A Unified Printed Circuit Board Routing Algorithm With Complicated Constraints and Differential Pairs

Ting-Chou Lin, Devon Merrill, Yen-Yi Wu, Chester Holtz and Chung-Kuan Cheng
UC San Diego, La Jolla, CA, USA
{til002, djmerril, yew001, chholtz, ckcheng}@eng.ucsd.edu

ABSTRACT

The printed circuit board (PCB) routing problem has been studied extensively in recent years. Due to continually growing net/pin counts, extremely high pin density, and unique physical constraints, the manual routing of PCBs has become a time-consuming task to reach design closure. Previous works break down the problem into escape routing and area routing and focus on these problems separately. However, there is always a gap between these two problems requiring a massive amount of human efforts to fine-tune the algorithms back and forth. Besides, previous works of area routing mainly focus on routing between escaping routed ball-grid-array (BGA) packages. Nevertheless, in practice, many components are not in the form of BGA packages, such as passive devices, decoupling capacitors, and through-hole pin arrays. To mitigate the deficiencies of previous works, we propose a full-board routing algorithm that can handle multiple real-world complicated constraints to facilitate the printed circuit board routing and produce high-quality manufacturable layouts. Experimental results show that our algorithm is effective and efficient. Specifically, for all given test cases, our router can achieve 100% routability without any design rule violation while the other two state-of-the-art routers fail to complete the routing for some test cases and incur design rule violations.

KEYWORDS

physical design, printed circuit board, routing

1 INTRODUCTION

In recent years, because of the growth of net/pin counts, extremely high pin density, and special physical constraints, the manual routing of printed circuit board (PCB) has become a time-consuming task. Moreover, the physical constraints required by the high performance PCB, such as length-matching routing, pairwise routing, and planar routing, are still a burden of the automation of PCB routing [1, 2]. Besides, for high-performance PCB designs, high-frequency signal transmissions are typically through differential pairs [3]. Differential pairs have several excellent properties, such as better tolerance of ground offsets, higher noise immunity, and improved reduction of electromagnetic interference, making it a popular technique for high-performance PCB designs. To realize the advantages of differential pairs, the two connections of each differential pair should be routed close to each other with matched wirelength and thus causes the routing of differential pairs nontrivial.

Furthermore, due to simplification, previous works of PCB routing usually consider the connections between the regular ball grid array (BGA) of chip packages [1]. However, from our observation of real PCB designs, it is common for a design to contain lots of components, such as passive devices, decoupling capacitors, and through-hole pin arrays, that are not BGA packages. Figure 1(a) shows an example of a partial real PCB design with components other than BGA packages. These non-BGA packages usually irregularly spread inside a PCB design, which incurs an uneven distribution of routing congestion and makes the routing task even harder. Thus, it is desirable to develop a PCB routing algorithm that can handle such designs while satisfying the unique physical constraints of high-performance PCBs.

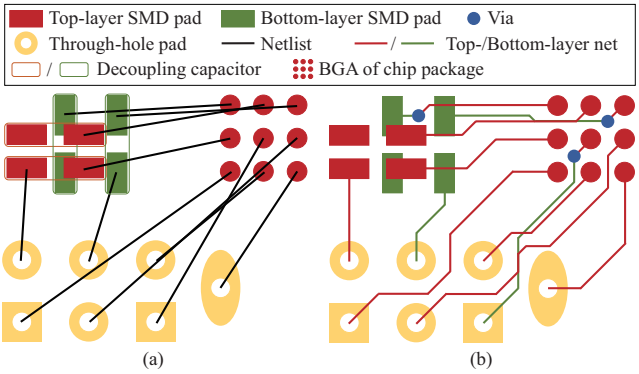

Figure 1: (a) A partial view of a typical PCB routing layout, including solder-mask-defined (SMD) pads, through-hole pads, and netlist. Note that an SMD pad appears on either top or bottom layer, while a through-hole pad is punched through all the layers from top to bottom. (b) A feasible routing result of the given PCB routing layout from (a).

1.1 Previous Works

The objective of the widely-studied PCB routing problem is to route all the connections between the BGA of chip packages while satisfying various physical constraints. Besides, the use of vias is often restricted, and thus the routing becomes planar. This planar routing style makes PCB routing a different problem of IC routing. Previous studies categorized the PCB routing problem into two types: (1) the escape routing problem and (2) the area routing problem [1, 2]. The escape routing problem is to route from the pads of BGAs to their array boundaries. The area routing problem is to connect the previously escaped routes of BGAs and usually subject to an upper/lower bound of routed length for each connection.

