

# IP-Matcher: An Efficient One-to-Many Matching Framework for Analog Circuit Design and Reusing

Shixin Chen<sup>1,†</sup>, Peng Xu<sup>1,†</sup>, Yapeng Li<sup>2</sup>, Tinghuan Chen<sup>2</sup>, Bei Yu<sup>1</sup>

<sup>1</sup>Department of Computer Science & Engineering, The Chinese University of Hong Kong

<sup>2</sup>School of Science and Engineering, The Chinese University of Hong Kong (Shenzhen)

**Abstract**—The design efficiency of analog circuits is generally lower than that of digital circuits, presenting a significant bottleneck in the current integrated circuit industry. One promising method to accelerate design processes is the modular design philosophy adapted from digital methodologies. However, there is a lack of an efficient framework for reusing mature analog circuit topologies and the corresponding layout designs. To achieve a rapid design iteration while utilizing specialized expertise in design, we propose IP-Matcher, an efficient IP-based analog circuit matching and reusing framework. The framework consists of three components: Analog Graph Converter, Analog IP Manager, and IP-based Matcher, which collaborate to enhance both matching accuracy and speed, thereby improving analog IP reusability. We leverage the unique characteristics of analog circuits to significantly prune the matching space, overcoming the limitations of traditional circuit matching strategies. Experimental results show that our work not only outperforms the state-of-the-art method by 32% in accuracy but also achieves a 16× speedup.

**Index Terms**—Analog Circuit, Design Methodology, IP Reuse, Graph Matching

## I. Introduction

Analog circuits play a crucial role in mixed-signal systems, with applications spanning sensor interfaces, power management, and high-speed communications [1]. Their design relies heavily on manual expertise due to sensitivity to layout parasitics and process variations, resulting in low automation [2]. For analog layout design, the engineers need to carefully place the devices considering symmetry constraints, the parasitic effects of the layout, and the manufacturing technology. Such a process is challenging and error-prone and the corresponding expert knowledge in analog circuit design is difficult to describe by clear rules. Thus, the level of automation in analog circuit design has remained insufficient [2].

To address this issue, some studies have proposed applying the design methodology of digital circuits to analog circuits to improve design efficiency. One of the promising directions is to increase the reusability of mature and well-tested analog circuits. Reusing such intellectual properties (IPs) can accelerate the design process of analog circuits and maintain the valuable expert knowledge embedded in the schematic design and parameter decision [3]. Another practical application is to reuse the layout design of corresponding analog circuits to generate the layout of a new analog circuit design [4]. All these design methodologies depend on precisely identifying the desired IPs from existing analog circuits. However, there does not exist an efficient strategy to match the most proper analog IPs and reuse them precisely, especially for many legacy designs without human notations.

<sup>†</sup>Equal contributors.

This work is partially supported by The Research Grants Council of Hong Kong SAR (No. CUHK14201624).

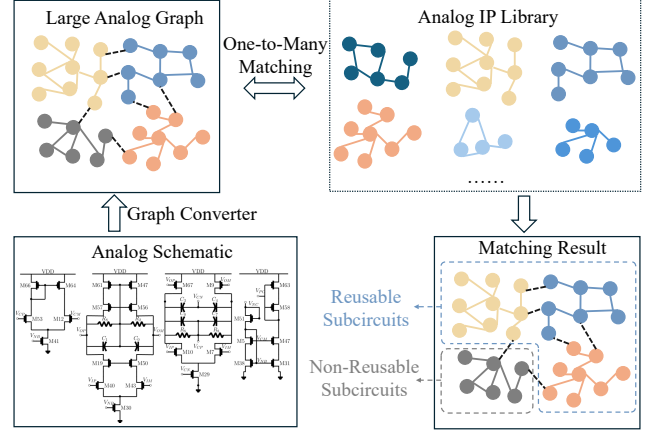


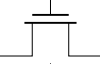
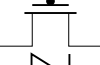
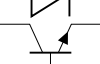
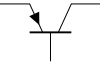

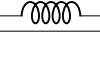
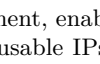
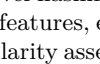
Fig. 1 Given one large query analog graph, the solution is to select a series of smaller analog IPs to cover the query circuit. Then, the expert knowledge embedded in the matched IPs can be reused in the large query circuit to improve design efficiency.

The matching process of two analog circuits aims to find corresponding transistor pairs, so as to share key parameters and reuse layout designs. Analog circuit topology can be modeled as a graph, and graph theory algorithms can be used to identify reusable device parts [5]. Notably, as the number of devices increases, a single comparison between two graphs with hundreds of nodes can take several hours to compute the match [6]. Although recent approaches [7]–[12] use graph neural networks with subgraph isomorphism, they cannot precisely output the matching pairs between two graphs. Some research [13]–[16] leverage circuit characteristics to accelerate matching, but these methods mainly support one-to-one matching between the query and target circuits. There is a lack of methods to handle large-scale comparisons between query circuits and a large number of IPs.

