Machine Learning based Structure Recognition in Analog Schematics for Constraints Generation

Rituj Patel
Applied Machine Learning, Infineon Technologies, Bangalore, India (*Rituj.Patel@infineon.com*)
Husni Habal
Analog Mixed Signal, Infineon Technologies, Munich, Germany (*Husni.Habal@infineon.com)*
Konda Reddy Venkata
Applied Machine Learning, Infineon Technologies, Bangalore, India (*Venkata.KondaReddy@infineon.com*)

***Abstract*- The analog mixed-signal integrated circuit (AMS-IC) design flow has many manual steps. In a constraint-driven design flow (CDD), it is necessary to manually identify and label various circuit structures, known as functional blocks, as separate design and verification constraints (rules) apply to each structure. Each block is composed of one or more devices and is recurrent and largely invariant across many analog circuits. Simple examples are MOSFET differential pairs and current mirror banks as illustrated in Fig. 1. In this work, a scheme is presented for accurate and computationally fast automatic structure recognition. AMS-IC functional blocks are identified using a combination of K-Means clustering (a non-parameter algorithm) and graphical convolutional neural networks (GCNNs). The algorithm is applied to disparate industrial schematics from several technology nodes, and of varying complexity. The new framework has an accuracy of over 95% on the applied examples. The average execution time for 30 example schematics is 0.5 seconds per schematic.**

Keywords- analog design automation; constraint-driven design; machine learning; clustering algorithm; graph convolutional neural network; structure recognition

I. Introduction

Despite the advancements in computer-aided circuit design, the analog mixed-signal integrated circuit (AMS-IC) design flow still has many manual steps. The process heavily relies on designer expertise to create and dimension the topological circuit structures (schematics) to meet design specifications.

In a constraint-driven design flow (CDD), various schematic structures – called functional blocks – must be identified and labeled, as disparate design and verification rules apply to each structure. A simple example is shown in Fig. 1(a): transistors are grouped into a differential pair and a current mirror. The current mirrors MOSFETs must have equal channel lengths. Practical circuits can have many sub-circuits and levels of hierarchy, as shown in Fig. 1(b): Manual structure recognition (SR) is cumbersome and error-prone, especially in large hierarchical AMS circuits.

Several approaches for automatic SR exist in literature. Library-based methods [1], [2] involve the manual creation of an exhaustive library of structures in the form of templates. This method is successful when a schematic block matches a library structure. It lacks robustness, however, as matching against the library is rigid. Any small variation needs to be coded as a new structure.

Learning-based approaches [3], [4] require manual inspection to learn new structures as initial training data must be manually labeled. In addition, complexity increases exponentially with the number of devices. While less rigid than library-based methods, the application of these methods is also limited to a small set of predefined (pre-trained) functional blocks.

The ML-based method in [5] performs SR automatically, but uses the x and y coordinate of each device in the schematic. This requires the schematic to be drawn in a specific manner. For example, differential pair transistors should be drawn symmetrically. It also lacks automation in identifying the structures in a hierarchical schematic.

A new scheme for automatic SR is proposed in this paper. Recent advances in machine learning (ML) are applied – including the use of K-Means clustering [6] and graph convolutional networks (GCNs) [7]. The circuit schematic is encoded in a connection graph, and several feature metrics are extracted to use with the aforementioned algorithms. The method proposed in this paper makes no assumptions about the schematic drawing or levels of hierarchy. Metal-oxide-semiconductor field-effect transistors (MOSFETs) were considered in this paper. In general, any other schematic device can be included.

Section II describes the overall methodology for data processing, graph connectivity, and feature encoding. Section III describes the overall methodology flow and the GCNs architecture used for the proposed implementation. Section IV describes the results of the methodology on tested schematics.

II. Primitives Structure Recognition

A novel SR scheme is outlined in this section to identify analog functional blocks in an input schematic. This scheme is inspired by the algorithm in [5], which is outlined in Fig. 2 and Fig. 3.

