S^{3}DET : Detecting System Symmetry Constraints for Analog Circuits with Graph Similarity

Mingjie Liu, Wuxi Li, Keren Zhu, Biying Xu, Yibo Lin, Linxiao Shen, Xiyuan Tang, Nan Sun, and David Z. Pan
ECE Department, The University of Texas at Austin, Austin, TX, USA
{jay_liu, wuxi.li, keren.zhu, biying, yibolin, lynn.shenlx, xitang}@utexas.edu, nansun@mail.utexas.edu, dpan@ece.utexas.edu

***Abstract*—Symmetry and matching between critical building blocks have a significant impact on analog system performance. However, there is limited research on generating system level symmetry constraints. In this paper, we propose a novel method of detecting system symmetry constraints for analog circuits with graph similarity. Leveraging spectral graph analysis and graph centrality, the proposed algorithm can be applied to circuits and systems of large scale and different architectures. To the best of our knowledge, this is the first work in detecting system level symmetry constraints for analog and mixed-signal (AMS) circuits. Experimental results show that the proposed method can achieve high accuracy of 88.3% with low false alarm rate of less than 1.1% in large-scale AMS designs.**

I. Introduction

High-performance analog circuits are susceptible to process variations and layout-dependent effects [1]. The current state-of-the-art analog layout synthesis tools rely on well-formulated constraints that are strictly followed by the placement and routing engines, such as symmetry, net shielding and minimum parasitics. These constraints have to be obeyed to ensure the post-layout performance and robustness of analog circuits [2].

Symmetry constraints are one of the most essential and widely adopted constraints applied during analog layout synthesis [3], [4]. Analog designs frequently use differential topologies to reject common-mode noise and enhance circuit robustness [1]. Mismatch in the devices of these topologies would significantly degrade the circuit performance. In layout designs, these device pairs need to be placed and routed symmetrically to enforce matching.

The matching for parasitics between critical building blocks is especially important for high-speed and high-precision analog system designs. The experiment on a time-interleaved analog-to-digital converter (TI-ADC) shown in Fig. 1 illustrates the degradation on system performance caused by mismatch. The timing skews between different sampling channels have a significant impact on the signal-to-noise and distortion ratio (SNDR) [5]. As shown in Fig. 2, a sampling mismatch of 0.1% relative to the clock period could lead to a degradation in SNDR of almost 15dB. Thus the timing paths highlighted in Fig. 1 need to be carefully matched in the layout implementation. In other words, symmetry placement and routing constraints need to be applied between the delay line cells (DL), sampling switches, and sub-ADC (SAR) channels.

Existing works for symmetry constraint detection focus on generating constraints for building block level circuits such as differential amplifiers and comparators. Charbon et al. [6] use ***sensitivity analysis*** to identify symmetry and matching constraints by circuit simulations. This approach is useful in extracting high-quality and critical matching constraints directly related to performance. Another type of algorithms use ***graph matching***. This type of algorithms represent circuits as different types of graphs and use various techniques to convert symmetry detection into graph matching. The works of [7], [8] convert analog circuits into bipartite graphs with signal flow analysis and use graph automorphism to detect graph symmetry. Wu et al. [9] further extend the graph representation by embedding pin connection information into edge weights and generate constraints based on subgraph isomorphism with a template circuit library. Eick et al. [10] directly generate constraints by pattern matching and group constraints hierarchically based on structural signal flow graphs. Nonetheless, previous works have only demonstrated success in generating symmetry constraints for simple and small building block circuits.

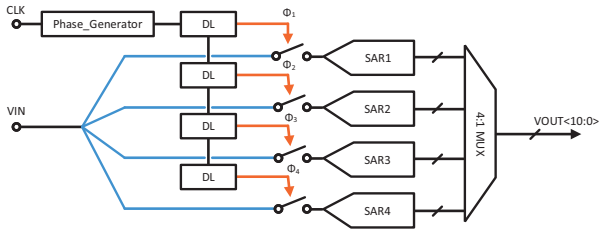


Fig. 1: TI-ADC timing sensitive paths.

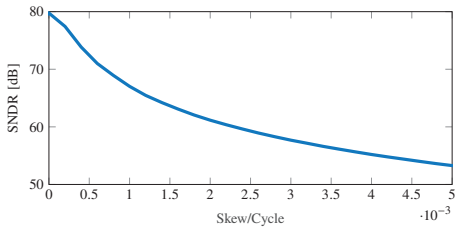


Fig. 2: Effect of timing mismatch on TI-ADC performance.