The works [4–10] focused on the escape routing problem. The escape routing problem can be further classified into unordered escape and ordered escape; the former needs not to route the connections with specific ordering on the boundary while the latter needs. Wang *et al.* [4] presented a model of triangular pattern and sequence to facilitate a network-flow-based approach to derive unordered escape routing solutions. Kubo *et al.* [5] developed an iterative via assignment method to efficiently minimize the wire congestion and the total wirelength. Fang *et al.* [6, 7, 9] also utilized network-flow-based algorithms

ASPDAC '21, January 18–21, 2021, Tokyo, Japan

ACM ISBN 978-1-4503-7999-1/21/01…$15.00
https://doi.org/10.1145/3394885.3431568

to perform the unordered escape routing. Fang *et al.* [8] presented an integer-linear-programming-based method with reduction techniques to accomplish the ordered escape routing in a reasonable runtime. Yan *et al.* [10] proposed a network model that can correctly reflect the capacity of routing resources and thus gave a better result of unordered escape routing.

The works [11–14] addressed the area routing problem with length-matching constraint. Ozdal *et al.* [11] adopted a Lagrangian-relaxation-based routing scheme to obtain length-matching routing solutions. Also, Ozdal *et al.* [12] presented an efficient river-routing-based algorithm to tackle the length-matching routing problem inside a channel with a theoretical constant factor to the optimal solution. Yan *et al.* [13] proposed an algorithm by the bounded-sliceline grid to retrieve routing solution without any routing topology limitation. Yan *et al.* [14] established an obstacle-aware routing framework based on routing region partitioning and max-flow algorithm.

In addition to the conventional escape routing and area routing problems, the works [15–21] handle the routing problem while considering the constraints of differential pairs. Fang *et al.* [16, 18] proposed a chip-package-board codesign algorithm considering differential pairs, but their assumption of monotonic routing limits the practical use of the algorithm. Yan *et al.* [17] presented a negotiation-based routing algorithm considering the routing congestion to finish the routing of differential pairs. Li *et al.* [19] adopted a two-stage scheme and min-cost max-flow algorithm to route all differential pairs simultaneously. Wang and Jiao *et al.* [20, 21] developed unordered and ordered escape routing algorithms of the staggered-pin-arrays with the consideration of differential pairs.

1.2 Differences from Previous Works

As mentioned above, we notice that there are irregularities of non-BGA packages within PCB designs. The conventional studies of escape routing and area routing may be inappropriate for dealing with these irregularities because they assume that the PCB designs contain only BGA packages. Thus, it is necessary to have a new unified algorithm that can handle this kind of routing problem. Figures 1(a) and (b) show an example of this routing problem and a feasible routing solution from (a), respectively. From this example, it is clear that the combination of escape routing and area routing is not suitable to handle such routing problems due to the unevenly scattered non-BGA packages. Furthermore, we observe that the routing of differential pairs becomes a different problem from what was assumed in the previous works. Due to the packages with very dense BGA, there may not have enough space for both nets of a differential pair to route through a very dense BGA on the same layer. Hence, the only viable way to perform escape routing on this very dense BGA is to assign a "*dog-bone*" via among the pads of BGA and then escape the nets on a different routing layer. Figure 2(a) shows a possible routing solution offered by an extension of the work [21]. This solution suffers from various design rule violations due to the limited space among the BGA and thus is not preferred. Therefore, it is desirable to propose a new routing algorithm that can handle the differential pair routing with very dense BGA. Figure 2(b) shows a routing result from our proposed algorithm, where "dog-bone" vias are assigned, and the differential pair is escaped through a different routing layer.

1.3 Our Contributions

We summarize our main contributions as follows:

- Most of the previous works have an assumption that PCB designs are with regular BGA, and thus they can handle PCB routing by a combination of escape routing and area routing. However, in reality, lots of PCB designs contain packages that are not in BGAs, making most of the previous works inapplicable to such designs. This paper is the first work in the literature to propose a comprehensive multi-layer PCB routing algorithm that can handle such designs with the routing of power/ground nets, differential pairs, and signal nets together under a unified routing framework. Besides, a novel cost function of obstacles for routed nets is presented to facilitate the negotiation of routing resources.
- We propose a scheme for multi-layer differential pair routing. Compared to previous works, which can only handle differential pairs in a single layer routing manner, our approach can search for better dog-bone via locations and perform multi-layer differential-pair routing. Moreover, our proposed scheme allows differential pairs to negotiate the routing resources with all the other signal nets to improve the overall routability.
- We present a pad-entry-aware post-processing method that can further refine our routing solution to enhance the manufacturability. By modifying routing segments within the clearance region of a pad or adding fillers, we can effectively remove design rule violations of acute-angle and sliver.
- Experimental results based on the real academic and industry PCB designs show that our algorithm is effective and efficient. In particular, our router can achieve 100% routablility without any design rule violation for all test cases while the other two state-of-the-art routers fail to complete the routing in several test cases and incur design rule violations jeopardizing the manufacturability.