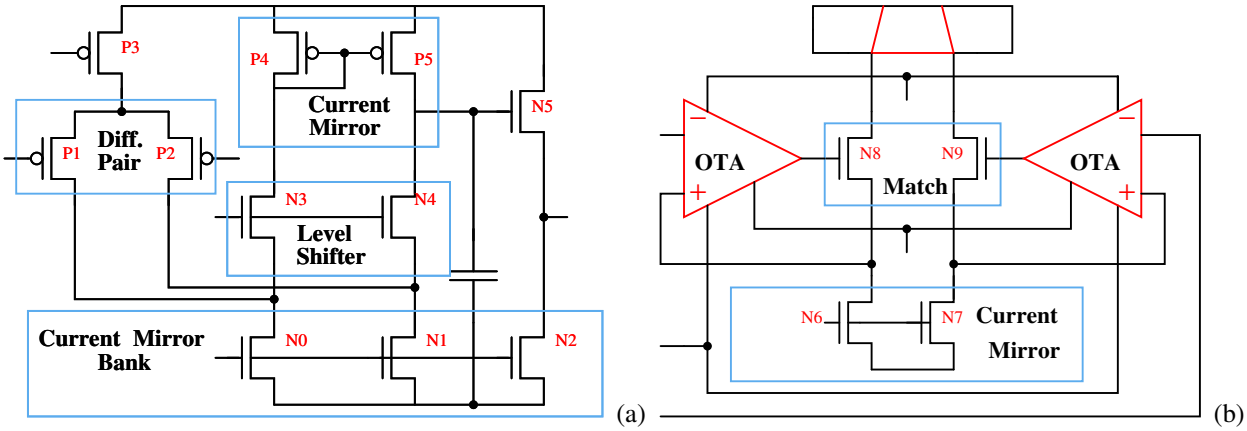

Fig. 1: (a) A novel ML algorithm is used to identify *functional blocks*. (b) Structures may transcend hierarchical boundaries: branches in two OTAs may form a single current mirror bank.

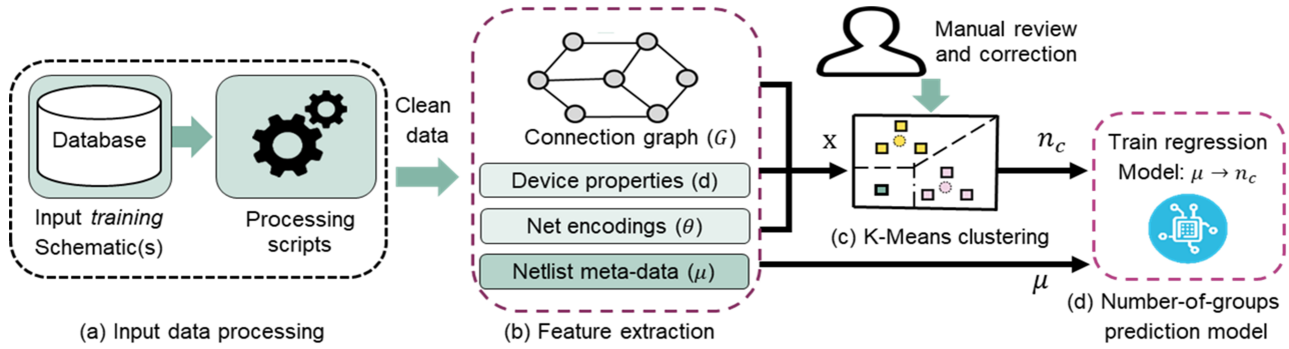

Fig. 2: Flow A (*a*) Input data processing, and the flattening of hierarchical schematics. (*b*) Feature extraction: Create schematic connection graph, extract device properties, net-encoding, and schematic meta-data. (*c*) K-Means clustering: Training clustering algorithm and manual review for the *number of groups* correction. (*d*) Number-of-groups prediction model: Trains regression model to predict *number of groups* using meta-data.

A. Input data processing

AMS circuit schematics are read by a data processing engine to generate clean data in a specific format for analysis. Hierarchical schematics (containing sub-circuits) are flattened during processing. This allows the methods of this paper to recognize structures across hierarchical bounds in the original (unflattened) circuit. The clean (output) data is written to a file in the a JavaScript Object Notation (JSON) format. After this step, the original schematic is no longer needed.