To solve the challenges mentioned above, we first propose an efficient matching framework for analog design from reusing patterns in legacy designs. Instead of one-to-one matching methods at a large scale, the analog circuit graphs are converted into several IP-based comparisons as shown in Fig. 1. The framework is constructed based on modular design philosophy and it explores a series of optimization strategies to improve the matching accuracy and efficiency. Our matching algorithm is customized to fit the analog circuit graph, which greatly prunes the search space to avoid waste during the matching process. To conclude, our contributions can be summarized as follows:

- Propose a customized analog graph converter to extract

TABLE I Devices of analog and basic information

Device	Symbol	#Pins	Pin types	Hash ID
NMOS		3	nd, ng, ns, nb	3
PMOS		4	pd, pg, ps, pb	5
Diode		2	n+, n-	7
NPN		3	nb, nc, ne	11
PNP		3	nb, nc, ne	13
Resistor		2	n+, n-	17
Capacitor		2	n+, n-	19
Inductor		2	n+, n-	23

key circuit characteristics and standardize data for efficient management, enabling historical design exploration to generate reusable IPs.

- Introduce a novel hashing strategy to label nodes with local and global features, enhancing IP library management and query similarity assessment in one-to-many matching.
- Develop an efficient matching flow with IP filters to prune unnecessary searches, accelerating the process and recommending high-reusability IPs for specific circuits.
- Validate our method through industrial designs to prove that our method outperforms baselines in quality and efficiency for analog design and reusing.

## II. Problem Formulation

A graph  $\mathcal{G} = (\mathcal{V}, \mathcal{E})$  consists of nodes  $\mathcal{V}$  and edges  $\mathcal{E}$ . A subgraph  $\mathcal{G}_s = (\mathcal{V}_s, \mathcal{E}_s)$  satisfies  $\mathcal{V}_s \subseteq \mathcal{V}$ ,  $\mathcal{E}_s \subseteq \mathcal{E}$ , and preserves all edges between nodes in  $\mathcal{V}_s$ .

**Definition 1 (Analog Circuit Graph).** An analog circuit graph  $\mathcal{G} = (\mathcal{V}, \mathcal{E})$  contains device nodes  $\mathcal{V}_d$  (e.g., transistors, resistors) and net nodes  $\mathcal{V}_n$ , with edges  $\mathcal{E}$  connecting  $\mathcal{V}_d$  and  $\mathcal{V}_n$ .

**Definition 2 (Analog IP Library).** An IP library  $\mathcal{G}_{\text{LIB}} = \{\mathcal{G}_0, \dots, \mathcal{G}_N\}$  stores pre-tested modular circuits for design reuse.

It is noted that previous work focuses on one-to-one matching, and our framework focuses on one-to-many matching. We give the definition to distinguish two scenarios.

**Definition 3 (One-to-One Matching).** Find the maximal common subgraph (MCS)  $MCS(\mathcal{G}_q, \mathcal{G}_t)$  between query  $\mathcal{G}_q$  and target  $\mathcal{G}_t$ .

**Definition 4 (One-to-Many Matching).** Cover a large query graph  $\mathcal{G}_q$  with multiple pre-tested IPs  $\mathcal{G}_{\text{IP}}$ .

**Problem 1 (IP-Matching Objective).** Develop a framework to efficiently manage analog IPs and rapidly match them against large query circuits.

## III. Overall Framework

To efficiently reuse analog IP through one-to-many matching between analog graphs, we propose an IP-based matching framework, as illustrated in Fig. 2. The main framework is

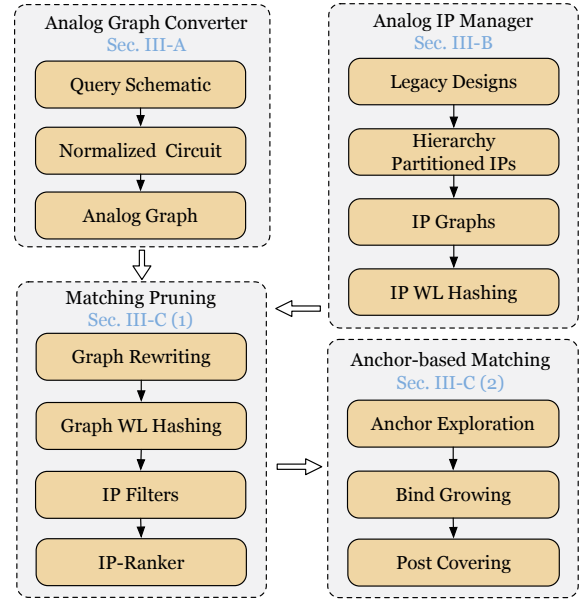


Fig. 2 The overall flow of the proposed one-to-many matching strategy of the analog graph.

divided into three parts: the Analog Graph Converter, the IP Library Manager, and the IP-based Matcher (Matching Pruning and Anchor-based Matching). Specifically, the Analog Graph Converter extracts raw analog circuit characteristics and converts circuits into a standard graph format. The Analog IP Manager employs a customized hashing method to index analog IPs and efficiently determine matching order, enabling simultaneous acquisition of local and global features of circuits. The IP-based Matcher is a heuristic matching flow that prunes the process via IP filters to save matching resources. Details of each component are elaborated in subsequent sections.

