IPSJ Transactions on System LSI Design Methodology Vol.8 75–84 (Aug. 2015)

[DOI: 10.2197/ipsjtsldm.8.75]

Regular Paper

Layer Assignment and Equal-length Routing for Disordered Pins in PCB Design

Ran Zhang[1,a) Tieyuan Pan[1,b) Li Zhu[1,c) Takahiro Watanabe[1,d)

Received: December 2, 2014, Revised: March 12, 2015,
Accepted: April 29, 2015, Released: August 1, 2015

Abstract: In recent printed circuit board (PCB) design, due to the high density of integration, the signal propagation delay or skew has become an important factor for a circuit performance. As the routing delay is proportional to the wire length, the controllability of the wire length is usually focused on. In this research, a heuristic algorithm to get equal-length routing for disordered pins in PCB design is proposed. The approach initially checks the longest common subsequence of source and target pin sets to assign layers for pins. Single commodity flow is then carried out to generate the base routes. Finally, considering target length requirement and available routing region, R-flip and C-flip are adopted to adjust the wire length. The experimental results show that the proposed method is able to obtain the routes with better wire length balance and smaller worst length error in reasonable CPU times.

Keywords: PCB routing, equal-length routing, single commodity flow, EDA

1. Introduction

In recent PCB design, the routing is still achieved manually to meet the high performance. As integrated circuit technology advances rapidly, the dimensions of packages and PCBs are reduced while the pin counts and routing layers keep increasing [1]. Due to the high density of integration, the signal propagation delay or skew has become an important factor for a circuit performance. In addition, in PCB, a lot of cells are required to receive the signal at the same time point. Hence, the signal propagation delay and skew have been taken into consideration in the PCB routing design [2], [3]. For one net, the signal propagation delay includes the routing delay and the gate delay, and is decided by lots of parameters. As the gate delay is often fixed in the PCB design, we can control the signal propagation delay by adjusting the routing delay. As the routing delay is proportional to the wire length, the controllability of the wire length is usually focused on. If the routing area is large enough, it is not different to control the wire length of the net. However, the routing area is usually limited and multi-nets should be considered in the dense area. Hence, how to balance the wire length of the multi-nets becomes a very important problem, which is formulated as equal-length routing problem in PCB design.

A modern PCB usually hosts several chip packages whose footprints are pin arrays which are expected to be routed by non-crossing nets [1]. There are two important problems called escape routing and river routing in PCB, as shown in **Fig. 1**. Escape routing is to route from the pins inside the pin arrays to the boundary of the arrays, like helping the pins "escape" the pin array. River routing is to connect the escaped routes between the pin arrays with length constraints. Escape routing and river routing have their different tasks. The major task of escape routing is to escape a set of pins using as few layers as possible because it usually dominates the number of layers. On the other hand, river routing's major task is to connect pin pairs to meet the length constraints while maintaining the planar topology generated by the escape routing.

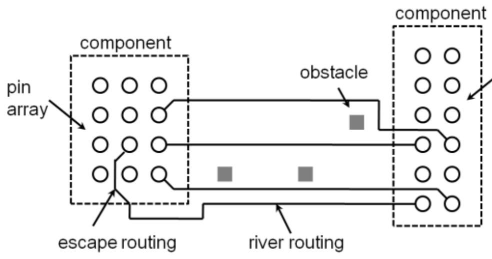
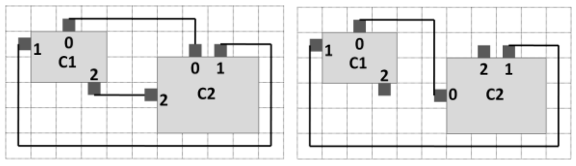


Fig. 1 Illustration of PCB routing problems [2].



(a) Ordered pins (b) Disordered pins

Fig. 2 Order of pins.

For river routing problems, in general the positions of pins are fixed on the component before routing starts. If the order of source pins around a component is reverse of the order of target pins around another component, they are called 'ordered' (**Fig. 2** (a)); otherwise, called 'disordered' (Fig. 2 (b)). Assuming

1 Graduate School of Information, Production and Systems, Waseda University, Kitakyushu, Fukuoka 808–0135, Japan
a) zhangran@toki.waseda.jp
b) tieyuan_pan@fuji.waseda.jp
c) zhuli2543@ruri.waseda.jp
d) watt@waseda.jp

routing area is large enough, if the source and target pins are ordered, the routing can be completed in single layer without crossing. However, as usually the source and target pins are disordered, some inevitable crossing cannot be solved in single layer, as the example of net 2 in Fig. 2 (b). Therefore, multi-layers are adopted for routing disordered pins, and a practical problem that how to assign layers for these pins and in what order to route them needs to be solved.

In this research, we consider the multi-layer equal-length routing problem for disordered pins in PCB. The objective of this problem is to minimize the wire length skew between obtained routes and reduce the worst length error. In other words, we aim to get a better wire length balance. The whole design process is composed of three phases. In the first phase, we assign layers for pins by checking the longest common subsequence (LCS) between source and target pins. In the second phase, single commodity flow is used to generate the base routes and the components are merged. This routing is carried out for multi-nets simultaneously. Finally, considering the equal target length requirement and available routing region, R-flip and C-flip [4], [5] are employed to adjust the wire length. The experimental results show that the proposed method is able to obtain the routes with better wire length balance and smaller worst length error in reasonable CPU times.

The remainder of this paper is organized as follows: Section 2 describes some previous works related to this research. Section 3 describes the problem definition of this work. Section 4 details the three phases of proposed routing algorithm. Section 5 illustrates the experimental results and analysis. Finally, the conclusion is given in Section 6.

2. Related Works

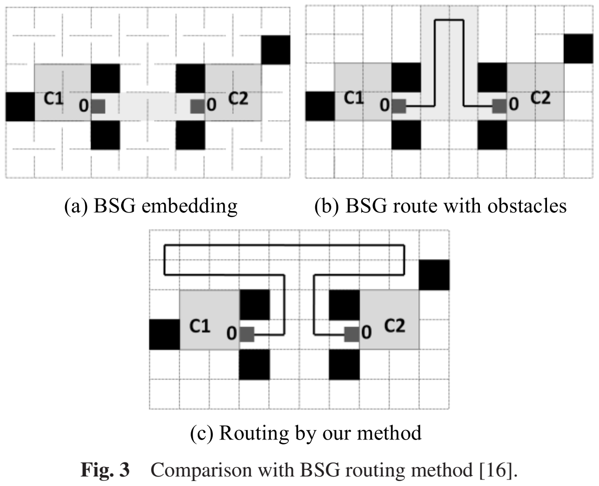
Some researches for river routing problem have been proposed. Reference [6] proposed an automatic bus planner for dense PCBs. In Refs. [7] and [8], a Lagrangian-relaxation framework was used to allocate routing resources during routing to control the length of each net. In Refs. [9] and [10], a river routing based algorithm was proposed to detour the net inside its bounded area. The length matching routing inside a channel was considered in Ref. [11], which used symmetric-slant grid interconnect to transform the length matching problem into a general grid routing problem. In Ref. [12], a length matching routing method was presented with no restriction on routing topology using bounded slice-line grid [13]. However, these works mentioned above do not consider the obstacles in routing area.