B. Feature extraction

The inputs features used are as follows:

1) Connection graph, $G = (\mathbf{V}, \mathbf{A})$: A schematic *connection graph* is created using the method defined in [3, Fig. 4]. The vertices, $\mathbf{V}$, correspond to the schematic transistor devices, and $|\mathbf{V}| = n_d$. Adjacency matrix $\mathbf{A} \in \mathbb{R}^{n_d \times n_d}$ defines the connection between vertices. An edge exists between vertices v_i and v_j, if the corresponding transistors are connected by a net in the schematic. In this case, element a_{ij} of $\mathbf{A}$ is non-zero. The specific value of a_{ij} is calculated using a universal coding scheme, and is based on the net connectivity definition in [3, Table II].
2) Device properties, $\mathbf{d}$: Vector ordering device properties. Effective width, W, is defined as a function of transistor finger width W_f, number of fingers N_f, and multiplicity factor M_f. MOSFET channel length is denoted by L. Device *model* is a reference to the transistor simulation (behavioral) model. Transistor *type* is $NMOS$ or $PMOS$. Since *model* and *type* are typically labels. For placement in a feature vector, they are encoded as one-hot vectors, as is done in [8, eq.(3)].

$$\mathbf{d} = [W, L, model, type], \; W = \frac{W_f}{N_f \times M_f}. \tag{1}$$

3) Netlist meta-data, $\boldsymbol{\mu}$: Vector of schematic statistics not specific to individual transistors. Five elements are counted and ordered: the number of transistors n_d; the number of device models n_{models} used in the schematic; the number of nets n_{nets}; and number of NMOS n_{NMOS} and PMOS n_{PMOS} transistors (2).

$$\boldsymbol{\mu} = [n_d, n_{models}, n_{nets}, n_{NMOS}, n_{PMOS}]. \tag{2}$$

4) Net encodings, $\boldsymbol{\theta}$: Each schematic net is labeled as a power, ground, or signal net. Limiting the analysis to MOSFET transistors with drain (d), gate (g), and source (s) terminals, a 9-bit terminal-to-net encoding vector is constructed for each transistor:

$$\begin{aligned}\boldsymbol{\theta} =&[\theta_{(g-power)}, \theta_{(s-power)}, \theta_{(d-power)}, \theta_{(g-ground)}, \theta_{(s-ground)}, \theta_{(d-ground)},\\ &\theta_{(g-signal)}, \theta_{(s-signal)}, \theta_{(d-signal)}],\ \boldsymbol{\theta} \in 2^9.\end{aligned} \tag{3}$$

The transistor bulk connection is not used as a feature. For example, if the drain terminal is connected to a power net, then $\theta_{(d-power)} = 1$ and is 0 otherwise. A combined *feature vector* is concatenated from the items above. Binary and integer values are treated as real:

$$\mathbf{x}_i = [\mathbf{a}_{i-}; \mathbf{d}_i, \tau{\cdot}\boldsymbol{\theta}_i], \tau \in (0.35, 0.75), \mathbf{x}_i \in \mathbb{R}^{n_p}, n_p = n_d + n_{models} + 4 + 9. \tag{4}$$

Schematic transistors are ordered and indexed by i, such that $1 \leq i \leq n_d$,

C. K-Means clustering algorithm

A non-parametric K-means clustering algorithm is applied to the *feature vectors* defined in Section II-B. The goal is to find the minimum-variance clustering of the vectors into n_k clusters. Each cluster corresponds to a schematic functional block. Let groups $\mathcal{G}_k, 1 \leq k \leq n_c$, denote schematic clusters. K-means clustering is employed twice in this paper:

1) Section II-D: Employ K-Means clustering to automatically predict the number of clusters per training schematic. This is used to build a regression model that predicts the *number of groups* from the meta-data: $\boldsymbol{\mu} \mapsto n_k$.
2) Section II-E: K-Means clustering is used to generate the labels of a device per cluster. The partial labels is later be utilized by Mask-GCN for grouping.

K-mean clustering of the feature vectors is equivalent to solving the following minimization problem [1, eq.(1)]:

$$\begin{aligned}&\text{Find } [\dot{\mathbf{x}}_1, \ldots, \dot{\mathbf{x}}_j, \ldots, \dot{\mathbf{x}}_{n_c}] \text{ to minimize } \mathbb{C},\\ &\mathbb{C} = \frac{1}{n_d}\sum_{i=1}^{n_d} \min_{j=1\ldots n_c} ||\mathbf{x}_i - \dot{\mathbf{x}}_j||^2,\end{aligned} \tag{5}$$

$\dot{\mathbf{x}}_j \in \mathbb{R}^{n_p}$, $1 \leq j \leq n_c$, denote the cluster center vectors. The cluster groups, $\mathcal{G}_k$, are also determined. A local minima for this problem can be found using Lloyd's algorithm [9]. An additional problem is to determine the optimal number of clusters (n_c). Equation (5) can be solved for different values of n_c between 1 and n_p.

