Waseda University Doctoral Dissertation

A Study of Routing Algorithms for PCB Design

Ran ZHANG

Graduate School of Information, Production and Systems
Waseda University

February, 2016

Abstract

A printed circuit board (PCB) supplies the connections of electronic components with tracks, pads and other features. It is almost used in all the electronic products and plays a very important role. The complexity of PCB becomes higher and higher since the integrated circuit technology advances rapidly. Such high density of pins makes the routing of PCB a time consuming and error-prone work. Therefore the routing in PCB design is usually dealt with by electronic design automation (EDA) tools to achieve optimizations.

In recent PCB design, due to the high density of integration, the signal propagation delay or skew has become an important factor for a circuit performance. We can control the signal propagation delay by adjusting the wire-length. If the routing area is large enough, it is not difficult to control the wire-length of the net. However, the routing area is usually limited and multiple nets should be considered in the dense area. Hence, how to balance the wire-length of the multiple nets becomes a very difficult problem. Moreover, for river routing problems, in general the positions of pins are fixed on the components, and usually the source and target pins are disordered. Therefore, multi-layers are used 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.

Besides, in a modern PCB, a flip-chip package is widely used to meet the higher integration density and the larger I/O counts of circuits. In the flip-chip design, redistribution layer (RDL) is often used to redistribute the I/O pads to the bump balls without changing the placement of them. For the pre-assignment RDL routing problem, how to assign the I/O pads to bump balls and minimize the total wire-length are usually focused on. Furthermore, 3D IC has become the good choice for high-performance circuits, since recently it is hard to solve some interconnection problems by traditional 2D IC. Thus, the I/O pad assignment and RDL routing problem in both 2D and 3D IC should be solved to improve the whole circuit performance.

In this research, we propose a series of routing algorithms for PCB design to solve the

above-mentioned different problems. This thesis mainly includes the following three points.

Firstly, to assign layers for disordered pins and get equal-length routing results, a region-aware routing algorithm in PCB design is proposed. In the layer assignment process, the longest common subsequence (LCS) algorithm is adopted between source and target pin sets to determine the layers for pins. In the routing process, virtual boundaries need to be set if the pins sequence does not satisfy trunk routing topology condition. The base routes for multiple nets are generated by the single commodity flow method. In addition, considering target length requirement and routing region coefficient α, R-flip and C-flip techniques are used to adjust the wire-length. This proposed routing algorithm is able to obtain the routes with better wire-length balance and smaller worst length error in reasonable CPU time.

Secondly, to minimize the total wire-length, a sorting-based I/O pad assignment and non-Manhattan RDL routing algorithm are proposed for area I/O flip-chip design. By sorting the Manhattan distance between I/O pads and bump balls, the pre-assignment and its revision are carried out to determine the initial assignment. Three kinds of pair-exchange procedures for shortening wire-length, overlapping and crossing connections are proceeded out respectively to improve the initial assignment. The exchange order is according to the descending Manhattan distance between the assigned I/O pad and bump ball pairs. To shorten the wire-length, non-Manhattan RDL routing with 90 degrees and 45 degrees wire segments is adopted to connect the I/O pads and bump balls. Moreover, some un-routed connections should be ripped-up and rerouted. The proposed design method is effective on reducing total wire-length no matter of the I/O pad locations and package sizes, and improves the routability.

Finally, we apply the same sorting method in the above to the I/O pad assignment and RDL routing method in 3D IC design. Similarly, we assign the same numbered I/O pads in two RDLs to micro-bumps by sorting the sum of Manhattan distance between them. A pair exchange modification of a route is considered for shortening wire-length, and the single layer routing in two RDLs are carried out respectively. Some un-routed connections are ripped-up and rerouted at last. This method for 3D IC is also able to obtain the routes with shorter total wire-length in reasonable CPU time.

In conclusion, the equal-length routing problem for disordered pins and the RDL routing problem for flip-chip in PCB design can be well solved using the proposed routing algorithms. Furthermore, some optimizations, such as better wire-length balance, smaller worst length error and shorter total wire-length, can be well realized by using them.

This thesis is organized as follows:

In Chapter 1, the architecture and package of PCB and the structure of RDL are firstly

summarized. Then three typical routing problems in PCB design are introduced, and the research proposals of this paper are given.

Chapter 2 reviews some fundamentals of PCB routing for discussion at the succeeding chapters. Firstly, four types of signal net routing problems are explained. Then, two kinds of basic routing method and their representative algorithms are respectively discussed.

Chapter 3 describes a proposed routing method called a region-aware layer assignment and equal-length routing method for disordered pins in PCB design. By using this algorithm, it is able to obtain the routes with better wire-length balance and smaller worst length error. The experimental results show that, the proposed method could be applied in both no-obstacle routing and obstacle-ware routing problems. Compared with another greedy method for disordered pins, the proposed method gets a smaller standard deviation, in other words, a better wire-length balance among the nets, by adopting coefficient α to adjust the wire-length skew. Besides, our method is effective in reducing worst length error, and the average reduction is 36.69%.

Chapter 4 introduces the second proposed method which is a sorting-based I/O pad assignment and non-Manhattan RDL routing method for area I/O flip-chip design. Our proposed method is effective on reducing wire-length no matter of the I/O pad locations and package sizes. Compared with a partition-based method, the proposed method can reduce the total wire-length by 23.4% using Manhattan routing, and 39.6% using non-Manhattan routing. Compared with another Delaunay-triangulation method, the proposed method can reduce the total wire-length by 3.8% using Manhattan routing, and 20.0% using non-Manhattan routing in the reasonable CPU time.

Chapter 5 presents an application of I/O pad assignment and RDL routing method to 3D IC, on the basis of the sorting method in Chapter 4. Compared with a matching-based method, the proposed method is able to obtain the routes with shorter total wire-length in reasonable CPU times. For small scale package, the average wire-length reduction is 17.52%. Then for large scale packages, the maximum and minimum wire-length reductions are 23.66% and 14.87%, respectively.

Chapter 6 concludes this thesis and discusses the future work.

Acknowledgements

I spend five years on studying towards Ph.D. at Graduate School of Information, Production and Systems, Waseda University. It is a long way filled with complex emotions. This thesis can never be accomplished without many people's help. I want to express my sincere thanks for their consistent support.

First of all, I would like to express my deepest gratitude to Prof. Takahiro Watanabe. His enthusiastic guidance helps me through all the stages of completing this thesis. He is not only my supervisor, but also my father and friend. His respectable personality and erudite knowledge will guide me in my whole life.

I also want give my deep thanks to Prof. Takeshi Yoshimura and Prof. Shinji Kimura for their valuable comments on my study. Their precise attitude and professionalism of research is worthy for us to study.

I am grateful to Dr. Xin Jiang and Dr. Candidates Tieyuan Pan, Yang Tian, Lian Zeng and Huatao Zhao for their constructive discussions of my research. Thanks are also due to all my friends in Watanabe Lab, and many other students and faculty for their constant encouragement.

Finally, my thanks would go to my parents, who support me all the way. They always encourage me and stand at my back whenever I meet difficulty. Thank you.

Ran ZHANG
Kitakyushu, Japan
February, 2016

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

This paper focuses on the study of routing algorithms for printed circuit board (PCB) design. Three major items relating to PCB routing problems are presented. In this chapter, firstly the architecture and package of PCB and the structure of redistribution layer (RDL) are summarized. Then three typical routing problems in PCB are introduced, and the proposals of this paper are described. The organization of this thesis is also shown in this chapter.

1.1 Overview of PCB and Flip-chip

A PCB is a board made of fiberglass, epoxy or other materials, and supplies the connections of electronic components with tracks, pads and other features. It is almost used in all the electronic products and plays a very important role. In recent PCB design, since the integrated circuit technology rapidly advances, the complexity of PCB becomes higher and higher. Such high density of pins makes the routing of PCB a time consuming and error-prone work [1]. Thus, recently a variety of researches have focused on the PCB routing problems. To precisely understand PCB routing problem, some fundamentals of PCB are firstly introduced in this section.

1.1.1 PCB Architecture

In the modern PCB design, it usually hosts several components such as multi-chip modules (MCMs), memories, and I/O modules [2]. These components are mounted onto the board as a set of dense pin arrays, as shown in Figure 1.1. These pin arrays should be connected by some non-crossing wire segments [1].

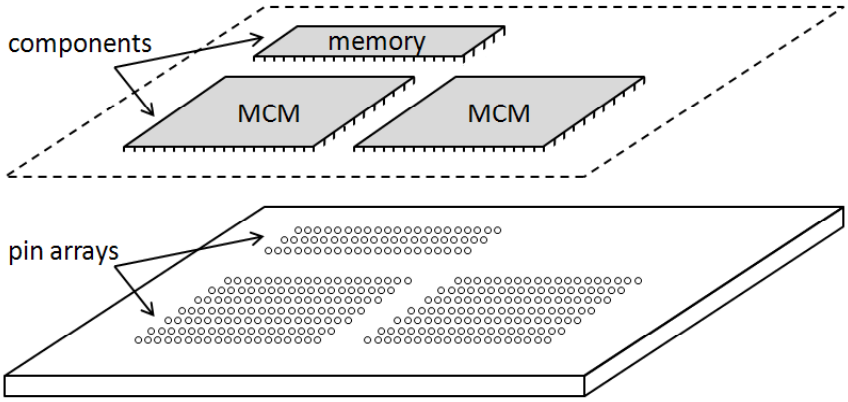
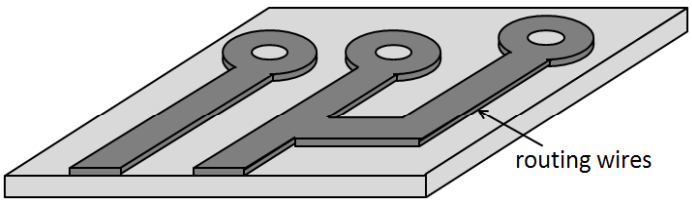
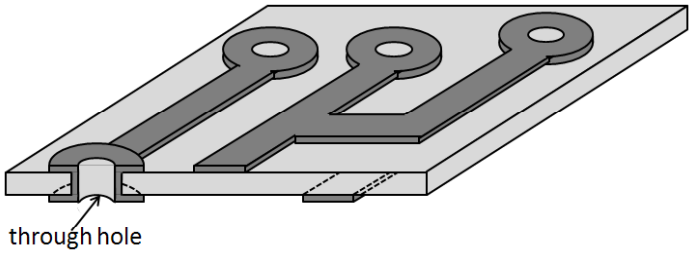


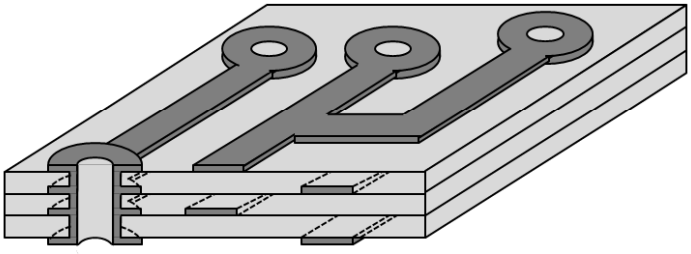
Figure 1.1 PCB architecture



(a) Single-sided board



(b) Double-sided board



(c) Multi-layer board

Figure 1.2 Three types of PCB [3]

According to the layer number of board, there are three types of PCB: single-sided board, double-sided board, and multi-layer board, as shown in Figure 1.2 [3].

Single-sided Board

In the single-sided board, the routing wires are concentrated only in one side, as shown in Figure 1.2 (a). Since there are many strict restrictions of the routing designs, for example, the wire segments on the board cannot cross with each other, the routing on single-sided board often fails. Furthermore, the high density of pins in modern routing designs makes the routing on single-sided board more and more difficult. Thus, only early circuit uses this type of board.

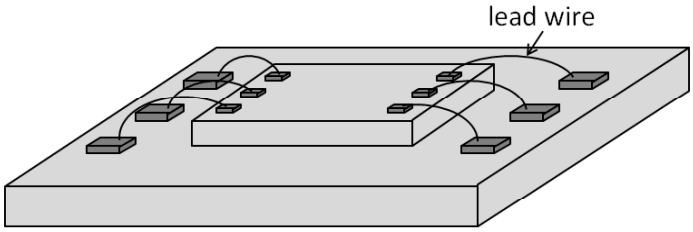
Double-sided Board

Different from single-sided board, double-sided board has wire segments on both sides of it, as shown in Figure 1.2 (b). It is a two-layer board, and the conductors on different layers are connected by the through holes. These through holes are usually called vias, which are coated with metal. Since the routing area of double-sided board is larger, and the wires can be routed in different layer through vias, some problems of single-sided board, such as crossing wires, routing resource limitation, can be solved by double-sided board. It is usually used in more complex circuits.

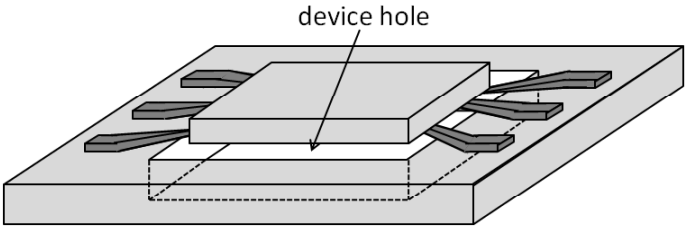
Multi-layer Board

Multi-layer board uses more single-sided boards or double-sided boards to increase the available area, as shown in Figure 1.2 (c). All the outer and inner layers can be used to route wires. Same as double-sided board, multi-layer board uses vias to connect the conductors on different layers. In present applications, the most common PCBs have 4-10 layers. The multi-layer boards allow for higher density of components and pins.

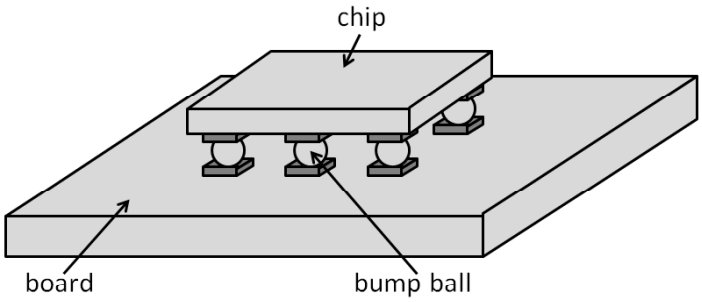
Recent years, because of the increasing of pins number and strict design rules, the routing resource for the components is usually limited, both within the pin arrays and between different components [2]. Thus, multi-layer PCBs are widely applied. In this paper, our research on PCB routing designs also relates to multiple layers.



(a) Wire bonding



(b) Tape-automated bonding



(c) Flip-chip

Figure 1.3 PCB package [3]

1.1.2 Flip-chip Architecture

In the PCB manufacture, the components should be packaged onto the board. Since the shorter wire connections produces higher performance of the circuit, there is a package technique that directly mounts the chips onto the board to connect with other conductors [3]. This technique is called bare chip package, and it includes three patterns: wire bonding, tape-automated bonding and flip-chip, as shown in Figure 1.3 [3].

Wire Bonding

Wire bonding is a method that makes the interconnection between device and substrate during

fabrication, as shown in Figure 1.3 (a). It is a main technology for electrical connections between components and board in PCB package. In the wire bonding, the chip is upright mounted, and lead wires are used to interconnect the components to external conductors [4]. Wire bonding has fast bonding process, excellent electrical and chemical property, and it is considered as a flexible and cost-effective interconnect technology. Great majority of PCB packages use this technique.

