Escaped Boundary Pins Routing for High-Speed Boards

Ching-Yu Chin, Chung-Yi Kuan, Tsung-Ying Tsai, Hung-Ming Chen, and Yoji Kajitani, *Life Fellow, IEEE*

***Abstract*—Routing for high-speed boards is still achieved manually today. There have recently been some related works to solve this problem; however, a more practical problem has not been addressed. Usually, the packages or components are designed with or without the requirement from board designers, and the boundary pins are usually fixed or advised to follow when the board design starts. In this paper, we describe this fixed ordering boundary pin routing problem, and propose a practical approach to solve it. Not only do we provide a way to address, we also further plan the wires in a better way to preserve the precious routing resources in the limited number of layers on the board, and to effectively deal with obstacles. Our approach has different features compared with the conventional shortest-path-based routing paradigm. In addition, we consider length-matching requirements and wire shape resemblance for high-speed signal routes on board. Our results show that we can utilize routing resources very carefully, and can account for the resemblance of nets in the presence of the obstacles. Our approach is workable for board buses as well.**

***Index Terms*—Length matching, printed circuit board (PCB), route.**

I. Introduction

Today, we have increasing pin count on the very dense boards, and current computer-aided design (CAD) tools for board routing are very few and cannot provide automatic solutions for board-level routing. There have been many works addressing this problem, such as [5], [6], [9]–[11], [13]–[17], [20], [21]. Board-level routing can be separated into two categories: the escape routing and the area routing. The former routes inner pins of a component to the boundary of that component, while the latter connects those escaped boundary pins between components.

Manuscript received October 8, 2011; revised February 7, 2012 and June 8, 2012; accepted August 28, 2012. Date of current version February 14, 2013. This work was supported in part by the National Science Council under Contracts NSC 101-2220-E-009-040. A portion of this work was published in IEEE Design, Automation & Test in Europe Conference & Exhibition, 2011 [19]. This paper was recommended by Associate Editor E. Young.

C.-Y. Chin is with the Institute of Electronics Engineering, National Chiao Tung University, Hsinchu 30010, Taiwan (e-mail: alwaysrain.gr@gmail.com).

C.-Y. Kuan is with ASUSTek Computer, Inc., Hsinchu, Taiwan (e-mail: aabgdbeaia@gmail.com).

T.-Y. Tsai is with Global Unichip Corporation, Hsinchu, Taiwan (e-mail: charming91@hotmail.com).

H.-M. Chen is with the Department of Electronics Engineering, National Chiao Tung University, Hsinchu 30010, Taiwan (e-mail: hmchen@mail.nctu.edu.tw).

Y. Kajitani is with the Japan Advanced Institute of Science and Technology, Kanazawa, Japan (e-mail: kajitani@env.kitakyu-u.ac.jp).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCAD.2012.2221714

References [9], [13], [16], [17] address escape routing. Luo and Wong [9] propose a problem in which there is a constraint on pin ordering from the requirement of the board designers, and the others address simultaneous escape routing. Simultaneous escape routing negotiates both sides of escaping terminals and explores a larger solution space. Two very recent works [10] and [11] also focus on this problem. Reference [11] uses a negotiated congestion based router to achieve the routing; this technique is widely used in modern academic global routers. On the other hand, [10] features a new concept boundary routing to solve the simultaneous escape routing problem, which achieves a better solution compared to previous work. The area routing [printed circuit board (PCB) routing] connects the escaped slots between components or packages on the board [5], [15], [20].

In the preliminary version of this paper [19], we address a different yet practical problem: how we route the wires or buses on the board with the fixed component or packages boundary pins. Usually, the components or packages are designed separately with boards; even with few interactions in codesign, they are shipped with fixed boundary pins. Another occasion is that those close-to-center pins in the component are pre-escaped by the provider, such as [21] and [22], and board designers are advised to use the given order of the escaped boundary pins. Our work is a two-stage planner. The first stage is to try to obtain a planar topology for all the connections with more routing space available in subsequent pin pair routing. The second stage is to obtain a refined routing with length-matching constraint awareness and wire shape resemblance. In addition, our approach is workable for board buses as well, even with routing obstacles.

In this paper, we first restate the proposed board routing algorithm. Next, we propose a heuristic approach to handle the inevitable crossing nets. The approach utilizes Supowit's channel route algorithm [18] to find the largest subset of nets that can be routed on one layer. Furthermore, an integer linear programming (ILP) formulation is proposed to adjust net length within each bus such that the net length differences can match the length or delay constraints that high-speed boards demand. Overall, the experimental results show that our approach can utilize the routing space effectively, and meet wirelength and shape requirements while taking obstacles into consideration.

The remainder of this paper is organized as follows. Section II describes the board routing issues. Section III describes the first stage of our routing methodology for obtaining planar topology considering obstacles. Section IV illustrates the routing refinement for length constraints and wire shape resemblance. Section V shows our results followed by the conclusion in Section VI.

II. Routing Issues for High-Speed Boards

Manual work is adopted in board-level routing due to many issues and constraints. The first concerning issue is impedance. Considering the IR drop, crosstalk, and impedance matching, the net shape and the net length must be controlled based on certain criteria. Many original ideas have been created to solve these problems, such as in [2], [7], [14], [15], [20]. Reference [2] proposes a topological planar routing algorithm. The BSG router [20] proposes an effective routing method. This method can match the net length and save a large amount of memory for data under a given routing topology. However, it consumes enormous routing area and ignores the shape of nets, creating a lot of jogged nets (net jogging). These net joggings may cause signal integrity issues.

The second problem is bus planning. The bus router handles a set of nets and routes these nets in parallel. An automatic bus planner is proposed in [5], which provides good results in bus planning. It is based on a good escaped pin assignment; therefore, the bus planning can be automated. However, we observed in the results that the buses are planned in a bunch, which seems unfriendly to obstacles and routing resources.

Therefore, the consequent problem is routing resources in board routing. Due to manufacturing costs, we should keep the number of layers minimal, which generates the constraint of planar routing for board-level planning. Most of the routing algorithms are based on the concept of a shortest path. This kind of router cannot take the whole set of nets into account; in each step, only the present net will be considered and routed. Since the routing of the present net may block the routing path (splitting the routing space) in a sequential routing framework and planar requirement, the routing of the following net or other nets is not favored due to an early routing decision. Since the shortest path algorithm usually causes the failure in such sequential routing, we are forced to go back to the previous step and reroute the net. Reference [3] provides a good framework for topological routing; however, it must use partitioning and Delaunay triangulation for applications, and it also has a net ordering problem. A very recent study [10] showed that boundary routing is a very good concept, but it may be limited by only several escape patterns.

Considering the fixed-boundary pin planar routing, our approach will try to mitigate the aforementioned effects. In order to not spoil the routing space for the remaining nets and to avoid the net crossing during board wire planning, our first stage routing will optimize the routing order to eliminate the conflicts of net routing and to better utilize the routing resource, which is very different from the traditional integrated

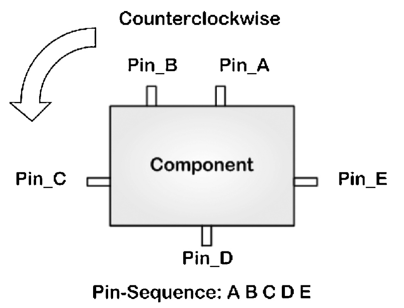


Fig. 1. Illustration of pin-sequence generation of a component for PCB routing. Pin *A* is chosen as the first pin. By walking along the component boundary counterclockwise, the other pins are listed as the pin sequence in order.

circuit global router.[1] During the second stage refinement, we propose a length-constraint ILP formulation to achieve min-max length bounds of nets, and take care of wire shape resemblance.

