CDLS: Constraint Driven Generative AI Framework for Analog Layout Synthesis

Prasanth Mangalagiri
Intel Corporation
Santa Clara, CA, USA
prasanth.mangalagiri@intel.com

Praveen Mosalikanti
Intel Corporation
Hillsboro, OR, USA
praveen.mosalikanti@intel.com

Farrukh Zafar
Intel Corporation
Santa Clara, CA, USA
farrukh.zafar@intel.com

Lynn Qian
Intel Corporation
Santa Clara, CA, USA
lynn.qian@intel.com

Phoebe Change
Intel Corporation
Santa Clara, CA, USA
phoebe.chang@intel.com

Arun J. Kurian
Intel Corporation
Folsom, CA, USA
arun.j.kurian@intel.com

Vinay Saripalli
Intel Corporation
Santa Clara, CA, USA
vinay.saripalli@intel.com

ABSTRACT

In advanced process technology nodes, analog circuit performance is intrinsically linked to layout parasitics and layout dependent effects (LDE). In contrast to digital designs, layout generation for analog mixed signal circuits remains predominantly a slow manual task, impeding rapid design convergence. To address this bottleneck, we introduce CDLS - a Constraint Driven Generative AI Framework for Analog Layout Synthesis. CDLS is fundamentally a constraint driven framework that enables analog circuit designers to auto-generate simulation-ready layout. Unlike traditional algorithmic approaches, CDLS uses generative AI and machine learning techniques to generate key design constraints that drive the quality of autogenerated placement and routing. Using CDLS, on average we reduce layout iteration time by 2-3X on industrial designs. By reducing the turn-around-time on layout iterations we estimate a 30% reduction to overall design convergence cycle. We also demonstrate that the quality of results achieved through CDLS is on par with manually drawn layout on state-of-the-art analog designs developed on an Intel sub-10nm process technology node.

KEYWORDS

Analog Layout Synthesis, Generative AI, Machine Learning, Placement, Routing, Analog Design Convergence

ACM Reference format:

Prasanth Mangalagiri, Praveen Mosalikanti, Farrukh Zafar, Lynn Qian, Phoebe Chang, Arun J. Kurian, and Vinay Saripalli. 2024. CDLS: Constraint Driven Generative AI Framework for Analog Layout Synthesis. In *Proceedings of the 61st ACM/IEEE Design Automation Conference (DAC '24). ACM, New York, NY, USA, 6 pages.*

1 MOTIVATION AND BACKGROUND

The past few decades have witnessed a paradigm shift in consumer computing demands, triggered by the proliferation of mobile technology. The shift has steered the device manufacturing industry away from a CPU-centric approach towards a peripheral-centric, connectivity-driven paradigm augmented by heterogeneous sensor integration. This has re-emphasized Analog Mixed Signal (AMS) designs in System-on-Chip (SoC) and core computing. Such a resurgence has notably expanded the on-die footprint of AMS IP content, including high-speed analog-to-digital converters (ADCs), high-speed clocking subsystems (DLLs, PLLs), and multi-GHz transceivers. Availability of key analog IPs such as DDR, PCIE, and USB, has become pivotal to product differentiation, influencing project schedules and time-to-market.