### A. Customized Analog Graph Converter

Fig. 3 shows an example of converting an analog schematic into an analog graph, where each device and net has a corresponding node in the analog graph. Analog circuits consist of devices (PMOS, NMOS, PNP, NPN, Resistor, Capacitor, Diode, Inductor), listed in TABLE I. The device type is the primary attribute, therefore nodes are classified by device type. Furthermore, node degree (connected edges) varies by device type: PMOS or NMOS typically have 4 pins (S, D, G, B), power and ground nodes (GND and VCC) have high degrees (connecting most devices), while common nets have low degrees. Considering these characteristics of the analog circuits, our analog graph converter takes the netlist of the schematic as input and performs the following stages to extract the normalized analog graph.

**Flatten Hierarchical Circuit Structure.** Analog schematics adopt a hierarchical structure, where devices and nets form functional subcircuits (e.g., inverters, amplifiers) that may nest to form a tree. Leaf nodes represent non-flattenable basic devices. Given the unsuitability of a deep hierarchy for graph processing, the analog graph converter first flattens the topology, maintaining internal-external pin connections for consistency. It also stores hierarchical information, which

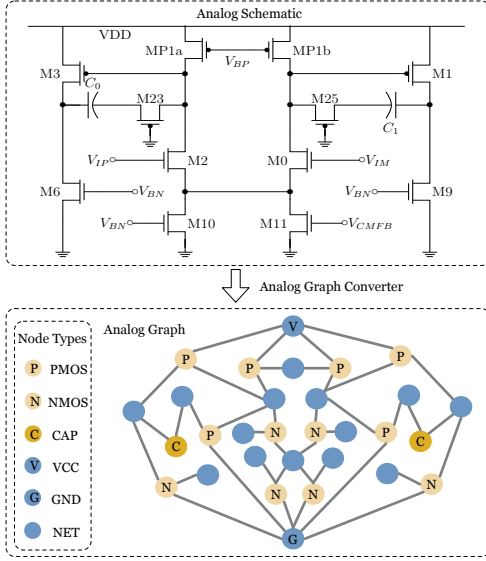


Fig. 3 The schematic of an analog design and the corresponding analog graph. The net nodes set  $\mathcal{V}_n$  are in blue and the device nodes set  $\mathcal{V}_d$  can be classified into several types, i.e., PMOS, NMOS, CAP.

forms the basis for the parent-child filter in Section III-B to enable hierarchical-aware matching.

**Normalize Device Pins to Standard Form.** As listed in TABLE I, the typical device pins define the standard format for devices. However, some library technologies may include extraneous pins that do not affect device functionality but introduce extra connections, altering the topology of the analog graph. To address this, we identify and remove these pins to normalize each device to the standard format.

**Preprocess Parallel and Serial Topologies.** In analog circuits, identical devices often connect to the same ports to form parallel structures, which can be aggregated into a single representative structure to simplify the analog graph. This operation allows matching results to cover one-to-many device combinations. The same principle applies to serial structures, both of which streamline the graph topology by reducing redundancy.

## B. Analog IP Library Manager

The analog IP library requires systematic processing to enhance matching efficiency, involving two core stages: IP preprocessing and IP indexing. As illustrated in Fig. 2, historical analog designs from experienced engineers embed domain-specific optimizations for functional modules, along with associated layout constraints—such as parasitic effect symmetry, parameter coupling, and shielding design. To fully exploit these engineering insights, we need to detect the promising analog IPs that can be reused and index them to construct the analog IP library.