Tape-automated Bonding

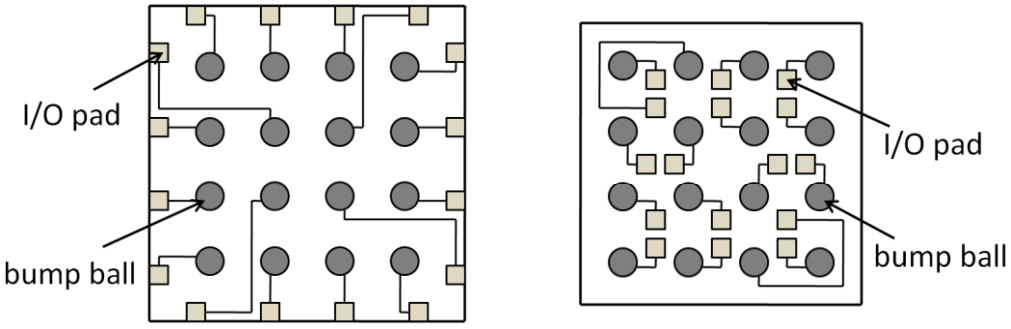
Tape-automated bonding (TAB) is a process directly placing the chips onto PCB by using etched copper beam, as shown in Figure 1.3 (b). One end of the etched copper beam leads to the conductors while another end leads to the PCB [5]. TAB is created as an alternative of wire bonding and finds its common use in the display driver circuits. TAB offers some advantages such as smaller bonding pad, less molding and lower costs. In addition, it improves the heat transfer, performs high frequency, and requires for less PCB surface area [6].

Flip-chip

Flip-chip is a method for interconnecting components to external conductors by using bump balls. The structure of flip-chip is shown in Figure 1.3 (c), where the active side of the chip is faced down towards and mounted onto a substrate [7]. The bump balls are deposited on the bump pads on the top side of the chip. The chip is flipped over to align with the matching pads on the board for connection.

The flip-chip structure can be classified into two types: a peripheral I/O flip-chip and an area I/O flip-chip, illustrated in Figure 1.4 [8]. In the peripheral I/O flip-chip, the I/O pads are placed along the boundary of a die, and the I/O pads should be routed from boundary to the bump balls inside. While in the area I/O flip-chip, the I/O pads are placed in the whole area of the flip-chip package. Compared with a peripheral I/O flip-chip, an area I/O flip-chip can generate shorter wire-length and smaller package size, so it is more popularly applied.

Compared with the conventional package technologies, flip-chip offers a number of advantages: higher I/O density, higher throughput, better heat dissipation, shorter interconnects, smaller footprint, lower profile and so on. These strong points have made flip-chip one of the most attractive techniques in modern package for PCB design [9].



(a) Peripheral I/O flip-chip (b) Area I/O flip-chip

Figure 1.4 Flip-chip structure [8]

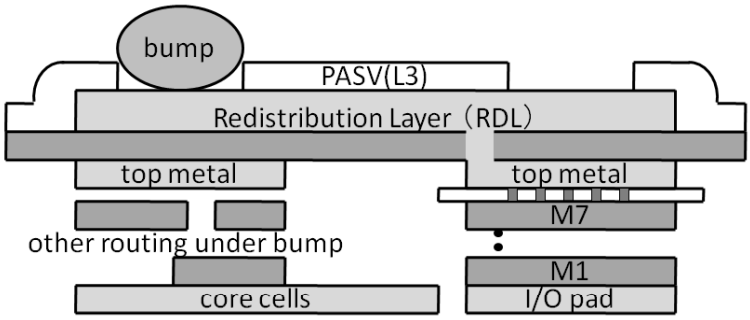


Figure 1.5 RDL structure [11]

Redistribution Layer

Although flip-chip technology is widely used in the PCB designs, sometimes the placement of I/O pads cannot be well mapped onto the bump balls [10]. As a result, a top metal or an extra metal layer, RDL is used to redistribute the I/O pads to the bump balls without changing the placement of the I/O pads, as shown in Figure 1.5 [11]. Bump balls are placed on the RDL and used to connect to I/O pads. In a RDL, the wires can be routed in either 90 degrees or 45 degrees wire segments by current technology.

Re-distributing the I/O pads to the bump balls offers higher density, greater flexibility and lower cost, and improves the circuit performance. Besides, it is an effective method to contact the power and ground in some applications, and transform off-chip connections from chip scale to board scale.

Recently, because of the increasing I/O pads in the large scale flip-chip package, only single RDL may be not enough to complete the routing. Furthermore, routing in a single RDL may bring out longer wire-length. Therefore, the multiple RDLs have been considered for the

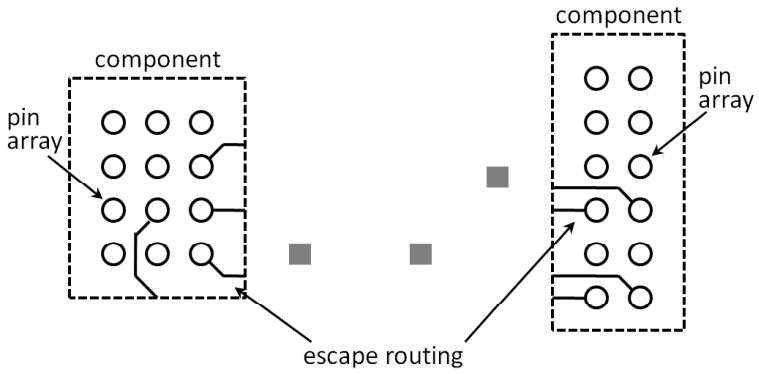


Figure 1.6 Escape routing problem

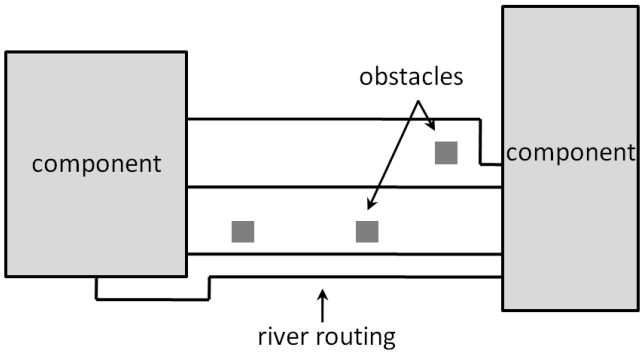


Figure 1.7 River routing problem

connection between I/O pads and bump balls in some researches [12]. In this paper, our research on flip-chip designs relates to both single RDL and multiple RDLs.

1.2 Typical Routing Problems

Based on the fundamentals of PCB described above, we discuss the PCB routing problem in this section. A modern PCB usually hosts several chip packages whose footprints are pin arrays which are expected to be routed by non-crossing nets [1]. Three typical routing problems of PCB are introduced: escape routing, river routing, and RDL routing.

1.2.1 Escape Routing

Escape routing is a routing problem among pin arrays inside the components. As illustrated in Figure 1.6, it is to route the pins from inside of the pin arrays to the boundary, just like helping

the pins “escape” the pin arrays.

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. Moreover, sometimes it should offer the matching pin order along the boundaries of two components for later river routing. There are mainly three types of escape routing problem: unordered escape routing, ordered escape routing and simultaneous escape routing [1], and they are all applied in PCB routing but not discussed in this paper.

1.2.2 River Routing

Relative to escape routing, river routing is a routing problem between two or more components. It is to connect the escaped pins on the boundaries of components with some length constraints, as shown in Figure 1.7. Usually the obstacles are aware in the river routing problem.

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. There are two typical categories of river routing problem: min-max length routing and equal-length routing. In the min-max length routing, each wire-length should meet the given minimum and maximum length bounds. While the equal-length routing aims to generate wires with the same length.

In recent PCB design, due to the high density of integration, the signal propagation delay or skew has become an important factor for a circuit performance. We can control the signal propagation delay by adjusting the wire-length. If the routing area is large enough, it is not difficult to control the wire-length of the net. However, the routing area is usually limited and multiple nets should be considered in the dense area. Hence, how to balance the wire-length of the multiple nets becomes a very difficult problem. Moreover, for river routing problem, in general the positions of pins are fixed on the components, and usually the source and target pins are disordered. Therefore, multi-layers are used 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.

1.2.3 RDL Routing

Besides, as a special structure in flip-chip, the routing in RDL is also often discussed for PCB design. It is to connect the I/O pad to the bump ball with wire-length minimization.

There are two main RDL routing problems. One of the problems is free-assignment routing

problem, in which any I/O pad is not assigned to any bump ball before routing. Another problem is pre-assignment routing problem. In this problem, the connections between the I/O pads and the bump balls are defined before routing. Since the pre-assignment of the connections has more routing constraints, the pre-assignment routing problem is much harder than the free-assignment one.

For the pre-assignment RDL routing problem for area flip-chip, how to assign the I/O pads to bump balls and minimize the total wire-length are usually focused on. Furthermore, 3D IC has become the good choice for high-performance circuits, since recently it is hard to solve some interconnection problems by traditional 2D IC. Thus, the I/O pad assignment and RDL routing problem in both 2D and 3D IC should be solved to improve the whole circuit performance.

1.3 Research Proposals

In this paper, we propose a series of routing algorithms for PCB design to solve the above-mentioned different problems. This thesis mainly includes the following three points.

Firstly, to assign layers for disordered pins and get equal-length routing results, a region-aware routing algorithm in PCB design is proposed. In the layer assignment process, the longest common subsequence (LCS) algorithm is adopted between source and target pin sets to determine the layers for pins. In the routing process, virtual boundaries need to be set if the pins sequence does not satisfy trunk routing topology condition. The base routes for multiple nets are generated by the single commodity flow method. In addition, considering target length requirement and routing region coefficient α, R-flip and C-flip techniques are used to adjust the wire-length. This proposed routing algorithm is able to obtain the routes with better wire-length balance and smaller worst length error in reasonable CPU time.

Secondly, to minimize the total wire-length, a sorting-based I/O pad assignment and non-Manhattan RDL routing algorithm are proposed for area I/O flip-chip design. By sorting the Manhattan distance between I/O pads and bump balls, the pre-assignment and its revision are carried out to determine the initial assignment. Three kinds of pair-exchange procedures for shortening wire-length, releasing overlap and avoiding cross are proceeded out respectively to improve the initial assignment. The exchange order is according to the descending Manhattan distance between the assigned I/O pad and bump ball pairs. To shorten the wire-length, non-Manhattan RDL routing with 90 degrees and 45 degrees wire segments is adopted to connect the I/O pads and bump balls. Moreover, some un-routed connections should be ripped-up and rerouted. The proposed design method is effective on reducing total wire-length

no matter of the I/O pad locations and package sizes, and improves the routability.

Finally, we apply the same sorting method in the above to the I/O pad assignment and RDL routing method in 3D IC design. Similarly, we assign the same numbered I/O pads in two RDLs to micro-bumps by sorting the sum of Manhattan distance between them. A pair exchange modification of a route is considered for shortening wire-length, and the single layer routing in two RDLs are carried out respectively. Some un-routed connections are ripped-up and rerouted at last. This method for 3D IC is also able to obtain the routes with shorter total wire-length in reasonable CPU time.

In conclusion, the equal-length routing problem for disordered pins and the RDL routing problem for flip-chip in PCB design can be well solved using the proposed routing algorithms. Furthermore, some optimizations, such as better wire-length balance, smaller worst length error and shorter total wire-length, can be well realized by using them.

1.4 Organization of This Thesis

The rest of this paper is organized as follows:

Chapter 2 reviews some fundamentals of PCB routing. Firstly, four types of signal net routing problems are explained. Then, two kinds of basic routing method and their representative algorithms are respectively discussed.

Chapter 3 describes a region-aware layer assignment and equal-length routing method for disordered pins in PCB design. By using this algorithm, it is able to obtain the routes with better wire-length balance and smaller worst length error.

Chapter 4 introduces a sorting-based I/O pad assignment and non-Manhattan RDL routing method for area I/O flip-chip design. The proposed method is able to obtain the routes with shorter wire-length in reasonable CPU time.

Chapter 5 presents an application of I/O pad assignment and RDL routing method to 3D IC, on the basis of the sorting method in Chapter 4. This method can also reduce the total wire-length by comparing with other design method.

Chapter 6 concludes this thesis and discusses the future work.

Chapter 2

Fundamentals of PCB Routing

This chapter reviews some fundamentals of PCB routing. Firstly, four types of signal net routing problems are explained. Then, two kinds of basic routing methods and their representative algorithms are respectively discussed. Based on these routing architectures and methods, we do our research for PCB routing design.

2.1 Signal Net Routing

In the modern PCB routing design, since the complexity becomes higher and higher, electronic design automation (EDA) has been widely used to automate optimizations. EDA is a class of software tools to design the electronic systems. It is heavily depended on in the complex PCB routing design process [13]. After the placement process, all the nets should be routed in the routing area. A net is a set of pins with the same electric potential that should be connected. For the signal nets, the routing usually includes three stages: global routing, detailed routing, and timing-driven routing. In addition, some specialized routing problems are also considered in the routing process.

2.1.1 Global Routing

In general, the pins on the component should be globally routed before detailed routing. The global routing does not route wires but just plans the connections [14].The input of the global routing are the locations of the components and pins. During global routing, the wire segments of the net are tentatively assigned in the routing area. Usually, coarse grids are used to represent the routing area. The edges in the grid graph represent the available routing resources, which are

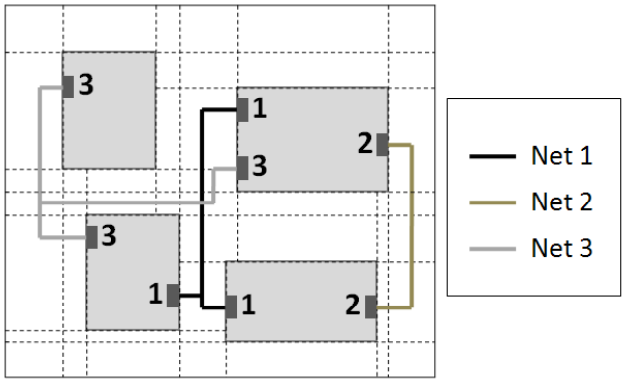


Figure 2.1 Example of global routing [13]

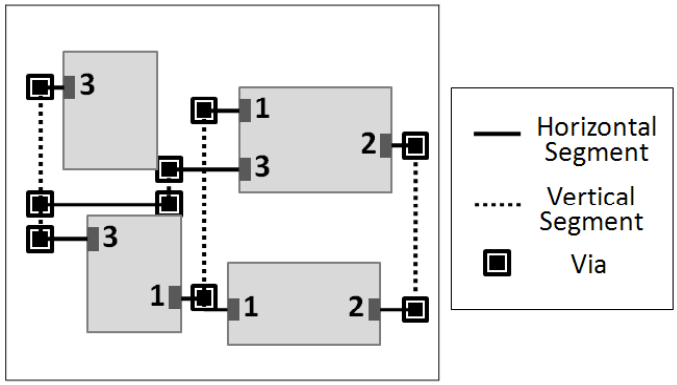


Figure 2.2 Example of detailed routing [13]

then used to assign the nets.

Global routing aims to provide detailed routing with where to route the nets. Typically, the objectives of global routing are to reduce the total wire-length, reduce the routing delay, or improve the probability for further detail routing. In the modern designs, due to millions of nets, the global routing has become a major challenge.

The global routing flow includes three steps. Firstly the routing area is formed as some region types, such as channels, switchboxes and so on. Then nets are mapped to the routing area. Finally some cross points are assigned. As shown in Figure 2.1 [13], there are four components and three nets to be routed, and the routing area is represented as coarse grids. The diagrammatic presentation of net1, net2 and net3 are globally routed in the divided routing regions.

2.1.2 Detailed Routing

After the global routing, the routing regions for nets have been determined. The detailed routing uses this information to decide the exact wire connections and layers for each net [14]. During detailed routing, the wire segments of the nets are assigned to specific routing tracks. In addition, the design rules must be considered in the detailed routing.

Detailed routing aims to complete the wire connections between the components. Commonly, its objectives are to reduce the total wire-length, layer number, or the routing delay. As the increasing of modern IC scaling, the impact of manufacturing faults, such as via defects [15]-[16], interconnect defects [17], etc., also should be considered during the detailed routing process.