In fact, there are several obstacles in PCB, such as device and IC package, etc. Thus, consideration of obstacles is important in PCB design. For obstacle-aware routing problems, Ref. [14] explored a length matching routing method based on region partition. A transactional parallel routing algorithm was studied in Ref. [15]. In Refs. [4] and [5], an obstacle-aware routing algorithm was proposed to expand the wave-front of all nets to obtain routes with target wire lengths. However, they are adopted in single layer routing and do not work well in the case of disordered pins.

In Ref. [16], a length matching routing method was presented with no restriction on routing topology using bounded slice-line grid (BSG) in multi-layer. This BSG routing method firstly embeds the given topology onto a BSG, and then sizes the cells to make the total area of the cells occupied by a net satisfying its target length. When in the routing area there is no obstacle, BSG routing method is able to achieve the target length by sizing cells and performing detail routing inside each cell to turn the assigned area into the expected length. However, if obstacles exist, the target length may not be achieved, because the embedded topology onto the BSG dominates the available wire length, where not only obstacles but also other nets would impact on the sizing of the BSG cells. For example in **Fig. 3** (a), an input topology is embedded onto BSG with obstacles. In this case, the longest wire generated by BSG routing method is shown in Fig. 3 (b), which cannot make full use of routing space. Hence, achieving the target length seems difficult when obstacles exist, and this defect is also discussed in Ref. [5].

A practical approach to solve the fixed disordered pins routing problem was discussed in Refs. [17], [18]. However, this work used a greedy way to assign layers for pins and merge all multi-components at the same time, which was not efficient. Moreover it didn't consider the wire length balance between nets in dense routing problems. For example, in **Fig. 4** (a), given a target length as 19, for the periphery nets, net 0 and net 1, there are enough area for detouring to achieve the target length. But for the inner net, net 2, whose routing area is insufficient, it only reaches length 13. It leads to a worst length error as 4 and unbalanced nets routing result.

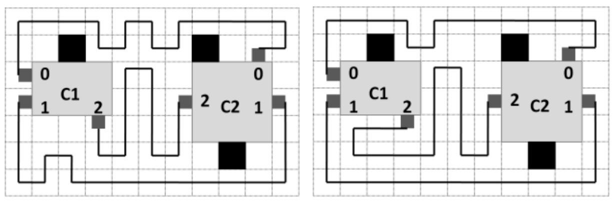
In this paper, we focus on the routing for disordered pins in dense routing problems, where the target length requirement and available routing region are taken into consideration. Compared with BSG routing method, our method can take advantage of effi-



(a) BSG embedding (b) BSG route with obstacles

(c) Routing by our method

Fig. 3 Comparison with BSG routing method [16].



(a) Unbalanced routing (b) Balanced routing

Fig. 4 Balance wire length of nets.

ciently using routing area to achieve a longer target length when obstacles exist, as shown in Fig. 3 (c). Moreover, we balance the nets by revising the adjusted wire length. The proposed method can generate routes with better wire length balance as shown in Fig. 4 (b), all the nets are with the same length 17 and the worst length error is reduced to 2. In Ref. [19], we already discussed the length matching routing for disordered pins in dense routing problems. However, the method for pin sets selection and wire length adjustment were not so efficient. In this paper, the algorithm for pin sets selection is improved, and the standard deviation of all nets' wire lengths is used to revise the adjusted wires. In addition, more experimental data are tested to further validate the proposed routing method. The following sections discuss the proposed algorithm in detail.

3. Problem Definition

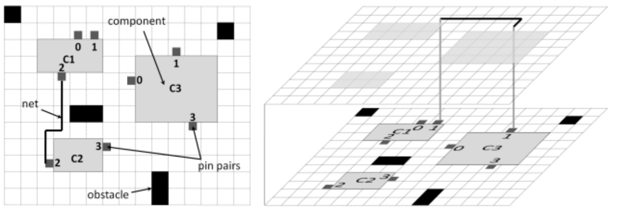
In this paper, the multi-layer equal-length routing problem is defined as follows: the input includes a grid graph $G(V, E)$, pins on each component, obstacles, and target length; it outputs the routes of pin pairs. The objective is to effectively assign layers for the disordered pins and generate routes with a better wire length balance of all the nets. In this paper, standard deviation is used to evaluate the routing results, which shows how much dispersion exists from the expected value (the value of average length of each net). It is defined as Eq. (1), where $x_1, \ldots, x_N$ are the values of sample items, $\overline{x}$ is the average value of $x_1, \ldots, x_N$, and N stands for the size of the sample.

$$S_N = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \overline{x})^2} \tag{1}$$

We give an example in **Fig. 5** (a), where the routing area is defined by routing grids. The white and black grids stand for the available routing resource and obstacles, respectively. The gray grids represent the components with pins on their boundaries. Let $C_1, C_2, \ldots, C_n$ be n components and P_i be a set of pins on $C_i (i = 1, 2, \ldots, n)$, called "pin set" of this component. The same labeled elements in different sets should be connected, called pin pairs. As illustrated in Fig. 5, there are three sets of pins, $P_1 = \{0, 1, 2\}$, $P_2 = \{2, 3\}$, $P_3 = \{0, 1, 3\}$. A net consists of a sequence of grids, and the wire length is defined as the number of grids used in the path.

Basically, the proposed method proceeds routing in single-layer. However, if net crossing is unavoidable, another layer should be used. In such a multi-layer model, to simplify the problem, the components are mapped to the added layer as obstacles and the impact of via included in the route is not considered in the wire length (Fig. 5 (b)). In this research, at most three layers are permitted, as adding layers without limitation is not much sense.

In this research, trunk routing problem between two components is dealt with. Trunk routing problem is introduced in Refs. [4] and [5], which is a sub-problem of river routing problem, The trunk routing topology condition is defined as follows: (1) all the pins are put on the boundary of the routing area; (2) the boundary pins sequence can be divided into the source pins sequence and the target pins sequence, where source pins sequence is in the reverse order of target pins sequence and vice versa. In



(a) Single layer (b) Multi-layers

Fig. 5 Routing model.