III. Routing Under Fixed-Ordering Pin Locations

In general, most of the routing algorithms are based on the concept of shortest path, which is usually referred to as maze routing. This kind of router cannot take the whole set of nets into account; in each step, only the present net will be considered and routed. Without the consideration in future routing, the shortest-path algorithm usually causes the failure or much longer detour in routing the next net since the routing space has been split (due to the planar routing constraint), and we are forced to go back to the previous step and reroute the net.

In this paper, we propose our solution of routing under fixed-ordering boundary pins, which takes the whole routing plane into account when routing a single net. First, we decide the net ordering statically by dynamic pin sequence (DPS) to avoid the net crossing in one-layer planar routing. Second, we perform the topological routing under the aforementioned net order. We develop an against-the-wall routing for better preservation of routing resources for consequent routings.[2] Finally, we describe how we handle the inevitable crossing nets.

For the rest of the presentation, a bus is defined as all common nets between a certain pair of components. The nets in this context are all two-pin nets. It is reasonable since most of the board-level connections are two pin nets.

A. Static Net Ordering With a Dynamic Pin Sequence

Before we present the routing algorithms, we first describe the approach for net ordering determination. Given the placement of the components on the board, routing space is formed

[1]A similar work [2] also targets topological planar routing. However, the proposed strategy of generating planar routing in a general routing region is different from this paper.

[2]B-escape [10] is a very recent work with very good results in solving the simultaneous pin assignment and routing problem. However, we make the following observations. First, our concept of against-the-wall routing is somewhat similar to the boundary routing but different. Second, our approach has no limit on monotonic routing, and has few detours.

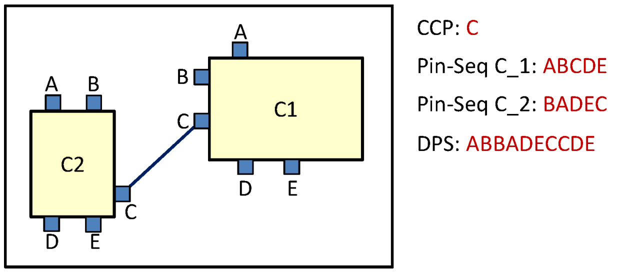


Fig. 2. Illustration of a CCP, which can be seen as the first connection among all nets in components. Pin C is set to be the CCP and is connected to the corresponding component.

with boundary pins on the periphery of the components. The pins are either fixed by package providers or escaped using methods, such as those used in [21] and [22]. We use a number sequence to represent the relative position of pins on one component. In order to include these pins in a number sequence, we randomly select a pin among the pins on one component as the start point of the sequence. From this start point, other pins of this component are inserted into the sequence one by one in a counterclockwise fashion. We apply the same process to the other components and get a series of pin sequences. The main idea is to use a series of pin sequences to represent the locations of pins. As Fig. 1 shows, this component contains five pins; we choose pin A as the start point and make a pin sequence in a counterclockwise fashion, and the sequence is $ABCDE$.

With components transforming to pin sequences, we will concatenate those sequences into one sequence; we name it DPS. The idea is to decide one pin sequence among all pin sequences as a basis sequence, and then we put other pin sequences into this basis by finding a common pin in the two sequences that are going to be concatenated. The selected common pin is referred to as component connecting point (CCP). As illustrated in Fig. 2, pin C is treated as a CCP, and we can concatenate two sequences as one. In this example, the pin sequence of component C_1 is $ABCDE$ and the pin sequence of component C_2 is $BADEC$. After merging with CCP pin C, the concatenated sequence becomes $ABBADECCDE$ when choosing the former sequence as a basis sequence (base DPS). There are many ways or combinations for sequence concatenation; we choose a greedy way. By applying the longest pin sequence (the component with a largest number of pins) as a basis, we then integrate the second longest pin sequence that contains common pin(s) with the basis into the basis, and then the third-longest one, and so forth.[3]

Two consequent problems remain: how do we use this concatenated DPS for net ordering, and which position or pin location is the best? We will first describe how we process the DPS for the first question, then we will present how we choose the CCP for sequences concatenation. Using Fig. 2 as an example, we use the pin sequence of C_1 as base DPS, and pin C is chosen as the CCP for the concatenation of two sequences. The resultant DPS is $ABBADECCDE$.

[3]It is possible to have more than one DPS since there are many components that connect to different components. Larger components (containing more pins) get to form other DPSs.

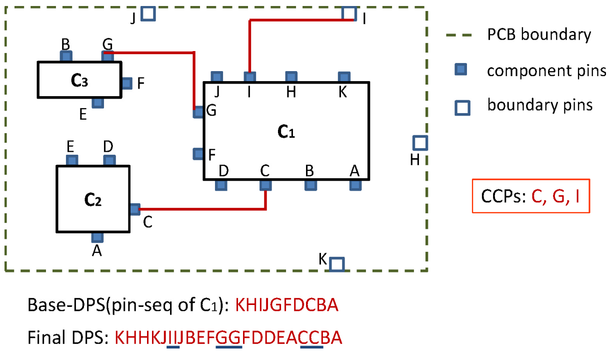


Fig. 3. Example of the first pin set CGI for routing. The underlined and repeated symbols in final DPS II, GG, and CC will be erased and routed first. The remaining DPS is $KHHKJBEFFDDEABA$, and we then connect the adjacent pins according to the DPS and perform the consequent routing.

We observe that symbol C (CCP) is repeated due to the concatenation; this means we can have the routing connection of the two pins of net C in the netlist. We then erase these two symbols in the sequences and the sequence becomes $ABBADEDE$; this means that net B can be connected without crossing. After erasing B, A is the next repeated symbol, and finally the DPS become $DEDE$ without any repeated symbol.

The first erasure net C acts as the start point of topological routing, and the remaining symbol erasure suggests us a routing order that produces a routing result without pin/net blocking. By following the symbol erasure order, prerouted paths will not block unrouted pins if each net is routed along the previous paths, forming a centralized bus, which is called the against-the-wall routing that is introduced in the next subsection. In addition, the interleaving symbols in the remaining DPS, $DEDE$, represent that there exists inevitable net crossing when routing on a single layer.

When multiple components on a board, the generation of DPS will start with a base DPS, and then concatenate each pin sequence into the base DPS according to the selected CCPs one by one until all pin sequences are combined into one final DPS. Fig. 3 illustrates CCP connections of a PCB with three components and four board boundary pins. Pins C, G, and I are selected as CCPs, and the resultant DPS is $KHHKJIIJBEFGGFDDEACCBA$.

We can, in fact, choose any position or pin location as our CCP during each concatenation; however, the choice itself will produce different configurations and then different ordering. This will possibly cause the crossing in some nets that could have been avoided. In the preliminary version, we use a greedy-based approach to minimize crossings for topological routing. For each concatenation, we evaluate each pin in the smaller pin sequence (being concatenated) to check the number of erasures that can be performed. For example, in Fig. 3, after the erasure of the repeated symbols II, we can continue to erase JJ and HH. Therefore, the topological routing for this subset can be performed without crossing. We then choose this pin with a largest number of erasures to be the CCP. In Section III-C, we present a new approach that replaces this greedy erasure process to further improve performance.

The advantages of this static net ordering are as follows. First, through the steps of pin-sequence concatenation, we

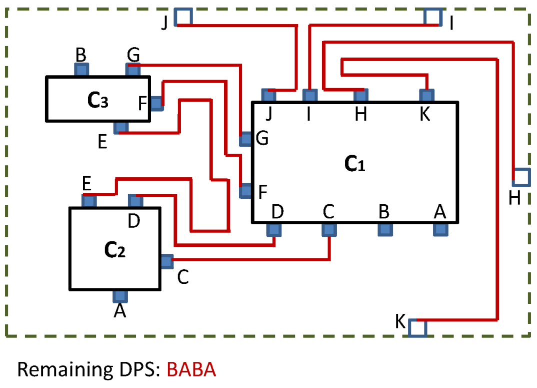


Fig. 4. Routing result of against-the-wall routing. All nets are connected in order and routed against the walls. The results show that the nets will be close to the wires of the connection of the CCPs and have resembled wire shape for bus requirement.

can decide the backbone of the topological routing. Second, we use this ordering to preserve the routing space so that it will not be split into many small and broken pieces, avoiding the possible crossing during sequential routing, especially if the shortest path or mazing routing is used. Third, we have a considerably good static order for routing nets, compared with blind ordering.