On the other hand, methods for generating system symmetry constraints have rarely been studied. There still exists a gap in automatic constraint detection and constraint management for system level analog designs [11]. Previous methods of constraint detection have difficulties in scalability and expressiveness when directly migrating to analog circuit systems. ***Sensitivity analysis*** based approaches require a large number of simulations and can not scale with design complexity. For example, while the performance simulation for an amplifier only takes minutes, the transistor level simulation for an entire ADC can take several hours. ***Graph matching*** algorithms are computationally expensive and infeasible for large circuits in practice. Generating a pattern library on the system level is also extremely difficult due to the flexibility of custom-designed analog circuits. Furthermore, the circuit graph abstractions of prior works are limited to only consider CMOS or bipolar transistors. In real-world analog designs, passive devices such as resistors and capacitors are also commonly used and critical in matching.

In this paper, we propose **S^{3}DET**, a general method of detecting system symmetry constraints for analog circuits leveraging graph similarity. In contrast to previous works, **S^{3}DET** detects hierarchical

978-1-7281-4123-7/20/$31.00 © 2020 IEEE

symmetry constraints among subcircuit building blocks for complete systems. Our main contributions are summarized as follows:

- We propose **S^{3}DET**, a general method of detecting system symmetry constraints for analog circuits.
- We propose a graph representation of analog circuits with the extended capability of handling passive devices.
- We propose a subgraph extraction technique using graph centrality to improve the constraint quality and reduce false alarms.
- We detect symmetry with graph similarity leveraging spectral analysis compared to unscalable graph matching algorithms.
- Experimental results demonstrate the effectiveness of our proposed method with practical taped-out designs.

The rest of the paper is organized as follows. Section II formulates the system symmetry constraint detection problem and gives background of graph similarity; Section III explains the detailed implementation of **S^{3}DET**; Section IV demonstrates the experimental results; Section V concludes the paper.

II. Preliminaries

In this section, we first define the system symmetry constraint detection problem for analog circuits (Section II-A). Then we give a brief overview of graph matching algorithms commonly used in previous works of constraint generation (Section II-B). Finally, we introduce a non-parametric statistical test for comparing sample probability distributions (Section II-C), which will be later revisited in Section III-B as a scalable metric for graph similarity.

A. System Symmetry Constraint Detection for Analog Circuits

We define the system symmetry constraint detection problem for analog circuits as follows. The hierarchical circuit netlist N is given as input. The circuit hierarchy is abstracted into a tree T, with each node representing a subcircuit. For each subcircuit $v \in T$, its children $G = \{g | g \subseteq v\}$ are subcircuits referenced in v. Symmetry constraint pair (g_i, g_j) represents critical matching is needed between the subcircuits, where $g_i, g_j \in G$. The system symmetry constraint detection problem is to generate constraint pairs for every subcircuit $v \in T$.

A simplified example of system symmetry constraints is shown in Fig. 3. The circuit COMP consists of 3 subcircuits: COMP_CORE, INV1 and INV2. The inverters in COMP need to be matched with symmetry constraint pair (INV1, INV2). System constraints do not consider transistor device symmetry in subcircuits, such as COMP_CORE. If COMP_CORE further contains other subcircuits, the system constraints between these subcircuits should also be considered.

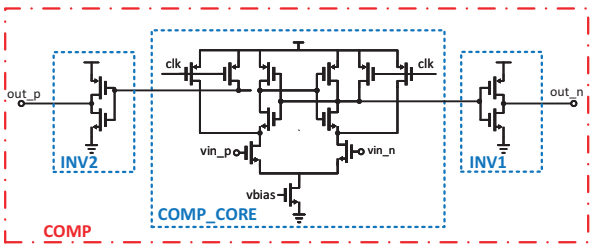


Fig. 3: Hierarchical circuit example.

B. Graph Similarity and Graph Matching

Graph similarity evaluates the similarity of two graph objects. A commonly used graph similarity technique is graph matching, which maps similar vertices and edges of the two involved graphs so that a comparison of the graphs could be conducted.

Graph matching algorithms can be classified into two categories: exact and inexact graph matching. Exact graph matching is also known as the graph isomorphism problem. Its generalization, the subgraph isomorphism problem, determines whether the larger graph G_1 contains a subgraph that is isomorphic to G_2. Inexact graph matching attempts to find the best possible match of the two graphs. Graph edit distance [12] is a commonly used metric for inexact matching. It is defined as the length of the cheapest sequence of graph edit operations (e.g., node insertion, node deletion, etc.) to transform one graph to the other.