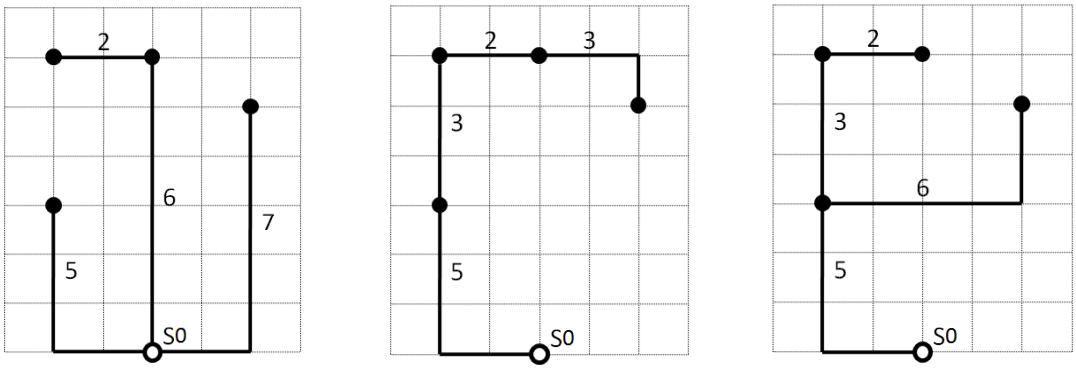
Detailed routing depends on the global routing result that, usually the configuration of nets is not changed. Thus, if the global routing result is good, the detailed routing result will be good likewise. For example, based on the global routing result in Figure 2.1, the detailed routed paths are shown in Figure 2.2 [13]. This detailed routing result assumes that, the horizontal wires and vertical wires are on the separate two layers, and vias are used to connect the wires in different layers.

2.1.3 Timing-driven Routing

Sometimes, timing-driven routing is necessary since the interconnection delay is concerned in the routing stages. The objectives of timing-driven routing are to reduce the maximum source-sink delay or the total load-dependent delay.

Usually, the source-sink delay is reflected by the source-sink wire-length, and the load-dependent delay is expressed as total wire-length [13], and we use depth and cost in name of them respectively. Therefore, an ideal routing tree could reduce both the maximum source-sink wire-length and the total wire-length. However, it is difficult to minimize these two items at the same time in most cases.

Figure 2.3 [13] illustrates a tradeoff of maximum source-sink wire-length and total wire-length. In the figure, s_0 is the source pins, and the black points stand for the sinks. The routing tree in Figure 2.3 (a) obtains the minimum depth = 8. It is a shortest-paths tree that is constructed by Dijkstra's algorithm [18]. However in this tree, the cost = 20 is very large. In Figure 2.3 (b), the routing tree obtains the minimum cost = 13, and it is a minimum spanning tree (MST) constructed by Prim's algorithm [19]. But the depth = 13 in this tree becomes larger.



(a) Shortest-paths tree (b) Minimum cost tree (c) Compromise of depth and cost

Figure 2.3 Example of timing-driven routing [13]

Figure 2.3 (c) shows a routing tree compromising of depth and cost, since no matter large cost or large depth in the routing tree are undesirable in practice.

2.1.4 Specialized Routing

Besides of global routing, detailed routing and timing-driven routing, some specialized routing problems are considered in modern PCB routing design. Two typical specialized routing problems, net order in area routing and non-Manhattan routing, are discussed in this subsection.

Net Order in Area Routing

In some types of design, the global routing and detailed routing are not performed separately. Instead, the area routing connects signal pins directly, and it aims to achieve crossing minimization. Area routing is usually constrained by the technology, electricity, and geometry factors [13], such as layer number, signal integrity and so on.

When multiple nets are routed, the net order in area routing will affect the final routing results and the total runtime. Greedily route multiple nets by minimizing each net's wire-length at a time may lead to a number of un-routable nets or large total wire-length. Moreover, relative to two-pin nets, the complexity of routing for multi-pin nets increases, which is more dependent on the net order. For example, in Figure 2.4, there are two nets to be routed. If we route one net at a time to optimize its wire-length, no matter net1 (Figure 2.4 (a)) or net2 (Figure 2.4 (b)) is firstly routed, it may fail to route another net because of the routing area limitation. However, these two nets can be routed at the same time if we do not minimize wire-length for each net, as

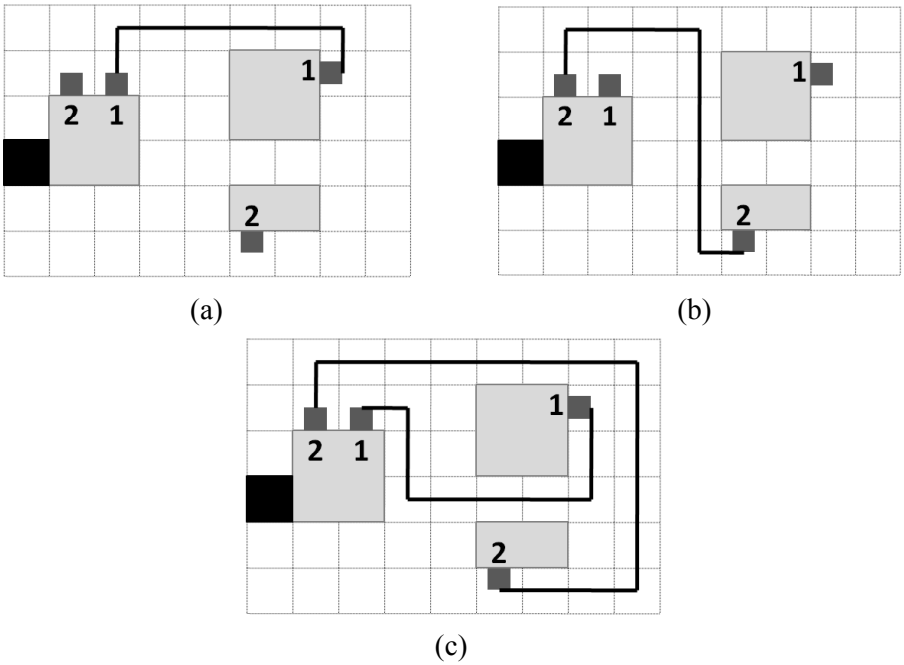


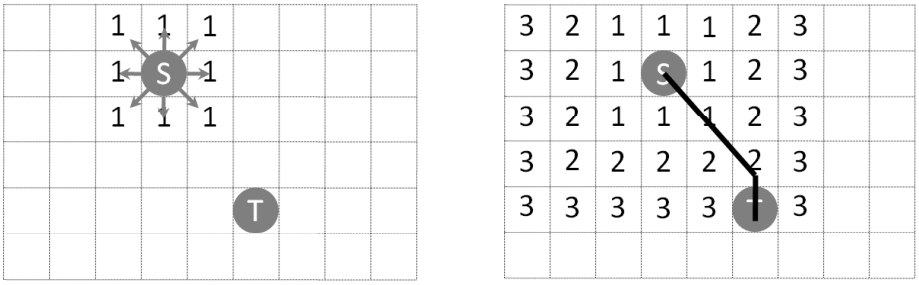
Figure 2.4 Net order in area routing

shown in Figure 2.4 (c).

As a result, usually the net order in area routing is determined by some routing algorithms in advance. Different applied routing algorithms may cause different of net order. Some Steiner tree-based algorithms [20]-[21] that resolve the multi-pin nets into two-pin nets, and some geometric criteria are used to optimize the net order. For instance, the nets can be ordered according to the x-coordinate of pins, and then be routed from left to right.

Non-Manhattan Routing

In the traditional Manhattan routing, it allows vertical and horizontal wire segments only. However, using diagonal wire segments may generate shorter wire-length. Since the diagonal wire segments cannot be arbitrary, in general 45 degrees or 60 degrees wire segments are added to horizontal and vertical wire segments. Such routing models are commonly described by a parameter λ, which indicates the number of routing directions and the angle of wire segments. When $\lambda = 2$, there are four routing directions and wires are 90 degrees, and it is the traditional Manhattan routing. When $\lambda = 3$, there are six routing directions and wires are 60 degrees, and it is called Y-routing. When $\lambda = 4$, there are eight routing directions and wires are 45 degrees, and it is called X-routing. The latter two types of routing are non-Manhattan routing [13].



(a) (b)

Figure 2.5 Non-Manhattan maze routing

The advantages of non-Manhattan routing model are reflected in the reduction of total wire-length and via number. However, the long time physical verification and optical lithography limitations make non-Manhattan routing difficult to achieve. Thus non-Manhattan routing is mainly adopted in PCB routing problems.

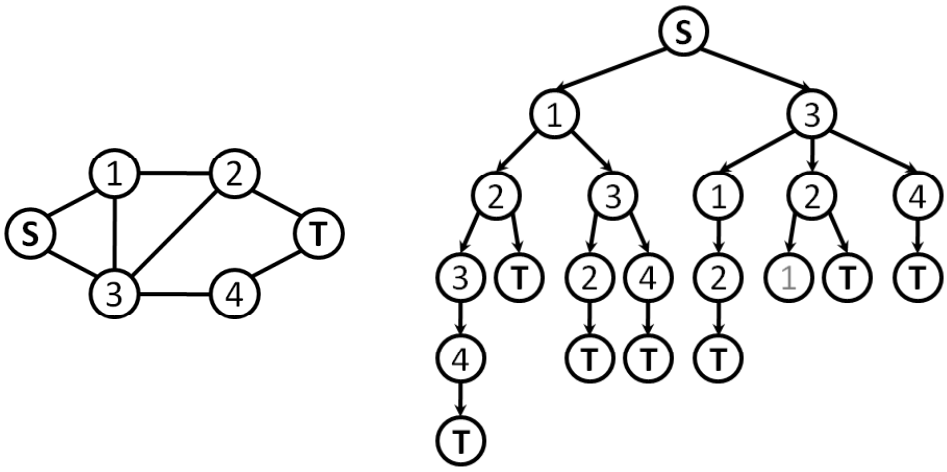
For example, Figure 2.5 illustrates a non-Manhattan maze routing method. This method is based on the traditional maze routing algorithm [22], but expands nodes in eight directions instead of four directions. It starts from source pin *S* and marks all unsearched neighbor grids with number 1 (Figure 2.5 (a)). Then, it restarts from each node marked by number 1, and marks all unsearched neighbor grids with number 2. This expansion continues until the target pin *T* is reached. Finally, from the target pin *T* to source pin *S*, a path including 45 degrees wire segments is traced back, as shown in Figure 2.5 (b).

2.2 Basic Routing Methods

During the PCB routing design, graph traversal based routing methods are usually considered. Graph traversal visits the vertices of a graph in some order [23]. Depth-first search (DFS) method and breadth-first search (BFS) method are two kinds of basic techniques to solve the graph-related problems. Both of these two methods construct the spanning trees in certain manners [24]. Based on them, a number of routing algorithms are proposed.

2.2.1 Depth-first Routing Method

DFS is a method for traversing a finite graph. For routing problems, it starts at the source vertex, and iteratively explores from current vertex to the unvisited neighbor vertex, until target vertex is reached or no unexplored vertex left. Then it backtracks along the visited vertices to the



(a) (b)

Figure 2.6 Depth-first routing method

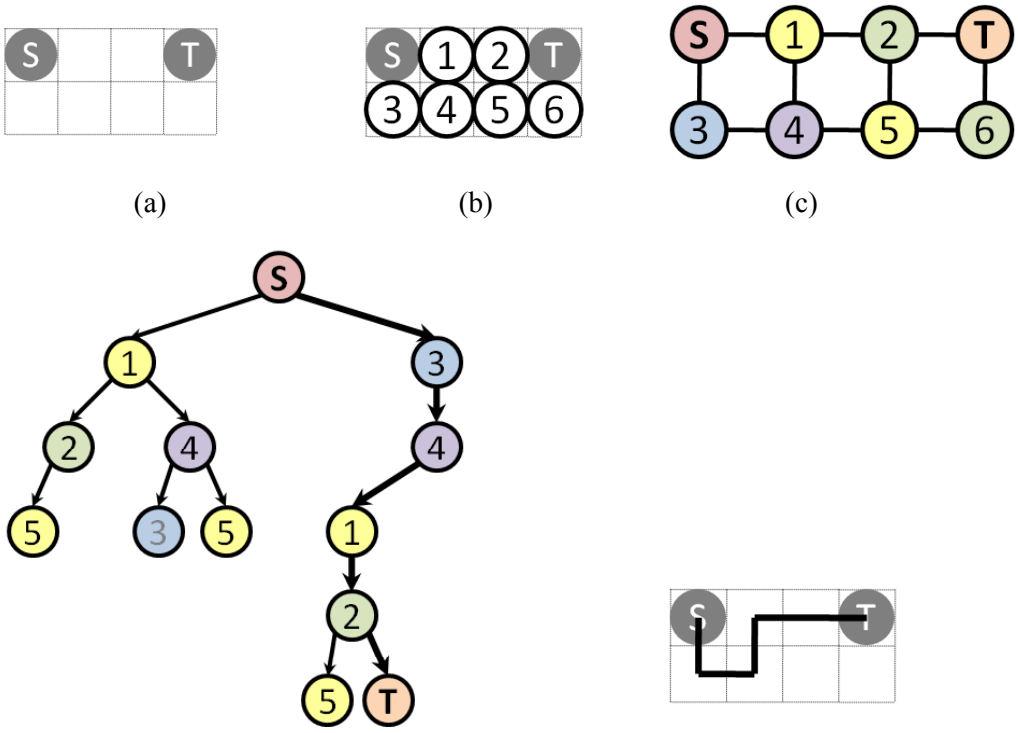
original source vertex. For example in Figure 2.6 (a), there are six vertices in a finite graph including the source *S* and target vertex *T*. DFS starts at *S*, and explores to its neighbor vertices 1 and 3. Then respectively from 1 and 3, the search continues to their next unvisited adjacent vertices. This exploration is carried on until *T* is reached or no unvisited vertex left. The spanning tree of this DFS is shown in Figure 2.6 (b). From this spanning tree, we can get all the possible paths from *S* to *T*.

The time analysis of depth-first search method is according to its applicated graph. The iterative deepening increases the running time, due to the geometric growth vertex number. Given a graph *G(V, E)*, the time complexity of DFS is $O(|V|^k)$, where *V* represents for the number of vertices, and *k* is the depth of spanning tree. Appling DFS, all the possible paths can be listed. In addition, it is effective in solving length-matching routing problems, which is a sub-problem of *k*-path problem [25]. A typical depth-first search based routing algorithm is color coding routing algorithm.

Color Coding Routing Algorithm

Color coding is a randomized method to find a path in graph with fixed length [26]-[27]. It firstly paints the vertices of the graph with random colors. The number of colors is equal to the fixed length, and one color should be used at least once. Then an improved depth-first search is carried on. Once the path with target length is found, the process is over.

Take an example for illustration. Figure 2.7 (a) shows a grid routing problem with a required



(a) (b) (c)

(d) (e)

Figure 2.7 Color coding routing algorithm

length five. Firstly we mark all the grids with different numbers (Figure 2.7 (b)), and this problem can be represented as the graph shown in Figure 2.7 (c). The vertex of this graph stands for the grid and the edges shows the connection relationship between these grids. Then we paint the vertices with five colors randomly. DFS starts from *S*, and iteratively explores to the unvisited neighbor vertex. If one vertex has the same color with previous visited vertex, such as vertex 5 and 1 with the same color yellow, the exploration of this path will stop. This process continues until the first *T* is reached, as shown in Figure 2.7 (d), or no unexplored vertex left. Then it tracks back from *T* to *S*, and the routing path with the target length five can be generated (Figure 2.7 (e)).

2.2.2 Breadth-first Routing Method

BFS is another method for traversing a finite graph. For routing problems, it starts at the source vertex, and explores first to the unvisited neighbor vertex in the same level, then to the next level neighbors. This level depends on the distance between the current vertex and the source

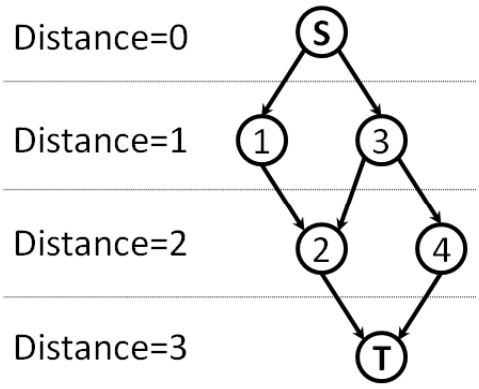


Figure 2.8 Breadth-first routing method

vertex. The exploration ends if target vertex is reached or no unexplored vertex left. Then it backtracks along the visited vertices to the original source vertex. BFS is often used to find the shortest path. For the same example in Figure 2.6 (a), BFS starts as *S*, and since there is only one vertex in the first level (Distance = 0), it explores to vertices 1 and 3 of the next level. This exploration is carried on until *T* is reached or no unvisited vertex left. The spanning tree of this BFS is shown in Figure 2.8. From this spanning tree, we can get the shortest path from *S* to *T*.

