "GNN-based hierarchical annotation for analog circuits," *IEEE T. Comput. Aid. D.*, 2023. (early access).

[7] A. K. Sharma, *et al.*, "Performance-aware common-centroid placement and routing of transistor arrays in analog circuits," in *Proc. ICCAD*, 2021.

[8] N. Karmokar, *et al.*, "Common-centroid layout for active and passive devices: A review and the road ahead," in *Proc. ASP-DAC*, 2022.

[9] N. Karmokar, *et al.*, "Constructive common-centroid placement and routing for binary-weighted capacitor arrays," in *Proc. DATE*, pp. 166–171, 2022.

[10] N. Karmokar, *et al.*, "Constructive placement and routing for common-centroid capacitor arrays in binary-weighted and split DACs," *IEEE T. Comput. Aid. D.*, 2023. (early access).

[11] N. Karmokar, *et al.*, "Minimum unit capacitance calculation for binary-weighted capacitor arrays," in *Proc. DATE*, 2023.

[12] A. K. Sharma, *et al.*, "Common-centroid layouts for analog circuits: Advantages and limitations," in *Proc. DATE*, pp. 1224–1229, 2021.

[13] Ramprasath S., *et al.*, "Analog/mixed-signal layout optimization using optimal well taps," in *Proc. ISPD*, pp. 159–166, 2022.

[14] Ramprasath S., *et al.*, "A generalized methodology for well island generation and well-tap insertion in analog/mixed-signal layouts," *ACM T. Des. Automat. El.*, 2023. (in press).

[15] Y. Lin, *et al.*, "Are analytical techniques worthwhile for analog IC placement?," in *Proc. DATE*, 2022.

[16] T. Dhar, *et al.*, "A charge flow formulation for guiding analog/mixed-signal placement," in *Proc. DATE*, 2022.

[17] T. Dhar, *et al.*, "Fast and efficient constraint evaluation of analog layout using machine learning models," in *Proc. ASP-DAC*, pp. 158–163, 2021.

[18] Y. Li, *et al.*, "Exploring a machine learning approach to performance driven analog IC placement," in *Proc. ISVLSI*, 2020.

[19] Y. Li, *et al.*, "A customized graph neural network model for guiding analog IC placement," in *Proc. ICCAD*, 2020.

[20] Y. Li, *et al.*, "A circuit attention network-based actor-critic learning approach to robust analog transistor sizing," in *Proceedings of Workshop on Machine Learning for CAD*, pp. 1–6, 2021.

[21] Y. Li, *et al.*, "Performance-driven wire sizing for analog integrated circuits," *ACM T. Des. Automat. El.*, vol. 28, Dec. 2022.

[22] J. Liu, *et al.*, "From specification to silicon: Towards analog/mixed-signal design automation using surrogate NN models with transfer learning," in *Proc. ICCAD*, 2021.

[23] X. Liu, *et al.*, "A digital LDO in 22nm CMOS with a 4b self-triggered binary search windowed flash ADC featuring automatic analog layout generator framework," in *Proc. A-SSCC*, pp. 2–4, 2022.

[24] J. Poojary and R. Harjani, "A 1-to-3GHz co-channel blocker resistant, spatially and spectrally passive MIMO receiver in 65nm CMOS with +6dBm in-band/in-notch B1dB," in *Proceedings of the IEEE International Solid-State Circuits Conference*, vol. 64, pp. 96–98, 2021.

[25] S. Kamineni, *et al.*, "AuxcellGen: A framework for autonomous generation of analog and memory unit cells," in *Proc. DATE*, 2023.