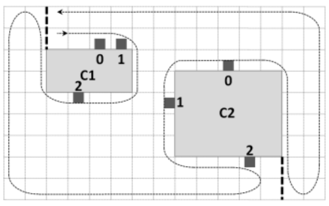


Fig. 6 Example of trunk routing topology.

this research, all the pins are on the boundaries, but the pins sequence do not satisfy the above-mentioned topology condition. Therefore virtual boundaries are introduced to solve this problem. For example in **Fig. 6**, after adding the virtual boundaries, represented by dotted lines, source pins sequence is $0 \rightarrow 1 \rightarrow 2$, and target pins sequence is $2 \rightarrow 1 \rightarrow 0$, which satisfy the trunk routing topology condition.

4. Proposed Routing Algorithm

In this research, since multi-components are given, the routing for multiple pin sets is considered. As mentioned in Section 2, Refs. [17] and [18] introduced a method to solve the fixed disordered pins routing problem, which routes nets as many as possible in a current layer, and then routes crossing nets in added layers to release the crossings. In this paper, we adopt the similar idea but implement in different way. Instead of dealing with the routing among all the components at the same time, our basic idea is to firstly handle the routing problem between two components and merge them as a new one. Then this process is repeated until all the pins are handled. It is easier to implement the crossings minimization between two components than that among all the components by adopting LCS (longest common subsequence) algorithm [20]. Another difference is that, in the initial routing phase of Refs. [17] and [18], against-the-wall routing method is adopted, which may generate some long wires, and it increases the workload for wire length adjustment. In our method, single commodity flow is used for initial routing, and it makes the initial routing result easier for further adjustment. The proposed routing algorithm includes three phases: pin sets selection and layer assignment, initial routing, and wire length adjustment. The flow chart of the whole routing process is shown in **Fig. 7**. Algorithms 1 to 4 are described in detail in the following sections.

4.1 Pin Sets Selection and Layer Assignment

Given the placement of components and disordered pins, initially we assign layers for pins in this phase. Note that, if there are two or more than two parts of components not related with each other, in other words, there are no nets to be routed between them, they are considered as two or more than two sub-problems.

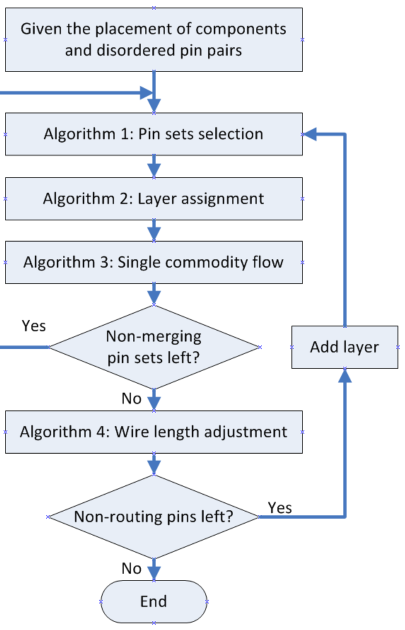

Given the placement of components and disordered pin pairs
Algorithm 1: Pin sets selection
Algorithm 2: Layer assignment
Algorithm 3: Single commodity flow
Non-merging pin sets left?
Yes
No
Add layer
Algorithm 4: Wire length adjustment
Non-routing pins left?
Yes
No
End

Fig. 7 Flow chart of routing process.

The layer assignment is processed by the following two steps:

Step 1: Select two pin sets;

Step 2: Find longest common subsequence between two sets to determine layer for pins.

[Step 1] Initially, if there are more than two pin sets in one sub-problem, these sets should be handled one by one. Hence, in Step 1 we need to select which two pin sets to be handled. To make full use of the available routing area, we need to assign the pins as much as possible in current layer. Hence the two pin sets selected to construct a new set should include pins as much as possible. However, if no nets to be routed between the selected two pin sets, it is impossible to merge them by routing. Hence, the selected two pin sets should have common elements. The pseudo-code of this process is shown in Algorithm 1.

In Algorithm 1, $P_1, P_2, \ldots, P_n$ are the pin sets of components. The two pin sets to be handled are noted as P_s and P_t. $P_i \bigcap P_s$ means the intersection set of P_i and P_s.

If there are more than two pin sets, to construct a new set including more pins, we need to find $P_s \cup P_t$ includes most elements in any two P_i, under the condition of P_s and P_t having common elements. $P_s \cup P_t$ means the union set of P_s and P_t. Firstly compare the number of elements in each P_i, and the largest set is selected as P_s. Then, for other sets, if their intersection set with P_s is not empty, the set whose $P_i - P_i \bigcap P_s$ includes most elements is selected as P_t. When comparing the elements number of pin sets, if there are more than one set have most elements, the smaller label set is chosen. If there are only two pin sets, they are noted as P_s and P_t.

Take the example in **Fig. 8** to explain Algorithm 1. As there are three pin sets, we need to select which two to be first handled. Each set is as follows: $P_1 = \{0, 1, 2, 3, 4, 5\}$, $P_2 = \{0, 6, 7\}$, and $P_3 = \{1, 2, 3, 4, 5, 6, 7\}$, and they are related with each other. We calculate the quantity of elements of each component set: P_1 is 6, P_2 is 3, and P_3 is 7. According to Algorithm 1, P_3 is determined as P_s. Then by calculating, both $P_1 \bigcap P_s = \{1, 2, 3, 4, 5\}$

Algorithm 1. Pin sets selection

```
Input: Pin sets
Output: The two sets to be handled
begin
  if $n>2$ then
    for $i=1$ to $n$ do
      calculate the quantity of elements of $P_i$;
    end for
    $P_s = P_i$ who is the largest set;
    for $i=1$ to $n$ do
      if $P_i \cap P_s$ != $\varnothing$
      $P_t = P_i$ whose $P_i - P_i \cap P_s$ is the largest set;
      end if
    end for
  else
    $P_s = P_1$;
    $P_t = P_2$;
  end if
end
```

Algorithm 2. Layer assignment

```
Input: $P_s$, $P_t$
Output: Layer assignment for pins
begin
  $Q = P_s \cap P_t$;
  $S$ = elements in $Q$ arranged in counterclockwise order of
      pins on the boundary of Ps's component;
  $T$ = elements in $Q$, arranged in clockwise order of pins on
      the boundary of Pt's component;
  $L$ = longest common subsequence of $S$ and $T$ by LCS
      algorithm;
  assign pins in $L$ to current layer;
  reserve pins in $C_Q L$ for other layers;
end
```

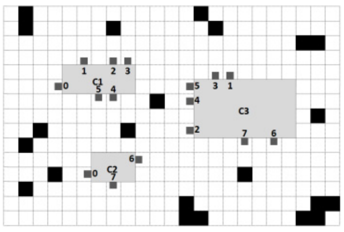


Fig. 8 Placement of components and disordered pins.

and $P_2 \bigcap P_s = \{6, 7\}$ are not empty, and the quantity of elements of $P_1 - P_1 \bigcap P_s$ is 1, $P_2 - P_2 \bigcap P_s$ is 1. Since the number of elements is the same, according to the algorithm, we choose the smaller label set P_1 as P_t.