The advantage of deciding the backbone includes preserving routing resources for unrouted nets and improving signal integrity due to similar routes for signal propagation. If routing paths of a bus are not grouped as a bunch, there must exist an unrouted area that is surrounded by routed paths or obstacles and can no longer be routed. Note that although the term bus used here indicates common nets between two components, the static ordering can be adapted on other user-defined sets of nets. The limitation of this static ordering is that it does not take routing capacity into consideration; therefore, it may fail to complete the routing of nets in the DPS. In this case, the unrouted nets will be assigned into another routing layer. In our testcases, which are based on real designs, the capacity issue does not affect the routability of the proposed approach.

B. *Against-the-Wall Topological Routing*

The DPS in Fig. 3 shows the first routing set (a collection of CCPs) in this example: CGI. Here, we perform the routing to connect those pins. $A*$ algorithm [4], [8] is used in our router to complete the connections. $A*$ algorithm connects the net by calculating the path cost. In the beginning of the $A*$ algorithm, we look for the lowest cost point near the start point. We refer the lowest cost point as the current point. For each of the four points (directions) adjacent to this current point, we find the lowest cost point and link this point to the current point. We can find the target point by doing this in a repeated manner. Fig. 3 shows the result of the CCP connections.

In order to fulfill the requirement of high-speed board signal routing, we intend to route them as close as possible and consider the length difference constraints. Given the routing shown in Fig. 3, the three paths of CCP connections become the first three boundaries or walls on the routing space. By using the previously routed nets as boundaries or walls, we route the following nets against-the-wall. Following the previous discussion, the erasure of symbol pairs in the DPS will produce the next sets of routing. We repeat this process until no symbol pairs existed in the DPS. The final against-the-wall routing is shown in Fig. 4. From another point of view, we can treat the first routed nets as gravity centers, and the remaining nets are attracted to be placed besides them. We will discuss how we handle the rest of the DPS in the next subsection.

We describe the against-the-wall routing as follows. By using routed nets as a boundary of routing space, the routes walk against the wall or boundary. We follow the order that we described previously to route the remaining nets, each route has a fixed direction toward the destination pin. If there exist blockages (or previously routed nets), the direction will be modified to walk around. The aforementioned routing is achieved by applying $A*$ algorithm with carefully adjusting grid costs. Routing grids closer to the wall or boundary are assigned with lower costs, while grids in the center of an empty region are assigned with higher costs. This strategy allows us to route net individually such that the bus will not be blocked by obstacles due to insufficient capacity between two obstacles. Paths within a bus are still centralized as a bunch when there exist enough routing resources.

C. *Improvement on Routability and Layer Assignment of Nets*

It is possible that the design may not be routable in one layer. We can, in fact, arrange some routes to be routed in other layers; it is also mentioned in [6]. In the previous subsection, we introduce the concept of DPS. By transforming pin locations of a board design into its DPS, we can easily figure out whether a design is routable within one layer or not. We also mention that different selections of CCPs result in different DPS, which affects routability. In fact, not only the CCP selections, but the concatenation order of pin sequences and the layer assignment of nets under given DPS will affect the routability of topological routing as well. In this subsection, we will discuss the net crossing problems, and propose our new approach of routability improvement and layer assignment. The approach described here is an improved method of our preliminary version that will produce better performance.

The problem is divided into two sequential stages: 1) the CCP selection and DPS generation, and 2) layer assignment of nets and routing order determination. At the end of this subsection, we conclude our new method with a flowchart.

1) *CCP Selection and DPS Generation:* A greedy-based CCP selection method is used in the preliminary version. For each component pair, the approach chooses CCP that produces the largest erasure number. However, this strategy has two weak points: 1) it focuses on only the target bus and does not take other nets into consideration, and 2) it has difficulty in tackling designs with a large number of components and complex connections. Since CCP connections are done by $A*$ algorithm following the order of sequence concatenation, as described in Section III-B, the order of concatenation affects the routing result. In Fig. 5, seven nets belong to two buses. For bus ABC, even if we choose net A, B, or C to be CCP, the

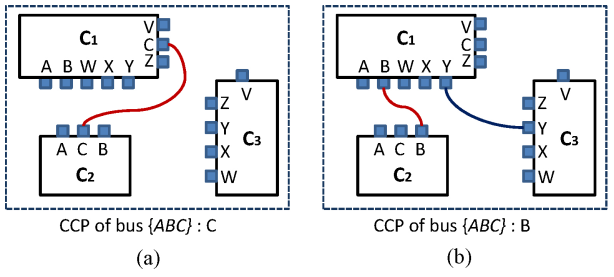


Fig. 5. Illustration of drawback in the CCP selection method of our preliminary version. The CCP connections in (a) result in worse routability compared to (b) a more efficient method. However, our preliminary selection method cannot distinguish the difference.

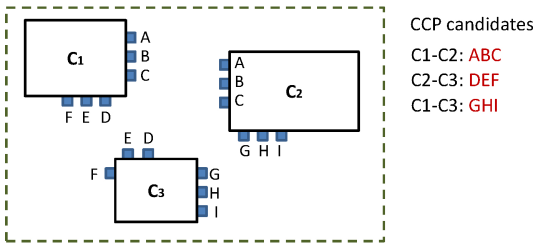


Fig. 6. Illustration of CCP candidate sets of a design with three components connected to each other.

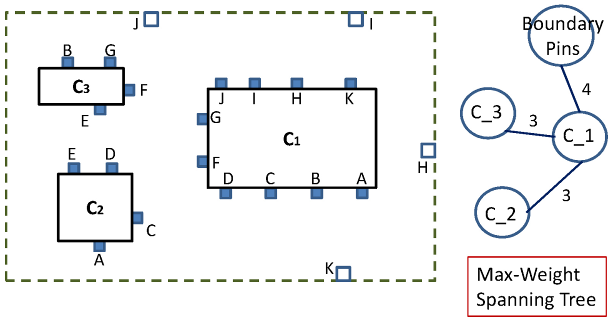


Fig. 7. Generation of a maximum weight spanning tree. The component with the largest connectivity is chosen to be the base DPS. According to the spanning tree, the concatenating process is from the boundary pins $\rightarrow C_2 \rightarrow C_3$.

number of routable nets is two. But when we take the other bus, $VWXYZ$, into consideration, we find that CCP connection shown in Fig. 5(a) has worse routability than in Fig. 5(b). In Fig. 5(a), the routing of bus $VWXYZ$ has more detours and consumes more routing resources than in Fig. 5(b). To avoid this situation, the DPS concatenation order should be arranged carefully, i.e., if the bus with a large number of nets is concatenated into DPS first, the resultant CCP connections may produce better routability.

Another common problem of CCP selection is illustrated in Fig. 6. Three components C_1, C_2, and C_3 are connected with each other, and three different sets of nets (three buses), in which nets ABC connect C_1 and C_2, nets DEF connect C_2 and C_3, and nets GHI connect C_1 and C_3, are going to be routed. Although there are three buses, only two CCPs should be selected to form the final DPS. The CCP connections cannot form any cycle on the routing plane; otherwise, the DPS generation would fail.