**Analog IP Detection.** Analog designs contain numerous subcircuits with specific functions, some of which are topologically and parametrically identical. These subcircuits, with the schematic and corresponding layout, are treated as reusable

## Algorithm 1 WL\_Node\_Hashing( $\mathcal{G}$ , depth)

---

Input:  $\mathcal{G}$  (graph),  $V_{\mathcal{G}}$  (nodes), *depth* (max WL depth), *hash\_T* (node type hash mapping)  
Output:  $\mathcal{H}$  (node hash codes per depth)

```

1:  $\mathcal{H} \leftarrow \{\}$ ; ▷ Initialize hash storage
2: for  $v \in V_{\mathcal{G}}$  do
3:    $\mathcal{H}[v][0] \leftarrow \text{hash\_T}[v.\text{type}] \times \prod_{u \in \text{neigh}(v)} \text{hash\_T}[u.\text{type}]$ 
4: end for
5: for cur_depth  $\leftarrow 1$  to depth do
6:   changed  $\leftarrow \text{False}$ 
7:   for  $v \in V_{\mathcal{G}}$  do
8:      $\mathcal{H}[v][\text{cur\_depth}] \leftarrow \prod_{u \in \text{Neigh}(v)} \mathcal{H}[u][\text{cur\_depth} - 1]$ 
9:     if  $\mathcal{H}[v][\text{cur\_depth}] \neq \mathcal{H}[v][\text{cur\_depth} - 1]$  then
10:      changed  $\leftarrow \text{True}$ 
11:     end if
12:   end for
13:   if changed == False then return  $\mathcal{H}$ 
14:   end if
15: end for
16: return  $\mathcal{H}$ 

```

---

analog IPs for new designs. By partitioning schematics according to the hierarchical tree structure, we ensure that extracted IPs are diverse, and finer-grained IPs significantly enhance matching efficiency. In the schematic of analog design, subcircuits at various levels exhibit a parent-child relationship (i.e., a subcircuit may contain its nested subcircuits). During partitioning, the ancestral nodes of each subcircuit are recorded to facilitate efficient matching by leveraging structural relationships. After extracting analog subcircuits, they are converted into graphs using the analog graph parser detailed in Section III-A. This stage normalizes circuit topologies, device types, and pin configurations to standardized formats, ensuring seamless compatibility and reusability within large-scale query graphs.

**Efficient Hash-based IP Indexing.** The IP library may contain a large number of entries, each of which must undergo a matching process with the query analog graph. Directly applying the matching algorithm to all IPs is infeasible due to its NP-hard property, which makes it inherently time-consuming to find an optimal solution. Therefore, an effective strategy is to prune unnecessary matchings as early as possible through lightweight preprocessing.

Among the pruning strategies, indexing all IPs with lightweight computations will be useful and efficient. The Weisfeiler-Lehman (WL) test [17] is a key graph theory method for isomorphism detection and feature learning. It iteratively aggregates node features to capture local and global graph properties, enabling efficient non-isomorphism determination with computational advantages, especially for large graphs. This process generates unique WL hashing codes, effectively capturing graph characteristics for comparison and classification. As shown in Fig. 4, an analog graph can use the customized WL hashing algorithm to label all nodes of the graph with various WL depths. Algorithm 1 shows the customized WL hashing algorithm for the analog graph.

Lines 1-4 initialize the hashing, where  $\text{neigh}(v)$  will obtain all neighborhoods of node  $v$ . For the first iteration, WL codes are derived from node and their neighbor device types, with initial hashes stored in *hash\_T* (prime numbers represent device types in TABLE I). Hash codes are products of primes



---

**Algorithm 2** IP\_Ranker( $\mathcal{G}_q, \mathcal{IP}, \text{depth}$ )

---

Input:  $\mathcal{G}_q$  (query graph),  $\mathcal{IP}$  (IP graphs), depth (WL depth)  
Output: *scores* (similarity scores for each IP)  
1:  $\mathcal{H}_q \leftarrow \text{WL\_Node\_Hashing}(\mathcal{G}_q, \text{depth})$   $\triangleright$  Algorithm 1  
2: *scores*  $\leftarrow \{\}$   $\triangleright$  Store the scores for each IP  
3: for  $\mathcal{G}_{IP} \in \mathcal{IP}$  do  
4:    $\mathcal{H}_{IP} \leftarrow \text{WL\_Node\_Hashing}(\mathcal{G}_{IP}, \text{depth})$   
5:   *common*  $\leftarrow 0$   
6:   for  $d \leftarrow 0$  to depth do  
7:      $\mathcal{H}_{IP_d} \leftarrow \{\mathcal{H}_{IP}[v][d] \mid v \in \mathcal{G}_{IP}\}$   
8:     *common*  $\leftarrow \text{common} + |\{v_q \in \mathcal{G}_q \mid \mathcal{H}_q[v_q][d] \in \mathcal{H}_{IP_d}\}|$   
9:   end for  
10:   *scores*[ $\mathcal{G}_{IP}$ ]  $\leftarrow \text{common} / (|\mathcal{G}_q| \times \text{depth})$   
11: end for  
12: return *scores*