Previous works for symmetry detection based on graph matching do not scale to system designs. In practice, graph representations of entire designs can consist thousands of nodes. However, no known polynomial algorithm exists for graph isomorphism. Subgraph matching and graph edit distance are also proven to be NP-complete [13]. Thus, graph matching algorithms are infeasible for measuring graphs similarity of large system designs.

C. Kolmogorov-Smirnov Test

S^{3}DET uses the two-sample Kolmogorov-Smirnov (K-S) test to measure the similarity between the eigenvalue distributions of two graphs. The K-S test is a non-parametric statistical test used to compare a sample with a reference probability distribution [14]. The extended two-sample K-S test is used to test whether the underlying probability distributions differ between the two samples.

The K-S statistic of two empirical cumulative distribution function $F_{1,n}(x)$ and $F_{2,m}(x)$ with sample size n and m is defined as:

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

It quantifies the difference between the two distributions. The probability value (p-value) from the K-S test measures the probability that the K-S statistic would be as large as observed if sampled from the same distribution. A small p-value concludes that the two samples are from different distributions, while a large p-value infers that the distributions match.

III. S^{3}DET Algorithm

In this section, we first give an overview of **S^{3}DET** in Section III-A. Then the detailed implementations are explained in the following subsections.

A. S^{3}DET Algorithm Overview

S^{3}DET detects system symmetry constraints for analog circuits based on the existing subcircuit hierarchies of the netlist. **S^{3}DET** consists of two main parts: (1) netlist pre-processing and (2) symmetry detection.

Netlist pre-processing generates constraint candidates and the flattened graph of the circuit. Constraint candidates are generated while traversing the hierarchy tree (Section III-B). Invalid constraint candidates are removed based on the subcircuit type labels propagated in the hierarchy tree. Since global structural information is needed, the flattened graph representation of the entire system is generated prior to symmetry detection (Section III-C).

Constraint candidates are classified as valid or invalid during symmetry detection. Symmetry constraints are detected if the circuit graphs are similar. To improve constraint quality and reduce false alarms, the neighboring circuit topologies are extracted on the entire circuit graph (Section III-D). The sizes of extracted subgraphs are determined by graph centrality (Section III-F). The similarities of

the extracted subgraphs are measured with a scalable graph similarity metric using spectral graph analysis (Section III-E).

Algorithm 1 S^{3}DET Algorithm

Input: Hierarchical Netlist N
Output: Hierarchical Symmetry Constraint

```
1: Generate hierarchy tree T and propagate label        ▷ Section III-B
2: Construct flattened graph representation G           ▷ Section III-C
3: Node v ← T.root
4: function S3DET (v)
5:     while v is not leaf cell do
6:         for valid subcircuits g1 and g2 of v do
7:             g1', g2' ← Extract(G, g1, g2)            ▷ Section III-D
8:             if GraphSim(g1', g2') then               ▷ Section III-E
9:                 Add symmetric constraint g1, g2 to hierarchy v
10:        for child node c of v do
11:            S3DET (c)
12: end function
```

Algorithm 1 shows the overall flow of **S^{3}DET** . First the netlist is processed. The hierarchical tree is generated with label propagation (line 1). Then the flattened graph representation of the entire circuit is constructed (line 2). Next, the neighboring subcircuit topologies for constraint candidates are extracted (line 7) while traversing the hierarchy tree. Finally, symmetry constraints are detected based on the subgraphs similarities (line 8).

B. Hierarchy Tree and Label Propagation

S^{3}DET abstracts the hierarchies of the netlist into a tree representation. Each subcircuit instantiation is abstracted with a node in the tree. The root of the hierarchy tree is the entire analog circuit system. The leaf cell nodes are device level instantiations of transistors, resistors, capacitors, and diodes. Figure 4 shows an example of the extracted hierarchy tree of the corresponding circuit in Fig. 3.

We label primal subcircuits as analog or digital in the netlist to improve constraint quality. Primal subcircuits are subcircuits only consisting of device instantiations and do not refer to other subcircuits. The amount of effort needed in labeling is trivial since most designs consistently reuse components. The labels are then propagated in the hierarchy tree, with parent node labeled as: (1) analog (digital) if all its children subcircuits are analog (digital), (2) mixed-signal if otherwise. As shown in Fig. 4, primal subcircuits are labeled as digital (blue) or analog (yellow). The propagated label to the parent node accordingly is mixed-signal (red).