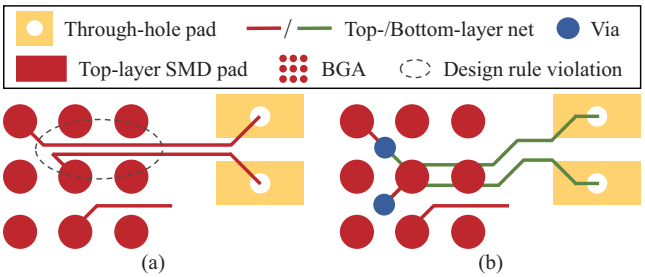

Figure 2: The comparison between an extension of the previous work [21] and our algorithm. (a) A possible routing solution offered by an extension of the work [21], where design rule violations (i.e., spacing violations) are introduced due to the limited space among the BGA. (b) A routing result provided by our proposed algorithm, where "dog-bone" vias are assigned, and the differential pair is escaped through a different routing layer to avoid spacing violations.

The rest of this paper is organized as follows. Section 2 formulates the addressed problem and states the PCB routing design rules. Section 3 details our proposed algorithm. Section 4 shows the experimental results. Section 5 concludes this paper.

2 PRELIMINARIES

In this section, we provide the formal definition of the PCB routing problem and depict the routing design rules corresponding to our problem.

2.1 Problem Formulation

In this paper, we focus on a unified multi-layer PCB routing problem. We first provide the following terminologies and notations used throughout this paper:

- *Solder-mask-defined (SMD) pad:* a top- or bottom-layer pad that has a predefined assignment within the netlist.
- *Through-hole pad:* a pad that punched through all routing layers that has a predefined assignment within the netlist.
- $N = \{n_1, n_2, ..., n_m\}$ is a netlist, where m is the number of nets and n_i defines the connection between SMD pads or through-hole pads.

Figure 1(a) shows an example of the partial structure of the considered PCB routing planes. Unlike the previous work, there are SMD pads and through-hole pads scattered around the routing plane that are not BGAs. Because of these irregularities, the algorithms proposed by previous works are generally not applicable to such routing problems. Figure1 1(b) shows a sample of two-layer routing results satisfying all the design rules. Here, we formally define the problem as follows:

- **The Unified Multi-Layer PCB Routing Problem:** Given a set of SMD pads, a set of through-hole pads, a netlist N, and design rules, connect all the nets $n_i \in N$ so that there is no design rule violation and the total wirelength is minimized.

2.2 Routing Design Rules

In this paper, to ease the design complexities while maintaining better manufacturability, our algorithm should handle several routing design rules. The five major design rules in our problem are detailed as follows:

- *Non-crossing and spacing constraint:* Net crossing is not allowed on the same layer. Moreover, routed nets on the same layer should maintain a specific amount of spacing between each other.
- *Routing-angle constraint:* Routing angles of 90-degree or 135-degree can be performed. However, a routing angle of 45-degree is not allowed.
- *Pad-entry constraint:* Routed nets entering a pad should not introduce an acute-angle between each other; otherwise, it is called an *acute-angle* design rule violation. In addition, if a routed net does not enter a pad, it should maintain a specific amount of spacing between the routed net and the pad; otherwise, it is called a *sliver*, a design rule violation. Figure 3(a) and (b) show examples of acute-angle and sliver design rule violations, respectively.
- *T-junction constraint:* Routed nets with three segments connected together on the same layer should maintain a Y-junction but not a T-junction. Figure 3(c) shows an example of T-junction design rule violation.
- *Differential-pair constraint:* Routed nets of a differential pair should be as close as possible to each other with a similar routed wirelength.

3 ALGORITHMS

In this section, we present our routing algorithm. First, we give an overview of our proposed algorithm. Then, we detail the methods used in each stage of the algorithm.

3.1 Algorithm Overview

Figure 4 summarizes our grid-based routing algorithm, which consists of three major stages: (1) routing grid initialization considering arbitrary pad shape, (2) iterative printed circuit board routing, and (3) design-rule-aware post processing. In the first stage, we capture the routing obstacles by rasterizing the arbitrary shapes of obstacles and pads to our routing grid. The second stage is iterative PCB routing.

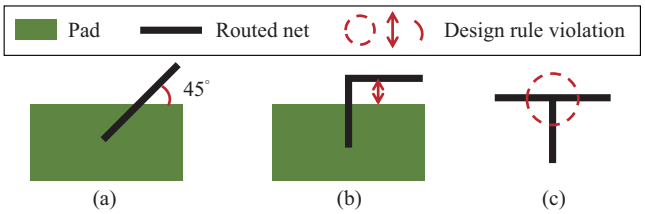

Figure 3: Examples of routing design rule violations. (a) An acute-angle violation of pad-entry constraint: the angle between a routed net and a pad should be 90-degree or 135-degree. (b) A sliver violation of pad-entry constraint: the spacing between a routed net and a pad should be larger than a designated value. (c) A violation of T-junction constraint: three segments of a routed net connected together on the same layer should maintain a Y-junction but not T-junction.