For the first application (determine the number of clusters per training schematic), K-means clustering results are subjected to a manual (human) review. This step is needed to verify the correct devices in each group.

To expedite this procedure, a software tool was created to overlay cluster results on a schematic canvas, and to plot $\min \mathbb{C}$ vs. n_c. Invalid groups that be quickly edited or discarded from the same tool.

D. Number-of-groups prediction model

Fig. 4 shows the existence of some outlier schematics. It is essential either to remove or put less weights on them during model construction. In order to counter the outliers effect on model parameters estimation, a robust regression algorithm (6) is trained on netlist meta-data to predict the *number of groups* n_c for an input schematic.

$$n_c = \alpha_1(n_d)^{\gamma_1} + \alpha_2(n_{models})^{\gamma_2} + \alpha_3(n_{nets})^{\gamma_3} + \alpha_4(n_{NMOS})^{\gamma_4} + \alpha_5(n_{PMOS})^{\gamma_5} + \beta \tag{6}$$

Where α_i, β are regression parameters of the model trained on 25 circuits out of 30 available samples. The remaining data is used for testing. $\gamma_i's > 0$, which was determined experimentally during the model training and testing.

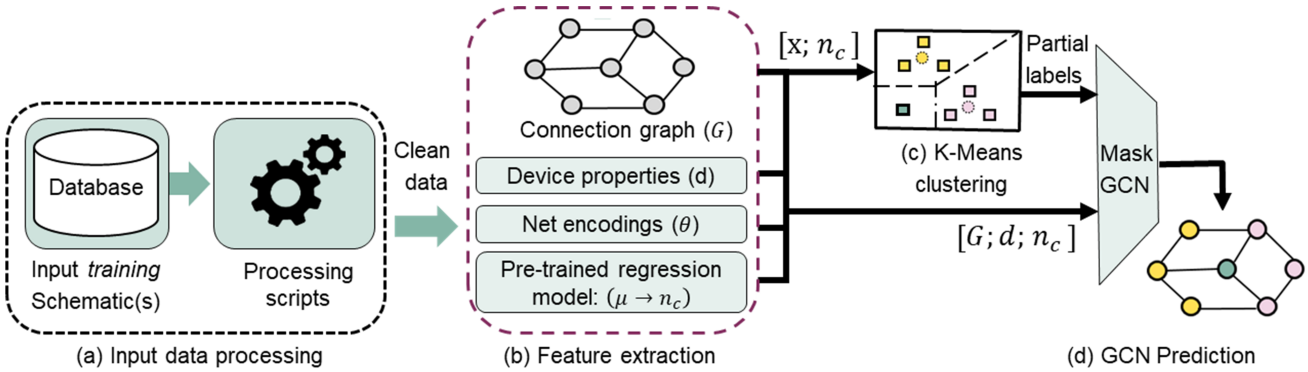

Fig. 3: Flow B (*a*) Input data processing, and the flattening of new hierarchical schematics. (*b*) Feature extraction: Create schematic connection graph, extract device properties, net-encoding, and meta-data for *number of groups* prediction using pre-trained linear model. (*c*) K-Means clustering: Train clustering algorithm to generate the partial labels for GCNs. (*d*) Mask GCN: Train GCNs model to figure out unlabeled devices.

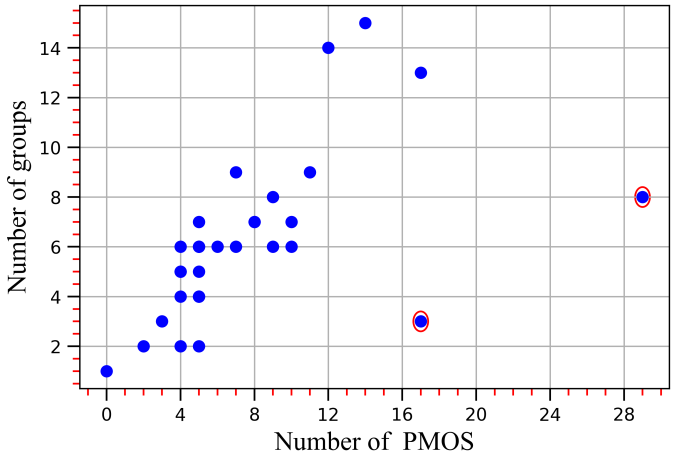

Fig. 4: Bi-Variate plot between meta-data versus *number of groups* from reviewed K-Means result. Each dot represents a schematic. The red circle can be designated as an outlier.