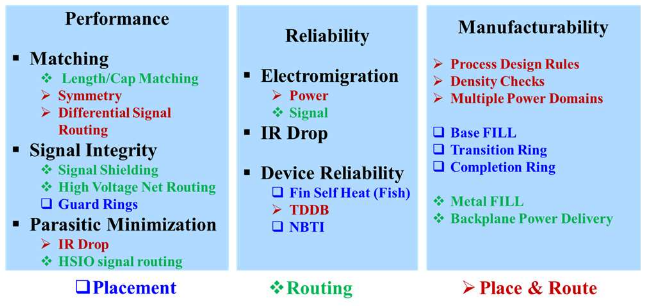


Figure 1: Analog design constraints

In contrast to digital design, where Electronic Design Automation (EDA) tools and methodologies have achieved full backend design automation, analog design, especially layout generation, remains a highly manual and iterative process. In advanced technology nodes, the performance of analog circuits is highly dependent on layout parasitics and a growing list of layout dependent effects (LDE). Automating analog layout generation is a very challenging task, requiring place and route solutions to simultaneously satisfy electrical, physical, and manufacturing constraints across different stages of the design hardening. Figure 1 illustrates this complexity with a selected sub-set of analog design constraints common to most advanced process node designs. The existing solutions in EDA vendor space are oriented mostly towards assisted automation and require circuit designers to master highly complex layout generation cockpits. In recent years, there has been a resurgence in academia [1] [2] towards providing end-to-end layout synthesis solutions. Despite recent research advancements in analog layout automation, existing module generation-based approaches, while promising quality results, fall short in industrial design settings due to their extensive design knowledge requirements and long preparation times for module creation. Rule based placement approaches relying heavily on standard circuit topologies [12] [13] often struggle getting adoption in industrial design settings.

Industrial circuits frequently deviate from standard topologies, evolving into complex variants that are tailored to specific applications and performance requirements. Addressing these gaps, this paper presents CDLS, a constraint driven analog layout synthesis framework for automating the layout generation task of analog circuit designs. The CDLS framework integrates generative AI and machine learning techniques to extract design intent from historical project data, enabling the automatic generation of essential placement and routing constraints that enhance the quality of auto-generated layouts. By providing fast accurate feedback on layout parasitics and layout dependent effects, CDLS significantly enhances analog designer productivity, and helps reduce time-to-convergence of analog design cycles. The core contributions of our work are as follows:

1. We present an industry-first, fully automated, end-to-end constraint driven layout synthesis framework for analog circuit design.
2. We introduce a novel transformer-based constraint generation method for analog device placement generation.
3. Through industrial design case studies, we demonstrate how CDLS facilitates early parasitic feedback and improves circuit designer productivity through fast design convergence.

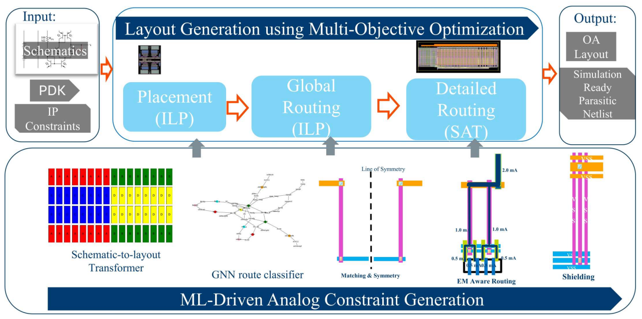


Figure 2: Constraint Driven Layout Synthesis (CDLS)

Figure 2 provides a detailed view of the CDLS framework, including its core components and the end-to-end data flow. Inputs to the CDLS system include the design schematic netlist integrated with PDK primitives, constraints specific to the IP addressing area and pin connectivity, and manufacturability constraints dictated by design rules. The output of the CDLS flow is a simulation-ready layout, designed for efficient integration into subsequent simulation and verification stages. It should be noted that, due to limitations in space, this document does not delve into the technical specifics of several sub-tasks such as device-level routing, the creation of power grids, filling of base and metal layers for manufacturability purposes, and the insertion of well taps. These tasks are integral to guaranteeing the manufacturability of the layouts produced, yet their detailed implementation processes are beyond the scope of this current discussion.

The remainder of the paper is organized as follows: Section II discusses in detail the architecture and training methods employed in our transformer-based device placement generation. Section III presents an overview of the analog placement challenges and how they are addressed in the CDLS placement engine. Section IV delves into the details of global and detailed routing engines that power CDLS. Section V demonstrates the quality-of-results (QOR) and turn-around-time improvements achieved through CDLS. Finally, Section VI concludes the paper with a summary of our contributions and outlines prospective enhancements and future work.

2 MACHINE LEARNING DRIVEN CONSTRAINT GENERATION

Automating analog layout design requires translating analog design objectives in circuit electrical domain into physical design placement and routing constraints, which is a complex task often resulting in one-to-many constraint mapping. In this section we delve into the details of how we employ machine learning techniques to generate constraints that guide CDLS placement and routing engines to infer design intent from circuit schematics and subsequently implement it in layout.