[Step 2] Then, the longest common subsequence between two pin sets is used to determine layer for pins. The pseudo-code of this process is shown in Algorithm 2.

In Algorithm 2, Q is a set of the common elements in P_s and P_t. S is defined as an array of elements in Q arranged in counterclockwise order of pins on the boundary of P_s's component. Similarly, T is an array of elements in Q, arranged in clockwise order of pins on the boundary of P_t's component. Note that S and T have the same elements but different order. Here, the boundary of a component is defined as the passed path that starting from any point on the component, going along the edge of this component or the merged components and the periphery routed wires and ultimately returning to that point. L is an array of longest

Algorithm 3. Single commodity flow

```
Input: Pin pairs in $L$, obstacles
Output: Initial path of pin pairs
begin
  flag = 0;
  for i=1 to n do
    set virtual boundary before the i-th element of L;
    generate path between pin pairs by single commodity
    flow method;
    if routing is feasible then
      flag = 1;
      break;
    end if
  end for
  if flag = 0 then
    reserve the last pin in L to other layers;
    repeat the process until routing is feasible;
  end if
end
```

common subsequence of elements of S and T. $C_Q L$ stands for the complementary set of L in Q.

Because of the disordered pins, the longest common subsequence between two components is used to determine a layer for pins. First we store the same labeled elements of P_s and P_t in Q. In trunk routing topology, the source pins sequence should be the reverse ordering of the target pins sequence. Hence, the elements of array S is in counterclockwise order of pins on the boundary of a component having P_s, while the elements of array T is in clockwise order of pins on the boundary of a component having P_t. Note that, the first element of S and T are the same. Then we obtain the longest common subsequence of elements in S and T by LCS algorithm [20], and put the result into array L. LCS algorithm is a well-known method to find the longest common subsequence in two sequences. The reason why we find the longest common subsequence is to make full use of the available routing area in the current layer. Finally, assign pins in L to the current layer, and reserve pins in $C_Q L$ for other layers.

We also take the case in Fig. 8 to explain Algorithm 2. From the last step, we know $P_s = \{1, 2, 3, 4, 5, 6, 7\}$ and $P_t = \{0, 1, 2, 3, 4, 5\}$. According to Algorithm 2, we can obtain $Q = \{1, 2, 3, 4, 5\}$, and then $S = [1, 3, 5, 4, 2]$, $T = [1, 2, 3, 4, 5]$. By LCS algorithm, we can get the longest common subsequence is $L = [1, 3, 4]$ and then $C_Q L = \{2, 5\}$. As a result, we assign net1, net3 and net4 in layer1, and reserve net2 and net5 to layer 2.

4.2 Initial Routing by the Single Commodity Flow Method

After layer assignment for some pin pairs, single commodity flow is used to generate the path of assigned pin pairs in current layer. This routing is carried out for multi-nets simultaneously. The pseudo-code of this phase is shown in Algorithm 3.

The routing by single commodity flow method is shown in **Fig. 9**. All the available routing grids are treated as the vertices and the edges connected vertices are represented in bi-direction. The capacity of each direction is set as 1, shown in Fig. 9 (a). Augmenting paths are explored by breadth first search, shown as net1 in Fig. 9 (b). In the path, the residual capacity of directions from source to target is changed to 0. Then, repeat this process until no augmenting path exists. If it crosses with the already existing nets, the reverse flow is applied as shown in Fig. 9 (c). The edges whose both bi-directions are used need to be deleted and non-crossing nets can be generated, shown in Fig. 9 (d). In this way, we obtain the base routes of pin pairs.

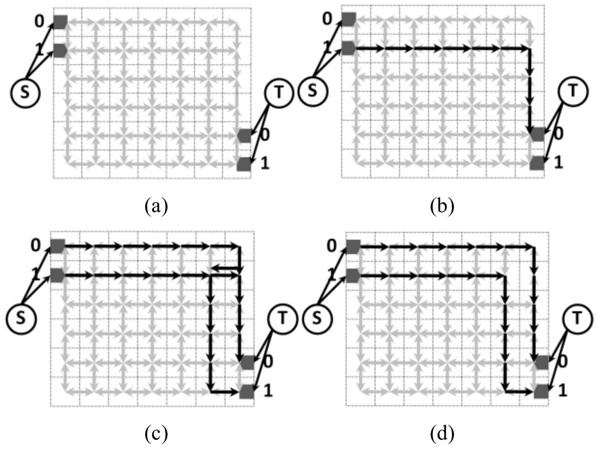


Fig. 9 Single commodity flow method.

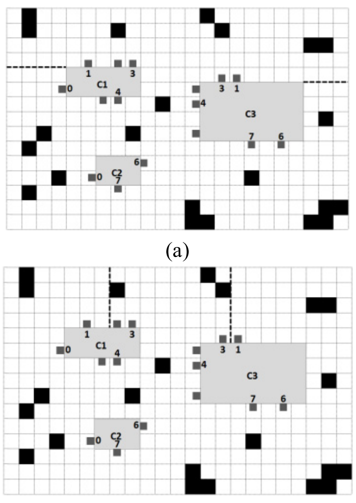


Fig. 10 Virtual boundary setting.

In addition, before the routing, virtual boundaries need to be set if the pins sequence do not satisfy trunk routing topology condition. Virtual boundaries are added as paired straight lines between the source or target components and edge of routing region, shown as dotted lines in **Fig. 10**. The function of virtual boundary is to cut off the connection between the two grids in the right and left sides of it. We set the virtual boundaries in a greedy way. Initially, the virtual boundary is set between the two pins same numbered with the first one and last element in array L. The position of virtual boundary is on the counterclockwise side from the first one to last one of source pins, and clockwise side of target pins. If the component corner exists between these two pins, the virtual boundary is set on the corner. The direction of a virtual boundary (horizontal or vertical) is decided by whose capacity is less. If there is more than one corner, the first one is chosen (Fig. 10 (a)). If between two pins no component corner exists, the virtual boundary is set in the middle of them (Fig. 10 (b)).

Then, we generate routes by a single commodity flow method. If the routing cannot be completed using current virtual boundary, we reset another virtual boundary between the two pins

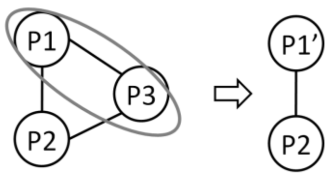


Fig. 11 Pin sets merging.