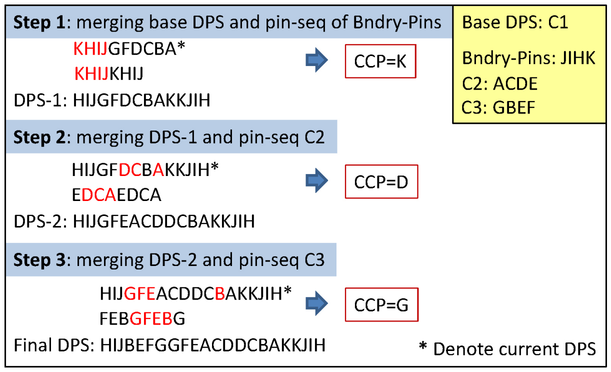

Step 1: merging base DPS and pin-seq of Bndry-Pins
KHIJGFDCBA*
KHIJKHIJ
CCP=K
DPS-1: HIJGFDCBAKKJIH
Step 2: merging DPS-1 and pin-seq C2
HIJGFDCBAKKJIH*
EDCAEDCA
CCP=D
DPS-2: HIJGFEACDDCBAKKJIH
Step 3: merging DPS-2 and pin-seq C3
HIJGFEACDDCBAKKJIH*
FEBGFEBG
CCP=G
Final DPS: HIJBEFGGFEACDDCBAKKJIH
* Denote current DPS
Base DPS: C1
Bndry-Pins: JIHK
C2: ACDE
C3: GBEF

Fig. 8. Example illustrating the whole process of DPS generation of case shown in Fig. 7. All pins are concatenated into a single sequence (final DPS) after K, D, and G are selected as CCPs.

For better performance, we propose a new DPS generation strategy to produce DPS with better routability. For a given board, the component with the largest connectivity with other components is selected to be the base DPS. The CCP selection and sequence concatenation then begin from the base DPS. In order to decide the order of concatenation (and to avoid cycles produced by CCP connections), we define a vertex set $V = \{v : v \in \text{components}\}$, and an edge set $E = \{e(u, v) :$ comp. u and comp. v have common net(s)$\}$, in which edge weight $w(e(u, v))$ equals the number of nets between components u and v. A maximum weight spanning tree T_{max} of $G(V, E)$ is then generated by Prim's algorithm with the vertex(component) representing base DPS as the start point. The concatenation order of DPS is set to the same order of the tree expanding order in Prim's algorithm. Fig. 7 illustrates the MST generated by the original board design with three components and a set of board boundary pins. Because a tree structure produces no cycles, the DPS generation can be processed successfully.

After the determination of a merging order, each pin sequence (except base DPS) is reversed and duplicated to form a sequence with double length (denoted as $PS*$), e.g., pin sequence ABC becomes $CBACBA$. The longest common subsequence [12] is then applied to the base DPS and the first $PS*$ to be connected, which is the reversed and duplicated pin sequence of the first visited component when growing T_{max}. This is to find the longest common net ordering in a cyclic manner. Fig. 8 illustrates the generation of DPS: the first matched symbol K is considered the CCP between boundary pins and component C_1. Once the CCP is decided, the base DPS then becomes $HIJGFDCBAKKJIH$. The current DPS can then be concatenated with $PS*$ of component C_2 (in this case, C_2 and C_3 have the same priority since the number of nets connecting C_1 and C_2 equals the number of nets connecting C_1 and C_3). The merging process continues until all edges in tree T_{max} are processed.

Despite the similarity to topological routing in [2], the major difference is that our approach is bus oriented. The proposed approach intends to route nets within a bus in a bunch in one layer by our CCP selection method, instead of taking net as an individual as [2] did. Under this premise, we generate DPS heuristically rather than exhaustively searching, as in [2].

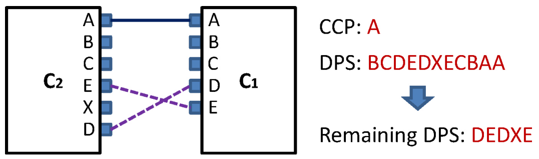


Fig. 9. Example of inevitable crossing situation.

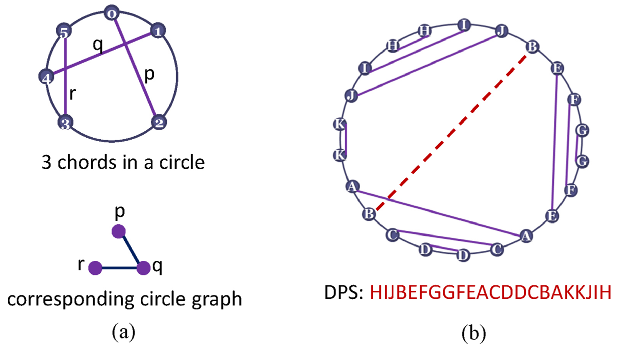


Fig. 10. (a) Transforming chords of a circle to the corresponding circle graph. (b) Illustration of routing net selection calculation by Supowit's channel route algorithm.

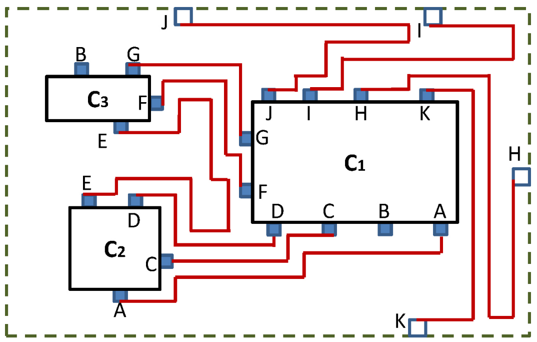


Fig. 11. Routing result using the modified net ordering of against-the-wall routing.

2) *Layer Assignment and Routing Order Determination:* Given a DPS, it is not obvious to determine which net(s) should be routed to maximize routability in the current layer, or to minimize the overall layer number. For example, the nets in Fig. 9 cannot be routed in one layer. By erasing the repeated symbols in DPS, we have sequence $DEDXE$ left, in which X is a net belonging to component C_2 but not C_1. If we route net D, then nets E and X are blocked and thus unroutable. On the other hand, if we route net E, although net D is unroutable, net X is not blocked. The selection of net to be routed is not straightforward in this situation.

Fortunately, the problem is similar to channel routing with dogleg paths. Through CCP connections, DPS is actually a geometric shape changing of channel routing. Here, Supowit's algorithm [18] is applied to find the maximum number of nets that can be routed on one layer.

Supowit's algorithm is originally developed to solve the dogleg-path channel route problem. Given N two-pin nets in a channel, the algorithm finds the largest subset that can be routed on one layer in $O(N^2)$ time. We describe the algorithm here. Given a set C of N chords of a

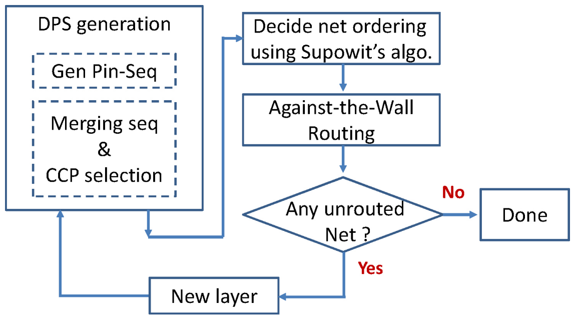


Fig. 12. Overall flow of topological routing.

circle, as shown in Fig. 10(a), number the endpoints of these chords from 0 to $2N - 1$ clockwise around the circle. A circle graph G can be constructed by a vertex set $V = \{v_{ab} : a < b \text{ and } \overline{ab} \in C\}$ and an edge set $E = \{(v_{ab}, v_{cd}) : a < c < b < d \text{ or } c < a < d < b\}$. By observation, we can find that a maximum independent set, MIS, of circle graph G corresponds to a maximum set of crossing-free chords in the original chord set, which equals the largest subset of routable nets on one layer. Although finding MIS in a general graph is NP-hard, Supowit proposed an $O(N^2)$ algorithm for this special circle graph. Defining $G_{i,j}$ as the subgraph of G induced by vertex set $V_{ij} = \{v_{ab} \in V : i \leq a, b \leq j\}$, the maximum independent set of graph $G_{i,j}$, denoted as $MIS(i, j)$, can be calculated by a simple-recursive formula as follows:

Initial condition

$$MIS(i, i) = \phi.$$

Recursive formula

$$MIS(i, j) = \begin{cases} MIS(i, k-1) \cup v_{kj} \cup MIS(k+1, j) \\ \qquad\qquad\qquad , \text{ if (1) holds} \\ MIS(i, j-1) \\ \qquad\qquad\qquad , \text{ else} \end{cases}$$

where (1) goes to

$$\exists \ i \leq k < j \text{ s.t. } \overline{kj} \in C \quad \text{and}$$

$$|MIS(i, k-1)| + |MIS(k+1, j-1)| + 1 > |MIS(i, j-1)|.$$

The MIS of graph G can thus be calculated by $MIS(0, 2N-1)$ due to $G = G(0, 2N - 1)$.