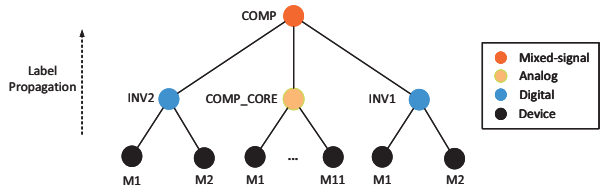


Fig. 4: Hierarchy tree and label propagation of circuit in Fig. 3

During symmetry constraint detection, **S^{3}DET** only generates constraints for valid candidates. Subcircuit pairs are invalid if the labels differ or are digital blocks not connected to analog blocks. **S^{3}DET** also considers symmetry constraints between critical passive devices in mixed-signal blocks, such as feedback capacitors of amplifiers.

C. S^{3}DET Graph Representation of Analog Circuits

Compared to previous works, **S^{3}DET** simplifies the graph circuit representation and improves generalization for commonly used passive device elements. Device instances and device pins are represented as graph vertices. Edges connect device instances to their corresponding pins. Pin vertices are connected in the graph accordingly if there exists net connections in the netlist. Power, ground and clock connections are removed in the graph representation. These high degree net connections cause unnecessary constraints due to large extracted subgraphs (Section III-D). Figure 5 (a) shows an example of the proposed graph representation on a simple circuit.

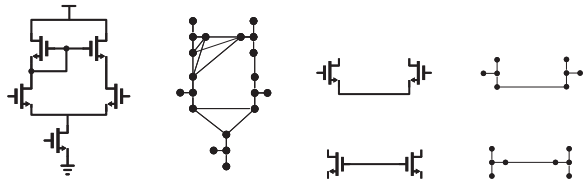


Fig. 5: Graph representation. (a) An example on a sample circuit. (b) Different circuit toplogies resulting in isomorphic graphs.

Our simplified graph representation is adequate to differentiate subcircuit graphs. Even though different pins of the same device are treated as the same type of vertex, the graph is still able to capture key topologies of entire circuits. Different device connections may have isomorphic graph representations, as shown in Fig. 5 (b). However, our algorithm intends to extract system level symmetry constraints rather than the device level. Experimental results show that the proposed graph representation effectively preserves similarities and differences for subcircuits of analog systems.

D. Subgraph Extraction

S^{3}DET extracts subgraphs of subcircuits with neighboring circuit topologies to improve constraint quality and reduce false alarms. Only comparing the subcircuits is not enough to fully characterize symmetry constraints. In AMS designs, extensively used digital standard cells cause false alarms. These false alarms are unnecessary constraints that can lead to performance degradation in subsequent layout design stages. Figure 6 shows an example of a false alarm based on the design of a second order continuous time delta-sigma ADC (CT-ΔΣ-ADC). The digital loop filters on different feedback paths cause false alarms if only considering the subcircuits. Only filter pairs (A,B) and (C,D) should be matched, while symmetry constraints such as (A,C) and (A,D) are unnecessary.

The extracted neighboring circuit size is critical to the quality of symmetry constraint detection. Extracting small subgraphs would not include enough information. On the other hand, large extracted subgraphs also cause false alarms. Large subgraphs of closely connected subcircuits would fully include both subcircuits and all their neighboring circuit topologies. In this case, the subgraphs would be detected as similar and create unnecessary constrained symmetry. Thus, to improve constraint quality, the extracted subgraphs need to be carefully sized in consideration of (1) the proximity of circuit connections, and (2) the subcircuit size.

To resolve such issues, **S^{3}DET** uses graph centrality to determine the extracted subgraph radius in Algorithm 2. The subcircuit graph centers and radius are calculated using graph centrality introduced in Section III-F (line 2). The extracted subgraph radius is then defined

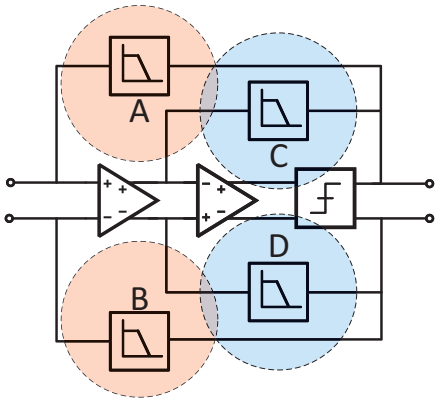


Fig. 6: CT-ΔΣ-ADC. Digital filters on different feedback paths cause false alarms. Circles enclosing cells represent extracted subgraphs. Only (A,B) and (C,D) need matching, while constraints such as (A,C) and (A,D) are unnecessary.