Device placement in analog layout design is a critical process that greatly influences circuit performance. This is especially true in designs requiring precise matching and symmetry, such as those found in precision amplifiers, data converters, and radiofrequency (RF) circuits. Effective device placement must ensure that the layout is resilient to process variations and maintains device matching, which is essential for the circuit's functional integrity. Two common strategies for achieving this are interdigitated and common centroid layouts, which are designed to counteract process gradients and mismatches across devices. In the interdigitated approach, devices are alternated in a way that averages out the variations over the layout, while the common centroid method places devices symmetrically around a central point to ensure that any process gradient has a symmetrical effect on the matched devices. These techniques are not merely layout patterns but are foundational strategies to achieve the desired analog performance and reliability.

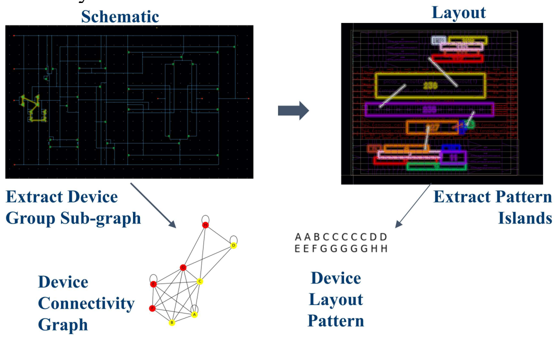


Figure 3: Data Labeling for SLX

2.1 Data Labeling The first step in training the schematic layout transformer is a comprehensive data extraction and labeling engine. This system harnesses layout and schematic information from 12 distinct IPs, spanning across three sub 10nm advanced process nodes. As illustrated in Figure 3, the labeling engine identifies recurring device patterns within the layouts, correlating them with their respective schematic sub-graphs. These device pattern "islands" are pinpointed using a set of heuristics that account for elements like diffusion breaks and well partitions—indicators that traditionally demarcate separate devices or groups of devices. This heuristic analysis is augmented by an examination of schematic device connectivity, ensuring that the identified patterns are not only geometrically isolated but also electrically connected. Geometrical methods are then employed to extract these device patterns from the layout, thereby creating a map of schematic sub-graph to device pattern pairings.

2.2 Graph-To-Sequence Mapping The first step in schematic to layout translation task is mapping the device sub-graph to a sequence of tokens that serves as input to an auto-regressive transformer. To capture the complex interactions between schematic devices, we employ a stack of GNN layers to encode the structural information of the device connectivity graph into a D-dimensional embedding space.

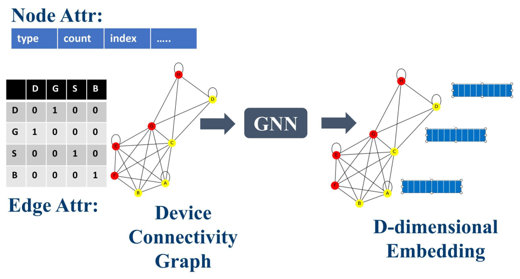


Figure 4: GNN embedding of device attributes

This embedding effectively represents nodes (devices) and edges (connections), critical for understanding the physical constraints and relationships necessary for predicting the design intent. As depicted in Figure 4, node attributes include device type (PMOS/NMOS), number of fingers (M/NF), device index and model types. The edge attributes are represented as a multi-bit binary vector that captures the type of connectivity between device terminals drain, gate, source, and bulk. These initial attributes are passed through 3 layers of Gated Attention Convolutions (GAT) [8]. To provide the transformer encoder with a clear understanding of each node's identity and its occurrence within the target pattern, two specific embeddings are used: node index and node count embeddings. The node index embeddings uniquely identify each node, like an alias, thereby ensuring that the transformer can distinguish between different nodes even when they share similar attributes and graph structure. The node count embeddings, on the other hand, inform the model of the frequency of each node's appearance within the target pattern, which is essential when generating device patterns with devices having differing mult-factors and fingers. As shown in Figure 5, the final node embeddings are formed by adding the node index and count embeddings to the GNN generated node embeddings. The sharing of node index embeddings between the encoder and decoder stages is a crucial design choice. It establishes a direct correspondence between the input nodes and their respective positions in the output placement pattern. This shared embedding space enables the model to maintain coherence between the schematic's electrical properties and the physical layout's spatial configuration.