For our board routing problem, we have shown that if two identical symbols can be erased through the erasure process, the corresponding net can be routed without net crossing. By arranging symbols around a circle, we can easily find that the erasure process is actually a simplified version of Supowit's algorithm, in which inevitable crossing nets correspond to crossing chords in the circle. Fig. 10(b) illustrates the result of applying Supowit's algorithm on the DPS generated in Fig. 8. Symbols connected with solid lines correspond to nets that are selected to be routed on the current layer with no crossing occurs between each other, and dotted lines indicate nets that are not selected to be routed on the current layer. Fig. 11 shows the routing result with Supowit's algorithm; only net B is unroutable.

3) *Overall Flow of the Proposed Topological Routing:* In this paper, we propose a new algorithm to handle the topological routing problem, which utilizes the concept of dynamic pin sequence in the preliminary work, but with a different approach. Fig. 12 shows the complete steps to perform the global topological routing. Given component placement and netlist, we first generate the DPS of the design using the new algorithm. Next, according to the DPS, which nets should be routed and a static net order are both produced by Supowit's channel route algorithm. Third, we perform our against-the-wall routing to generate a topological routing result of the first layer. If there is any unrouted net left, a new routing layer is needed, and the whole procedure is repeated again with respect to the remaining nets. The procedure will repeat until no unrouted net is left.

IV. Length-Constraint-Aware Routing Refinement

After against-the-wall routing, our router produces a series of bus nets. In this stage, we adjust the routing without changing the completed interconnection to match the length constraints on the board to resemble the net shape and to shorten (sometimes lengthen) the wirelength for longer (shorter) nets. Recall that during the process of the against-the-wall routing, a central (backbone) net is generated and the other nets are routed close to the central net, and thus more unused area can be reserved for later use. This unused area is useful for net length adjusting, in which the length differences should be adjusted to meet the min-max length constraint of the design.

In this section, we propose an ILP formulation to adjust the net length. Due to obstacles or the location of components, the edge shifting used in the preliminary version sometimes fails to find a feasible solution. Compared with the edge shifting technique, our new proposed ILP approach expands more solution space in finding paths.

A. ILP Formulation for General Length-Matching Problem

We observe that, once all nets in a bus are merged as a bunch, the length differences within the bus are relatively small. As shown in Fig. 13(a), the separated pin locations contribute most of the length mismatch. To compensate length differences caused by pin locations while maintaining net shape for signal integrity, we adjust routing paths inside a certain region, referred to as the adjust region. The adjust region is a ring-shaped area around a component, as shown in Fig. 14. Outside the adjust region, routing paths remain the same as in against-the-wall routing. Inside the adjust region, prerouted paths will be removed and rearranged by the proposed ILP length-matching algorithm. The position where paths pass through the boundary of adjust region is called the cut point. The objective of length matching is to rearrange paths inside the adjust region with each net, from its pin to the cut point, having equal length (or satisfying the min–max constraint).

Each adjust region is split into routing tracks, and each track of the region is considered a 1-D virtual axis, as illustrated in Fig. 14. Pin locations (or any point inside adjust region) are transformed into coordinates on the corresponding axis. Any

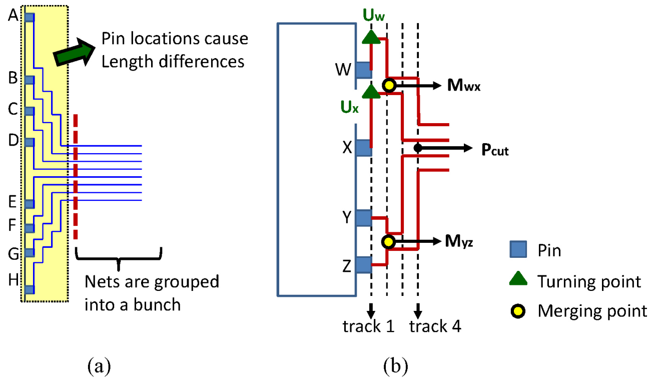


Fig. 13. (a) Pin locations contribute most of the length mismatch. (b) Example of length adjusting.

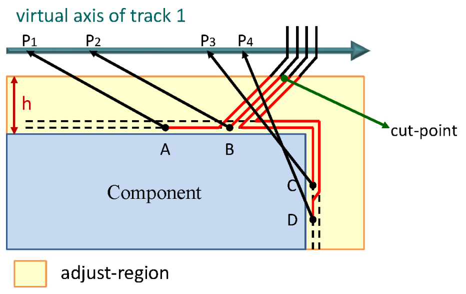


Fig. 14. Transforming 2-D length-matching problem into 1-D coordinate determination. The four pins are mapped to four coordinates on the virtual axis, respectively, for length adjusting. The ILP length adjusting will be performed within the adjust region.

path inside the adjust region can be represented by a series of coordinates on each axis (tracks). Through this transformation, the 2-D length matching is reduced to a 1-D problem.

Fig. 13(b) illustrates the idea of our net adjusting strategy. Four nets are demanded to adjust their length inside the adjust region. We divide the problem into small subproblems. First, consider only nets Y and Z, the path equalization can be done by using track 1 and track 2. The paths of the two nets are merged at point M_{YZ} with equal length $length(Y, M_{YZ}) = length(Z, M_{YZ})$. After merging, nets are demanded to route as a bunch. Similarly, nets W and X are merged at M_{WX}. However, due to the fixed location of P_{cut}, wirelengths of W and X need to be adjusted by detoured paths such that their total wirelengths can be the same as Y and Z. This adjusting makes net W have a U-turn shape path turning at U_W, and net X turns at U_X as well. U_W and U_X are called the turning points. Turning points are used to adjust net length when paths go from track 1 to track 2. Now, starting from M_{WX} and M_{YZ}, the merged (or grouped) nets WX and YZ are merged at P_{cut} with all four nets having equal length.

In general cases, this hierarchical merging and length adjusting is obtained through steps, as shown in Fig. 15. A binary merge tree is first constructed to determine the merging pair of nets, and the number of merging points is then decided by a tree height. If the number of nets in a bus, N, equals 2^k, a k-level tree is constructed with each leaf node representing a net. Otherwise, a $\lceil log_2(N)\rceil$-level tree is constructed with some leaves being empty. In Fig. 15(a), the six-net bus is transformed into a three-level tree rooted

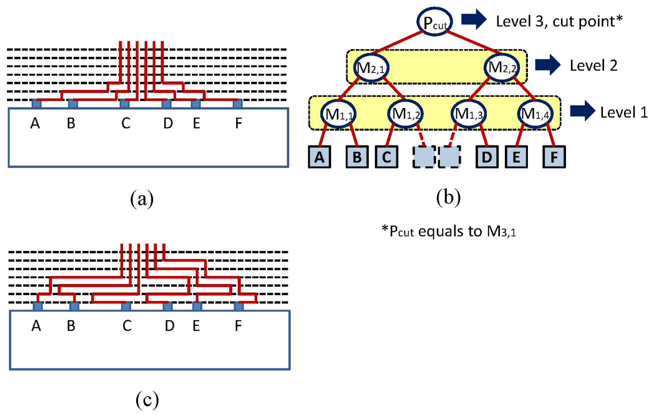


Fig. 15. (a) Routing results of the against-the-wall routing. (b) Merging tree of (a). (c) Length-matching routing results.

at P_{cut}, as shown in Fig. 15(b) with six merging points. Fig. 15(c) shows the length-matching routing results produced based on the merging tree.