Algorithm 2 Subgraph Extraction

Input: Graph of entire circuit G, Subcircuit graphs g_1 and g_2
Output: Extracted subgraphs g_3 and g_4

1: **function** EXTRACT(G, g_1, g_2)
2: $c_{1,2} \leftarrow$ Center($g_{1,2}$) ▷ Section III-F
3: $rad_{1,2} \leftarrow$Eccentricity($c_{1,2}$) on $g_{1,2}$ ▷ Equation 2
4: $sub_rad \leftarrow \frac{1}{2}$Shortest_path_distance($c_1$, c_2)
5: $sub\_rad \leftarrow max(sub\_rad, \alpha_{min} \cdot rad_{1,2})$
6: $sub\_rad \leftarrow min(sub\_rad, \alpha_{max} \cdot rad_{1,2})$
7: $g_{3,4} \leftarrow$ Subgraph(G, $c_{1,2}$, sub_rad)
8: **return** g_3, g_4
9: **end function**

as half of the shortest path distance between the two graph centers (line 4). Line 5 provides a lower bound on the subgraph size to be at least α_{min} times of the subcircuit size. Similarly, line 6 provides an α_{max} upper bound. Line 7 extracts the subgraph with the newly defined subgraph radius. The extracted subgraphs are supergraphs of the original subcircuits which contain neighboring circuit toplogies.

E. Graph Similarity with Spectral Graph Analysis

Spectral graph theory analyzes the properties of the graph in relationships to the matrices associated with the graph. Specifically, we use the graph *Laplacian* matrix $L = D - A$, where D is the degree matrix and A is the *adjacency* matrix.

The graph *Laplacian* matrix is closely linked to the structure of the graph. It contains the degree distribution as well as the adjacency information from the graph. Its eigenvalues measure the node cluster cohesiveness and algebraic connectivity of the graph [15]. The graph *Laplacian* has also been used to approximate sparsest cuts and partition circuit netlists in VLSI designs [16].

S^{3}DET applies a scalable graph similarity algorithm using graph spectral analysis. Since the eigenvalue distribution of the graph *Laplacian* is closely linked with the graph structure, the K-S statistic of the eigenvalue distribution can be used as a graph similarity metric [15]. **S^{3}DET** proposes the following graph similarity test:

1) The eigenvalues of the two graph *Laplacian* matrices are calculated and sorted.
2) The two-sample K-S test is conducted to test whether the underlying distributions differ for the two sets of eigenvalues.
3) The resulting p-value of the K-S test is used as the graph similarity score.
4) The two graphs are identified as similar if the similarity score is larger than a preset tolerance tol.

F. Graph Centrality

S^{3}DET uses graph centrality to determine the size of the extracted subgraphs. The selected graph centers directly affect the quality of the detected constraints. In this section, we introduce three common centrality metrics in graph and network analysis. **S^{3}DET** extracts different subgraphs based on the three centrality metrics and uses the average graph similarity score for system symmetry detection.

1) Jordan Center: The Jordan center is closely related to the radius and size of the graph [17]. The Jordan center of a graph is the vertex where the greatest distance to any other vertex is minimal. Define the eccentricity of a vertex v as the greatest distance to any other vertex:

$$Eccentricity(v) = \max_{u \in V} d(v, u), \tag{2}$$

where the distance of two nodes is the number of edges in the shortest path. The Jordan center is thus the vertex with minimal eccentricity.

2) Eigenvector Centrality: Eigenvector centrality is closely related to the graph *adjacency* matrix A [17]. It measures the influence of a node in a network based on graph connections. A vertex with high eigenvector centrality score is connected to many other vertices whom themselves have high scores. The centrality score x for a vertex v is defined as:

$$x_v = \sum_{u \in N(v)} x_u = \sum_{u \in V} a_{vu} x_u, \tag{3}$$

where $N(v)$ are the neighbor vertices of v and a_{vu} is the corresponding entry in the *adjacency* matrix A of the graph. The eigenvector center is the vertex with the highest eigenvector centrality score.

3) PageRank Center: PageRank extends the idea of eigenvector centrality by normalizing the score of vertices by its output degree. It is related to the normalized graph *Laplacian*. Brin and Page [18] originally proposed PageRank as a link analysis algorithm for assigning numerical weights to hyperlinked objects. The PageRank adaptation for undirected graphs is defined as:

$$PR(v) = \zeta \sum_{u \in N(v)} \frac{PR(u)}{deg(u)} + \frac{1 - \zeta}{|V|}, \tag{4}$$

where ζ is the damping factor. The PageRank center is assigned to the vertex with the highest PageRank score.