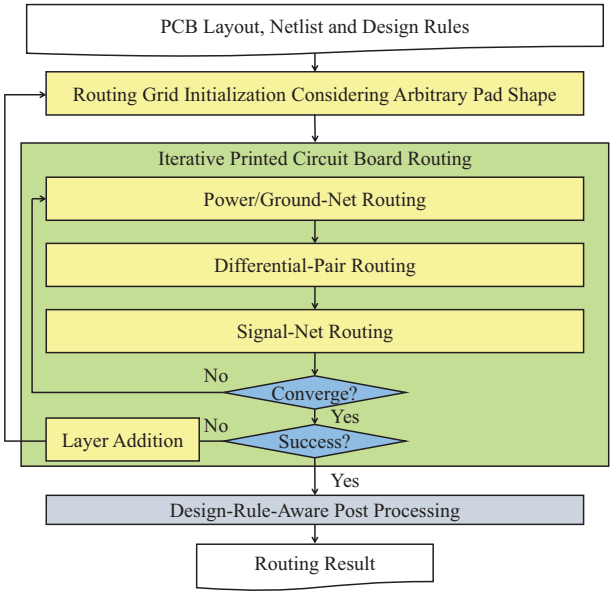

Figure 4: Routing flow of our proposed algorithm.

Each iteration consists of the routings of power/ground nets, differential pairs, and signal nets. Power/ground-net routing routes all the power/ground-net on their designated routing layers. Differential-pair routing routes all the differential pairs while satisfying the differential pair constraints. Signal-net routing routes all the remaining nets. The negotiation-based router [22] is adopted to perform all the above three types of routing while minimizing the number of design rule violations. In the last stage, post-processing is performed to further minimize the number of design rule violations.

3.2 Routing Grid Initialization

PCB designs may contain various predefined obstacles region, called *routing keepout*, in which designated nets should not be routed. These routing keepouts are defined by PCB designers for various purposes and may come in arbitrary shapes. To comply with these routing keepouts, we rasterize these arbitrary shapes along with the SMD and through-hole pads to a 3-D routing grid as obstacles to facilitate later stages to derive a design-rule-violation free routing solution. Figure 5(a) shows an example of routing keepouts and pads represented

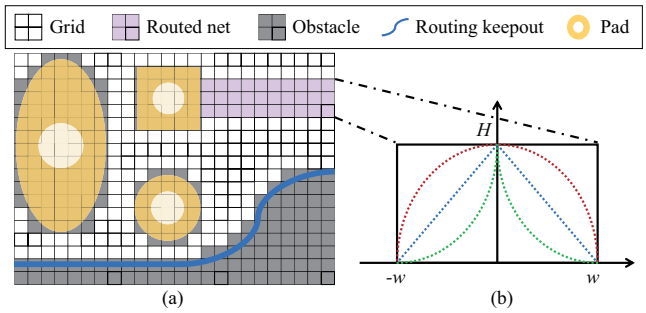

Figure 5: (a) An example of routing keepouts and pads as obstacles in the routing grid. (b) A cross-sectional view of a cost function of obstacles for routed nets.

by obstacles in the routing grid. Routing keepouts and pads are fully enclosed and covered by the obstacles to avoid design rule violations.

3.3 Iterative Printed Circuit Board Routing

In the section, we detail our iterative PCB routing scheme. To perform the tasks of both escape routing and area routing, our proposed unified routing scheme is based on a grid-based A* searching algorithm [23]. By adjusting the cost function of an A* searching algorithm, we are able to adapt the A* searching to fulfill the needs of different routing objectives. Different from previous proposals, which route each type of nets separately and then combine them together with dedicated methods to resolve the conflicts of resources. In our routing scheme, because all the different types of nets are routed in the same routing grid, all of the nets, including power/ground nets, differential pairs, and signal nets, can negotiate the routing resources together without any manual adjustment or overly-complex approaches.

An A* searching cost function $f(x)$ representing a cost of a path x can be defined as:

$$f(x) = g(x) + h(x), \tag{1}$$

where $g(x)$ is the cost from the source to the current location x, and $h(x)$ is the estimated cost from the current location x to the target. Since our router supports the routing angle of 135-degree and 90-degree, in this work, we design the estimation function $h(x)$ to be:

$$\begin{aligned} minD &= min(|x.x - t.x|, |x.y - t.y|), \\ maxD &= max(|x.x - t.x|, |x.y - t.y|), \\ h(x) &= maxD - minD + \sqrt{2} * minD, \end{aligned} \tag{2}$$

where $minD$ is the minimum difference of a location x and a target t along with x-axis or y-axis and $maxD$ is the maximum one. The estimation function $h(x)$ here is admissible, which would never overestimate the actual cost from the current location x to the target t and thus the A* searching gives an optimal solution.