We formulate the binary merge style length-matching problem into an ILP. Given pin set = $\{P_1, P_2, ..., P_N\}$ and their routing paths (generated by the against-the-wall routing), the objective of the ILP is to find the positions of merging points that satisfy length constraints while minimizing the total wirelength.

The notation used in the ILP formulation is as follows:

1) N: number of nets in a bus;
2) L: number of merging levels, $L = \lceil log_2(N) \rceil$;
3) h: number of tracks needed to merge all nets, $2^{L-1} \leq h \leq 2^L$;
4) P_{cut}: the coordinate of the cut point of the bus;
5) P_i: the coordinate of the ith pin with $P_i < P_{i+1}$, $1 \leq i \leq N$;
6) U_i: the coordinate that the ith pin goes from track 1 to track 2, $1 \leq i \leq N$;
7) $M_{lv,j}$: the coordinate of the jth merging point in level lv, $1 \leq lv \leq L$ and $1 \leq j \leq 2^{L-lv}$;
8) $len(u, v)$: the total path length from point u to point v.

Here, N, L, h, P_{cut}, and P_i are fixed numbers,[4] and U_i and $M_{lv,j}$ are variables in ILP formulation, except $M_{L,1}$, which is equal to P_{cut} and therefore it is a fixed number.

To make the formula more meaningful and easy to explain, we use an example to represent the following constraints. Using Fig. 13(b) as an example, pins W, X, Y, and Z map to parameters P_1, P_2, P_3, and P_4, respectively. Then, $M_{1,1}$ represents the merging point of W and Y, and $M_{1,2}$ represents the merging point of Y and Z. In order to have the same wire length, the following path-equivalent constraints should be satisfied:

$$M_{1,1} - 2U_1 + P_1 = -(M_{1,1} - 2U_2 + P_2) \quad (1)$$
$$M_{1,2} - 2U_3 + P_3 = -(M_{1,2} - 2U_4 + P_4) \quad (2)$$
$$P_{cut} - 2M_{1,1} + 2U_1 - P_1 = -(P_{cut} - 2M_{1,2} + 2U_3 - P_3). \quad (3)$$

Equation (1) ensures that paths from P_1 and P_2 to merge point M_1 have equal length, and (2) ensures that paths from P_3 and P_4 to M_2 have equal length. Equation (3) ensures that all paths from pins to cut points have the same length.

[4] P_{cut} will be set to movable variable when the proposed ILP fails to find a feasible solution; this does not happen in our testcases.

Locations of merging points and turning points have to be constrained to avoid illegal solutions, such as overlaps of routes produced by ILP. These position constraints restrict values of M_i and U_i in a certain range

$$U_i < U_{i+1}\ ,\ 1 \leq i < N \quad (4)$$
$$P_{i-1} < U_i < P_{i+1}\ ,\ 1 < i < N \quad (5)$$
$$M_{lv,j} < M_{lv,j+1}\ ,\ 1 \leq j \leq 2^{L-lv}.$$

Inequalities (4) and (5) ensure feasible locations of turning points. Inequality (6) ensures feasible locations of merging points.

Therefore, the net or bus adjusting problem can be formulated as follows:

$$\text{minimize} \sum\nolimits_{i=1}^{N} len(P_i, P_{cut})$$

subject to

path-equivalent constraints
position constraints.

The objective function is to minimize the total wire length. However, minimizing total wire length is equivalent to minimizing length of an arbitrary path. Without loss of generality, we simplify the objective function to minimizing the path length of P_1 to P_{cut}.

For a bus of N nets, there are $2(N - 1)$ variables and $N-1$ constraint equations when there is no obstacles or blockage inside the adjust region. To handle obstacles, additional constraints are needed to ensure that the result is available. For example, if a small device locates near the component, resulting track 1 unrouted from coordinates u to v. Then, for each U_i, an extra constraint will be added to the ILP formula

$$U_i < u, \text{ if } P_i < u$$
$$U_i > v, \text{ if } P_i < v. \quad (6)$$

The inequalities in (6) are the obstacle-avoid constraints. The ILP formulation thus becomes

$$\text{minimize } len(P_1, P_{cut})$$

subject to

path-equivalent constraints
position constraints
obstacle-avoid constraints.

Reference [14] also proposed a length-matching algorithm on single-layer routing, which finds an optimal solution under a maximum-length constraint without obstacles inside the routing region. The proposed ILP approach, on the other hand, deals with obstacles when adjusting net length. Net shape is also taken into account for signal integrity issues, which improves the quality of the routing solution.

However, the proposed length matching may fail to find a solution even though there exists one due to its binary

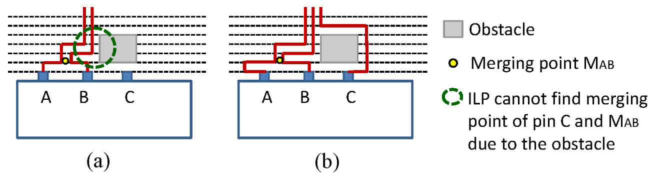

A
B
C
(a)
(b)
Obstacle
Merging point M_AB
ILP cannot find merging point of pin C and M_AB due to the obstacle

Fig. 16. Illustration of a situation in which the proposed ILP fails to find a solution. (a) ILP fails to find a valid position for a merging point of pin C and M_{AB} due to the obstacle. (b) Length matched solution produced by hand.

merging constitution. The ILP may fail when: 1) the locations of obstacles occupy part of the adjust region such that no available space for merging points, and 2) there does not exist enough space between pins such that ILP cannot satisfy position constraints. Fig. 16 shows the situation that ILP fail to find the location for merging points of C and M_{AB}, while there exists a feasible solution as shown in Fig. 16(b).

B. ILP Formulation Under Mismatch on Backbone of Bus

The aforementioned ILP constraints are based on a premise that the length differences between nets within a bus are small and thus can be ignored. However, there are small devices such as capacitors, resistors, and other components on boards that may block routing area. Although the proposed against-the-wall routing can handle the above situations and produce feasible results, the length differences between nets may become too large to ignore. In this case, the proposed ILP formulation can still maintain the routing quality with only a small modification by adding a constant number to the corresponding path-equivalent constraint. For example, in Fig. 13(b), if net W is one unit longer than net X outside the adjust region, (1) will become

$$M_1 - 2U_1 + P_1 = -(M_1 - 2U_2 + P_2) + 1. \qquad (7)$$

The additional +1 term compensates the original length difference between W and X. In addition, the length differences caused by against-the-wall routing can be compensated for by the two adjust regions of the bus (around the two components the bus connects), which means the length differences are shared out of the two adjust regions. As a result, the length matching inside each adjust region will produce little detours.

Solving an ILP problem is often time consuming. To accelerate the execution time, we solve the problem by an open source LP solver lp_solve_55, and round up the outputs to the nearest integer. Although the round-up process may produce length errors between nets, it is tolerable if all net lengths conform to the min–max length constraints. The min–max length constraints are easy to meet because the length error produced by roundup is relatively small compared to the scale of length bounds.

V. ROUTING RESULTS

We implemented our methodology in C++ and the platform is on Intel(R) Xeon(R) Linux workstation 2.5 GHz with 16-GB memory. We constructed the test cases that are based on cases shown in recent works and references (test cases I and II resemble the cases shown in [20] since the cases in [20] are not available, test cases III–V are from [1], provided from design houses). Table I shows the test cases and routing efficiency for our router. *Nets* and *Comp.* indicate the number of nets and components in test cases. *Layers* reports the number of routing layers used in test case. *Grid size* shows the size (the number of grid cells on the PCB board)[5] of the routing problem, *Total Time* includes the time for building the routing grid and the runtime spent by our router. All cases are finished within 1 s.