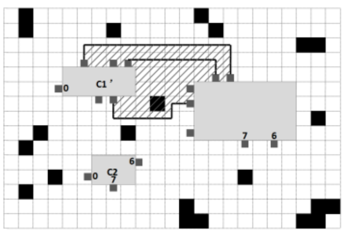


Fig. 12 Initial routing.

same numbered with the second element and first one in array L(Fig. 10 (b)), then between the third and second one, and so on, until the routing is completed. In addition, if the routing is not feasible with any virtual boundary, we should reserve the last pin in L to other layers and repeat the process until the routing between selected two pin sets P_s and P_t is completed. After the routing, the first chosen two pin sets are merged as new pin set, as shown in **Fig. 11**. For the new pin set, its component is in irregular shape, and the pins of it do not include the routed pins and reserved pins. The area surrounded by peripheral routed wires, illustrated as the shaded part in **Fig. 12**, is treated as the interior of the new component.

We repeat first two processes of layer assignment and initial routing for other pin sets until all the routing in current layer are accomplished.

Take the example of Fig. 8 again. Since $L = [1, 3, 4]$ in the last phase, we set the virtual boundary between pin pair 1 and 4, as shown in Fig. 10 (a). Then the routes are generated by single commodity flow method. After the routing between P_1 and P_3 is finished, they are then treated as a new pin sets $P_1' = \{0, 6, 7\}$. Repeat the above processes until all the pin sets are handled.

4.3 Wire Length Adjustment

Based on the generated paths of all the nets, we need to adjust the wire length of each net to satisfy the length constrains. Both the target length requirement and available routing region are considered. The pseudo-code of this phase is shown in Algorithm 4.

In Algorithm 4, l_t stands for the given target length and A means the available routing area after routing; L_i and La_i represent the current wire length and the adjusted wire length of net i respectively; α means utilized coefficient of A.

To leave space for other nets, the peripheral nets are firstly handled. An array N_i stores the nets id sorted according to position from periphery to the inner counterclockwise. The net closest to the up border of the routing area is first stored, then the net closest to the bottom border. If there is more than one net has the same vertical ordinate, we choose the net whose horizontal ordinate is smaller. Then store the second closest nets and continue this process until all the nets are completed. For the wire length

Algorithm 4. Wire length adjustment

```
Input: Initial path of pin pairs, obstacles, $l_t$, $A$
Output: Adjusted path of pin pairs
begin
  $N_i$= an array of nets id stored according to position from
  periphery to the inner counterclockwise;
  for $i=1$ to $n$ do
    reroute wire along the edge;
    $L_i$ = current length of net i;
    $La_i = [(l_t - L_i)/2]*2$;
    adjust $La_i$ units of length by R-flip or C-flip;
  end for
  calculate $S_{N0}$;
  if $S_{N0} > 1.15$ then
    for $j=10$ to $1$ do
    $\alpha = j/10$;
      if $[\alpha*A]/n < l_t$ then
        for $i=1$ to $n$ do
          $La_i = [([\alpha*A]/n - L_i)/2]*2$;
          revise wire length with $La_i$;
        end for
      calculate $S_{Nj}$;
      end if
    end for
  end if
  output the result with smallest $S_{Nj}$;
end
```

adjustment of each net, it is processed by the following two steps:

Step 1: Reroute wires along the boundary of routing area;

Step 2: Adjust length by R-flip or C-flip;

Step 3: Revise adjusted wire length.

[Step 1] Initially, we extend the source pin and target pin to the outer boundary of a routing region. If the pin is on the horizontal boundary of the component, the extend direction is horizontal, otherwise, the extend direction is vertical. Then the wires are rerouted along the boundary. This process can reserve space for the succeeding inner nets. If we directly adjust the wire length based on the initial path, there may be not enough space for other nets.

For the example in **Fig. 13**, $N_i = [1, 0, 3, 6, 4]$. So, the first net to be adjusted is net 1, and the last one is net 4. The extend wire of net 1 is shown in Fig. 13 (a).

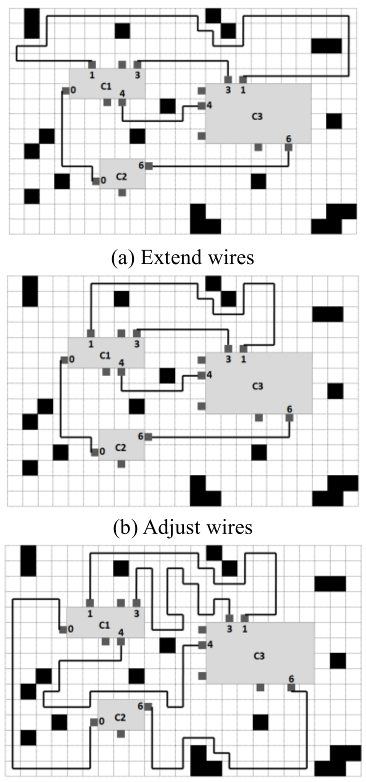
[Step 2] Then, based on the extended wires, we adjust the wire length to meet the target length using R-flip or C-flip operations [4], [5]. R-Flip detours a partial route of length two to four by searching a rectangle along the initial route from the source to target. C-Flip, a generalization of R-Flip, replaces a partial route by another route with the same terminals to increase the wire length, and vice versa to shorten length. Note that, either lengthening or shortening a wire is a first-go-then-back process, where the adjustment of wires takes even number not odd. Hence, the adjusted wire length La_i is calculated by $[(l_t - L_i)/2] * 2$. If La_i is negative, then we need to shorten the wire. Otherwise, we lengthen the wire.

For the example in Fig. 13 (a), l_t is set to 25, $L_1 = 46$. According to the definition, $La_1 = [(25 - 46)/2] * 2 = -20$. So, the wire length of net 1 is shortened by seven times R-flip and once C-flip operations. The adjustment result is shown in Fig. 13 (b). Similarly, the other nets as adjusted one by one until all the nets are completed.

[Step 3] As the aim of this research is to generate routes with a better wire length balance of all the nets, after the adjustment, we check the standard deviation S_{N0} of all nets' wire lengths according to Eq. (1), to decide whether further revise. As mentioned above, the adjustment of wires takes even number not odd, therefore sometimes one unit length error is inevitable, where length error is defined as $|L_i - l_t|$. As a result, even though all the nets are successfully adjusted, S_{N0} may be not 0, and the maximum value is 1.15. If $S_{N0} > 1.15$, we revise adjusted wire length.