---



---

**Algorithm 3** Binding\_Growing( $\mathcal{G}_q, \mathcal{G}_{ip}$ )

---

Input:  $\mathcal{G}_q$  (query graph),  $\mathcal{G}_{ip}$  (IP graph)  
Output:  $\mathcal{M}$  (maximal common subgraph node pairs)  
1:  $\mathcal{M} \leftarrow \emptyset$ ;  $\triangleright$  Store matched node pairs  
2:  $\mathcal{A} \leftarrow \text{FindAnchors}(\mathcal{G}_q, \mathcal{G}_{ip})$ ;  $\triangleright$  Get anchor pairs via WL hashing  
3: for  $(q_a, ip_a) \in \mathcal{A}$  do  $\triangleright$  Process each anchor pair  
4:    $\mathcal{M} \leftarrow \mathcal{M} \cup \{(q_a, ip_a)\}$ ;  $\triangleright$  Add anchor to matches  
5:    $Q_q \leftarrow \text{Neigh}(q_a, \mathcal{G}_q)$ ;  $Q_{ip} \leftarrow \text{Neigh}(ip_a, \mathcal{G}_{ip})$ ;  
6:   while  $Q_q \neq \emptyset$  and  $Q_{ip} \neq \emptyset$  do  
7:      $q \leftarrow Q_q.\text{pop}()$ ;  $ip \leftarrow Q_{ip}.\text{pop}()$ ;  $\triangleright$  Pop nodes to match  
8:     if  $\text{DeviceType}(q) = \text{DeviceType}(ip)$  and  
9:      $\text{NetDegree}(q) = \text{NetDegree}(ip)$  then  
10:        $\mathcal{M} \leftarrow \mathcal{M} \cup \{(q, ip)\}$ ;  $\triangleright$  Add valid match  
11:        $Q_q \leftarrow Q_q \cup \text{Neigh}(q, \mathcal{G}_q)$ ;  $Q_{ip} \leftarrow Q_{ip} \cup \text{Neigh}(ip, \mathcal{G}_{ip})$ ;  
12:     end if  
13:   end while  
14: end for  
15: return  $\mathcal{M}$ ;

---

is because any incorrect node binding during the matching process will lead to a sub-optimal matching solution, and the back-tracing process will incur unnecessary overhead.

We can leverage the customized WL-hashing methods for analog circuits to explore the best initial binding nodes, i.e., the Anchors. According to Algorithm 1, if two nodes have the same hash codes in the  $d$ th iteration, they will have  $d$  hops of identical connections and devices in their vicinity. The deeper the iteration of the hash codes of two nodes, the more similar the structures around the nodes are. Therefore, based on the WL-hashing methods, we can easily identify the optimal anchors between two graphs.

**Bind Growing.** Matching results are extended from the anchors in both the query graph and the IP graph. The solution of the matching takes the form of a series of node pairs, where the two nodes are from the query graph and the IP graph, respectively.

First, we bind the anchors themselves. Then, based on the device types and the connectivity of neighboring nodes, we bind more node pairs. It should be noted that this method is efficient and reduces the probability of incorrect binding. The advantages are derived from the pruning strategies already employed, and the neighboring structures around the anchors are guaranteed to be identical due to the WL-hashing method.

Moreover, there may be multiple anchors in the large query graph. The presence of multiple anchors implies that the IP can be used in multiple locations. In this case, the binding growth process will start from these anchors simultaneously to enhance efficiency. Previous one-to-one matching cannot

handle such a complex situation. The detailed algorithm is shown in Algorithm 3.

**Post Covering.** After the binding growth of the matching pairs, a post-covering stage is carried out to maximize the solution coverage. IPs that are largely covered by the large query graph will be prioritized to cover the query graph. During this process, overlapping parts are avoided when two IPs have the same structural mapping. The post-covering process continues until the large query graph is completely covered by a series of IPs or all the matched IPs are exhausted (i.e., no more IPs can be used to cover the graph). Once the covering is finished, we can obtain the coverage ratio of the query graph. This ratio serves as a metric indicating the proportion of subcircuits that can be reused from the IP library. The larger the cover ratio is, the better the IP reuse strategy is.

#### IV. Experiment & Analysis

##### A. Implementation & Settings

We conduct a series of experiments to demonstrate the validity of the proposed strategies. To ensure generality, we carefully constructed the benchmarks by selecting analog circuits from industrial designs, with device numbers ranging from 50 to 300, as shown in TABLE II. Using hierarchical partitioning to construct IP library, we detected IPs and generated 1745 distinct IPs from in-house analog design, as shown in Fig. 7. We choose depth= 3 in Algorithm 1 for WL hashing indexing and set the similarity threshold as 0.9 for the Similarity Filter.

We implement all our strategies using Python 3.12, the analog graph circuits will be stored in JSON format during IP library construction. The graph operations are based on the Python package NetworkX 3.2.1 [20]. All experiments are conducted on a Linux server with an Intel Xeon CPU (E5-2630 v2@2.60GH) and 256 GB RAM.