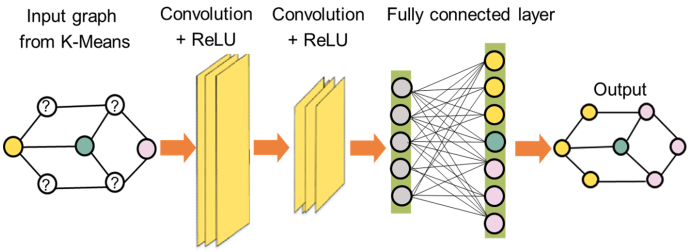

Fig. 5: GCN topology used in device clustering.

E. K-Means + GCN-based recognition

Fig. 3 shows the framework overview to cluster the devices at the transistor level for a given input schematic. It was done at the intersection of K-Means and GCNs algorithms. Further, to train the GCNs, three types of information are needed: connection graph, node features, and one labeled device per cluster in the circuit.

To generate the labels automatically, the K-Means algorithm (5) is run on the input *feature vector* defined in Section II-B. The K-Means generates as many clusters as the total number of devices in the schematic, which makes the problem sub-optimal and computationally expensive. To avoid this, the pre-trained regression model (6) was used to predict the *number of groups* n_c. This is then taken as an input to K-Means to generate the groups of the devices. The device closest in distance to each group centroid is used as the partial labels $\mathbb{L}$ for the GCNs. The adjacency matrix $\mathbf{A}$, device properties μ, net encodings θ and designated partial labels are then fed to a GCNs classifier and trained until the loss converges to a saturation value.

To further clarify, the GCNs trains off only on one schematic at a time using the partial labels information given to it by K-Means. The trained GCNs model is then responsible for figuring out the rest of the unlabeled devices in the schematics in the respective cluster they might fall into. In this approach, the framework can be restricted to only using K-Means to figure out the labels of every single device. However, it was concluded that this method lacks accuracy in preserving the device connectivity information for certain schematic (see Fig. 7 and Fig. 8). Therefore, the K-Means algorithm was only used to get the device closest to the cluster centroid location.

Algorithm 1: K-Means + GCN

Input:
  Initialize GCN with parameters w, pre-trained regression model (l_m), adjacency matrix (A), device properties (d), netlist meta-data (μ), net encodings (θ)
Output:
  Transistors grouping result of an input schematic
$n_c \leftarrow l_m(\mu)$;
$\mathbf{x} = [A, \mathbf{d}, \theta]; \mathbf{x} \in \mathbb{R}^{n_d \mathbf{x} n_p}$
$[\dot{\mathbf{x}}_1, \ldots, \dot{\mathbf{x}}_j, \ldots, \dot{\mathbf{x}}_{n_c}] \leftarrow$ K-Means($n_c, \mathbf{x}$)
$\mathbb{L} \leftarrow \emptyset$; ▶ Initialize an empty list for partial labels
for $j = 1, n_c$ **do**
  num=MAX_NUMBER;
  for $i = 1, n_d$ **do**
    if $||\mathbf{x}_i - \dot{\mathbf{x}}_j|| <$ *num* **then**
      num=$||\mathbf{x}_i - \dot{\mathbf{x}}_j||$; $(\mathbf{x}_i, \dot{\mathbf{x}}_j) \in \mathbb{R}^{n_p}$
      $x_{i_{min}} = x_i$
  $\mathbb{D} \leftarrow$ cluster label of $x_{i_{min}}$;
  $\mathbb{L} \leftarrow \mathbb{L} \cup \{\mathbb{D}\}$; ▶ Append label to the list
  return $\mathbb{L}$
$\mathbf{y} = [\mathbf{d}, \theta]; \mathbf{y} \in \mathbb{R}^{n_d \mathbf{x} n_{p_{gcn}}}$ where, $p_{gcn} = dim$(d)+9
for $epoch = 1, M$ **do**
  Train(GCN,$A, \mathbf{y}, \mathbb{L}|w$)
Predict labels for other devices:
Transistors grouping $\leftarrow$ Test(GCN,$A, \mathbf{y}|w$);