Fig. 17 shows the test case IV that has eight components. The results show that almost all nets are routed in a certain region and the routing area is better utilized. Fig. 18 is the enlarged left-top view of Fig. 17. The result shows that our router can adjust net paths in a certain region to meet the length constraints. Test case VI is modified from test case IV, where some pin locations are alternated such that it is not routable on one layer. Fig. 19 shows routing results of test case VI, which uses two routing layers.

Fig. 20 shows test case III that has five components on boards. The net distribution between each component is similar to the result of bus routing. Our router can also detour the nets to avoid crossing and confirm with the length constraints. Even when the components are not face to face, our router can

TABLE I
SUMMARY OF TEST CASES AND EXPERIMENTAL RESULTS

	#Nets	#Comp.	#Layers	Grid Size	Total Time
Test Case I	17	2	1	250 × 250	<1 s
Test Case II	17	4	1	300 × 230	<1 s
Test Case III	57	5	1	300 × 230	<1 s
Test Case IV	81	8	1	640 × 560	1 s
Test Case V	104	9	1	900 × 900	1 s
Test Case VI	81	8	2	640 × 560	1 s

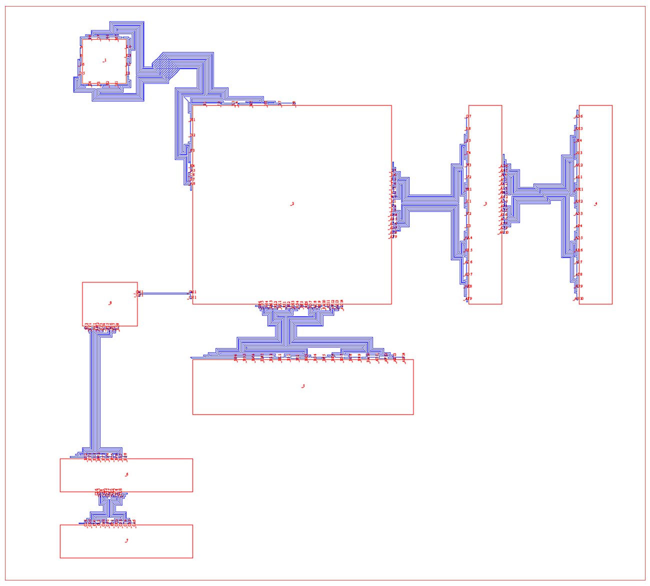


Fig. 17. Routing result of test case IV. The result shows that the routing nets are centralized as buses and the routing area is better utilized.

[5]The grid sizes of the test cases in this paper are larger than the BSG grid sizes in [20], which shows the granularity of our approach.

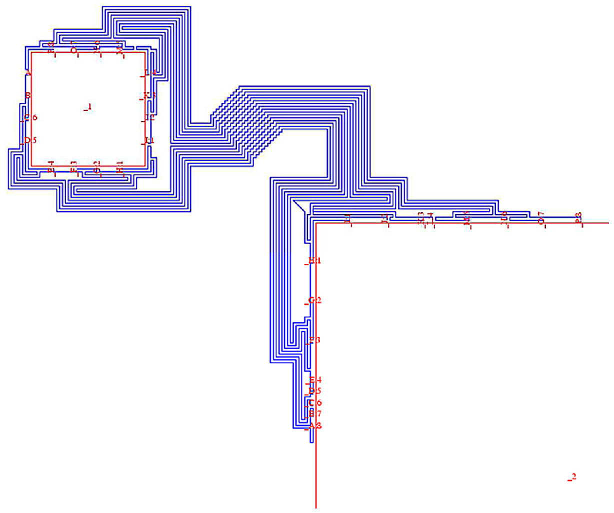


Fig. 18. Enlarged view of length adjusting result of test case IV.

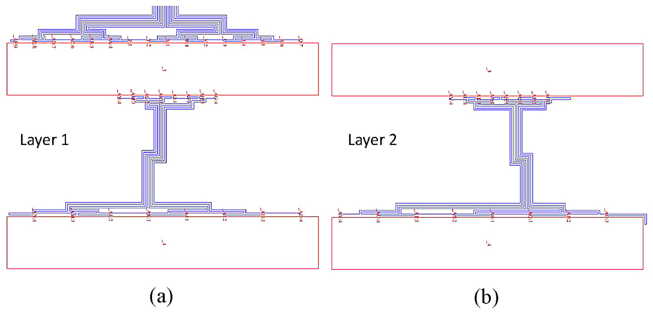


Fig. 19. Enlarged view of test case VI. Two layers are used to resolve net crossings. (a) Layer 1. (b) Layer 2.

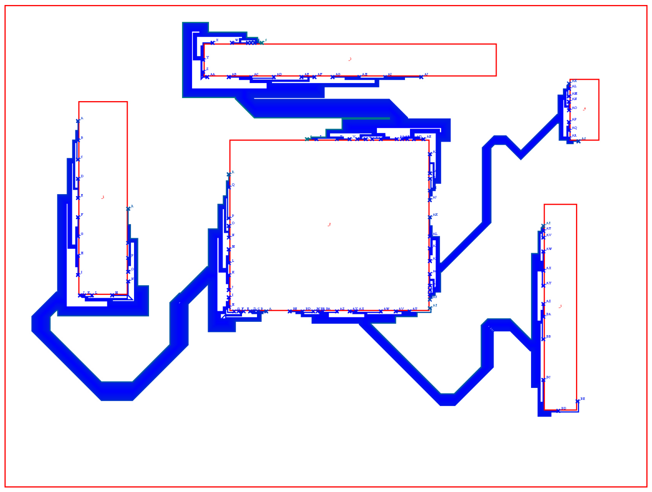


Fig. 20. Routing result of test case III.

still find a good solution. Fig. 21 (test case IV with obstacles) shows that our approach is effective in handling obstacles and the length constraints can still be confirmed.

VI. Conclusion

In this paper, we proposed an efficient obstacle-aware PCB router that can better utilize the routing resource and consider the length-matching issue. The static net ordering using DPS and Supowit's algorithm generates a good routing order in a topological routing step. The against-the-wall routing was then applied to obtain the actual routing path of nets with a resembled shape. The ILP length adjusting approach is used to meet the min–max length constraints. Our methodology not only overcomes the PCB area-routing problem effectively, but also provides a prediction of routing feasibility. Unlike the conventional router, which applies the shortest path concept, our router can better utilize the routing space and consider the wire length and shape requirements. The experimental results showed that we can preserve routing space in sequential planar routing under the given boundary pin assignment and conform to the length constraints even when many obstacles occupy the routing area.

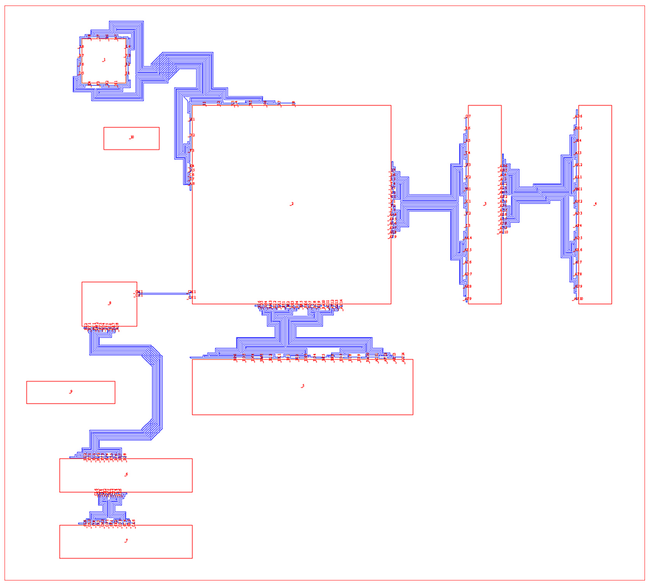


Fig. 21. Routing result of test case IV with obstacles. The length constraints are conformed.