2.3 Sequence-to-Sequence Transformer Once the input schematic graph is embedded through GNN and translated into a sequence, we feed this as input to an auto-regressive transformer with encoder-decoder architecture. The output of the transformer (decoder-input) is a sequence that represents the 2D device placement in a sequential format. In addition to the device aliases, it consists of three special tokens: <SOS>, <NL>, and <EOS>, representing the start of sequence, beginning of a new device row, and end of sequence, respectively. By translating the input graph and output 2D placement into sequences, we can map the device placement problem into a sequence-to-sequence translation task that auto-regressive transformer models are adept at solving. We use a combination of torch-geometric [4] for training GNN encoder and Pytorch [5] framework for building the transformer encoder-decoder networks. The Schematic-to-Layout Transformer is trained using the AdamW [3] optimizer and Cross Entropy Loss as the loss

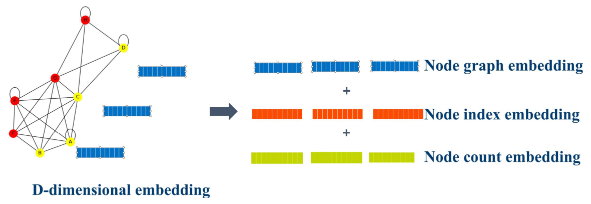


Figure 5: Auxiliary index and count additive embedding

function, with a batch size of 512. All the models are trained on a single NVIDIA A100 Tensor Core GPU with 80GB memory. The end-to-end data flow during the training process, includes GNN encoder, index and count embedding augmentation, transformer encoder-decoder network, and finally auto-regressive output sequence prediction. We first sample a batch of graphs from the dataset, and auto-regressively train the output sequence generation. The autoregressive prediction is critical, as it allows the model to consider the partial layout as it evolves, ensuring that each new device placement is consistent with the placement of previous devices.

3 ROUTING CONSTRAINTS PREDICTION

In this section we give an outline of some of the routing constraints automatically generated as part of CDLS. Please note that we omit the details of training, and model architecture to conserve space.

3.1 Signal Matching Classification Identifying signals that need matched parasitics in layout is a key challenge for a fully automated synthesis solution. Traditionally, methods like sub-graph isomorphism have been employed on the schematic graphs to identify nets with matching topologies. However, we note that in practice, these methods fall short due to lack of strict isomorphism in design. For example, the presence of additional dummy devices in one signal path added for DFM purposes can disrupt the detection of symmetry using strict algorithmic approaches like sub-graph isomorphism. In addition, for signals with small fanout, the sub-graph isomorphism can cause unintended false positives. In such scenarios we need to leverage other design features, such as terminal connectivity, size, and type of connected instances. To address these challenges, the CDLS framework employs a GNN-based net pair classification approach. For brevity, we omit the training and inference details of GNN based symmetry net pair classification.

3.2 Top Layer and Aspect Ratio Prediction One of the design choices that is critical to the RC profile of a signal is its choice of top routing layer. This is typically driven by the allowed top-layer of a given cell and the metal congestion in each net routing region. However, in practice, designers use their prior knowledge to determine signals that are critical to circuit performance. In CDLS constraint generation, we leverage the choices made in past designs to predict the cell and net top for routing layer. Top layer and cell aspect ratio predictions are formulated as multi-class classification and regression tasks, respectively. Input features to these models include number of nets, terminals, cell area, dimensions, number of devices, and intermediate hierarchies. XGBoost [12] was used to model the classification of cell top layer and net top layer models, which achieve a prediction accuracy of 0.78 and 0.74, respectively.