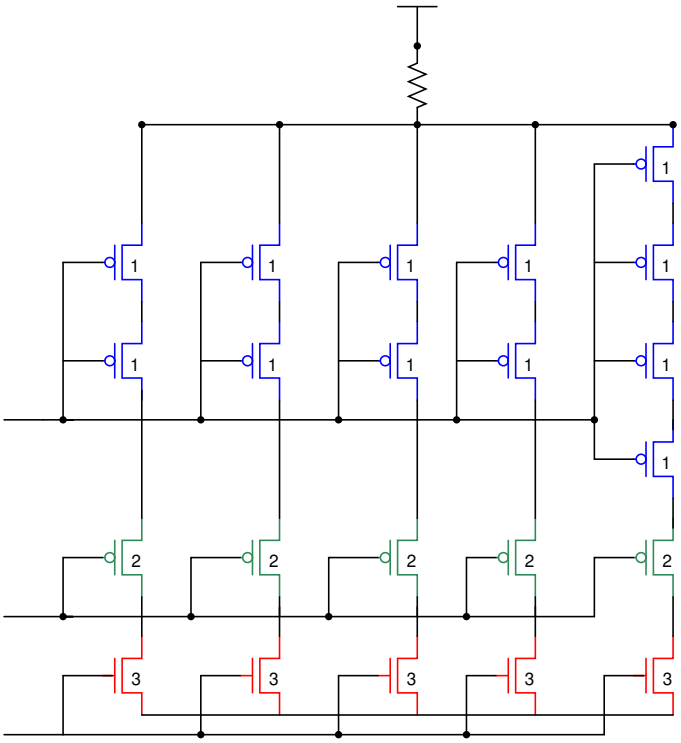

Fig. 6: DAC trim circuit. Num. 1-2-3 represnt the group ID.

III. Graph Convolutional Neural Network

A. GCN Custom Layer Operation

To train a semi-supervised model that can accurately make the device grouping prediction on unseen schematic, a novel graph neural network layer architecture is developed (modified from [10, eq.(3)] by transforming into weighted graph convolution as shown in (7) and (8) with α_{ji}). The modified algorithm was found to be performing 30% better in term of accuracy than the other open source graph-based libraries described in [7]. The role of graph neural networks is to encode and extract the node information from its neighbors. It uses convolution based on the connectivity of a node within a large graph and generates vector representation which can be used in subsequent tasks such as node and graph classification. The node features include a concatenated vector representation of device properties and net encodings. The schematic graph information and one label per cluster generated by K-Means are also passed as inputs to the algorithm. Thereafter, the following updates are performed: 1) each edge updates its representation by applying a fully connected network (φ) to the concatenated representation of connecting node embeddings (by applying another fully connected network θ to node feature) with trainable parameter w corresponding to the edges to adjust the learning, and 2) each node updates its embedding by taking the mean of adjacent edge embeddings connected to its source nodes. The node and edge updates are shown in (7) and (8).

$$e_{ji} = \varphi\{concat(\theta(v_j)\alpha_{ji}, \theta(v_i), w_{ji})\} \tag{7}$$

$$v_i = \frac{\sum_{j=N(v_i)} e_{ji}}{N(v_i)} \tag{8}$$

Here, j and i are source and destination node respectively

θ: shared MLP across the node for node embedding generation
φ: shared MLP across the edge for edge embedding generation
N: total number of neighboring nodes to source node
w_{ji}: learnable weights parameter
e_{ji}: edge vector representation
v_i: node vector representation
α_{ji}: edge weight from source to destination node

Node embeddings are denoted by $v_i s$ for $1 \leq i \leq n_d$, where, n_d is the total number of transistor devices in the schematic. The outputs of this convolutional operation is node embeddings.

B. GCN Topology for Technology Node Classification

For the layer defined in (7) and (8), a GCNs is constructed that uses multiple convolutional stages. The GCNs topology used in this work is shown in Fig. 5. It has two convolutional layers which are fed to a fully connected (fc) layer whose outputs provide the classification results. To clarify further, the final layer is a fully connected layer of dynamic size (as it depends on the *number of groups* predicted by the model in (6) for a given input schematic) followed by a softmax function for node classification.

IV. Experimental Results

TABLE I: Netlist meta-data description.

Commercial Technology	#Circuits	#Inputs	#Outputs
120 nm node	20	5	1
40 nm node	10	5	1

TABLE II: Robust regression model result on test schematics. The DAC trim circuit is an outlier as described in Section IV-C.

Circuits	Predicted	Actual	Rounded Pred.	Error
OTA-1	4.33	5	4	1
Comparator	5.41	6	5	1
OTA-2	8.52	9	9	0
DAC trim*	9.29	3	9	-6
OTA-3	5.86	7	6	1

TABLE III: Grouping algorithm results on multiple schematics.