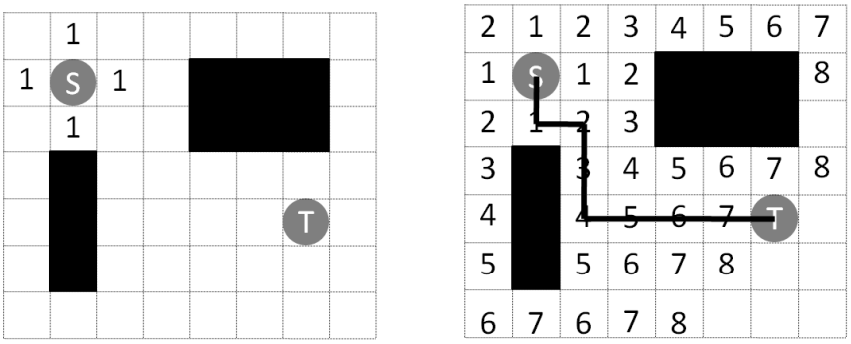
The time analysis of breadth-first search method is according to the number of vertices and edges of the graph. Given a graph *G(V, E)*, the time complexity of DFS is O(|*V*|+|*E*|), where *V* and *E* represents for the number of vertices and edges respectively. A typical breadth-first search based routing algorithm is maze routing algorithm to find the shortest path.

Maze Routing Algorithm

Maze routing algorithm aims to solve the shortest path in the grid routing problem [22]. It starts at the source vertex, and marks current vertex's neighbor grids in four directions with number from small to large. Once the target vertex is reached, the process is over.

For example in Figure 2.9, there is a grid routing problem to find the shortest path from source *S* to target *T*. In this example, the obstacles, indicated as black blocks, are also taken into consideration. The algorithm starts from *S* and marks all unvisited neighbor grids in four directions with number 1 (Figure 2.9 (a)). Then, it restarts from each grid marked by number 1, and marks all unvisited neighbor grids with number 2. This expansion continues until *T* is reached. Finally, from *T* to *S*, a shortest path is traced back, as shown in Figure 2.9 (b).

Both DFS and BFS have their advantages and disadvantages. Although DFS can find all the



(a) (b)

Figure 2.9 Maze routing algorithm

possible paths or generate a path with fixed length, the time complexity of it is very large. On the other hand, the running time by BFS is short, but it usually consumes more computer memory. Some researches about combining BFS with DFS have been studied in recent years [28]-[29]. For our research in this paper, breadth-first search based routing algorithms are mainly adopted.

2.3 Conclusions

In this chapter, some fundamentals of PCB routing are reviewed. Four types of signal net routing problems are explained firstly. Then two kinds of basic routing methods and their representative algorithms are discussed respectively. Based on these points, we develop the routing methods for PCB design.

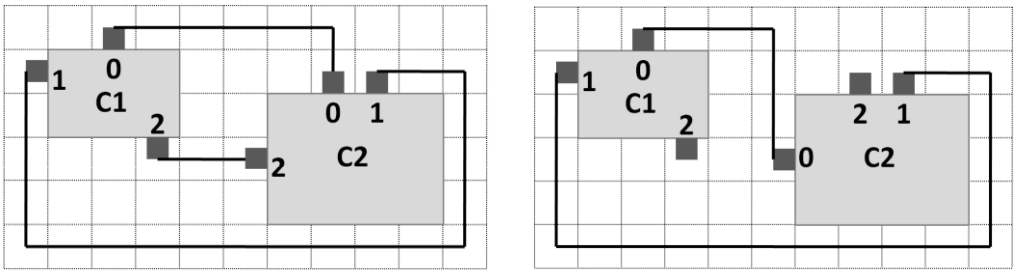
Chapter 3

Region-aware Layer Assignment and Equal-length Routing

In this chapter, a region-aware layer assignment and equal-length routing method 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.

3.1 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 designs [30]-[31]. 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 difficult 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



(a) Ordered pins (b) Disordered pins

Figure 3.1 Order of pins

very important problem, which is formulated as equal-length routing problem in PCB design.

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’ (Figure 3.1 (a)); otherwise, called ‘disordered’ (Figure 3.1 (b)). Assuming 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 Figure 3.1 (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 chapter, 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 [32]-[34] 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 time.

The remainder of this chapter is organized as follows: Section 3.2 describes some previous works related to this study. Section 3.3 describes the problem definition of this work. Section 3.4 details the three phases of proposed routing algorithm. Section 3.5 illustrates the experimental results and analysis. Finally, the conclusion is given in Section 3.6.

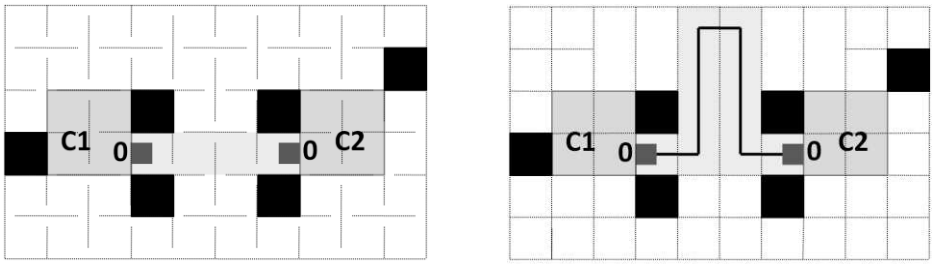
3.2 Related Works

Some researches for river routing problem have been proposed. [35] proposed an automatic bus planner for dense PCBs. In [36] and [37], a Lagrangian-relaxation framework was used to allocate routing resources during routing to control the length of each net. In [38] and [39], 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 [40], which used symmetric-slant grid interconnect to transform the length matching problem into a general grid routing problem. In [41], a length matching routing method was presented with no restriction on routing topology using bounded slice-line grid [42]. However, these works mentioned above do not consider the obstacles in routing area.

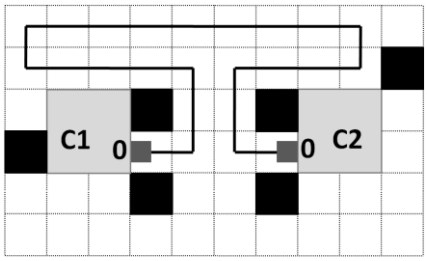
In fact, there are several obstacles in PCB routing area, such as device and IC package, etc. Thus, consideration of obstacles is important in PCB design. For obstacle-aware routing problems, [43] explored a length matching routing method based on region partition. A transactional parallel routing algorithm was studied in [44]. In [32]-[34], 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 [45], 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 Figure 3.2 (a), an input topology is embedded onto BSG with obstacles. In this case, the longest wire generated by BSG routing method is shown in Figure 3.2 (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 [34].

A practical approach to solve the fixed disordered pins routing problem was discussed in [46]-[47]. 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 Figure 3.3 (a),

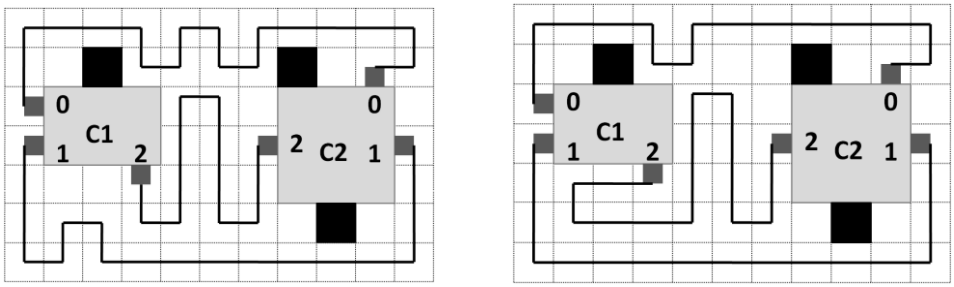


(a) BSG embedding (b) BSG route with obstacles



(c) Routing by our method

Figure 3.2 Comparison with BSG routing method

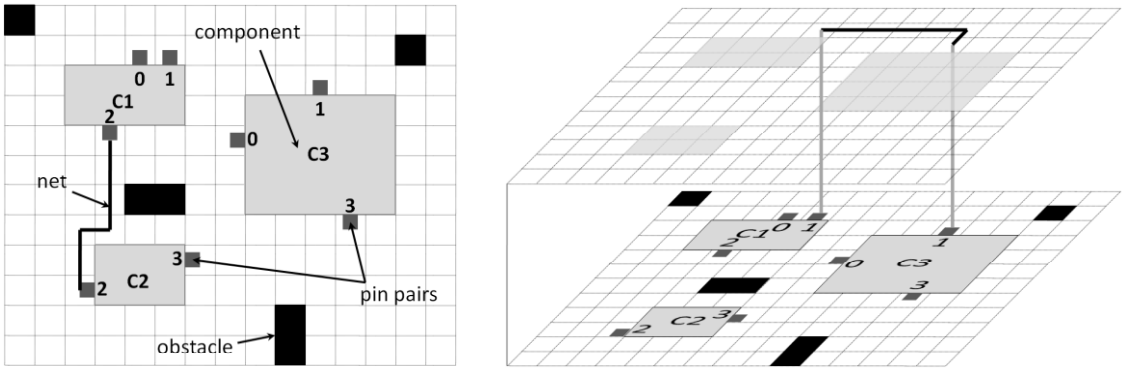


(a) Unbalanced routing (b) Balanced routing

Figure 3.3 Balance wire-length of nets

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 6 and unbalanced nets routing result.

In this chapter, 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 efficiently using routing area to achieve a longer target length when obstacles exist, as shown in Figure 3.2 (c). Moreover, we balance the nets by revising the adjusted wire-length. The proposed method can



(a) Single layer (b) Multi-layers

Figure 3.4 Routing model

generate routes with better wire-length balance as shown in Figure 3.3 (b), all the nets are with the same length 17 and the worst length error is reduced to 2. The following sections discuss the proposed method in detail.

3.3 Problem Definition

In this chapter, 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 study, 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, $\bar{x}$ is the average value of $x_1, \ldots, x_N$, and N stands for the size of the sample.

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

We give an example in Figure 3.4 (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, …, n), called "pin set" of this component. The same labeled elements in different sets should be connected, called pin pairs. As illustrated in Figure 3.4, 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.

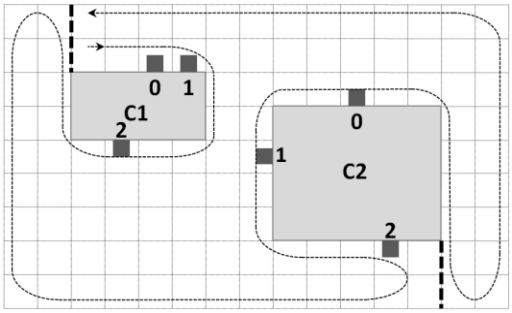


Figure 3.5 Example of trunk routing topology

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 (Figure 3.4 (b)). In this study, at most three layers are permitted, as adding layers without limitation is not much sense.

In this study, trunk routing problem between two components is dealt with. Trunk routing problem is introduced in [32]-[34], 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 this study, 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 Figure 3.5, after adding the virtual boundaries, represented by dotted lines, source pins sequence is 0→1→2, and target pins sequence is 2→1→0, which satisfy the trunk routing topology condition.

3.4 Routing Algorithm

In this study, since multi-components are given, the routing for multiple pin sets is considered. As mentioned in Section 3.2, [46] and [47] 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 study, 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

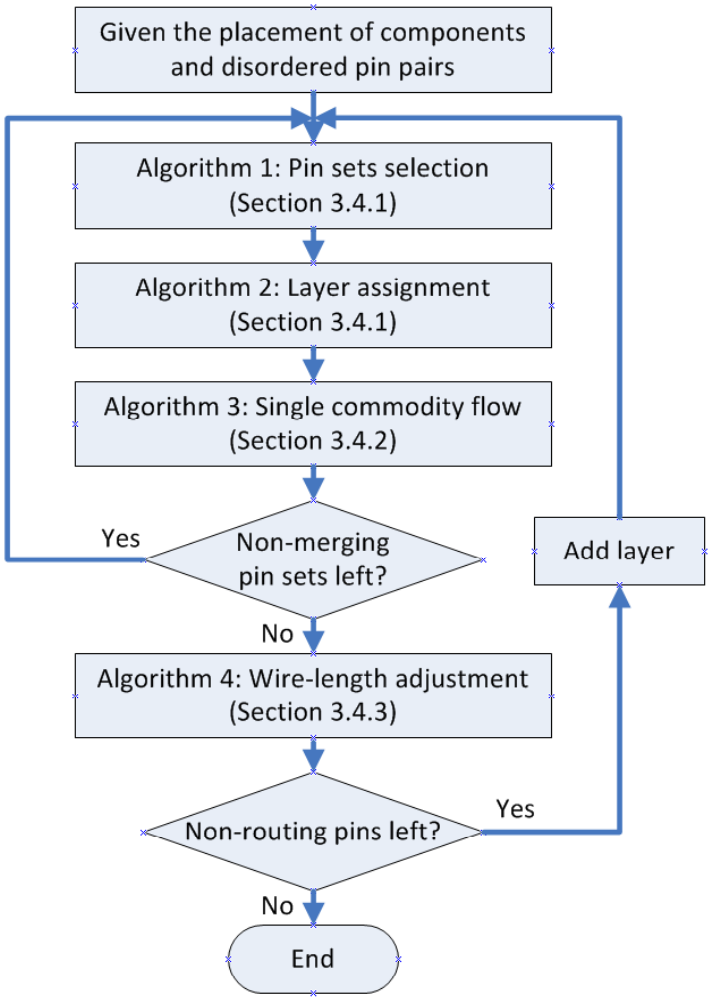


Figure 3.6 Flow chart of layer assignment and equal-length routing process

to implement the crossings minimization between two components than that among all the components by adopting LCS (longest common subsequence) algorithm [48]-[49]. Another difference is that, in the initial routing phase of [46] and [47], 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 Figure 3.6. Algorithms 1 to 4 are described in detail in the following subsections.

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

3.4.1 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. 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, …, P_n are the pin sets of components. The two pin sets to be handled are noted as P_s and P_t. $P_i \cap P_s$ means the intersection set of P_i and P_s.

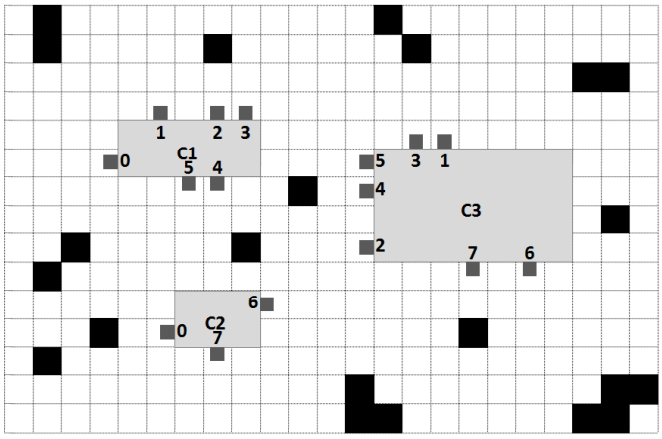


Figure 3.7 Placement of components and disordered pins

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-$P_i \cap P_s$ 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 Figure 3.7 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 $P_1 \cap P_s$ = {1, 2, 3, 4, 5} and $P_2 \cap P_s$ = {6, 7} are not empty, and the quantity of elements of P_1 - $P_1 \cap P_s$ is 1, P_2 - $P_2 \cap P_s$ 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 elements common 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

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 $\complement_Q L$ for other layers;
end

periphery routed wires and ultimately returning to that point. L is a set of longest common subsequence of elements of S and T. $\complement_Q L$ 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 [48]-[49], and put the result into set 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 $\complement_Q L$ for other layers.

We also take the case in Figure 3.7 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 $\complement_Q L$ = {2, 5}. As a result, we assign net1, net3 and net4 in layer1, and reserve net2 and net5 to layer 2.

3.4.2 Initial Routing

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.