In this research, for equal-length routing, a coefficient $\alpha(0.1, 0.2, \ldots, 1)$ is defined to adjust the wire length skew between nets. The coefficient α represents the utilized region, since not all the available routing grids can be used due to the position of components and obstacles. Then the adjusted wire length is revised as $La_i = [([\alpha * A]/n - L_i)/2] * 2$. Under the condition of $[\alpha * A]/n < l_t$, apply different α to get a smaller S_N.

Also for the example in Fig. 13, Fig. 13 (c) shows its adjusted routing result. By calculating, $S_{N0} = 0.84 < 1.15$, hence we do not further revise. Then similarly, the other nets as adjusted one by one until all the nets are completed. After completing the routing in Layer 1, we should check if there are any non-routing pins left. If there are, repeat the whole process above in next layer until all the pins are routed.



(a) Extend wires

(b) Adjust wires

(c) Adjustment result

Fig. 13 Wire length adjustment.

4.4 Discussion on Time Complexity

As mentioned above, the proposed routing method is divided into three phases: layer assignment, single commodity flow, and wire length adjustment. For layer assignment, based on LCS algorithm, the time complexity of this phase is O(m^2), where m is the number of the total pins for routing. Since the complexity of one net flow in a grid graph is O(n), where n is the number of grids, the time complexity of routing all nets is O(mn). In wire length adjustment, for one net the adjusted length is O(n) in the worst case. So the time complexity of modifying all nets is O(mn). Therefore, the total time complexity of the proposed routing method is O(mn).

5. Experimental Results

We implemented our proposed method in C language, which is compiled by MinGW Developer Studio 2.06, and executed on a PC with 2.66GHz Intel Core 2 CPU and 2GB RAM. Six experimental data named from Data00 to Data05 for evaluation are synthesized by referring the test cases in Ref. [17]. We narrow the range of the routing area to simulate a dense routing problem. 10% of the routing area is randomly set with obstacles. The properties of each experimental data are listed in **Table 1**, where *Grid size* is the scale of the routing problem, and *#Obstacle* denotes the number of obstacles, *#Component* is the quantity of components, *#Nets* is the number of two-pin nets, and *A* means the available routing area after initial routing. Three experiments are carried out. Experiment 1 is executed on obstacles, Experiment 2 is on the same target length but different utilized coefficient α, and Experiment 3 is on comparison with another routing method based on Ref. [17] followed by some adjustment.

For Experiment 1, Data00 is executed without utilized coefficient α. Cases without obstacles and with obstacles are considered. The experimental results are listed in **Table 2**. *Std dev*

Table 1 Properties of experiment data.

	Grid size	*#Obstacle*	*#Component*	*#Nets*	*A*
Data00	30*20	60	2	5	411
Data01	40*30	120	2	8	962
Data02	45*30	135	3	12	1015
Data03	50*40	200	4	20	1554
Data04	55*40	220	5	28	1695
Data05	60*50	300	6	37	2215

Table 2 Experiment results on different target length for Data00 (without α).

Target length	Without obstacle					With obstacles				
	Std dev	*Average length*	*Worst length error*	*CPU time*	*#Layer*	*Std dev*	*Average length*	*Worst length error*	*CPU time*	*#Layer*
$l_t = 30$	0.55	30.60	1	<1s	2	0.55	30.60	1	<1s	2
$l_t = 130$	24.18	113.80	55	<1s	2	33.32	114.60	75	<1s	2

Table 3 Experiment results on different utilized coefficient for Data00 (l_t = 130).

Utilized coefficient	*Std dev*	*Average length*	*Worst length error*	*CPU time*	*#Layer*
$\alpha = 1$	43.61	117.00	91	<1s	2
$\alpha = 0.9$	28.40	109.80	71	<1s	2
$\alpha = 0.8$	20.35	99.40	67	<1s	2
$\alpha = 0.7$	0.55	94.60	36	<1s	2
$\alpha = 0.6$	0.55	81.40	49	<1s	2
$\alpha = 0.5$	0.55	67.40	63	<1s	2
$\alpha = 0.4$	0.84	53.80	77	<1s	2
$\alpha = 0.3$	0.55	41.40	89	<1s	2
$\alpha = 0.2$	0.55	27.40	103	<1s	2
$\alpha = 0.1$	4.76	17.80	117	<1s	2

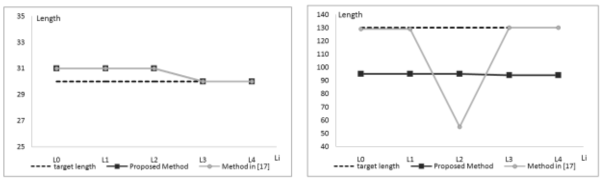
Table 4 Experiment results on comparison with another method.

		Method [17] + adjustment				Proposed method				
	Target length	*Std dev*	*Average length*	*CPU time*	*#Layer*	*Std dev*	*Average length*	α	*CPU time*	*#Layer*
Data00	$l_t = 30$	0.55	30.60	<1s	2	0.55	30.60	-	<1s	2
	$l_t = 130$	33.32	114.60	<1s	2	0.55	94.60	0.7	<1s	2
Data01	$l_t = 70$	2.17	70.88	<1s	2	0.64	69.88	-	<1s	2
	$l_t = 190$	54.50	165.63	<1s	2	0.52	152.38	0.8	<1s	2
Data02	$l_t = 70$	12.74	66.42	<1s	2	4.93	49.92	0.4	<1s	2
	$l_t = 120$	42.33	96.58	<1s	2	4.93	49.92	0.4	<1s	2
Data03	$l_t = 40$	2.34	39.70	<1s	2	1.48	33.10	0.3	<1s	2
	$l_t = 110$	39.77	83.70	<1s	2	0.83	76.80	0.7	<1s	2
Data04	$l_t = 40$	7.80	37.39	<1s	2	0.61	35.18	0.4	<1s	2
	$l_t = 80$	23.34	68.89	<1s	2	0.61	35.18	0.4	<1s	2
Data05	$l_t = 60$	15.96	52.32	<1s	3	7.42	31.73	0.3	<1s	3
	$l_t = 110$	40.02	81.46	<1s	3	7.42	31.73	0.3	<1s	3

denotes the standard deviation of the wire length, that shows how much dispersion exists from the average length of nets. The wire lengths are given in accordance with the number of unit grid. From this table, we note that, our proposed method could be applied in both no-obstacle routing and obstacle-ware routing problems. When obstacles are set, the maximum network becomes smaller, so some nets should be reassigned to Layer 2. From Table 2, we also note that, when target length is set larger, the standard deviation becomes larger, and the worst length error increases. The reason is that, as the peripheral nets are first adjusted, the inner nets do not have enough area to detour, which leads to the larger differences among the nets.