IV. Experimental Results

We implement the proposed algorithm in Python. Auxiliary functions for calculating graph eccentricity and eigenvector centrality are based on NetworkX [19]. The two sampled K-S test is implemented in the statistical tests toolbox in SciPy [20]. All experiments were performed on a Linux workstation with Intel 3.4GHz i7-3770 CPU and 32GB memory.

We conducted experiments on three different ADC architectures for system symmetry constraint detection. All three designs were previously taped-out and tested. For our experiments, we used the default parameters in Table I. **S^{3}DET** takes spice netlist as input with the primal subcircuits labeled as analog or digital. Statistics on the circuit graphs and ADC architectures are shown in Table II. Valid pairs are potential constraint candidates defined in Section III-B. The detected symmetry constraints by **S^{3}DET** are then verified and compared with constraints given by the circuit designer.

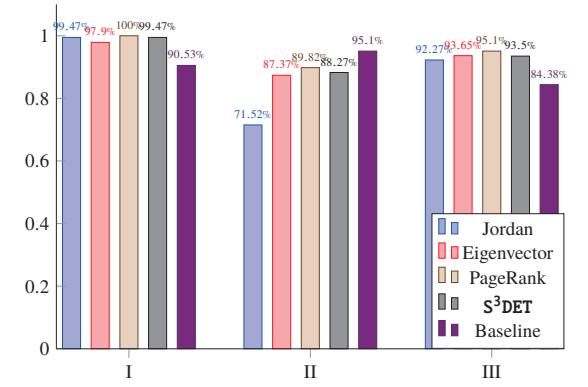
We evaluate the quality of generated constraints in terms of accuracy, precision, and false alarm rate. **S^{3}DET** takes the average score

TABLE I: Default Parameter Setup

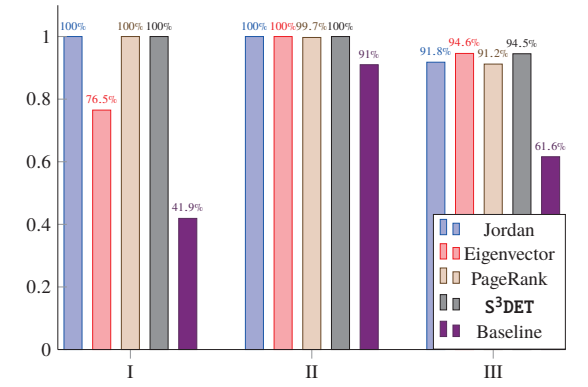
Parameter	Source	Value
$\alpha_{min}, \alpha_{max}$	Algorithm 2	1, 3
Damping facter ζ	PageRank	0.6
Threshold tolerance tol	Graph similarity test	0.9

TABLE II: Statistics of ADC Designs

Design	ADC Architecture	#Valid Pairs	#Nodes	#Edges
I	CT ΔΣ	190	1300	4158
II	SAR	776	2924	7427
III	CT ΔΣ SAR Hybrid	1229	4618	11674



(a) Accuracy



(b) Precision

(c) False alarm rate

Fig. 7: **S³DET** results with different graph centers and baseline.

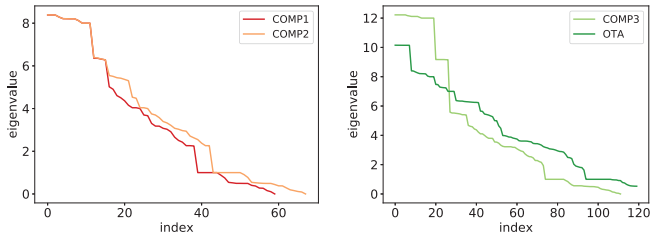
from the three proposed graph centrality definitions for graph similarity. We implement a baseline by replacing subgraph extraction and graph similarity with subcircuit name matching. Figure 7 compares the results of **S³DET** with separate graph centrality measurements and the baseline. Accuracy measures the percentage of correctly detected and filtered symmetry constraints out of valid subcircuit pairs. Our proposed method has higher accuracy compared with the baseline except in Design II. This is mostly because the extensive use of custom designed digital cells, such as inverters, have similar topology and create false alarms with similarity measures. Precision measures the percentage of correctly detected symmetry constraints out of the total proposed symmetry. Lower false alarm rate is better since this suggests that the algorithm is more efficient in reducing unnecessary constraints. Our method consistently outperforms the baseline by a large margin in terms of precision and false alarm rate. The results demonstrate that **S³DET** is capable of detecting critical system symmetry constraints with a low false alarm.