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

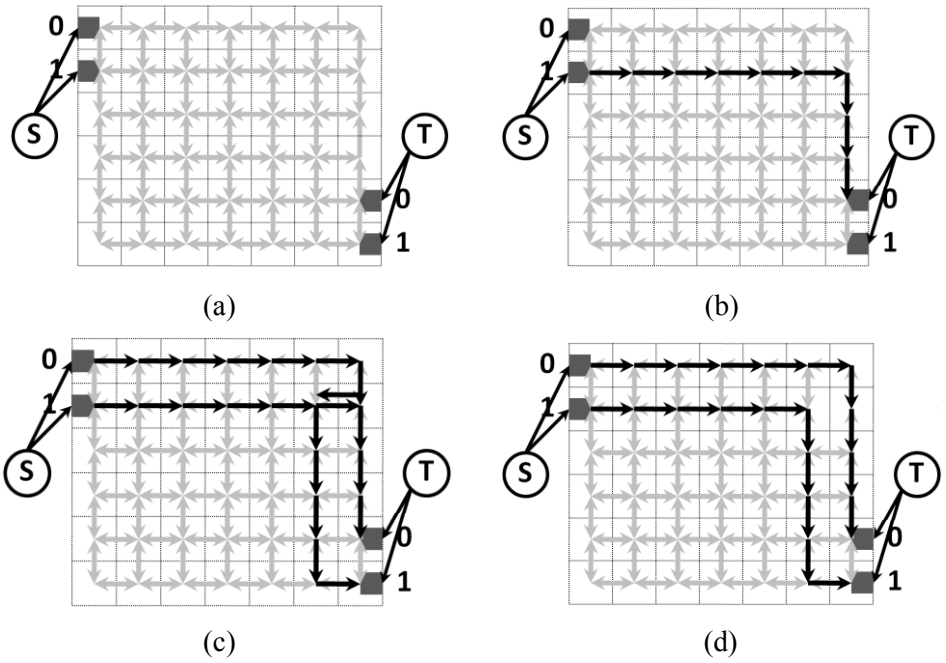
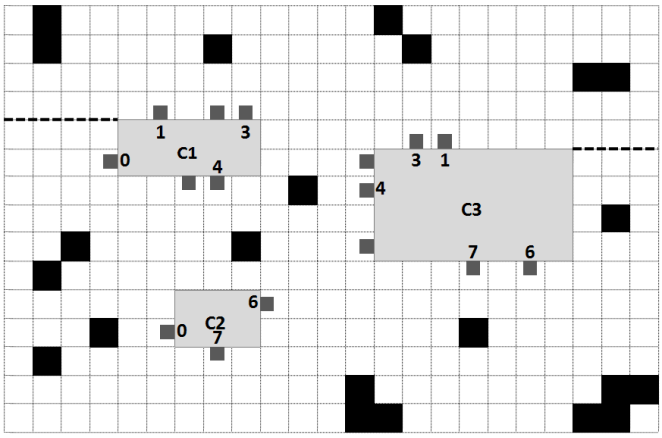
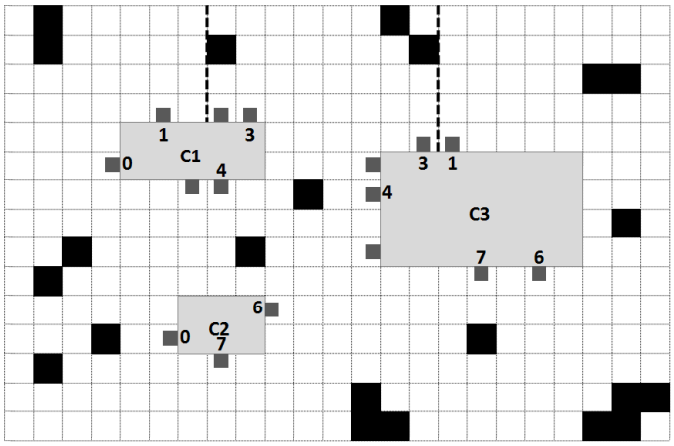


Figure 3.8 Single commodity flow method

The routing by single commodity flow method is shown in Figure 3.8. 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 Figure 3.8 (a). Augmenting paths are explored by breadth first search, shown as net 1 in Figure 3.8 (b). In the path, the



(a)



(b)

Figure 3.9 Virtual boundary setting

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 Figure 3.8 (c). The edge whose both bi-directions are used needs to be deleted and non-crossing nets can be generated, shown in Figure 3.8 (d). In this way, we obtain the base routes of pin pairs.

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 Figure 3.9. 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

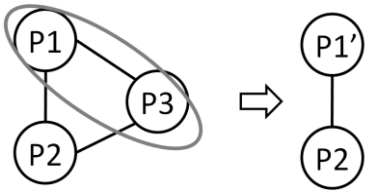


Figure 3.10 Pin sets merging

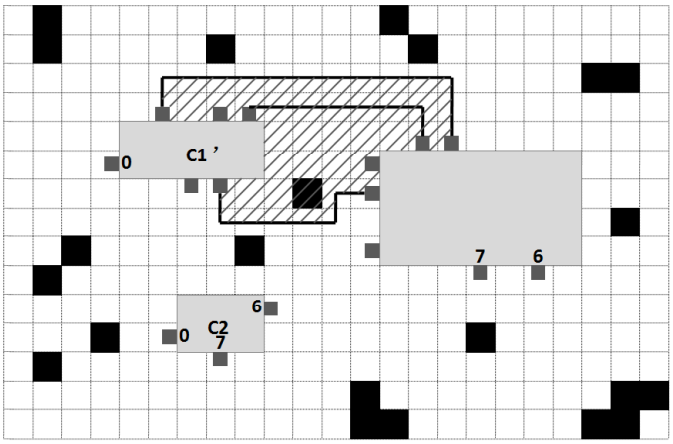


Figure 3.11 Initial routing

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 (Figure 3.9 (a)). If between two pins no component corner exists, the virtual boundary is set in the middle of them (Figure 3.9 (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 same numbered with the second element and first one in *L* (Figure 3.9 (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 Figure 3.10. 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 Figure 3.11, 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 Figure 3.7 again. Since L = {1, 3, 4} in the last phase, we set the virtual boundary between pin pair 1 and 4, as shown in Figure 3.9 (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.

3.4.3 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 wire-length skew 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 adjustment of each net, it is processed by the following three 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.

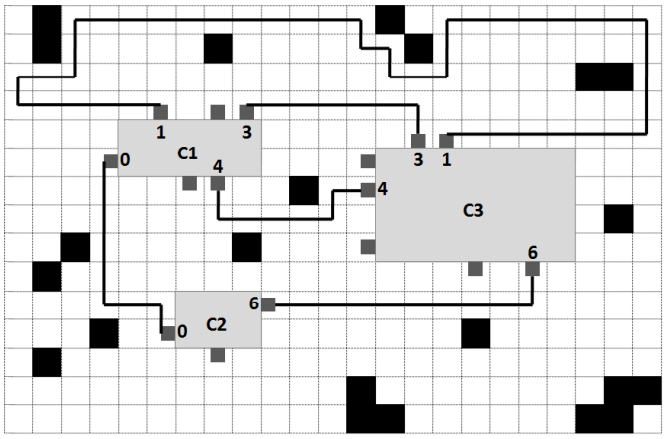
[Step 1] Initially, we extend the source pin and target pin to the outer boundary of a routing region. If the pin is on the horizontal boundary of the component, the extend direction is horizontal, otherwise, the extend direction is vertical. Then the wires are rerouted along the boundary. This process 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 Figure 3.12, 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 Figure 3.12 (a).

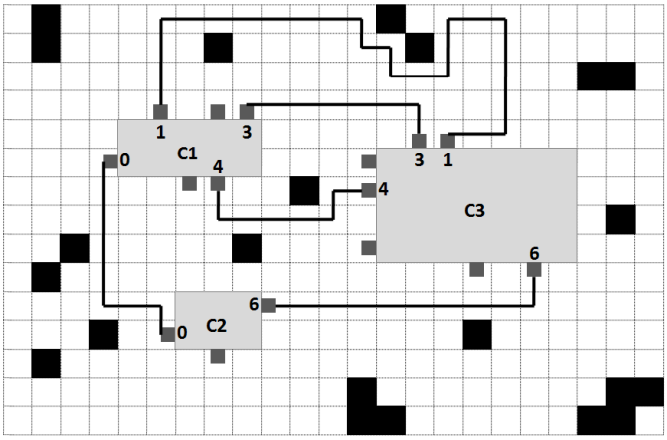
Algorithm 4. Wire-length adjustment

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

[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 [32]-[34]. 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



(a) Extend wires



(b) Adjust wires

Figure 3.12 Wire-length adjustment

number not odd. Hence, the wire-length skew La_i is calculated by $[(l_t\text{-}L_i)/2]*2$ ("[ ]" means omit decimals). If La_i is negative, then we need to shorten the wire. Otherwise, we lengthen the wire.

For the example in Figure 3.12 (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 Figure 3.12 (b). Similarly, the other nets as adjusted one by one until all the nets are completed.

[Step 3] As the aim of this study 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

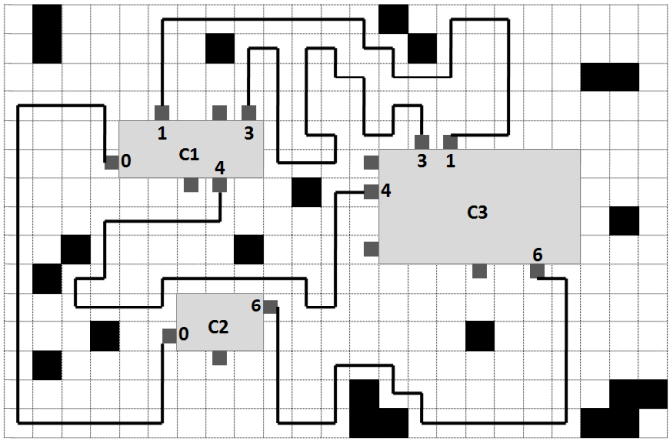


Figure 3.13 Adjusted routing result

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 study, for equal-length routing, a coefficient α (0.1, 0.2, …, 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 wire-length skew 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 Figure 3.12, Figure 3.13 shows the final 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.

3.4.4 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 $O(mn)$.

Table 3.1 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

3.5 Experimental Results and Analysis

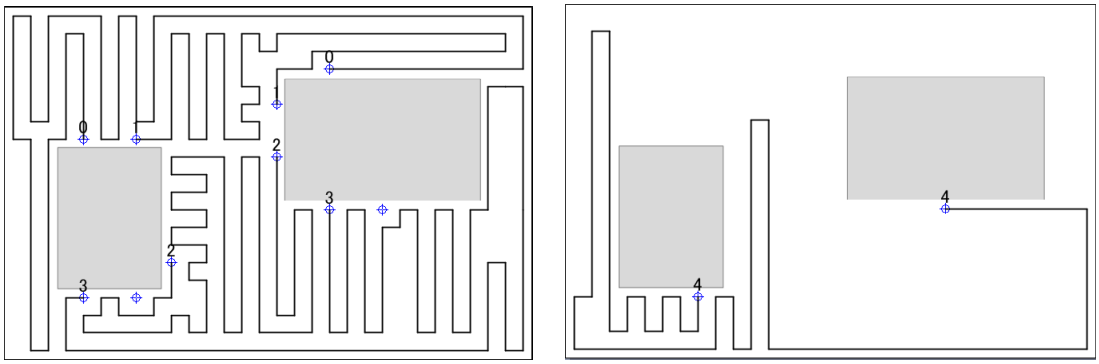
We implemented our proposed method in C language, which is compiled by MinGW Developer Studio 2.06, and executed on a PC with 2.66GHz Intel Core 2 CPU and 2GB RAM. Six experimental data named from Data00 to Data05 for evaluation are synthesized by referring the test cases in [46]. We narrow the range 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 3.1, 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 greedy routing method based on [46] followed by some adjustment.

Experiment 1

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 3.2 and the routing results of l_t = 130 without and with obstacles are respectively shown in Figure 3.14 and Figure 3.15. In this table, *Std dev* 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 the table and figures, 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, such as net 0 in Figure 3.14 (a) and Figure 3.15 (b). From Table 3.2, we also note that, when target length is set larger, the standard deviation becomes larger, and the worst length

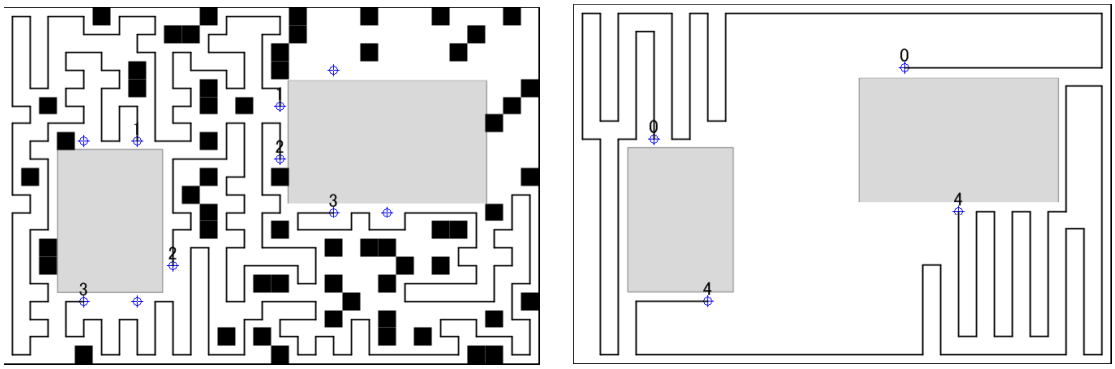
Table 3.2 Experimental results on different target length for Data00 (without α)

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



(a) Layer 1 (b) Layer 2

Figure 3.14 Routing results of $l_t = 130$ without obstacles in Experiment 1



(a) Layer 1 (b) Layer 2

Figure 3.15 Routing results of $l_t = 130$ with obstacles in Experiment 1

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.

Experiment 2

In Experiment 2, also for Data00, we change the value of α from 1 to 0.1 to analysis of the

Table 3.3 Experimental results on different utilized coefficient for Data00 (l_t = 130)

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

impact of α on standard deviation, while target length is not changed. From the experimental results in Table 3.3, when α = 0.7, 0.6, 0.5, 0.3, 0.2, a minimum standard deviation can be obtained. Moreover, when α = 0.7, the worst length error is smallest, thus α = 0.7 is considered as an optimal coefficient for wire-length balance.

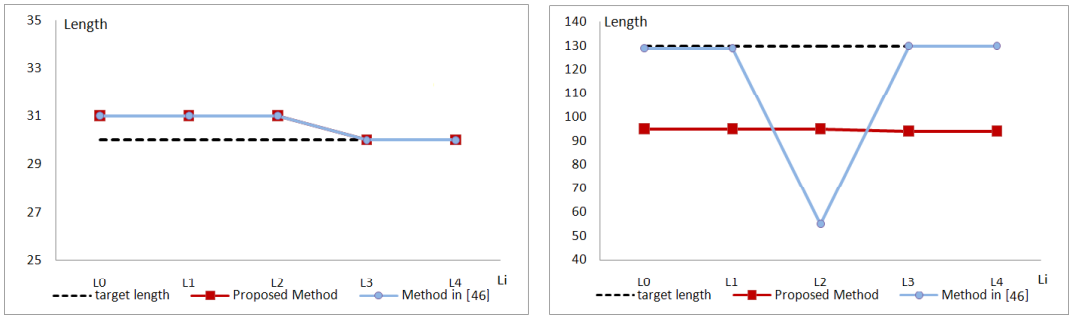
Experiment 3

We compare the proposed routing method with another greedy method [46] in Experiment 3. Since the method [46] 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*.

The experimental results are shown in Table 3.4. And the length comparison of each net is illustrated in Figure 3.16 to Figure 3.21, where Y-axis means a resultant net length for each net L_i mapped on X-axis, and a dotted line represents the target length. The experimental results show that, the method [46] 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.