Circuits	Num. Devices	Rounded Pred. (n_c)	Possible Num. Groups $[n_c-2, \cdot, \cdot, \cdot, n_c+2]$	K-Means+GCNs n_c	K-Means+GCNs Accuracy	K-Means+GCNs Best Possible Num. Groups	K-Means+GCNs Accuracy
OTA-1	9	4	2,3,4,**5**,6	4	78%	**5**	100%
Comparator	14	5	5,**6**,7,8,9	5	65%	**6**	100%
OTA-2	24	9	7,8,**9**,10,11	9	95.6%	**9**	95.6%
DAC trim*	44	9	7,8,9,10,11	9	63%	**3**	100%
OTA-3	14	6	4,5,6,**7**,8	6	71.5%	**7**	100%

TABLE IV: Comparison of this work with other methods in [5, Table I].

Algorithm	Maximum Num. Devices	Num. Circuits	Scalability	Automation?	Capture Hierarchy?
Library-based [1], [2]	22	2	No	No	No
Knowledge-based [3]	37	5	No	No	No
Learning-based [4]	34	34	Yes	No	Yes
ML-based [5]	35	6	Yes	Yes	No
This work	**44**	**30**	**Yes**	**Yes**	**Yes**

A. Training set for number-of-groups prediction model

The training set for the regression model was taken from the schematics of multiple commercial technologies. Using these sources, two labeled datasets for 120 nm and 40 nm technology have been generated with characteristics as listed in Table I. Only one schematic is kept in the data set when two or more schematics are sharing the same meta-data μ. Dummy MOSFETs devices (if all the terminals of a device is connected with the same net) were removed during the processing stage of the schematics.

B. Number-of-groups prediction model results

Table II visualizes the performance of trained robust regression model (6) on the test circuits in predicting the *number of groups*. Predicted values were rounded up to its nearest integer point value to get the correct grouping number. Model error is calculated based on the actual and rounded prediction values. In this case, $|Error| \leq 2$ for most of the tested circuits. So, given the predicted *number of groups* n_c and total number of devices n_d for a circuit, the designer has to only iterate through $[\max(1, n_c-2), \cdot, \cdot, \cdot, \min(n_c+2, n_d)]$ possible *number of groups* as shown in Table III for the given test circuits.

C. K-Means Vs K-Means + GCNs Results

Fig. 7 and Fig. 8 show the visualization of the grouping results with K-Means and K-Means+GCNs respectively. Each color encoding represents the devices grouped in the respective cluster. There are some false prediction cases in both the algorithms as shown inside the boundary box on to the schematic. Misclassified device in K-Means result fails to preserve the connectivity information with other devices falling in that cluster. It only considers the local vector representation of each device in the schematic. In contrast, K-Means+GCNs is taking care of the connections in grouping the devices. GCN can easily be able to learn the vector representation of a device based on its surrounding nodes. Table III shows the accuracy measured on device by device in (9) for input schematics when predicted and best number of groups is fed to the learning algorithm.

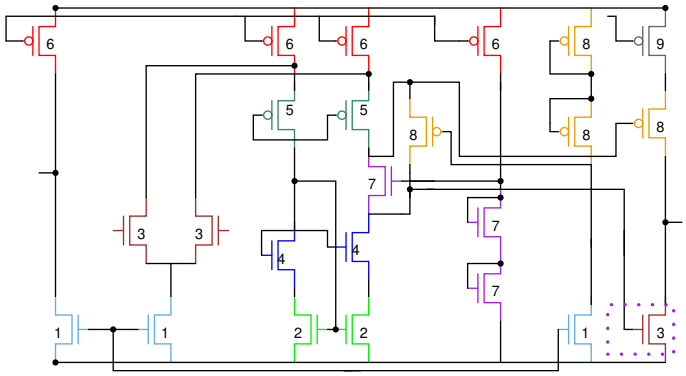

Fig. 7: Schematic clustering result. Numbers 1 to 9 represent the transistors grouping ID generated by K-Means clustering.