Furthermore, we propose a novel cost function of obstacles for routed nets within each routing iteration. Since our iterative routing scheme routes each net one-by-one, a formerly routed net would become an obstacle for the routing afterward. Depending on the width of a routed net, it may occupy several grid cells as obstacles (i.e., soft obstacles). Hence, from the cross-sectional perspective, this cost function is set to determine each grid cell's value (i.e., obstacle cost) according to its distance to the center of a routed net. To encourage the negotiation of routing resources, we design a novel cost function of obstacles for routed nets as:

$$\begin{aligned} OC_{routed\_net}(r) &= H - H * (|r|^c / w^c), \\ c_i &= c_{i-1} * (\lambda + 1), \end{aligned} \tag{3}$$

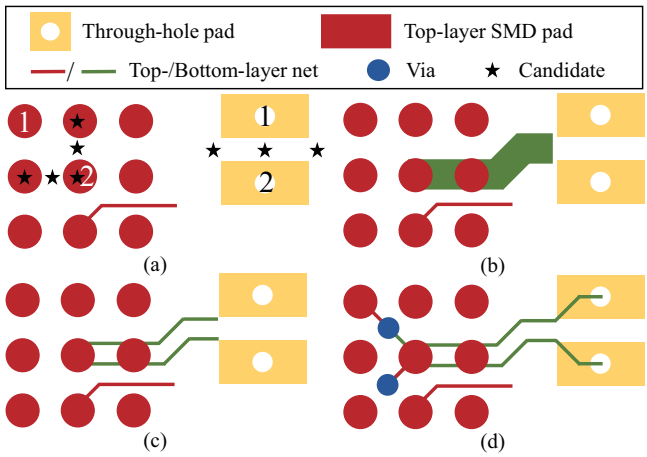

Figure 6: An example of differential-pair routing. (a) The first step is to decide candidate merging points for each side of the pad pairs of the differential pairs, where the routing of nets of a differential pair will be parallel to each other. (b) The second step is to route between the candidate merging points from each side of the pad pairs with enlarged wire width and clearance searching region. (c) The third step is to decouple the routed net from the second step into two routed nets. (d) The fourth step is to route each of the decoupled nets to its corresponding pads on each side.

where OC_{routed_net} is the obstacle cost, r is the distance to the center of a net, w is half of the net width, H is the cost that is large enough to avoid net crossing, c is designed to adjust the cost at the edge of a net (from the cross-sectional perspective) to manipulate the level of resource negotiation, and λ is a user-defined parameter. For example, Figure 5(b) gives a cross-sectional view of a cost function of obstacles for routed nets. From green to blue and then red dotted lines, the c is slightly increased. In other words, to encourage the negotiation of routing resources at the early stage, the cost function represented by green dotted lines may be selected. As the continuing of iterations, the cost function represented by red dotted lines can be utilized to avoid spacing violations.

3.3.1 Power/Ground-Net Routing. To adapt the general A* searching algorithm to the routing of power/ground-net routing, we have to adjust the cost function $g(x)$ to realize the need for routing power/ground nets on their designated routing layers. The cost function $g_p(x)$ is defined as follows:

$$\begin{aligned} g_p(x) &= \alpha * C_{layer} + \beta * C_{wl} + C_{obstacle}, \\ C_{wl} &= \gamma * C_{Euclidean} + C_{layer\_change}, \end{aligned} \tag{4}$$

where α, β, and γ are user-defined parameters, C_{layer} is the cost to route on a specific layer, C_{wl} is the cost of accumulated walked wirelength, $C_{Euclidean}$ is the cost of walking on the same layer, C_{layer_change} is the cost of walking across different layers (i.e., the cost of utilizing a via) and $C_{obstacle}$ is the cost of obstacles among the clearance region. With this cost function $g_p(x)$, we can find a minimal cost path of a currently routing power/ground net on a preferred routing layer.

3.3.2 Differential-Pair Routing. Our multi-layer differential-pair routing approach includes four steps. The first step is to decide the candidate merging points for each side of the pad pairs of the differential pairs, where the routing of nets of a differential pair will start to be parallel to each other. The second step is to route between the candidate merging points from each side of the pad pairs with an enlarged wire width and clearance searching region. The third step is

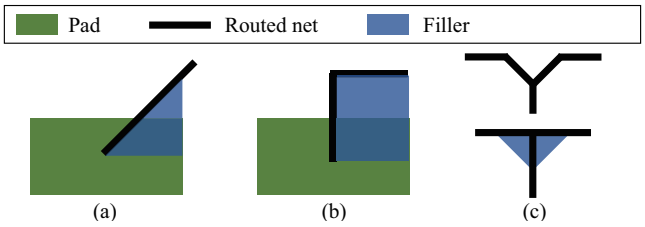

Figure 7: (a) An example of fixing the acute-angle violation of pad-entry constraint by adding a filler to cover the acute-angle. (b) An example of fixing the sliver, a pad-entry violation, by introducing a filler to cover the limited spacing. (c) An example of fixing the T-junction violation by introducing a filler or directly turning a T-junction into a Y-junction.

to decouple the routed net from the second step into two routed nets while the start and end points of the decoupled nets are still close to the candidate merging points. Finally, the fourth step is to route each of the decoupled nets to its corresponding pads on each side and correctly assign dog-bone vias among BGA. Note that in the second and fourth steps, the routing cost function is the same as the one used in signal-net routing and will be elaborated later.