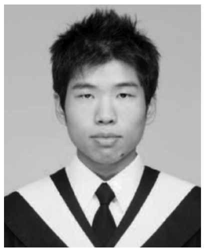
References

[1] Taiwan MOE IC/CAD Contest, 2009. [Online]. Available: http://cad_contest.cs.nctu.edu.tw/cad11/
[2] J. Cong, M. Hossain, and N. A. Sherwani, "A provably good multilayer topological planar routing algorithm in IC layout designs," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 12, no. 1, pp. 70–78, Jan. 1993.
[3] W.-M. Dai, T. Dayan, and D. Staepelaere, "Topological routing in SURF: Generating a rubber-band sketch," in *Proc. ACM/IEEE Design Autom. Conf.*, Jun. 1991, pp. 39–44.
[4] Z. Dong and M. Li, "A routing method of ad hoc networks based on A-star algorithm," in *Proc. Int. Conf. Networks Security Wireless Commun. Trusted Comput.*, 2009, pp. 623–626.
[5] H. Kong, T. Yan, and D.-F. Wong, "Automatic bus planner for dense PCBs," in *Proc. ACM/IEEE Design Autom. Conf.*, Jul. 2009, pp. 326–331.
[6] H. Kong, T. Yan, and D.-F. Wong, "Optimal simultaneous pin assignment and escape routing for dense PCBs," in *Proc. IEEE Asia South Pacific Design Autom. Conf.*, Jan. 2010, pp. 275–280.
[7] Y. Kubo, H. Miyashita, Y. Kajitani, and K. Tateishi, "Equidistance routing in high-speed VLSI layout design," in *Proc. Great Lakes Symp. VLSI*, 2005, pp. 439–449.
[8] P. Lester. (2005, Jul.). *A* Pathfinding for Beginners* [Online]. Available: (http://www.gamedev.net/reference/articles/article2003.asp)
[9] L. Luo and M. Wong. "Ordered escape routing based on Boolean satisfiability," in *Proc. IEEE Asia South Pacific Design Autom. Conf.*, Jan. 2008, pp. 244–249.

[10] L. Luo, T. Yan, Q. Ma, D.-F. Wong, and T. Shibuya, "B-Escape: A simultaneous escape routing algorithm based on boundary routing," in *Proc. ACM Int. Symp. Phys. Design*, 2010, pp. 19–25.
[11] Q. Ma, T. Yan, and M. Wong. "A negotiated congestion based router for simultaneous escape routing," in *Proc. Int. Symp. Quality Electron. Design*, 2010, pp. 606–610.
[12] D. Maier, "The complexity of some problems on subsequences and supersequences," *J. Assocmtton Comput. Mach.*, vol. 25, pp. 322–336, Apr. 1978.
[13] M.-M. Ozdal and D.-F. Wong. "Simultaneous escape routing and layer assignment for dense PCBs," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design*, Nov. 2004, pp. 822–829.
[14] M.-M. Ozdal and D.-F. Wong. "Algorithmic study of single-layer bus routing for high-speed boards," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 25, no. 3, pp. 490–503, Mar. 2006.
[15] M.-M. Ozdal and D.-F. Wong. "Two-layer bus routing for high-speed printed circuit boards," *ACM Trans. Design Autom. Electron. Syst.*, vol. 11, pp. 213–227, Jan. 2006.
[16] M.-M. Ozdal, D.-F. Wong, and P.-S. Honsinger, "An escape routing framework for dense boards with high-speed design constraints," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design*, Nov. 2005, pp. 759–766.
[17] M.-M. Ozdal, D.-F. Wong, and P.-S. Honsinger, "Simultaneous escape-routing algorithms for via minimization of high-speed boards," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 27, no. 1, pp. 84–95, Jan. 2008.
[18] K. J. Supowit, "Finding a maximum planar subset of a set of nets in a channel," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 6, no. 1, pp. 93–94, Jan. 1987.
[19] T.-Y. Tsai, R.-J. Lee, C.-Y. Chin, C.-Y. Kuan, H.-M. Chen, and Y. Kajitani, "On routing fixed escaped boundary pins for high speed boards," in *Proc. Design Autom. Test Eur.*, 2011, pp. 1–6.
[20] T. Yan and D.-F. Wong, "BSG-route: A length-matching router for general topology," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design*, Nov. 2008, pp. 499–505.
[21] T. Yan and D.-F. Wong, "A correct network flow model for escape routing," in *Proc. ACM/IEEE Design Autom. Conf.*, Jul. 2009, pp. 332–335.
[22] M.-F. Yu and W.-M. Dai. "Single-layer fanout routing and routability analysis for ball grid arrays," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design*, Nov. 1995, pp. 581–586.

Ching-Yu Chin received the B.S. degree in electronics engineering from National Chiao Tung University, Hsinchu, Taiwan, in 2008, where she is currently pursuing the Ph.D. degree with the Institute of Electronics Engineering.

Her current research interests include physical design automation, on-chip and off-chip routing algorithms, and very large scale integration testing.



Chung-Yi Kuan received the B.S. and M.S. degrees in electronics engineering from National Chiao Tung University, Hsinchu, Taiwan, in 2009 and 2011, respectively.

He is currently an Engineer with ASUSTek Computer, Inc., Taiwan. His current research interests include board-level routing and beyond-die integration.

Tsung-Ying Tsai received the B.S. and M.S. degrees in electronics engineering from National Chiao Tung University, Hsinchu, Taiwan, in 2007 and 2009, respectively.

He is currently an Engineer with Global Unichip Corporation, Hsinchu. His current research interests include physical design automation and board-level routing.

Hung-Ming Chen received the B.S. degree in computer science and information engineering from National Chiao Tung University, Hsinchu, Taiwan, in 1993, and the M.S. and Ph.D. degrees in computer science from the University of Texas, Austin, in 1998 and 2003, respectively.

He is currently an Associate Professor with the Department of Electronics Engineering, National Chiao Tung University. His current research interests include physical design automation in digital and analog circuits, beyond-die integration (off-chip electronic design automation), and 3-D integrated circuit design methodology.

Dr. Chen has served as a Technical Program Committee Member on ACM/IEEE ASP-DAC, IEEE SOCC, VLSI-DAT, and ACM ISPD.

Yoji Kajitani (F'92) received the Ph.D. in electrical engineering from the Tokyo Institute of Technology, Tokyo, Japan, in 1969.

He has been a Senior Professor with the Japan Advanced Institute of Science and Technology, Kanazawa, Japan, since April 2012 after his retirement as a Professor from the University of Kitakyushu, Kitakyushu, Japan, and the Tokyo Institute of Technology, Tokyo, Japan. He was a Guest Professor with the Department of Electronics Engineering, National Chiao Tung University, Hsinchu, Taiwan, from November 2011 to January 2012. He started his academic career on graph theory for electrical networks. One of his major contributions involved finding the innate structure of the graph by the name of principal partition, which is based on edge density over nodes. With this concept, he has solved several longstanding problems, including minimization of the variables in an electrical network and Shannon switching game. Since the 1980s, his research interests have included placement and routing. Major contributions are in formulation of via-number minimization problem and invention of placement code named BSG and its equivalent sequence pair (SP). The paper on SP was listed as one of the best papers by the IEEE/ACM International Conference on Computer-Aided Design (ICCAD) in its history of 20 years.

Dr. Kajitani received the IEEE 2009 Circuits and Systems Society Technical Achievement Award, the IEEE 1999 CAS Golden Jubilee Medal, and the Best Paper Award three times from the Institute of Electrical, Information, and Computer Engineers of Japan, in 1982, 1985, and 1988. He received the IEEE fellowship for his contribution to the development of graph theory and its applications in 1992.