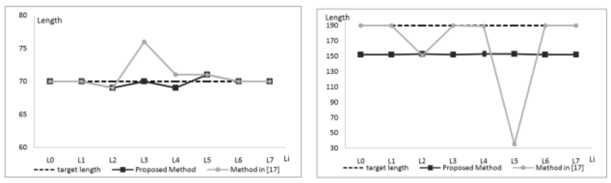
In Experiment 2, also for Data00, we change the value of α from 1 to 0.1 to analysis of the impact of α on standard deviation, while target length is not changed. From the experimental results in **Table 3**, when $\alpha = 0.7, 0.6, 0.5, 0.3, 0.2$, a minimum standard deviation can be obtained. Moreover, when $\alpha = 0.7$, the worst length error is smallest, thus α 0.7 is considered as an optimal coefficient for wire length balance.

We compare the proposed routing method with another routing method [17] in Experiment 3. Since the method [17] does not focus on the equal-length routing problem, we adopt R-flip and C-flip to adjust its routing result as well. There are six data Data00 to Data05 with different grid sizes from small to large. If we set the target length too small (less than the largest distance among pin pairs of all nets) or too large (larger than the average routing area for each net), it may make the achievement of the target length impossible for some nets. In this experiment, for each data, we only test two bound target lengths, one is a smaller target length which is set as the least common multiple of ten larger than the largest distance among pin pairs of all nets, and another is a larger target length which is set as the largest common multiple of ten less than the average routing area for each net, *A/#Nets*.



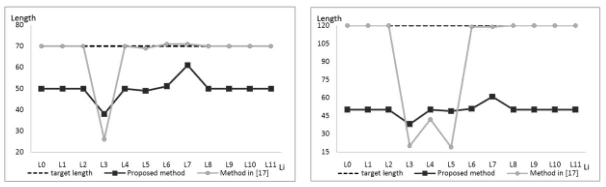
(a) $l_t = 30$ (b) $l_t = 130$

Fig. 14 Length comparison for Data00.



(a) $l_t = 70$ (b) $l_t = 190$

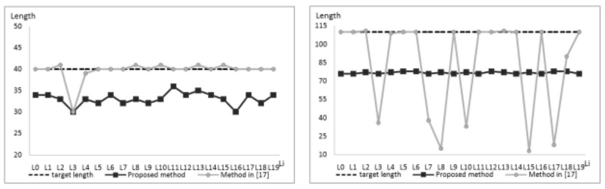
Fig. 15 Length comparison for Data01.



(a) $l_t = 70$ (b) $l_t = 120$

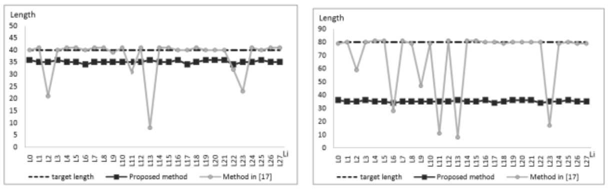
Fig. 16 Length comparison for Data02.

The experimental results are shown in **Table 4**. And the length comparison of each net is illustrated in **Fig. 14**, **Fig. 15**, **Fig. 16**, **Fig. 17**, **Fig. 18** and **Fig. 19**, where Y-axis means a resultant net length for each net mapped on X-axis, and a dotted line represents the target length. The experimental results show that, the



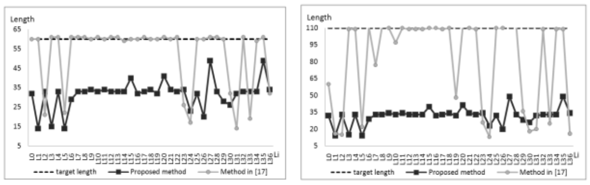
(a) $l_t = 40$ (b) $l_t = 110$

Fig. 17 Length comparison for Data03.



(a) $l_t = 40$ (b) $l_t = 80$

Fig. 18 Length comparison for Data04.



(a) $l_t = 60$ (b) $l_t = 110$

Fig. 19 Length comparison for Data05.

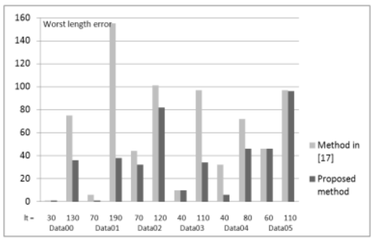
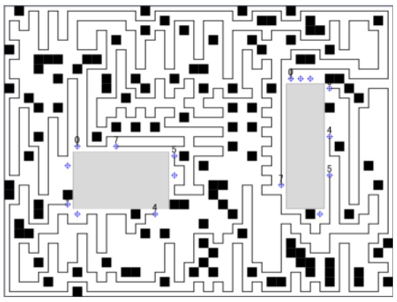


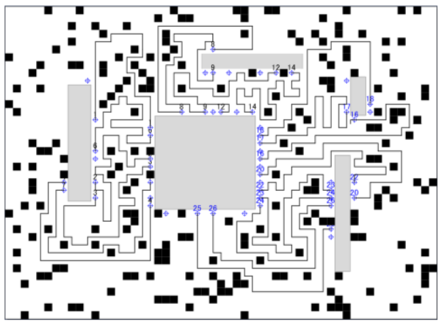
Fig. 20 Worst length error comparison.

method [17] has a smaller standard deviation for the small target length and a larger standard deviation for the large one. However, our method has the smaller standard deviation in spite of the target length. This mainly owes to the adoption of an appropriate α. Especially for the small target lengths in Data00 and Data01, we obtain an optimal result without using coefficient α. From the experimental results we also get that, for the same data with the same target length, the standard deviation obtained by our method is always less than or equal to that of the method [17], no matter the target length is small or large. As a result, our method gets a better wire length balance among the nets.

In addition, **Fig. 20** shows the worst length error comparison between the proposed method and the method [17]. From this figure we obtain that, executed no matter by the method [17] or our method, the worst length error is smaller when the target length is small, while it becomes larger when the target length is large. The reason is that, when target length is larger, some nets adjusted later do not have enough space to detour, which leads to the larger worst length error of net. For each data, the worst length error obtained by our method is always less than or equal to that by the method [17], which confirms that our method is effective in reducing worst length error. This means that the wire lengths of all nets obtained by our method are more concentrative. Though this concentration is at the expense of average length error (that is, | target length – average length |) increasing, our method is effective in getting better wire length balance, when we pay attention to the equal-length routing, which is the main objective in this study.



(a) Data01



(b) Data04

Fig. 21 Routing result (Layer 1).

Figure 21 (a) and (b) show the routing results of Data01 and Data04 in layer 1, respectively. Combining the results in Table 4 and Fig. 21, we can get that, the more intensive the pins are, the more difficult it is to get the ideal standard deviation of all the wires. It is because of the interacting between wires and the limitation of R-flip and C-flip. Hence, how to further optimize the sharing of limited routing region between nets, and how to make much fuller use of the routing space remain for our future works.