3.3.3 Signal-Net Routing. To conform the general A* searching algorithm to the routing of signal nets and differential pairs, we need to modify the cost function $g(x)$ to achieve a better routing solution. The cost function $g_s(x)$ is defined as follows:

$$g_s(x) = \delta * C_{layer} + \epsilon * C_{wl} + \eta * C_{bend} + C_{obstacle}, \quad (5)$$

where δ, ϵ, and η are user-defined parameters, C_{layer} is the cost to route on a specific layer, C_{wl} is the cost of accumulated walked wirelength, C_{bend} is the cost proportional to the accumulated number of bending, and $C_{obstacle}$ is the cost of obstacles among the clearance region. With this cost function $g_s(x)$, we can obtain a minimal cost path of a currently routing net while minimizing the number of bends and design rule violations, and assigning appropriate dog-bone vias.

3.4 Design-Rule-Aware Post Processing

In the last stage, we perform various post-processing to minimize the number of design rule violations further. After the iterative PCB routing, all of the nets are routed without spacing violations, and hence the rooms for post-processing have been reserved. We are free to adjust the segments of routed nets slightly without any concern about introducing new design rule violations. Alternatively, we can insert additional fillers. Figures 7(a) and (b) show a way to eliminate the pad-entry violations by adding fillers. Similarly, Figure 7(c) shows an example of fixing the T-junction violations by adding fillers or modifying routed segments.

4 EXPERIMENTAL RESULTS

We implemented our algorithm in the C++ programming language. Boost C++ libraries [24] were used to calculate the geometric computation within our proposed algorithm. All of the experiments were performed on an Intel Xeon 1.80GHz Linux machine with 256GB memory, except the ones with DeepPCB [25], which were performed by the services of DeepPCB. Table 1 lists the benchmark of real academic and industry PCB designs. "WxH" denotes the dimension of width and height for the designs. "$|L|$", "$|C|$", "$|P|$", and "$|N|$" denote the numbers of PCB routing layers, components, pads, and nets, respectively.

As mentioned in Section 1.2, most of the previous studies assume that the PCB designs are with regular BGA. However, in practice, some SMD pads and through-hole pads may not be in the arrangement of an array, making it challenging to apply the previous works to solve the routing task of such designs. Therefore, to justify the efficiency and effectiveness of our proposed algorithm, we compared our algorithm with two state-of-the-art non-academic routers, DeepPCB and FreeRouting. DeepPCB [25] is a commercial AI-driven cloud-based automatic PCB routing services launched recently.[1] FreeRouting [26] is a well-known open-source PCB routing software written in Java.[2] Both routers support the PCB routing with non-BGA packages. Besides, both routers were set with their default parameters and given the same input designs and design rules for our experiments. For our router, the width and height of a grid cell were 0.05 millimeters. Non-crossing, spacing, and routing-angle constraints were verified by KiCad [27], and we implemented a design rule checker for the remaining constraints.

Table 2 shows the experimental results and reports the total wirelength, the number of vias, the routability, the runtime, and the total number of design rule violations (DRVs).[3] Note that the number of DRVs here includes all types of violations from the constraints mentioned in Section 2.2. As shown from the table, compared with

Table 1: Benchmark PCB Design statistics.

Designs	WxH (mm)	$\|L\|$	$\|C\|$	$\|P\|$	$\|N\|$
PCB1	21x14	1	8	40	15
PCB2	51x23	2	18	77	34
PCB3	55x28	2	34	138	38
PCB4	23x60	2	28	140	52
PCB5	41x42	2	48	163	54
PCB6	62x57	2	48	190	64
PCB7	51x23	2	46	211	69
PCB8	57x87	2	36	188	70
PCB9	44x36	2	58	229	80
PCB10	102x54	2	57	319	99
PCB11	89x58	2	64	401	134
PCB12	58x60	4	58	233	35
PCB13	86x72	4	61	314	63
PCB14	86x54	4	1570	1638	386