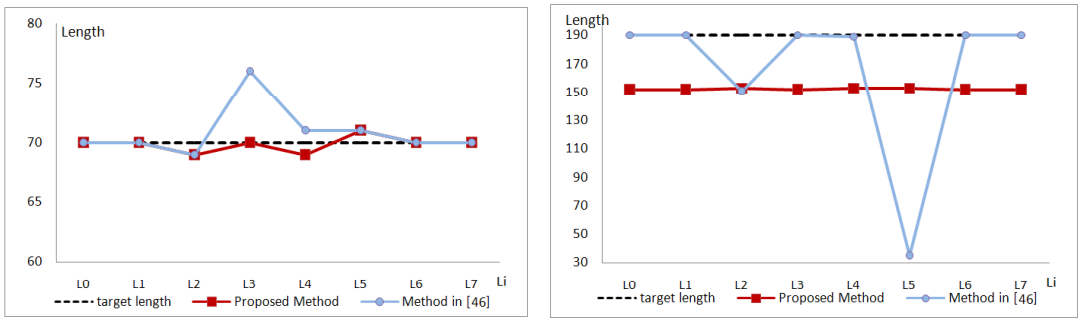
Table 3.4 Experimental results on comparison with another greedy method

		Method [46] + adjustment			Proposed method			
	Target length	*Std dev*	*Average length*	*CPU time*	*Std dev*	*Average length*	α	*CPU time*
Data00	$l_t = 30$	0.55	30.60	<1s	0.55	30.60	-	<1s
	$l_t = 130$	33.32	114.60	<1s	0.55	94.60	0.7	<1s
Data01	$l_t = 70$	2.17	70.88	<1s	0.64	69.88	-	<1s
	$l_t = 190$	54.50	165.63	<1s	0.52	152.38	0.8	<1s
Data02	$l_t = 70$	12.74	66.42	<1s	4.93	49.92	0.4	<1s
	$l_t = 120$	42.33	96.58	<1s	4.93	49.92	0.4	<1s
Data03	$l_t = 40$	2.34	39.70	<1s	1.48	33.10	0.3	<1s
	$l_t = 110$	39.77	83.70	<1s	0.83	76.80	0.7	<1s
Data04	$l_t = 40$	7.80	37.39	<1s	0.61	35.18	0.4	<1s
	$l_t = 80$	23.34	68.89	<1s	0.61	35.18	0.4	<1s
Data05	$l_t = 60$	15.96	52.32	<1s	7.42	31.73	0.3	<1s
	$l_t = 110$	40.02	81.46	<1s	7.42	31.73	0.3	<1s



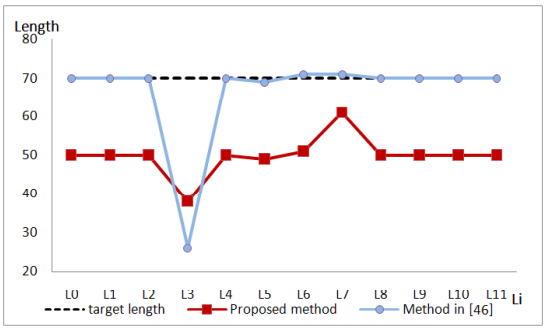
(a) $l_t = 30$ (b) $l_t = 130$

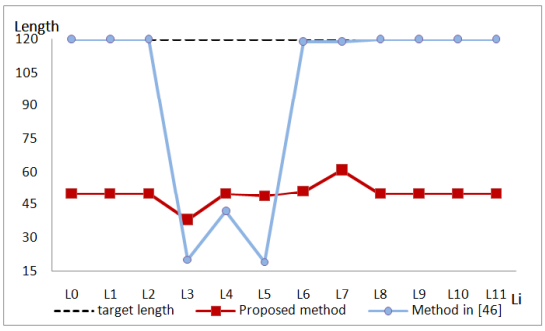
Figure 3.16 Length comparison for Data00 in Experiment 3



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

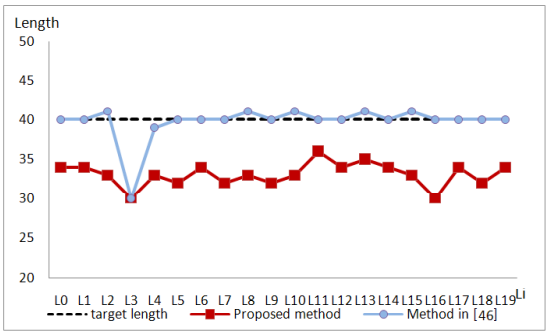
Figure 3.17 Length comparison for Data01 in Experiment 3

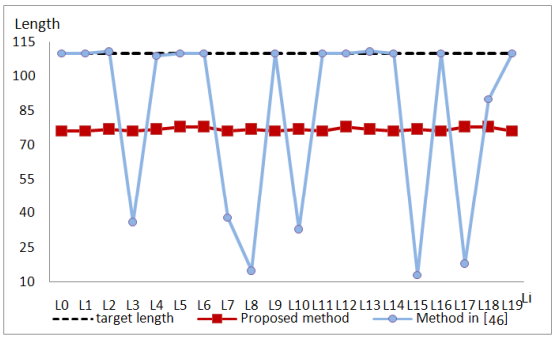




(a) $l_t = 70$ (b) $l_t = 120$

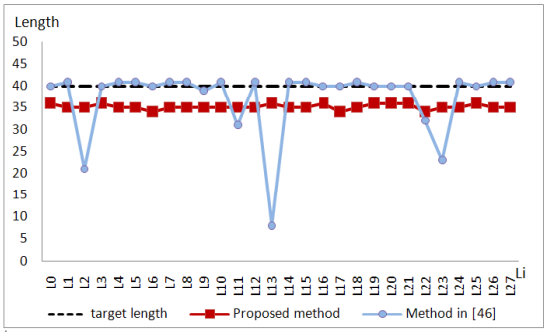
Figure 3.18 Length comparison for Data02 in Experiment 3

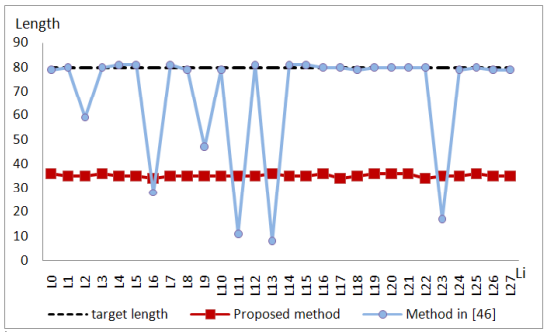




(a) $l_t = 40$ (b) $l_t = 110$

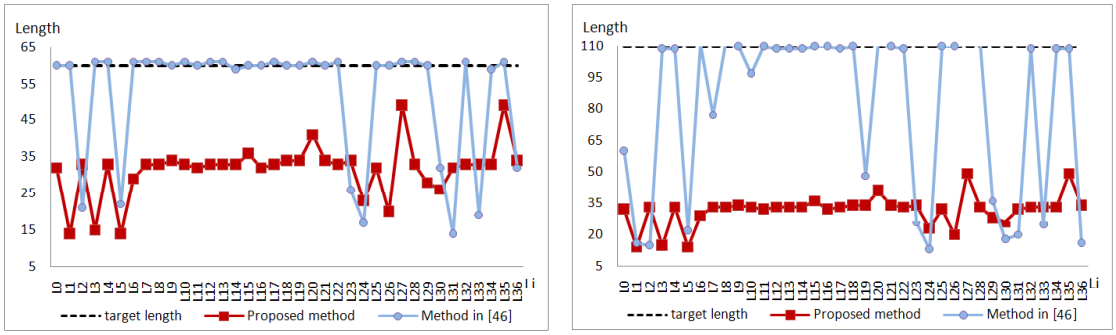
Figure 3.19 Length comparison for Data03 in Experiment 3





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

Figure 3.20 Length comparison for Data04 in Experiment 3



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

Figure 3.21 Length comparison for Data05 in Experiment 3

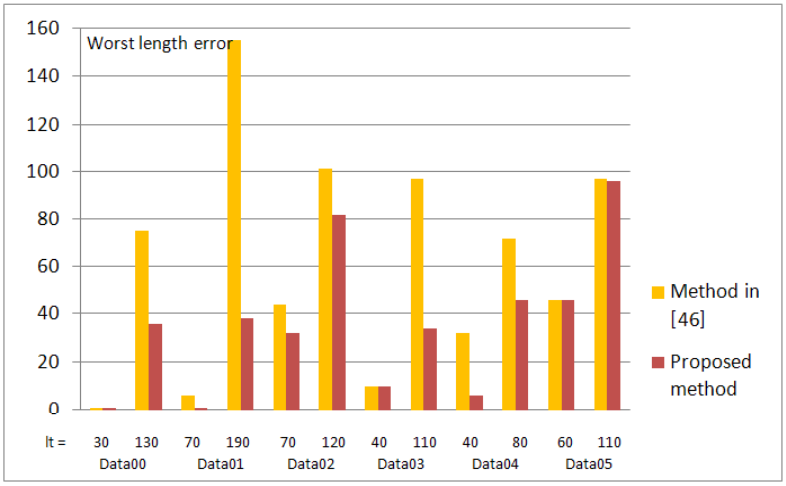
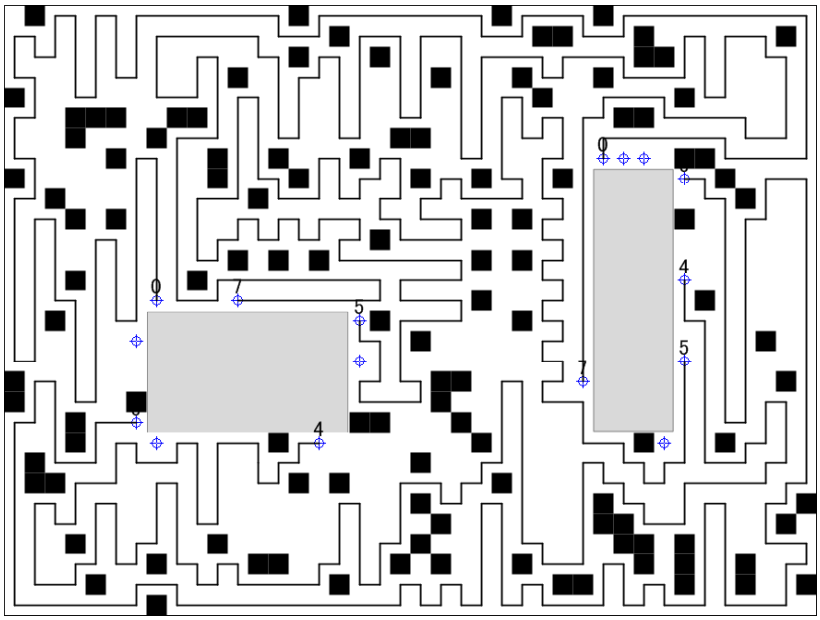


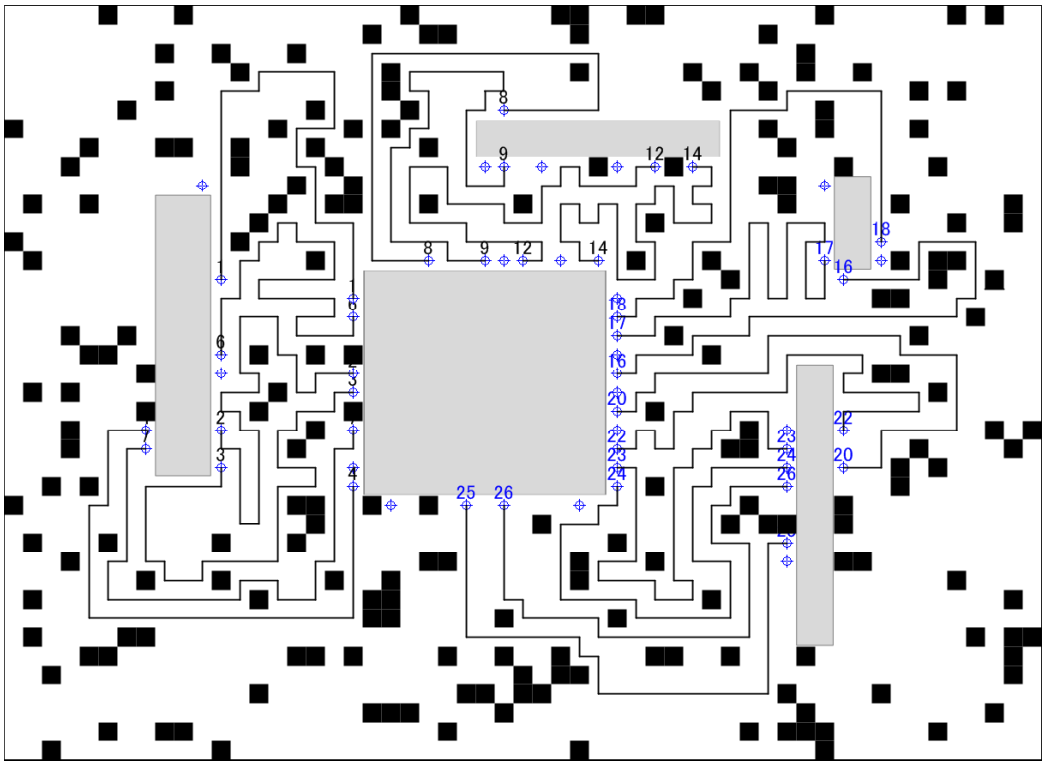
Figure 3.22 Worst length error comparison at each target length

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 [46], 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, Figure 3.22 shows the worst length error comparison between the proposed method and the method [46]. From this figure we obtain that, executed no matter by the method [46] 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



(a) Data01 of $l_t = 190$



(b) Data04 of $l_t = 80$

Figure 3.23 Routing results of layer1 in Experiment 3

than or equal to that by the method [46], which confirms that our method is effective in reducing worst length error, and the average reduction is 36.69%. 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, $|\text{target length} - \text{average length}|$) 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.

Figure 3.23 (a) and (b) show the first layer routing results of Data01 of l_t = 190 and Data04 of l_t = 80, respectively. Combining the results in Table 3.4 and Figure 3.23, 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.

3.6 Conclusions

In this chapter, a region-aware layer assignment and equal-length routing method for disordered pins in PCB design is proposed, where the longest common subsequence (LCS) algorithm and single commodity flow are effectively combined. The approach initially checks the LCS 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. By using the proposed method, it is able to obtain the routes with better wire-length balance and smaller worst length error in reasonable CPU time. The experimental results show that, the proposed method could be applied in both no-obstacle routing and obstacle-ware routing problems. Compared with another greedy method for disordered pins, the proposed method gets a smaller standard deviation, in other words, a better wire-length balance among the nets, by adopting coefficient α to adjust the wire-length skew. Besides, our method is effective in reducing worst length error, and the average reduction is 36.69%.

Chapter 4

Sorting-based I/O Pad Assignment and Non-Manhattan RDL Routing

In this chapter, a sorting-based I/O pad assignment and non-Manhattan RDL routing method is proposed for area I/O flip-chip design. The approach initially assigns the I/O pads to bump balls by sorting the Manhattan distance between them. Three kinds of pair-exchange procedures are then carried out to improve the initial assignment. Then to shorten the wire-length, non-Manhattan RDL routing is adopted to connect the I/O pads and bump balls. Finally some un-routed connections are ripped-up and rerouted.

4.1 Introduction

In modern PCB design, flip-chip package is widely used to meet the higher integration density and the larger I/O counts of circuits, which describes the method of electrically connecting the die to the package carrier. Flip-chip technology becomes the choice in high-speed applications because of its high speed, low power, smaller die size, higher signal density, lower thermal effect and so on.

As we have introduced in Chapter 1, the flip-chip structures can be classified into peripheral I/O flip-chip and area I/O flip-chip, and area I/O flip-chip is more popularly due to its shorter wire-length and smaller package size. For the flip chip designs, RDL is often used to redistribute the I/O pads to the bump balls and the wires can be routed in either 90 or 45 degrees wire segments. For the pre-assignment RDL routing problem, where the connections between I/O pads and bump balls are assigned before routing, wire-length minimization is usually focused on to improve the whole circuit performance.

In this study, given a set of I/O pads and bump balls, we propose a sorting-based I/O pad assignment and non-Manhattan RDL routing method in single layer for area I/O flip-chip design. The primary objective of this study is to reduce the total wire-length, meanwhile improve the routability as far as possible. The whole design process is composed of four phases. In the first phase, we generate the I/O pad assignment by sorting the Manhattan distance between I/O pads and their nearest bump balls. In the second phase, three kinds of pair-exchange procedures are carried out to modify the initial assignment. The exchanges are used to shorten the total wire-length and improve the routability. In the third phase, for each net, firstly the routing is determined by maze routing algorithm, and then non-Manhattan modification is applied to shorten the wire-length. Finally some un-routed connections are ripped-up and rerouted in the last phase. The experimental results show that the proposed method is able to obtain the routes with shorter wire-length in reasonable CPU time.