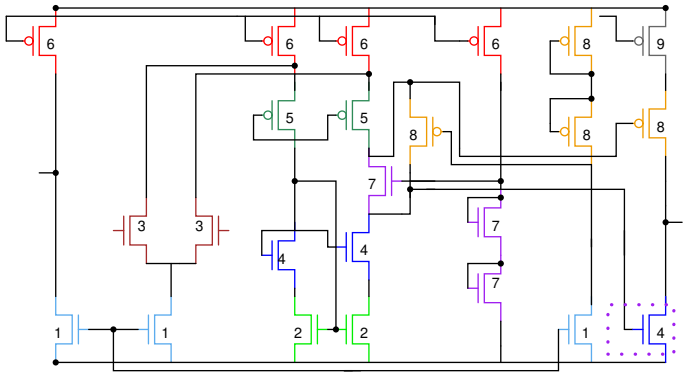

Fig. 8: Schematic clustering result. Numbers 1 to 9 represent the transistors grouping ID generated by K-Means + GCNs.

For example, "OTA-1" circuit requires 5/9 application runs considering [2,3,4,**5**,6] groups for K-Means+GCNs to figure out the best possible number of groups (5 groups in this case) and clustering results. DAC trim* circuit in Fig. 6 is a special type of circuit and can be considered as an outlier based on the training data. Structures in this circuit contain large amount of series connected devices. Hence, robust regression model over estimated the number of groups by 6 units during prediction (see Table II). Due to this high error-rate in this particular circuit, K-Means+GCNs algorithm is not able to find out the optimal number of groups (3 in this case) in the estimated list of values as shown in Table III. However, it makes accurate grouping prediction in Fig. 6 when correct number of groups is passed to the learning algorithm.

Table IV compares this work with the earlier methods on the basis of scalability, automation, and hierarchical structure recognition. Comparing accuracy, in this case, can be a topic of subjectivity. As few of these methods are applied to different circuits and pre-defined structures. A detailed comparative analysis is listed in Section I.

$$Accuracy = \frac{\#\ devices\ correctly\ classified\ in\ respective\ group}{\#\ devices\ in\ an\ input\ schematic} \tag{9}$$

V. Conclusion and Future Work

When designing AMS-IC, one of the most challenging and time-consuming tasks is to identify the structures in a hierarchical schematic. It also requires deep domain expertise for manual structure recognition. Currently, there is still no industrial tool which can automatically identify the structures at the schematic level for CDD flow. Prior state-of-the-art methods lack automation. As, it is constrained on pre-defined rules and correct drawing of the schematic. In this paper, we applied ML techniques to automatic structure recognition task across the hierarchical boundaries without making any prior assumption. The experimental result shows that the proposed method based on the K-Means and GCNs algorithm is promising. Future work lies in to develop a ML based classification algorithm to automatically designate the functional labels of identified simple and hierarchical structures into multiple categories such as current mirror, differential pair, cascode current mirror etc. This will be utilized further for constraints (rules) generation.

References

[1] T. Massier, H. Graeb, and U. Schlichtmann, "The sizing rules method for CMOS and bipolar analog integrated circuit synthesis," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 27, no. 12, pp. 2209–2222, 2008.

[2] M. Meissner and L. Hedrich, "FEATS: Framework for explorative analog topology synthesis," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 34, no. 2, pp. 213–226, 2015.

[3] 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 Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 34, no. 2, pp. 199–212, 2015.

[4] H. Li, F. Jiao, and A. Doboli, "Analog circuit topological feature extraction with unsupervised learning of new sub-structures," in *2016 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2016, pp. 1509–1512.

[5] K. Settaluri and E. Fallon, "Fully automated analog sub-circuit clustering with graph convolutional neural networks," in *2020 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2020, pp. 1714–1715.

[6] X. Jin and J. Han, *K-Means Clustering*. Boston, MA: Springer US, 2010, pp. 563–564. [Online]. Available: https://doi.org/10.1007/978-0-387-30164-8_425

[7] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.

[8] H. Wang, K. Wang, J. Yang, L. Shen, N. Sun, H.-S. Lee, and S. Han, "GCN-RL circuit designer: Transferable transistor sizing with graph neural networks and reinforcement learning," in *2020 57th ACM/IEEE Design Automation Conference (DAC)*. IEEE, 2020, pp. 1–6.

[9] S. Lloyd, "Least squares quantization in pcm," *IEEE Transactions on Information Theory*, vol. 28, no. 2, pp. 129–137, 1982.

[10] A. Mirhoseini, A. Goldie, M. Yazgan, J. Jiang, E. Songhori, S. Wang, Y.-J. Lee, E. Johnson, O. Pathak, S. Bae *et al.*, "Chip placement with deep reinforcement learning," *arXiv preprint arXiv:2004.10746*, 2020.