4 PLACEMENT

In this section we discuss the key design choices and techniques employed in the CDLS placement engine to address the analog physical design challenges. The analog placement task is modeled as a weighted constraint optimization problem. The auto-generated placement constraints are translated into physical placement through Mixed Integer Linear Programming (MILP) and optimized using a commercial solver [9]. The placement process begins with a top-down generation of hierarchical placement constraints, followed by a bottom-up block level placement. The MILP formulation shown below integrates several design constraints, which are critical for retaining analog design intent.

$$Minimize \sum_{k=0}^{n} |N_{xh} - N_{xl}| + |N_{yh} - N_{yl}| + \sum_{c=0}^{C} W_c * C_{dim} \quad (1)$$

The objective of the placement optimization process shown in (1) is to minimize the cell area and span of signals in the design, while adhering to the following constraints:

4.1 Analog Design Constraints: Core to the complexity of analog placement, these include design-specific requirements like differential signal path matching through common centroid layouts or interdigitation of device legs and minimizing parasitic capacitance for high-frequency IO signals. The schematic-to-layout transformer, and the GNN based signal classification modules presented in section II generate the device grouping and signal matching constraints respectively. As shown in (2) and (3), constraints are added to the x, y locations of instance pairs that require matching and symmetry, as identified by the auto-generated constraints.

$$\forall\ (i,j) \in VerticalAxisSymPairs{:}\ x_i = x_j \quad (2)$$
$$\forall\ (i,j) \in HorizontalAxisSymPairs{:}\ y_i = y_j \quad (3)$$

4.2 IP Specific Constraints: Derived from the SoC's top-level floorplan, these dictate the aspect ratio and area allocation for the IP, influencing hierarchical block levels and instance placements. The cell aspect ratio, width, and height targets are fed as soft-constraints, and are optimized as part of the objective function.

4.3 Design Rule Complexity: Process design rules necessitate modeling of spacing and density constraints for well regions and diffusion areas to ensure manufacturability. We leverage third-party EDA tools to calculate the design rule correct min-spacing between all pairs of placement components. As shown in (4), MILP constraints are then added between the x and y coordinates of the components, to model min-spacing, overlap avoidance, and cell halo constraints.

$$\forall\ (i,j) \in Instances{:}\ i \ \neq j{:}$$
$$x_i + width_i + x\_spacing_{ij} \ \leq x_j \ or \quad (4)$$
$$x_j + width_j + x\_spacing_{ij} \ \leq x_i \ or$$
$$y_i + height_i + y\_spacing_{ij} \ \leq y_j \ or$$
$$y_j + height_j + y\_spacing_{ij} \ \leq y_i \ or$$

5 ROUTING

Following the placement stage, the physical interconnect wiring between placed components is performed during routing stage.

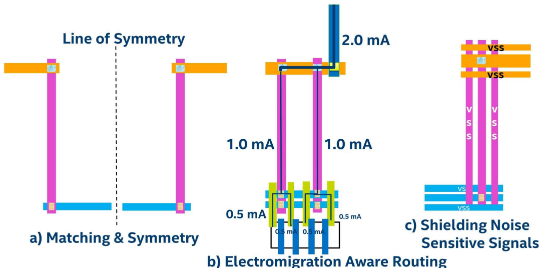


Figure 6: Constraint Aware Global Routing Topologies

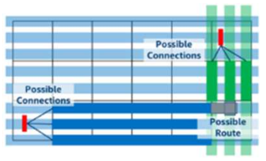
5.1 Global Routing

Due to the high complexity of this multi-objective optimization problem, typically the routing task is performed in two phases. The first phase is called Global Routing (GR), and handles key tasks such as constraint-based route planning, congestion aware template mapping, etc. Once the route plans/topologies are generated, detailed routing phase handles their design-rule aware realization. To simplify the complexity of the problem, the global router abstracts the available routing resources from individual wires into a 3-dimensional global routing grid-graph. While the first two dimensions model the 2D grid of routing resources on a given layer, the 3rd dimension spans across all the available routing layers. The nodes in the GR grid-graph model the tracks available in each grid-cell and the edges correspond to either the connections between neighboring tiles on the same area or vias between adjacent layers.

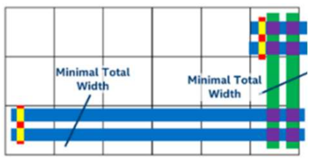
5.1.1 Topology Generation: The global routing stage communicates the design intent to the detailed router using routing topology. A global route topology models the analog constraint by encapsulating the necessary physical design information such as route shape, layers, wire widths, and signal connectivity. Multiple constraint-aware topologies are generated for each signal in the design, and subsequently, a feasible independent set is chosen based on MILP global optimization. The first stage in topology generation, prior to path finding, is the edge cost assignment in the 3D global grid graph. An edge that connects nodes from adjacent interacting layers are referred to as via-edge, and its cost is a combination of the average parasitic resistance (pre-characterized per-process), and its Electromigration (EM) S-factor metric. Edges that connect neighboring global grids on the same layer along its primary direction are referred as wire-edges. The cost of wire-edges is a combination of the relative layer parasitic contribution, and the minimum of the number of un-occupied tracks in edge-nodes.