The remainder of this chapter is organized as follows: Section 4.2 describes some previous works related to this study. Section 4.3 describes the problem definition of this work. Section 4.4 details the design algorithm of the four phases. Section 4.5 illustrates the experimental results and analysis. Finally, Section 4.6 concludes this chapter.

4.2 Related Works

As mentioned earlier, flip-chip package is widely used in modern PCB design. A series of works have been published to handle I/O pad assignment and RDL routing problems, for both a peripheral I/O flip-chip and an area I/O flip-chip.

For peripheral I/O flip-chip designs, [50] adopted a weighted bipartite matching algorithm to generate a set of I/O connections between I/O pads and bump balls. In [11] and [51], a network flow algorithm was used to solve the assignment of wire-bonding pads to bump pads and then complete the routing for each net. In [52], an integer-programming-based algorithm was presented to find an optimal solution for the problem. In [53], a heuristic approach based on routing sequence exchange for pre-assignment flip-chips was proposed.

For area I/O flip-chip designs, some researches focus on the chip-package co-design problem, which considers both the package-level RDL routing and the chip-level routing. In [8] and [12], a work to handle the multi-RDLs routing problem for chip-package co-design was introduced. An area I/O RDL routing problem considering wire-length minimization and chip-package co-design was discussed in [54].

Besides, for area I/O flip-chip designs, some researches talk about the package-level RDL

routing only. In [55], a partition-based assignment was proposed to assign I/O pads for a flip-chip design. In this method, based on the recursive partition of I/O pads and bump balls, the assignment can be obtained by using the geometrical mapping. However, this partition neither consider the wire-length in the I/O assignment phase, nor take measures to improve the assignment afterward, which may cause longer wire-length. In [56], a work used Delaunay-triangulation method to assign all the I/O pads. Although some pair-exchange modifications are carried out in this method, the ordering of the exchange is not so effective, and there still remain some crossing connections after the pair-exchange, which may lead to un-routable connections or redundant wire-length.

In this study, for the I/O pad assignment, we take the Manhattan distance between I/O pads and bump balls into consideration. Then three kinds of pair-exchange procedures are carried out to improve the initial assignment, meanwhile exchange order is considered. Moreover, as mentioned above, the wires can be routed in either 90 degrees or 45 degrees in a RDL, thus we adopt non-Manhattan routing to shorten the total wire-length. The following sections discuss the proposed method in detail.

4.3 Problem Definition

In this chapter, the I/O pad assignment and RDL routing problem is defined as follows: the input includes I/O pads and bump balls; it outputs the routes of assigned connections. The primary objective of this study is to reduce the total wire-length, meanwhile improve the routability as far as possible.

Let $P = \{p_1, p_2 \dots p_n\}$ and $B = \{b_1, b_2 \dots b_m\}$ be the set of n I/O pads and the set of m bump balls ($n<=m$), respectively. There is capacity constraint in the space between any pair of adjacent bump balls. Basically, each I/O pad should be assigned to a bump ball. Sometimes, two or more I/O pads should be assigned to the unique bump ball. If one bump ball connects to the unique I/O pad, this connection is considered as a two-pin net. On the other hand, if one bump ball connects more than one I/O pad, the connection is called a multi-pin net. In this study, we formulate that it is only allowed to route the wires in a single RDL.

As illustrated in Figure 4.1, there are 16 I/O pads and 16 bump balls for I/O pad assignment and RDL routing. It is assumed that the capacity constraint is 2. Note that, there are two I/O pads with the same number, p_2, which must be assigned to the same bump ball. In this study, all the distance and wire-length are defined as the number of grids.

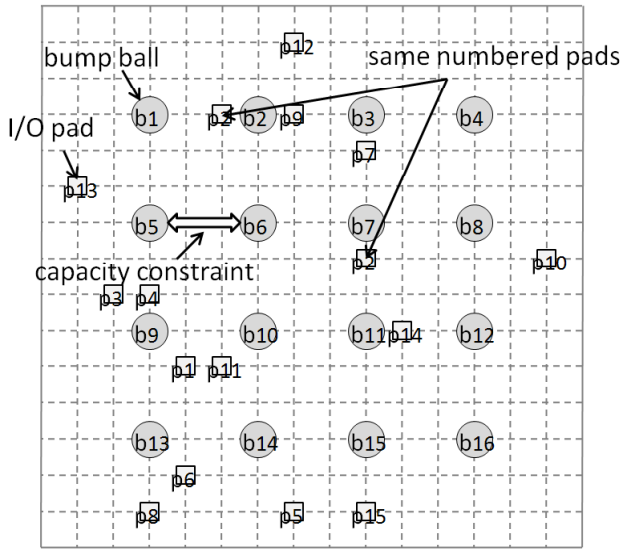


Figure 4.1 Problem definition

4.4 Design Algorithm

In this study, the proposed method includes four sequential phases: initial I/O pad assignment, pair-exchange modification, non-Manhattan RDL routing, rip-up and reroute. The flow chart of the whole design process is shown in Figure 4.2. Algorithm 1 to Algorithm 5 are described in detail in the following sub-sections.

4.4.1 Initial I/O Pad Assignment

Given the placement of I/O pads and bump balls, according to the location of them, initially the I/O pads are assigned to the bump balls. The Manhattan distance between I/O pad and bump ball is taken into consideration for the assignment. The pseudo-code of this phase is shown in Algorithm 1. Note that, if there are more than one I/O pads should be assigned to the same bump ball, we temporarily treat these I/O pads independently in this phase.

In Algorithm 1, $P = \{p_1, p_2 \dots p_n\}$ is the set of I/O pads, and $B = \{b_1, b_2 \dots b_m\}$ is the set of bump balls. The set of bump balls assigned to P is noted as *P.connectBall*. We define $MD(p_i, b_j)$ as the Manhattan distance between I/O pad p_i and bump ball b_j. Thus, $MD(p_i, B)$ is the set of Manhattan distances between p_i and each bump ball. We use a set of T to represent the I/O pads assigned to the same b_j. Similar to the definition of $MD(p_i, B)$, $MD(b_j, T)$ is the set of Manhattan distances between b_j and each pad in the set T. Algorithm 1 is processed by the following two

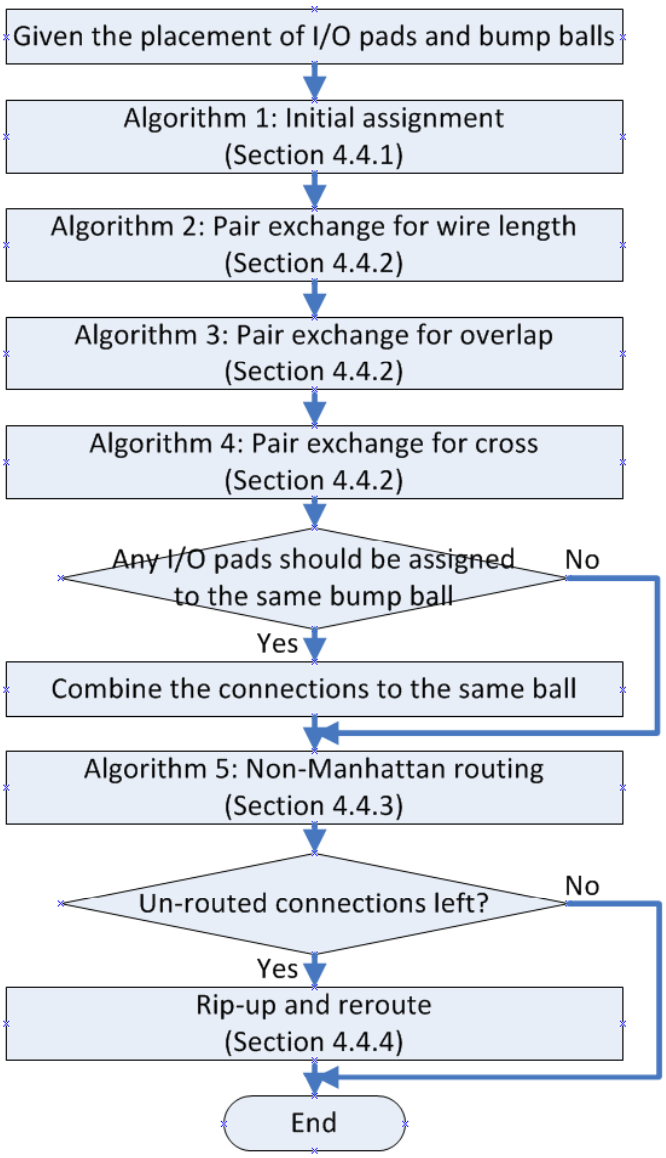


Figure 4.2 Flow chart of I/O pad assignment and RDL routing process

steps.

[Procedure: Initial I/O pad assignment]

Step 1: Pre-assignment between I/O pads and bump balls;

Step 2: Revision for the pre-assignment to make the connection for each I/O pad uniquely.

[Step 1] Initially, we pre-assign the I/O pads to one bump ball. By calculating and comparing the Manhattan distance between one I/O pad and each non-connected bump ball, we can get the bump ball with the shortest Manhattan distance for this I/O pad. This bump ball is pre-assigned to this I/O pad temporarily.

Algorithm 1. Initial I/O pad assignment

Input: Placement of I/O pads and bump balls
Output: I/O pad assignment to bump balls

```
begin
  for i=1 to n do
    sort MD(p_i, B) in ascending order;
    p_i.connectBall=the first bump ball in the sequence;
  end for
  for j=1 to m do
    for i=1 to n do
      if p_i.connectBall is b_j then
        put pi into a set T;
      end if
    end for
    sort MD(b_j, T) in ascending order;
    assign b_j to the first I/O pad in the sequence;
  end for
  delete assigned I/O pads and bump balls from both sequences;
  repeat the process until all I/O pads are assigned;
  output P and P.connectBall;
end
```

[Step 2] Then we revise the pre-assignment to make the connection for each I/O pad uniquely. If there is just one I/O pad assigned to the unique bump ball, the connection pair can be determined. Otherwise, if there is more than one I/O pad assigned to the same bump ball, we should sort these I/O pads in ascending order according to the calculated Manhattan distances. Then, choose the first I/O pad in the sequence since it is the nearest one to the same numbered bump ball, and put the other I/O pads into the next loop until all the I/O pads are assigned to non-connected bump balls.

Take the case in Figure 4.3 to explain Algorithm 1. As there are two I/O pads with the same number p_2, one of them is remembered as p_{16} to temporarily treat independently. In the first loop of initial I/O pad assignment, after the process of Step 1, the I/O pads and their pre-assigned bump balls are listed in Table 4.1. From this table we can obtain that p_2, p_5, p_7, p_{10}, p_{11}, p_{13}, p_{14} and p_{15} are assigned to their unique bump ball, so in Step 2, the connection of them can be firstly determined. Then the sorting for the set of p_9, p_{12} and p_{16} with the same b_2, the set

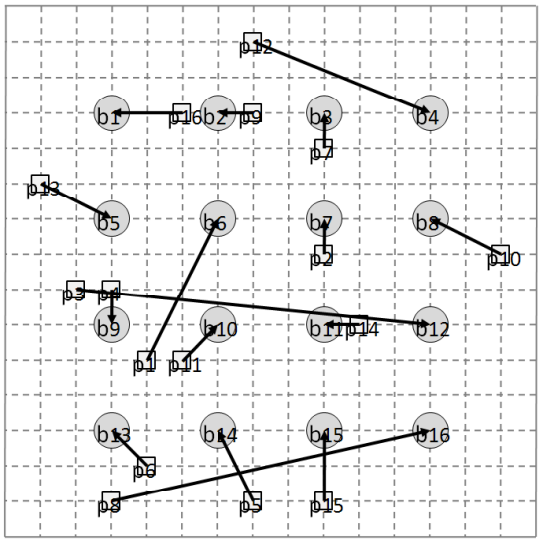


Figure 4.3 Initial I/O pad assignment

Table 4.1 First loop of pre-assignment

P	*P.connectBall*	*MD(P, P.connectBall)*
p_1	b_9	2
p_2	b_7	1
p_3	b_9	2
p_4	b_9	1
p_5	b_{14}	3
p_6	b_{13}	2
p_7	b_3	1
p_8	b_{13}	2
p_9	b_2	1
p_{10}	b_8	3
p_{11}	b_{10}	2
p_{12}	b_2	3
p_{13}	b_5	3
p_{14}	b_{11}	1
p_{15}	b_{15}	2
p_{16}	b_2	1

of p_1, p_3 and p_4 with the same b_9, and the set of p_6 and p_8 with the same b_{13} are shown in Table 4.2. According to the algorithm, we can determine the connection of p_9, p_4 and p_6 that are first in sequence. After the first loop, five I/O pads p_1, p_3, p_8, p_{12} and p_{16} have not been assigned, which are put into next loop. Finally after three loops, all the I/O pads are assigned to different bump balls.

Table 4.2 First loop of revision

P.connectBall	*P*	*MD(P, P.connectBall)*
b_2	p_9	1
	p_{16}	1
	p_{12}	3
b_9	p_4	1
	p_1	2
	p_3	2
b_{13}	p_6	2
	p_8	2

4.4.2 Pair-exchange Modification

To make the routing result better, the initial I/O pad assignment result should be further modified. Three kinds of pair-exchanges are carried out to modify the assignment. The exchange order is according to the descending Manhattan distance between the assigned pad and ball pairs. In other words, we first handle the connections whose distance is longer, then the shorter ones. The reason is that, when longer connections are exchanged, some other shorter connections that also should be exchanged may be solved in advance, especially for the crossing exchange. This exchange order may reduce the number of pair-exchange. For example in Figure 4.4 (a), there are three crossing connections. We first exchange the longer pairs, and after only once pair-exchange, all the crossing connections are solved, as shown in Figure 4.4 (b). However, if we first exchange the shorter pairs, there is still one crossing connection after once pair-exchange, as shown in Figure 4.4 (c), and then once more pair-exchange should be carried out. According to this pair-exchange order, we make the modification.

The first modification is pair-exchange for shortening wire-length, whose pseudo-code is shown in Algorithm 2.

Take the example in Figure 4.5 to illustrate this algorithm. We compare the sum Manhattan distance of every two assigned pairs with the sum Manhattan distance of these two pairs after pair-exchange. If the sum Manhattan distance after pair-exchange is less than the previous one, this pair-exchange is accepted. In this example, the sum Manhattan distance of the pair is 1+5=6, shown in Figure 4.5 (a). As the distance after exchange is 2+2=4, which is less than 6, the exchange for this pair is carried out as shown in Figure 4.5 (b).

The second modification is pair-exchange to release the overlap of the connection regions, and the pseudo-code is shown in Algorithm 3.