6. Conclusions

In this research, a heuristic algorithm to get equal-length routing for disordered pins in PCB design is proposed. The approach initially checks the longest common subsequence of source and target pin sets to assign layers for pins. Single commodity flow is then carried out to generate the base routes. Finally, considering target length requirement and available routing region, R-flip and C-flip are adopted to adjust the wire length. The experimental results show that the proposed method is able to obtain the routes with better wire length balance and smaller worst length error in reasonable CPU times.

References

[1] Yan, T., Ma, Q. and Wong, M.D.F.: Advances in PCB Routing, *IPSJ Trans. System LSI Design Methodology*, Vol.5, pp.14–22 (2012).

[2] Kohira, Y., Suehiro, S. and Takahashi, A.: A Fast Longer Path Algorithm for Routing Grid with Obstacles using Bi-connectivity based Length Upper Bound, *Proc. Asia and South Pacific Design Automa-*

tion Conf. 2009, pp.600–605 (2009).
[3] Kohira, Y., Suehiro, S. and Takahashi, A.: A Fast Longer Path Algorithm for Routing Grid with Obstacles using Bi-connectivity based Length Upper Bound, *IEICE Trans. Fundamentals*, Vol.E92-A, No.12, pp.2971–2978 (2009).
[4] Kohira, Y. and Takahashi, A.: CAFE Router: A Fast Connectivity Aware Multiple Nets Routing Algorithm for Routing Grid with Obstacles, *Proc. Asia and South Pacific Design Automation Conf. 2010*, pp.281–286 (2010).
[5] Kohira, Y. and Takahashi, A.: CAFE Router: A Fast Connectivity Aware Multiple Nets Routing Algorithm for Routing Grid with Obstacles, *IEICE Trans. Fundamental of Electron., Commun. and Comput.*, Vol.E93-A, No.12, pp.2380–2388 (2010).
[6] Kong, H., Yan, T. and Wong, M.D.F.: Automatic Bus Planner for Dense PCBs, *Proc. Design Automation Conf.*, pp.326–331 (2009).
[7] Ozdal, M.M. and Wong, M.D.F.: Length-Matching Routing for High-Speed Printed Circuit Boards, *Proc. Int. Conf. Computer-Aided Design 2003*, pp.394–400 (2003).
[8] Ozdal, M.M. and Wong, M.D.F.: A Length-Matching Routing Algorithm for High-Performance Printed Circuit Boards, *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, Vol.25, No.12, pp.2784–2794 (2006).
[9] Ozdal, M.M. and Wong, M.D.F.: A Provably Good Algorithm for High Performance Bus Routing, *Proc. Int. Conf. Computer-Aided Design 2004*, pp.830–837 (2004).
[10] Ozdal, M.M. and Wong, M.D.F.: 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 (2006).
[11] Kubo, Y., Miyashita, H., Kajitani, Y. and Takeishi, K.: Equidistance Routing in High-Speed VLSI Layout Design, *Integration, VLSI Journal*, Vol.38, No.3, pp.439–449 (2005).
[12] Yan, T. and Wong, M.D.F.: BSG-Route: A Length-Matching Router for General Topology, *Proc. Int. Conf. Computer-Aided Design 2008*, pp.499–505 (2008).
[13] Nakatake, S., Fujiyoshi, K., Murata, H. and Kajitani, Y.: Module Packing Based on the BSG-Structure and IC Layout Applications, *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, Vol.17, No.6, pp.519–530 (1998).
[14] Yan, J.T. and Chen, Z.W.: Obstacle-Aware Length-Matching Bus Routing, *Proc. Int. Symp. Physical Design*, pp.61–68 (2011).
[15] Watson, I., Kirkham, C. and Lujan, M.: A Study of a Transactional Parallel Routing Algorithm, *16th International Conference on Parallel Architecture and Compilation Techniques*, pp.388–400 (2007).
[16] Yan, T. and Wong, M.D.F.: BSG-Route: A Length-Constrained Routing Scheme for General Planar Topology, *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, Vol.28, No.11, pp.1679–1690 (2009).
[17] Tsai, T.Y., Lee, R.J., Chin, C.Y., Kuan, C.Y., Chen, H.M. and Kajitani, Y.: On Routing Fixed Escaped Boundary Pins for High Speed Boards, *Design, Automation and Test in Europe Conference and Exhibition 2011*, pp.1–6 (2011).
[18] Chin, C.Y., Kuan, C.Y., Tsai, T.Y., Chen, H.M. and Kajitani, Y.: Escaped Boundary Pins Routing for High Speed Boards, *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, Vol.32, No.3, pp.381–391 (2013).
[19] Zhang, R., Pan, T.Y., Zhu, L. and Watanabe, T.: A Length Matching Routing Method for Disordered Pins in PCB Design, *Proc. Asia and South Pacific Design Automation Conf. 2015*, pp.402–407 (2015).
[20] Bergroth, L., Hakonen, H. and Raita, T.: A survey of longest common subsequence algorithms, *SPIRE 2000*, pp.39–48 (2000).

Ran Zhang was born in Dalian, China in April, 1985. She received her B.E degree in Electronic Engineering in 2008, from Dalian University of Technology. In 2010, she received her M.S. degree in Graduate School of Information, Productions and Systems, from Waseda University. She is currently working toward a Dr. Eng. degree in Waseda University. Her current research is optimization algorithms for Electronics DA designs. She is a member of IEICE.

Tieyuan Pan was born in Donggang, China, in October, 1988. He received his B.E. and M.E. degrees from the University of Kitakyushu in 2012 and 2014, respectively. He is currently working toward a Ph.D. degree in Graduate School of Information, Production and Systems, from Waseda University. His research interests include combination algorithms of VLSI in Electronics DA designs.

Li Zhu was born in Danyang, China, in January, 1991. He received his B.E. degree from Nanjing University in 2013. He is a master student from Graduate School of Information, Production and Systems, Waseda University. His current research is optimization algorithms for Electronics DA designs.

Takahiro Watanabe was born in Ube, Japan in October, 1950. He received his B.E. and M.E. degrees in Electrical Engineering from Yamaguchi University, and the Dr. Eng. from Tohoku University. In 1979, he joined Research and Development Center of TOSHIBA Corporation, where he worked in the field of LSI design automation. In August 1990, he joined Yamaguchi University, the Department of Computer Science and Systems Engineering, and in April 2003, he moved to Waseda University, Graduate School of Information, Production and Systems. His current research interests are EDA algorithm, Microprocessor and MPSoC, NoC, FPGA and their applications. He is a member of IEICE, IPSJ and IEEE.

(Recommended by Associate Editor: *Atsushi Takahashi*)