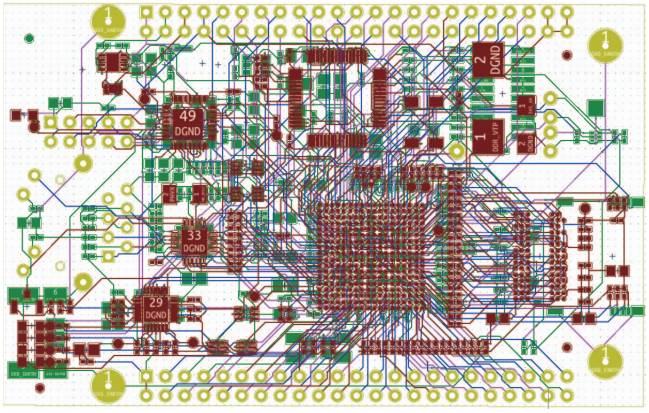

Figure 8: The PCB routing solution of PCB14 from our algorithm. The yellow pads are through-hole pads, the red nets and SMD pads are on the Top layer, the blue nets are on the second layer, the pink nets are on the third layer and the green nets and pads are on the Bottom layer.

[1]The version of DeepPCB was "11 Feb 2020: Version 0.9.0 Beta release".
[2]The version of FreeRouting was "Version 1.3.1".
[3]DeepPCB was under a beta trial when performing the experiments and can only support PCB designs with less than 250 pins and 2 layers. Hence, testcases PCB10 through PCB14 did not have routing results from DeepPCB.

Table 2: Comparison of the overall PCB routing results between DeepPCB(DEEP), FreeRouting(FR), and ours ("N/A": over 24 hours).

2*Designs	Total Wirelength (mm)			#Vias			Routability (%)			Runtime (sec.)			#DRVs		
	Deep	FR	Ours	Deep	FR	Ours	Deep	FR	Ours	Deep	FR	Ours	Deep	FR	Ours
PCB1	84	94	77	0	0	3	100.0	100.0	100.0	540	9	3	3	0	0
PCB2	309	281	272	5	7	13	100.0	100.0	100.0	1020	29	4	13	5	0
PCB3	527	558	464	26	17	44	97.4	97.4	100.0	N/A	N/A	193	21	11	0
PCB4	752	779	700	20	9	60	100.0	100.0	100.0	1620	166	110	22	17	0
PCB5	950	558	854	38	32	91	98.1	98.1	100.0	76620	N/A	542	19	7	0
PCB6	1027	1007	983	23	23	46	100.0	100.0	100.0	1980	302	151	34	11	0
PCB7	913	972	1046	70	57	145	94.2	100.0	100.0	74700	852	283	50	27	0
PCB8	1102	1121	1163	0	1	13	100.0	100.0	100.0	N/A	49	286	6	10	0
PCB9	1053	1069	1065	57	26	98	100.0	100.0	100.0	5760	415	172	38	17	0
PCB10	-	4124	4190	-	73	457	-	100.0	100.0	-	1759	2320	-	12	0
PCB11	-	3459	3341	-	131	317	-	99.3	100.0	-	N/A	11990	-	23	0
PCB12	-	1536	1429	-	36	88	-	100.0	100.0	-	502	742	-	15	0
PCB13	-	3147	3000	-	98	172	-	100.0	100.0	-	3372	2474	-	11	0
PCB14	-	8999	8056	-	922	1370	-	91.2	100.0	-	N/A	25247	-	92	0
Comp.	-	-	-	-	-	-	98.9	99.0	100.0	-	-	-	22.89	18.43	0

DeepPCB (Deep) and FreeRouting (FR), our algorithm can achieve 100% routability for all the fourteen benchmarks without any design rule violation, while DeepPCB and FreeRouting fail to obtain routing solutions for several benchmarks and incur undesired design rule violations. Note that the routing result of our router reflects a higher usage of vias because our router is set to route all the nets as a first objective and then minimize the wirelength. As shown in the results, for the test cases that all the routers accomplish the routing (i.e. 100% routability), our router gives the shortest total wirelength on average among all the routers. Furthermore, depending on the choices of designers and the applications of designs, our proposed routing framework could be easily adjusted to the preferences of minimizing the number of vias or minimizing the total wirelength. Hence, the experimental results show that our algorithm is flexible, effective, and efficient for the routing task of PCB designs with irregular arrangement of SMD and through-hole pads.

5 CONCLUSIONS

In this paper, we have presented a unified PCB routing algorithm for the PCB designs with irregular arrangement of SMD and through-hole pads. In addition, we are the first in the literature to propose a comprehensive multi-layer PCB routing algorithm that can handle power/ground nets, differential pairs, and signal nets together under a unified routing framework. Based on our proposed multi-layer differential pair routing scheme, we have effectively assigned the locations of dog-bone vias and accomplished the routing of differential pairs while negotiating the routing resources with all the other nets. To maintain better manufacturability, we have proposed an efficient post-processing method to further remove the undesired design rule violations that hazard the manufacturability. The experimental results have demonstrated the efficiency and effectiveness of our proposed algorithm.

ACKNOWLEDGMENTS

We would like to thank Dr. Andrew B. Kahng (Professor, University of California San Diego) for providing valuable suggestions for this research. Research at UCSD is supported by DARPA.