5.1.2 Parasitic Matching and Symmetrical Routing: Analog IPs frequently necessitate matching key performance parameters across signal paths, requiring layout parasitic matching. The layout must match the resistance, capacitance, and inductance to achieve this. For instance, to minimize the common-mode voltage of differential signals, matching routing parasitics is as essential as placing devices using variation reduction techniques like common centroid or interdigitated placements. Identifying such matching signals and ensuring routing with matched parasitics is a key challenge that we address using GNN based matching classifier detailed in section II. For such signals, as shown in Figure 6a, we ensure that the routing topologies are matched with respect to layer choices, segment lengths, and widths.

5.1.3 Signal Electromigration [EM]: As technology scales beyond nanometer regime, electromigration (EM) has become a paramount design constraint. With the significant reduction in the current-carrying capacity of wires at these scales, the reliability verification (RV) fixing process, which was traditionally a post-layout step, has now become a predominant aspect of layout creation. The CDLS global router, utilizing device pin currents obtained from pre-layout simulations, creates topologies that are inherently compliant with EM standards. It ensures that wire widths are designed to stay within the EM limits corresponding to the branch currents.



7a: SAT Formulation



7b: Potential Solution

Figure 7: Detailed Router SAT formulation and solution

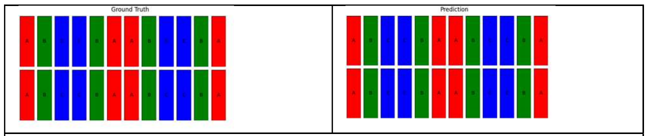
5.2 Detailed Routing

In this section, we provide an in-depth description of the SAT-based detailed router's implementation. Figure 8a illustrates two M1 terminals, highlighted in red. The routing framework is structured with fixed templates for all metal layers, including horizontal M2 and vertical M3 tracks. The global routing grid, configured to an optimal size, assigns global route segments to specific grid tiles and metal layers. The concept of a 'tie' is introduced as a logical link between disconnected layout elements, namely terminals and global routes. A 'route' is the physical implementation of this tie, comprising wires and vias that establish actual connections between layout components. The SAT solver is tasked with optimizing the routing process, constrained to select the most efficient combination of vias and wires necessary for connecting two targeted wires. Figure 7b demonstrates an example of such a route, connecting an M1 terminal wire to an M3 trunk wire. In addition to adhering to global router topologies, for layer and width choices, the detailed router tracks segments that need width matching across nets. This is an essential feature for handling matching and symmetrical routing constraints. This example illustrates the detailed router's capability in managing complex routing tasks while adhering to the specified design constraints and objectives. For brevity, we omit the details of design rule modeling, and SAT formulation in this discussion.

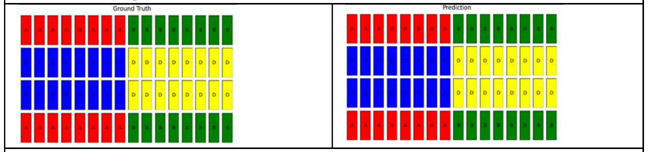
6 RESULTS

In this section we demonstrate the quality of results and faster design convergence by the mode-of-work enabled by CDLS framework through the following design case-studies.