##### B. Baselines

To evaluate the efficiency and accuracy of our methods against traditional one-to-one matching approaches, we select several representative algorithms as baselines. The VF2-based algorithm [18], [21] utilizes a recursive backtracking framework with state-space pruning to explore node mappings in graph isomorphism detection. VF3 [19] improves performance through more intelligent candidate node-pair selection, optimized state-space search heuristics, and refined node-priority mechanisms compared to VF2. To isolate the contribution of our proposed analog graph converter and pruning stages, we first convert circuits into graphs without these components and apply a basic WL method [17] termed Naive-WL for comparison. Given the growing significance of learning-based approaches in graph matching, we include the state-of-the-art SMART [6] as a baseline, which represents the current

TABLE II Query benchmark circuits information.

No.	Query Circuit	#PMOS	#NMOS	#C	#R	#NET	#DEV	#EDGE
1	signbit	27	27	0	1	32	55	199
2	dac_r2r	34	34	0	103	131	68	461
3	vbg_b0	58	58	1	187	258	117	706
4	resdiv	63	63	0	20	72	146	503
5	pll_post	76	76	0	0	76	152	497
6	spare_cell_33	86	84	0	2	131	172	548
7	spare_cell_18	107	106	0	2	111	215	682
8	core_pwm	83	138	10	59	185	290	864
9	ref_system	113	112	0	65	176	290	844
10	levelshift	148	119	0	30	159	297	861
11	testmode_entry	143	132	0	22	157	297	919

TABLE III Our framework outperform baselines in both cover ratio and runtime (higher cover ratio has better reusability).

Case	#DEV	VF3-based [18], [19]		Naive-WL [17]		SMART [6]		Ours	
		Cover ratio (%)	Time (s)	Cover ratio (%)	Time (s)	Cover ratio (%)	Time (s)	Cover ratio (%)	Time (s)
Case 1	55	0.945	33.9	0.545	194.4	0.885	8.78	0.981	0.769
Case 2	68	0.910	488.2	0.169	405.1	0.792	97.64	0.941	16.03
Case 3	117	0.603	1815.5	0.089	771.4	0.569	323.61	0.617	75.58
Case 4	146	0.521	712.5	0.336	359.5	0.452	142.82	0.782	59.03
Case 5	165	0.842	1166.4	0.203	349.8	0.753	233.58	0.921	107.29
Case 6	215	0.616	3600	0.279	472.6	0.579	3600	0.640	153.78
Case 7	172	0.670	3600	0.288	596.2	0.592	3600	0.679	204.30
Case 8	290	0.128	2217	0.148	617.6	0.473	420.35	0.617	158.7
Case 9	290	0.195	3600	0.249	738.6	0.286	3600	0.895	189.74
Case 10	297	0.615	3600	0.258	584.7	0.337	3600	0.823	270.46
Case 11	297	0.574	3600	0.382	678.4	0.482	3600	0.802	260.27
Average	-	0.60	2221.2	0.2678	524.38	0.563	1747.89	0.79 $\uparrow$	136.0 $\downarrow$
Ratio	-	0.759	16.33	0.339	3.86	0.713	12.85	1.0 $\uparrow$	1.0 $\downarrow$

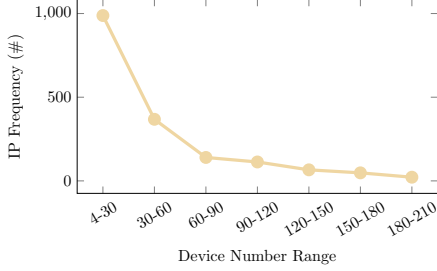


Fig. 7 The device number range of 1745 IPs in the IP library from 26 system-level analog designs.

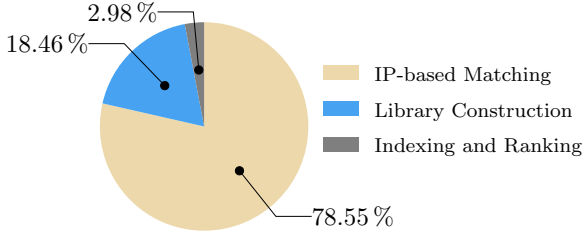


Fig. 8 The runtime distribution of the entire framework.

frontier of data-driven matching techniques. Since all baseline methods are inherently designed for one-to-one matching, we integrate them as iterative subroutines within our one-to-many framework, treating each as a reusable engine for repeated graph comparisons.

It is noted that in the practical analog design process, a long iteration time is unacceptable, so experiments will be terminated if the total matching time exceeds 3600 seconds. The coverage ratio recorded at that point will be used for evaluation, and a higher ratio indicates that more well-tested layout IPs can be used to reduce engineers' labor.