REFERENCES

[1] Tan Yan and Martin DF Wong. Recent research development in pcb layout. In *Proc. of ICCAD*, pages 398–403, 2010.
[2] Tan Yan, Qiang Ma, and Martin DF Wong. Advances in pcb routing. *IPSJ Tran. on SLDM*, 5:14–22, 2012.
[3] Christopher T Robertson. *Printed circuit board designer's reference: basics*. Prentice Hall Professional, 2004.
[4] Renshen Wang, Rui Shi, and Chung-Kuan Cheng. Layer minimization of escape routing in area array packaging. In *Proc. of ICCAD*, pages 815–819, 2006.
[5] Yukiko Kubo and Atsushi Takahashi. Global routing by iterative improvements for two-layer ball grid array packages. *IEEE Tran. on CAD*, 25(4):725–733, 2006.
[6] Jia-Wei Fang, I-Jye Lin, Yao-Wen Chang, and Jyh-Herng Wang. A network-flow-based rdl routing algorithmz for flip-chip design. *IEEE Tran. on CAD*, 26(8):1417–1429, 2007.
[7] Jia-Wei Fang and Yao-Wen Chang. Area-i/o flip-chip routing for chip-package co-design. In *Proc. of ICCAD*, pages 518–522, 2008.
[8] Jia-Wei Fang, Chin-Hsiung Hsu, and Yao-Wen Chang. An integer-linear-programming-based routing algorithm for flip-chip designs. *IEEE Tran. on CAD*, 28(1):98–110, 2008.
[9] Jia-Wei Fang, Martin DF Wong, and Yao-Wen Chang. Flip-chip routing with unified area-i/o pad assignments for package-board co-design. In *Proc. of DAC*, pages 336–339, 2009.
[10] Tan Yan and Martin DF Wong. Correctly modeling the diagonal capacity in escape routing. *IEEE Tran. on CAD*, 31(2):285–293, 2012.
[11] Muhammet Mustafa Ozdal and Martin DF Wong. A length-matching routing algorithm for high-performance printed circuit boards. *IEEE Tran. on CAD*, 25(12):2784–2794, 2006.
[12] Muhammet Mustafa Ozdal and Martin DF Wong. Algorithmic study of single-layer bus routing for high-speed boards. *IEEE Tran. on CAD*, 25(3):490–503, 2006.
[13] Tan Yan and Martin DF Wong. Bsg-route: A length-constrained routing scheme for general planar topology. *IEEE Tran. on CAD*, 28(11):1679–1690, 2009.
[14] Jin-Tai Yan and Zhi-Wei Chen. Obstacle-aware length-matching bus routing. In *Proc. of ISPD*, pages 61–68, 2011.
[15] Muhammet Mustafa Ozdal, Martin DF Wong, and Philip S Honsinger. Simultaneous escape-routing algorithms for via minimization of high-speed boards. *IEEE Tran. on CAD*, 27(1):84–95, 2007.
[16] Jia-Wei Fang, Kuan-Hsien Ho, and Yao-Wen Chang. Routing for chip-package-board co-design considering differential pairs. In *Proc. of ICCAD*, pages 512–517, 2008.
[17] Tan Yan, Pei-Ci Wu, Qiang Ma, and Martin DF Wong. On the escape routing of differential pairs. In *Proc. of ICCAD*, pages 614–620, 2010.
[18] Jia-Wei Fang and Yao-Wen Chang. Area-i/o flip-chip routing for chip-package co-design considering signal skews. *IEEE Tran. on CAD*, 29(5):711–721, 2010.
[19] Tai-Hung Li, Wan-Chun Chen, Xian-Ting Cai, and Tai-Chen Chen. Escape routing of differential pairs considering length matching. In *Proc. of ASP-DAC*, pages 139–144, 2012.
[20] Kan Wang, Sheqin Dong, Huaxi Wang, Qian Chen, and Tao Lin. Mixed-crossing-avoided escape routing of mixed-pattern signals on staggered-pin-array pcbs. *IEEE Tran. on CAD*, 33(4):571–584, 2014.
[21] Fengxian Jiao and Sheqin Dong. Ordered escape routing with consideration of differential pair and blockage. *ACM Tran. on DAES*, 23(4):1–26, 2018.
[22] Larry McMurchie and Carl Ebeling. Pathfinder: a negotiation-based performance-driven router for fpgas. In *Reconfigurable Computing*, pages 365–381. 2008.
[23] Peter E Hart, Nils J Nilsson, and Bertram Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Tran. on SSC*, 4(2):100–107, 1968.
[24] Boris Schäling. *The boost C++ libraries*. 2011.
[25] *DeepPCB*. https://deeppcb.ai/.
[26] *FreeRouting*. https://freerouting.org/.
[27] *KiCad EDA*. https://kicad-pcb.org/.