We observe that graph centrality affects the constraint quality. Jordan center has a low accuracy for Design II and eigenvector centrality has a low precision for Design I. Jordan centers have ambiguity due to ties in eccentricity, causing randomness in center selection and graph extraction. Eigenvector centrality tends to select "exterior" pin nodes with high degrees, resulting in large extracted subgraphs. The performance of PageRank centrality is better in consistency between the designs. This might be due to that the selected PageRank centers have more relevance to the graph *Laplacian*, which is also the core of our graph spectral analysis based graph similarity metric.

Table III shows the runtime results. To demonstrate the scalability of our method, we implement a graph edit distance (GED) based similarity metric. GED based similarity metric fails to complete for any design due to limited memory after 8 hours.

TABLE III: Runtime Results

Design	Jordan	Eigenvector	PageRank	GED
I	20.0s	9.5s	10.7s	-
II	8m37.7s	4m13.6s	10m29.0s	-
III	13m52.8s	8m34.2s	13m13.9s	-



(a) (b)

Fig. 8: Eigenvalue distribution of circuit graphs. (a) Similar Comparator circuits. (b) OTA and Comparator circuit.

Figure 8 demonstrates the effectiveness of our graph representation. The graph *Laplacian* eigenvalue distribution of different circuits are presented. We observe that circuits of similar topology have similar eigenvalue distributions in (a), while different topologies have large discrepancies in (b). The proposed similarity metric corresponds with our intuition, with (a) having a high score of 0.99 and (b) 0.02.

To gain further insight into the proposed similarity metric, we compare our similarity metric with GED. Figure 9 plots the similarity score and the normalized GED over 50 different simulations on the same circuit graph. For each individual simulation, we continuously make random graph edits to the same circuit graph. GEDs are normalized by the total number of edges in the graph.

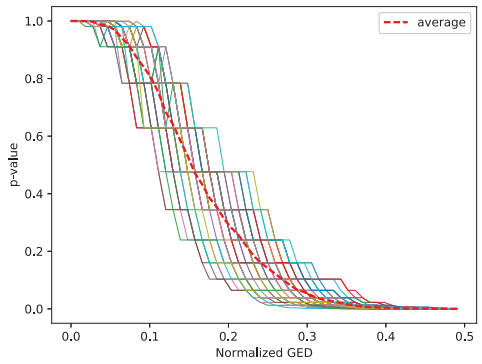


Fig. 9: Similarity score and GED on COMP1

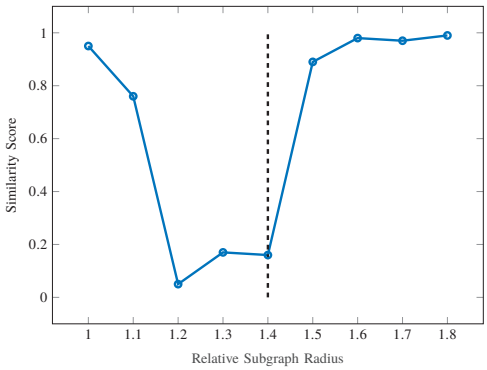


Fig. 10: Similarity score of two unmatched loop filters on different feedback paths. Dashed line is the proposed subgraph radius.

Figure 10 demonstrates the importance of selecting the extracted subgraph size. We select the two loop filters on different feedback paths of the CT-ΔΣ-ADC in Fig. 6 as the subcircuit candidate. The two loop filters have similar circuit topology but do not need to be matched. Thus having a high similarity score indicates false alarm. We extract subgraphs for the two candidates using Jordan centers and plot the similarity score with different subgraph radius. The relative subgraph radius is normalized with the original subcircuit radius. High similarity scores are observed with both small and large extracted subgraphs indicating unnecessary constraint detected. The dashed line is the subgraph radius of **S^{3}DET** in consideration of the distance of the two subcircuit Jordan center. Our proposed subgraph extraction leveraging graph centrality has a low similarity score and successfully filters this unnecessary constraint.

V. Conclusion

In this paper, we present **S^{3}DET**, a novel method of detecting system symmetry constraints for analog circuits. **S^{3}DET** extracts neighboring circuit topologies of subcircuits based on graph centrality and measures graph similarity with spectral graph analysis. Experimental results demonstrate the effectiveness of our proposed method across different ADC designs. Compared with a name matching baseline, **S^{3}DET** achieves high accuracy of around 88.3%, while reducing the false alarm rate by more than 10 times to 1.1%.