### C. Accuracy and Efficiency

TABLE III presents the one-to-many matching results of the benchmarks using different strategies. Experimental results show that our work not only outperforms the state-of-the-art method by 32% in accuracy but also achieves a 16 $\times$  speedup. Each query circuit was matched with every IP in the library, resulting in a total of 1745 matchings. For small-sized query circuits, all methods achieved a good cover ratio, with our approach attaining the highest ratio in the least time. As circuit size increased to medium and large, the baseline method could not complete within an acceptable time frame. Moreover, their cover ratios dropped signifi-

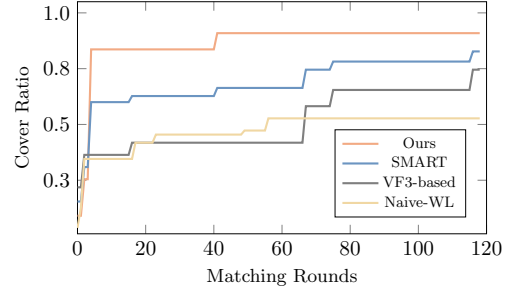


Fig. 9 The cover ratios increase with more matching rounds from different methods.

cantly, indicating these methods failed to identify reusable IPs effectively. Overall, the experiments demonstrate that our proposed method is more accurate and efficient for practical one-to-many matching tasks.

To validate the effectiveness of the proposed IP ranking strategy, we keep all other components identical but replace the hash-code-based indexing order with a random selection order. As shown in TABLE III, without the ranking method, the framework requires significantly more time to find reusable IPs while achieving a lower cover ratio. Fig. 8 shows the average runtime distribution of the entire benchmark, demonstrating that IP indexing and ranking introduce only acceptable overhead while significantly improving matching efficiency.

To illustrate the trend during the matching process, we analyze results using Case 1 as an example. As depicted in Fig. 9 for different methods, our approach eliminates most unnecessary matching steps and achieves a high cover ratio in the early matching stages, directly improving design efficiency. When a limited time is imposed for design matching, our framework can quickly identify more valid coverage solutions, increasing the likelihood of optimal design reuse.

### D. Case Study: IP-Matcher for Layout Design Reuse

Given a new analog design with a layout in the same technology, we can use our framework to identify the well-tested layout IPs in the library. To verify the efficiency of our framework for layout reuse in practical design processes, we use a layout of an analog design with 68 devices (Case 2). After the matching steps in Section III-C based on the IP library constructed in Section III-B, we load the initial layout of the design. As illustrated in Fig. 10, the large components in the blue box are matched reusable IPs from the IP library, and the devices in the gray box are not matched and are loaded

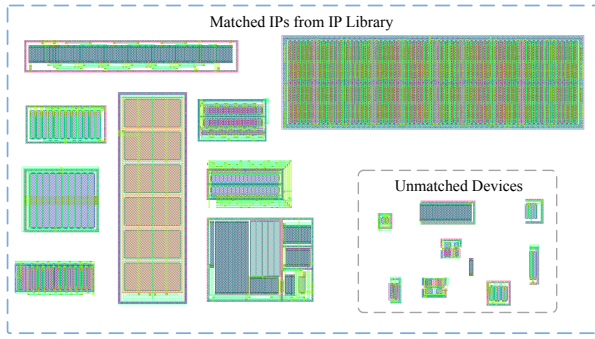


Fig. 10 The initial layout design based on the matching results (Case 2 in TABLE III). In this example, the matched devices can cover 94.1% of all the devices of the schematic.

from the technology library. By reusing the placement patterns and specific optimizations of the matched IPs, designers can leverage the validated expert knowledge (e.g., operations to decrease parasitic effects) in this IPs and thus improve the design efficiency. After further placement and routing stages, the layout design is completed, thereby reducing significant human labor.

## V. Conclusion

This paper presents an efficient framework for one-to-many matching of analog IPs in large-scale circuit design, addressing challenges posed by structural complexity and high computational cost. We introduce three key innovations: a customized graph converter that standardizes analog circuit representations, a specialized hashing method to enhance IP library management, and an optimized IP-based matcher to improve matching efficiency and accuracy. Industrial experiments demonstrate that our framework outperforms the previous baselines by reducing computational overhead and leveraging expert knowledge embedded in legacy designs. Our framework significantly advances IP-driven reuse and improves the modular methodology in analog circuit design.