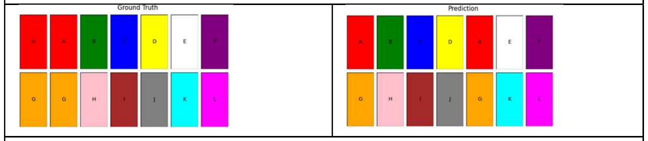
6.1 Device Placement using Schematic to Layout Transformer



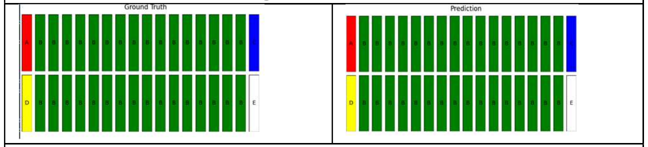
8a: Interdigitation



8b: Common Centroid



8c: Cross-Row Gate Alignment



8d: Dummy Device Padding

Figure 8: SLX design intent aware predictions

The dataset consists of close to 15k unique schematic sub-graphs and matching device patterns sourced from a wide range of custom analog designs. Table 1 lists some of key metrics used to evaluate the prediction quality of the schematic-to-layout transformer model. An average BLUE score of 0.76 indicates a good level of similarity between the transformer model's layout prediction to the reference sequences extracted from manual layout. A ROGUE-1 and ROUGE-2 score of 0.94 and 0.85 highlight the model's precision in predicting unigram and bigram sequences, hence its ability to capture neighboring layout patterns. Finally, a ROUGE-L score of 0.92 indicates the model's effectiveness in capturing longer sequences, that require the ability in modeling long range interactions based on signal connectivity.

Metric	Average BLUE	ROUGE-1	ROUGE-2	ROUGE-L
Score	0.76	0.94	0.85	0.92

Table1: SLX evaluation metrics

To illustrate the quality of results, we share the snapshots of successful cases where the schematic layout transformer was able to match the ground truth patterns. Figure 8a shows a scenario for device **interdigitation** for nodes in the sub-graph containing mult-leg devices sharing source/drain. The SLX module effectively learned and replicated **common-centroid** configurations shown in Figure 8b, critical to minimizing gradient mismatches in devices like differential pairs. In Figure 8c, we demonstrate the scenario, where the predicted pattern was not an exact match to the ground truth. However, upon further inspection the design intent to **vertically align** devices matching the same gate was captured and meets the designer criteria for this device group. Finally, figure 8d, depicts dummy device padding, a layout pattern common for isolating sensitive devices from the neighboring elements. In this scenario, devices with aliases A/C/D/E are dummies and B is the active mult-leg device. These validation examples demonstrate that by combining graph representative abilities of GNNs with predictive power of the auto-regressive transformer architecture, SLX model can capture analog design intent hidden in the training data.

6.2 Design Performance Iterations (LJPLL-VCO) A phase-locked loop (PLL) is employed for clock frequency multiplication and phase alignment. In this case study, CDLS is used to generate layout for a voltage-controlled ring oscillator (VCO), a key component in wide-range PLLs. The simulation process included sweeping the virtual supply (vcovro) and measuring the output frequency. Comparative analysis between manually produced layouts and those generated by CDLS under various constraints revealed interesting performance patterns. Initially, CDLS-generated layouts showed marginal frequency gains but underperformed compared to manual designs. This was attributed to the handling of the control voltage signal (vcovro), which was subsequently routed as a power supply with necessary gridding.

As illustrated in Figure 9 (purple), this iteration led to layout that not only met, but exceeded the target frequency, showcasing the effectiveness of CDLS in facilitating faster design iterations and design convergence. Each iteration, including layout generation and circuit simulation, was completed in about a day, in contrast to 1-2 weeks typically required through manual layout generation.

6.3 Comparisons To Manual Layout Quality In addition to enabling faster layout iterations for early convergence of critical analog building blocks, CDLS layout generation framework was used to generate seed layout for physical design teams. CDLS development team in collaboration with IP design's circuit and mask design teams worked towards automating the layout generation of the entire IP using constraints. The building blocks constraints were generated using the CDLS constraint generation framework. These constraints were used to generate floorplan and routing for all the custom-building blocks in PLL IP including charge pump, loop filter, VCO, current reference generator (IREF), and feedback divider. IP level constraints that dictate the top-level