Acknowledgement

This work is supported in part by the NSF under Grant No. 1704758, and the DARPA ERI IDEA program.

References

[1] B. Razavi, *Design of Analog CMOS Integrated Circuits*, 1st ed. New York, NY, USA: McGraw-Hill, Inc., 2001.

[2] M. P. Lin, Y. Chang, and C. Hung, "Recent research development and new challenges in analog layout synthesis," in *ASPDAC*, Jan 2016, pp. 617–622.

[3] P. Lin, H. Yu, T. Tsai, and S. Lin, "A matching-based placement and routing system for analog design," in *2007 International Symposium on VLSI Design, Automation and Test (VLSI-DAT)*, April 2007, pp. 1–4.

[4] R. Martins, N. Lourenco, and N. Horta, "Laygen ii-automatic layout generation of analog integrated circuits," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 32, no. 11, pp. 1641–1654, Nov 2013.

[5] J. Song, K. Ragab, X. Tang, and N. Sun, "A 10-b 800-ms/s time-interleaved sar adc with fast variance-based timing-skew calibration," *IEEE Journal of Solid-State Circuits*, vol. 52, no. 10, pp. 2563–2575, Oct 2017.

[6] E. Charbon, E. Malavasi, and A. Sangiovanni-Vincentelli, "Generalized constraint generation for analog circuit design," in *ICCAD*, Nov 1993, pp. 408–414.

[7] Qinsheng Hao, Sheqin Dong, Song Chen, Xianlong Hong, Yi Su, and Zhiyi Qu, "Constraints generation for analog circuits layout," in *2004 International Conference on Communications, Circuits and Systems (IEEE Cat. No.04EX914)*, vol. 2, June 2004, pp. 1339–1343 Vol.2.

[8] Zhe Zhou, Sheqin Dong, Xianlong Hong, Qingsheng Hao, and Song Chen, "Analog constraints extraction based on the signal flow analysis," in *2005 6th International Conference on ASIC*, vol. 2, Oct 2005, pp. 825–828.

[9] P. Wu, M. P. Lin, and T. Ho, "Analog layout synthesis with knowledge mining," in *European Conference on Circuit Theory and Design (ECCTD)*, Aug 2015, pp. 1–4.

[10] 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, Feb 2011.

[11] J. Scheible and J. Lienig, "Automation of analog ic layout: Challenges and solutions," in *Proceedings of the 2015 Symposium on International Symposium on Physical Design*, ser. ISPD '15. New York, NY, USA: ACM, 2015, pp. 33–40.

[12] A. Sanfeliu and K. Fu, "A distance measure between attributed relational graphs for pattern recognition," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-13, no. 3, pp. 353–362, May 1983.

[13] M. R. Garey and D. S. Johnson, *Computers and Intractability; A Guide to the Theory of NP-Completeness*. New York, NY, USA: W. H. Freeman & Co., 1990.

[14] T. B Arnold and J. W Emerson, "Nonparametric goodness-of-fit tests for discrete null distributions," *R Journal*, vol. 3, 12 2011.

[15] R. Gera, L. Alonso, B. Crawford, J. House, J. A. Mendez-Bermudez, T. Knuth, and R. Miller, "Identifying network structure similarity using spectral graph theory," *Applied Network Science*, vol. 3, no. 1, p. 2, Jan 2018. [Online]. Available: https://doi.org/10.1007/s41109-017-0042-3

[16] L. Hagen and A. B. Kahng, "New spectral methods for ratio cut partitioning and clustering," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 11, no. 9, pp. 1074–1085, Sep. 1992.

[17] I. McCulloh, H. Armstrong, and A. Johnson, *Social Network Analysis with Applications*, 1st ed. Wiley Publishing, 2013.

[18] S. Brin and L. Page, "The anatomy of a large-scale hypertextual web search engine," *Computer Networks*, vol. 30, pp. 107–117, 1998. [Online]. Available: http://www-db.stanford.edu/~backrub/google.html

[19] A. A. Hagberg, D. A. Schult, and P. J. Swart, "Exploring network structure, dynamics, and function using networkx," in *Proceedings of the 7th Python in Science Conference*, G. Varoquaux, T. Vaught, and J. Millman, Eds., Pasadena, CA USA, 2008, pp. 11 – 15.

[20] E. Jones, T. Oliphant, P. Peterson *et al.*, "SciPy: Open source scientific tools for Python," 2001–. [Online]. Available: http://www.scipy.org/