## References

- [1] J. Scheible, “Optimized is not always optimal-the dilemma of analog design automation,” in *ACM International Symposium on Physical Design (ISPD)*, 2022, pp. 151–158.
- [2] P. Xu, J. Li, T.-Y. Ho, B. Yu, and K. Zhu, “Performance-driven analog layout automation: Current status and future directions,” in *IEEE/ACM Asia and South Pacific Design Automation Conference (ASPDAC)*. IEEE, 2024, pp. 679–685.
- [3] T. Levi, J. Tomas, N. Lewis, and P. Fouillat, “Ip-based design reuse for analog systems,” in *VLSI Circuits and Systems III*, vol. 6590. SPIE, 2007, pp. 159–167.
- [4] E. Dalbudak, K. Baratli, Y. Fouzar, B. Lakhssassi, K. El Guemhioui, and A. Lakhssassi, “The Use of Agile Methodology for Porting Analog and Mixed-Signal Circuits Between Different Technology Nodes,” in *IEEE International IOT, Electronics and Mechatronics Conference (IEMTRONICS)*. IEEE, 2020, pp. 1–7.
- [5] M. Ohlrich, C. Ebeling, E. Ginting, and L. Sather, “Subgemini: Identifying subcircuits using a fast subgraph isomorphism algorithm,” in *ACM/IEEE Design Automation Conference (DAC)*, 1993, pp. 31–37.
- [6] J. Tu, Y. Li, P. Li, P. Xu, Q. Zhang, S. Wan, Y. Sun, B. Yu, and T. Chen, “SMART: Graph Learning-Boosted Subcircuit Matching for Large-Scale Analog Circuits,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, 2025.
- [7] C. T. Duong, T. D. Hoang, H. Yin, M. Weidlich, Q. V. H. Nguyen, and K. Aberer, “Efficient streaming subgraph isomorphism with graph neural networks,” *Proceedings of the VLDB Endowment*, vol. 14, no. 5, pp. 730–742, 2021.
- [8] T. Bücher, L. Alrahis, G. Paim, S. Bampi, O. Sinanoglu, and H. Amrouch, “AppGNN: Approximation-aware functional reverse engineering using graph neural networks,” in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2022, pp. 1–9.
- [9] L. Alrahis, A. Sengupta, J. Knechtel, S. Patnaik, H. Saleh, B. Mohammad, M. Al-Qutayri, and O. Sinanoglu, “GNN-RE: Graph neural networks for reverse engineering of gate-level netlists,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, vol. 41, no. 8, pp. 2435–2448, 2021.
- [10] Z. Wu, I. Song, and I. Savidis, “Hybrid utilization of subgraph isomorphism and relational graph convolutional networks for analog functional grouping annotation,” in *ACM/IEEE Workshop on Machine Learning CAD (MLCAD)*, 2023.
- [11] Z. He, Z. Wang, C. Bai, H. Yang, and B. Yu, “Graph learning-based arithmetic block identification,” in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2021.
- [12] K. Kunal, T. Dhar, M. Madhusudan, J. Poojary, A. Sharma, W. Xu, S. M. Burns, J. Hu, R. Harjani, and S. S. Sapatnekar, “GANA: Graph convolutional network based automated netlist annotation for analog circuits,” in *IEEE/ACM Proceedings Design, Automation and Test in Europe (DATE)*. IEEE, 2020, pp. 55–60.
- [13] M. Ohlrich, C. Ebeling, E. Ginting, and L. Sather, “SubGemini: Identifying subcircuits using a fast subgraph isomorphism algorithm,” in *ACM/IEEE Design Automation Conference (DAC)*, 1993, pp. 31–37.
- [14] H.-Y. Su, C.-H. Hsu, and Y.-L. Li, “SubHunter: A high-performance and scalable sub-circuit recognition method with prüfer-encoding,” in *IEEE/ACM Proceedings Design, Automation and Test in Europe (DATE)*, 2015.
- [15] M. Su, Y. Xiao, S. Zhang, H. Su, J. Xu, H. He, Z. Zhu, J. Chen, and Y.-W. Chang, “Subgraph matching based reference placement for PCB designs: late breaking results,” in *ACM/IEEE Design Automation Conference (DAC)*, 2022.
- [16] C. McCreesh, P. Prosser, and J. Trimble, “A partitioning algorithm for maximum common subgraph problems,” 2017.
- [17] B. Weisfeiler and A. Leman, “The reduction of a graph to canonical form and the algebra which appears therein,” *nti, Series*, vol. 2, no. 9, pp. 12–16, 1968.
- [18] M. Liu, W. Li, K. Zhu, B. Xu, Y. Lin, L. Shen, X. Tang, N. Sun, and D. Z. Pan, “S3DET: Detecting System Symmetry Constraints for Analog Circuits with Graph Similarity,” in *IEEE/ACM Asia and South Pacific Design Automation Conference (ASPDAC)*, 2020, pp. 193–198.
- [19] “VF3 algorithm,” <https://pypi.org/project/vf3py/>.
- [20] “NetworkX Library,” <https://github.com/networkx/networkx>.
- [21] H. Chen, K. Zhu, M. Liu, X. Tang, N. Sun, and D. Z. Pan, “Toward Silicon-Proven Detailed Routing for Analog and Mixed-Signal Circuits,” in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2020, pp. 1–8.