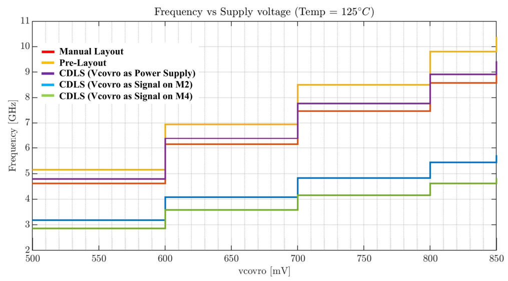
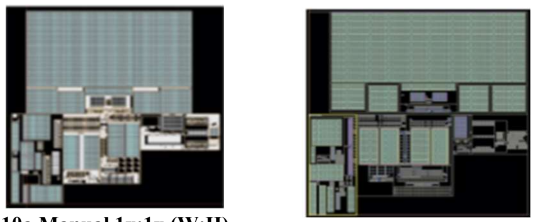


Figure 9: Fast Layout Iterations using CDLS

floorplan were manually entered by the design engineers to match the manual layout. Figure 10 compares the top-level floorplan and area scaling achieved for PLL analog components. We were able to achieve comparable target area while matching relative placement and alignment of the building blocks. In addition to matching the custom layout floorplan, we were also able to demonstrate the scalability and quality of the CDLS placement and routing solutions.



10a Manual 1x:1x (W:H) **10b CDLS 0.95x:1x (W:H)**

Figure 10: LJPLL CDLS vs Manual layout comparison

7 CONCLUSION

In this paper, we introduced CDLS, a Constraint Driven AI Framework for Analog Layout Synthesis. CDLS leverages generative AI and machine learning techniques to generate key design constraints, thereby enabling automatic generation of high-quality, simulation-ready layout. This approach significantly enhances the efficiency of analog circuit designers, reducing layout iteration time by approximately 2-3X, and contributing to an estimated 30% reduction in overall design convergence time. Our findings demonstrate that layouts generated by CDLS match the quality of manually drawn layouts on state-of-the-art analog designs implemented on an Intel sub-10nm process technology node. In the future, we plan to expand the use of generative AI techniques within CDLS to further enhance the quality and range of design constraint predictions, addressing a diverse array of analog circuits. Additionally, we envision utilizing CDLS to facilitate closed-loop analog design convergence and optimization, integrating layout synthesis with iterative design refinements.

REFERENCES

[1] Hakhamaneshi, Kourosh, Nick Werblun, Pieter Abbeel, and Vladimir Stojanović. "BagNet: Berkeley analog generator with layout optimizer boosted with deep neural networks." In *2019 IEEE/ACM International Conference on Computer-Aided Design (ICCAD),* pp. 1-8. IEEE, 2019.
[2] Kunal, Kishor, et al "ALIGN: Open-source analog layout automation from the ground up." In *Proceedings of the 56th Annual Design Automation Conference 2019,* pp. 1-4. 2019.
[3] Zhang, Zijun. "Improved adam optimizer for deep neural networks." In *2018 IEEE/ACM 26th international symposium on quality of service (IWQoS),* pp. 1-2. Ieee, 2018.
[4] Fey, Matthias, and Jan Eric Lenssen. "Fast graph representation learning with PyTorch Geometric." *arXiv preprint arXiv:1903.02428* (2019).
[5] Paszke, Adam, et al. "Pytorch: An imperative style, high-performance deep learning library." *Advances in neural information processing systems* 32 (2019).
[6] Vaswani et al "Attention is all you need." *Advances in neural information processing systems* 30 (2017).
[7] Kipf, Thomas N., and Max Welling. "Semi-supervised classification with graph convolutional networks." *arXiv preprint arXiv:1609.02907* (2016).
[8] Veličković, Petar, et al "Graph attention networks." *arXiv preprint arXiv:1710.10903* (2017).
[9] IBM (2017) IBM ILOG CPLEX 12.7 User's Manual (IBM ILOG CPLEX Division, Incline Village, NV)
[10] Veličković, Petar, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. "Graph attention networks." *arXiv preprint arXiv:1710.10903* (2017).
[11] Chen, Tianqi, and Carlos Guestrin. "Xgboost: A scalable tree boosting system." *In Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pp. 785-794. 2016.
[12] Eick, Michael, Martin Strasser, Kun Lu, Ulf Schlichtmann, and Helmut E. Graeb. "Comprehensive generation of hierarchical placement rules for analog integrated circuits." *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 30, no. 2 (2011): 180-193.
[13] Graeb, Helmut E., ed. *Analog layout synthesis: a survey of topological approaches*. Springer Science & Business Media, 2010.