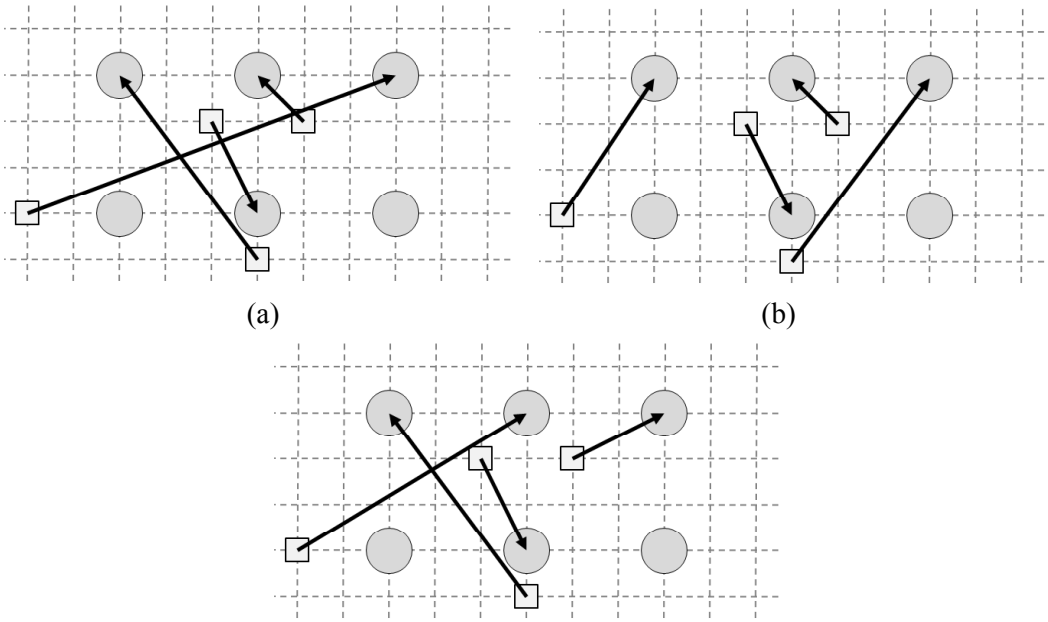


Figure 4.4 Pair-exchange order

Algorithm 2. Pair-exchange modification for shortening wire-length

Input: I/O pads, assigned bump balls

Output: new I/O pad assignment to bump balls

begin

for *i=1 to n* **do**

Sort p_i *in descending order of* $MD(p_i, p_i.connectBall)$;

end for

for *i=1 to n* **do**

for *j=1 to n* **do**

if $MD(p_i, p_j.connectBall)+MD(p_j, p_i.connectBall)<MD(p_i, p_i.connectBall)+MD(p_j, p_j.connectBall)$ **then**

exchange the pair of p_i *and* p_j;

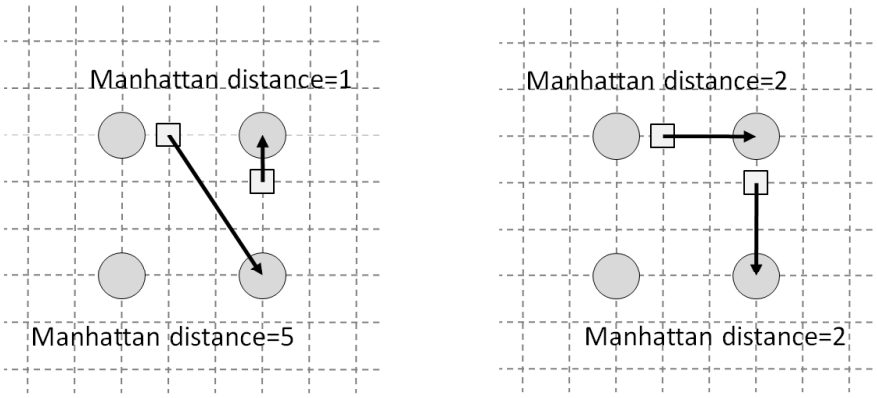
end if

end for

end for

output P and P.connectBall;

end



(a) Before pair-exchange (b) After pair-exchange

Figure 4.5 Pair-exchange for shortening wire-length

Algorithm 3. Pair-exchange modification for releasing overlap

Input: I/O pads, assigned bump balls

Output: new I/O pad assignment to bump balls

begin

  for *i=1 to n* **do**

    for *j=1 to n* **do**

      if *OL(p_i, p_i.connectBall, p_j, p_j.connectBall) is true && OL(p_i, p_j.connectBall, p_j, p_i.connectBall) is false* **then**

        exchange the pair of p_i and p_j;

      end if

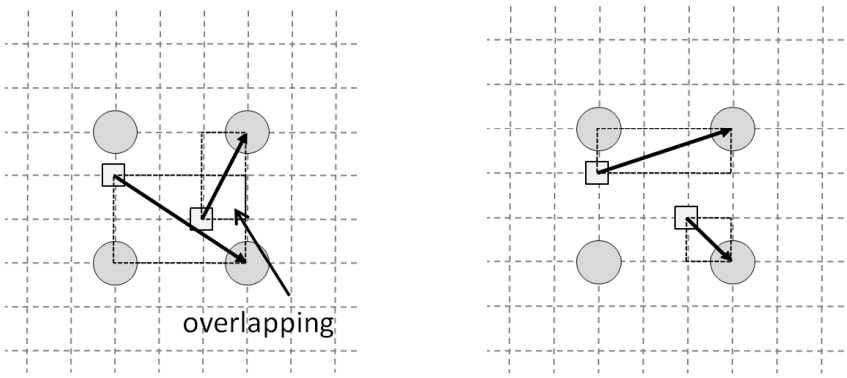
    end for

  end for

  output new P and P.connectBall;

end

The region of connection is defined as a rectangle whose one pair of diagonal vertices are the I/O pad and the bump ball of this connection. The overlap of the regions of two connections will affect the routability in the single layer routing. As a result, the pair of overlap should be exchanged. In Algorithm 3, we define *OL(p_i, p_i.connectBall, p_j, p_j.connectBall)* as the overlap relation between the region of p_i and p_i.*connectBall*, and the region of p_j and p_j.*connectBall*. If the regions of two connections overlap with each other (Figure 4.6 (a)), but after pair-exchange this overlap can be reduced (Figure 4.6 (b)), the pair-exchange for these two connections is



(a) Before pair-exchange (b) After pair-exchange

Figure 4.6 Pair-exchange for releasing overlap

Algorithm 4. Pair-exchange modification for avoiding cross

Input: I/O pads, assigned bump balls

Output: new I/O pad assignment to bump balls

begin

  for *i=1 to n* **do**

    for *j=1 to n* **do**

      if *connections of* p_i *and* p_j *have crossing point* **then**

        exchange the pair of p_i *and* p_j*;*

      end if

    end for

  end for

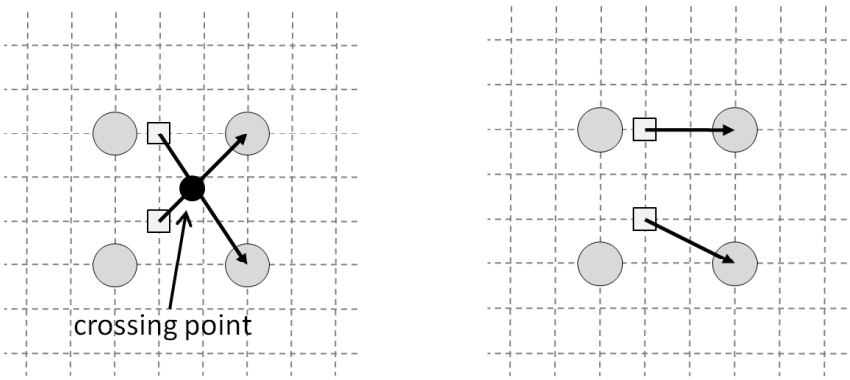
  output new P and P.connectBall;

end

carried out. Otherwise, if the overlap cannot be solved after the pair-exchange, we do not any process.

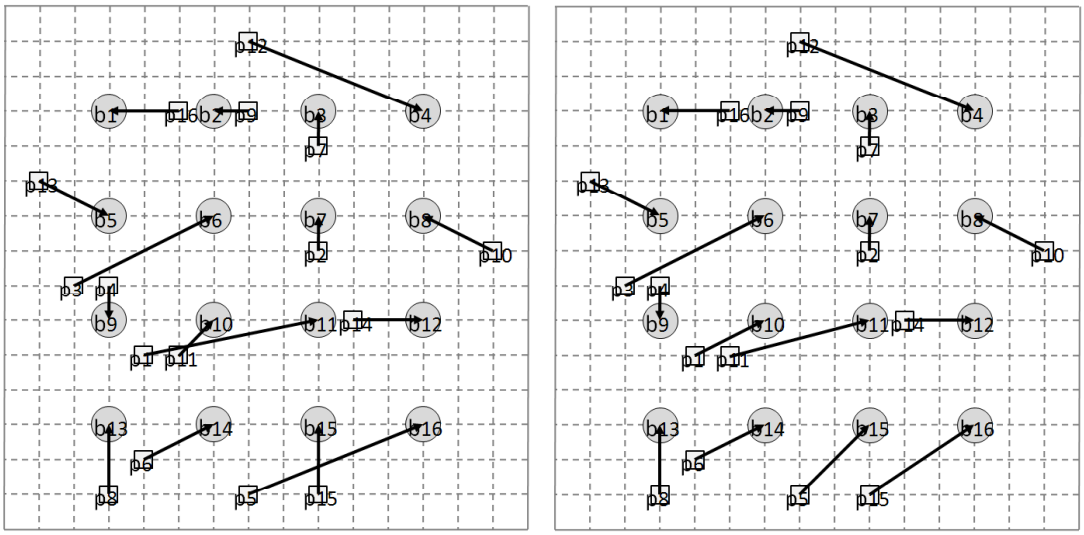
The third modification is pair-exchange to avoid crossing connections, and the pseudo-code is shown in Algorithm 4.

As mentioned earlier, we formulate that it is only allowed to route wires in single RDL in this study. If there are crossing connections, it will lead to un-routable result or detoured result with redundant wire-length. Hence, the pair of crossing connections should be exchanged. To solve the crossing problem, first we compare the slopes of every two connections' line segments. If the slopes are not the same, it means that the straight lines of these two connections may have



(a) Before pair-exchange (b) After pair-exchange

Figure 4.7 Pair-exchange for avoiding cross



(a) Shortening wire length (b) Releasing crossing connections

Figure 4.8 Pair-exchange modification

crossing point. Then we check whether the crossing point is on these two I/O connections line segments. If the crossing point is on the line segments as shown in Figure 4.7 (a), the pair-exchange should be done to remove the crossing, shown in Figure 4.7 (b). Otherwise, we do not any process.

Refer to the initial I/O pad assignment result in Figure 4.3, by Algorithm 2, the pair of p_1, p_3, p_5, p_6, p_8 and p_{14} are exchanged, the modified result is shown in Figure 4.8 (a). Then by Algorithm 3, the result does not change for this example. Finally by Algorithm 4, the pair of p_1, p_5, p_{11} and p_{15} are exchanged, the modified result is shown in Figure 4.8 (b).

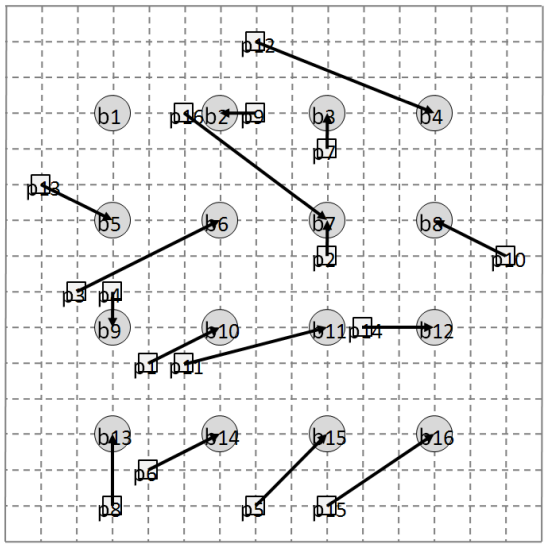


Figure 4.9 Combination of some connections

After completing the I/O pad assignment, we should solve the problem in the very beginning: combine the connection of the I/O pads which should be assigned to the unique bump ball. For the example above, p_2 and p_{16} are combined to b_7 as shown in Figure 4.9.

4.4.3 Non-Manhattan RDL Routing

In recent PCB technologies, non-Manhattan routing is usually utilized to reduce wire-length [57]. As we mentioned in above sections, in a RDL, the wires can be routed in either 90 degrees or 45 degrees by current technology. In this phase, we finish the RDL routing with non-Manhattan wires to get shorter wire-length. For all assigned connections, the routing order is determined by the calculated Manhattan distances between I/O pads and bump balls in ascending order. Algorithm 5 shows the pseudo-code of the routing procedure for each net.

In Algorithm 5, we define $S = \{s_1, s_2 \ldots s_n\}$ is the set of bending points in the initial connection path obtained by maze routing algorithm, c_0, c_1 and c_2 are defined as the crossing points of 45 or 135 degrees line segments generated from T (same as T in Algorithm 1) and the initial connection path. Algorithm 5 is processed by the following four steps.

[Procedure: Non-Manhattan RDL routing]

Step 1: Wire direction decision;

Step 2: Setting escape points for I/O pads and bump balls if necessary;

Step 3: Generate the initial connection path by maze routing algorithm;

Algorithm 5. Non-Manhattan RDL routing

Input: I/O pads, assigned bump balls

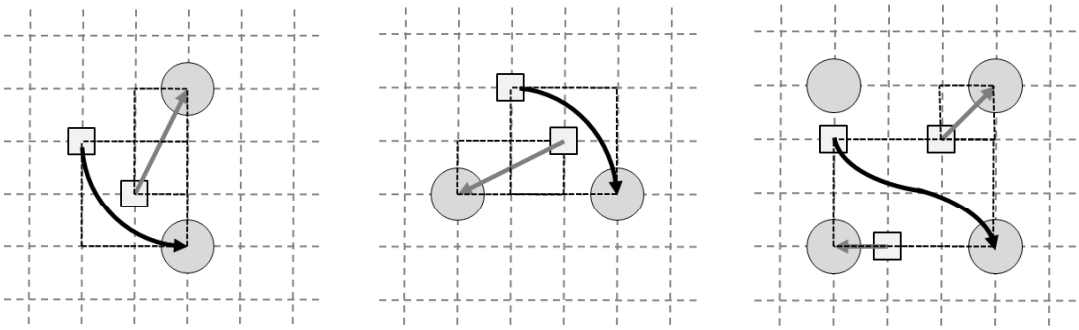
Output: connection path between I/O pads and bump balls

begin
  wire direction decision;
  set escape points;
  initial connection path generated by maze routing;
  for *i=1 to number of inflection points in connection path* **do**
    generate 45 and 135 degrees lines from s_i*;*
    if c_0 *on connection path* **then**
      if *obstacle on the line segment between* s_i *and* c_0 **then**
        parallel shift the crossing line;
        if *crossing point* c_1 *and* c_2 *on connection path* **then**
          add 45 or 135 degrees line segment between c_1 *and* c_2 *into connection path;*
          delete previous 0 and 90 degrees line segment between c_1 *and* c_2*;*
        end if
      else *add 45 or 135 degrees line segment between* s_i *and* c_0 *into connection path;*
          delete previous 0 and 90 degrees line segment between s_i *and* c_0*;*
      end if
      update S;
    end if
  end for
  output connection path between P and B;
end

Step 4: Utilize non-Manhattan modification to shorten the wire-length.

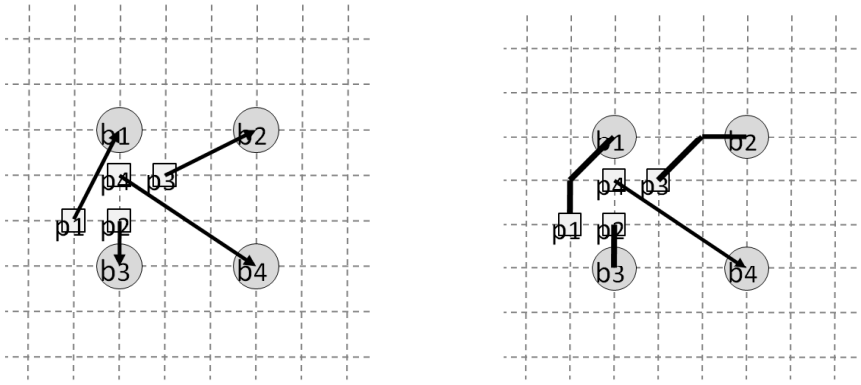
[Step 1] Initially, to avoid the congestion area for RDL routing, we decide the wire direction by global routing. As the I/O pad assignment has been obtained, according to the region overlap relation among all the connections, we assign the wire direction for each connection to avoid the congestion area. Take Figure 4.10 for example, assuming that the bump ball is in lower right of the I/O pad, as shown in Figure 4.10 (a), if there are obstacles (routing region of other connections, routed wires, or unassigned bump balls) in right side of this connection's region, the direction is decided as vertical-horizontal from pad to ball. Similarly, if there are obstacles in left side, the direction is decided as horizontal-vertical (Figure 4.10 (b)). Moreover, if both sides are obstructed, as shown in Figure 4.10 (c), it is decided as vertical-horizontal-vertical, in the middle of the region. Otherwise, the wire direction is decided as vertical-horizontal. By the same method, the wire direction of other positions of I/O pads and bump balls also can be decided.

[Step 2] Then, we set escape points for some I/O pads and bump balls if necessary. Figure

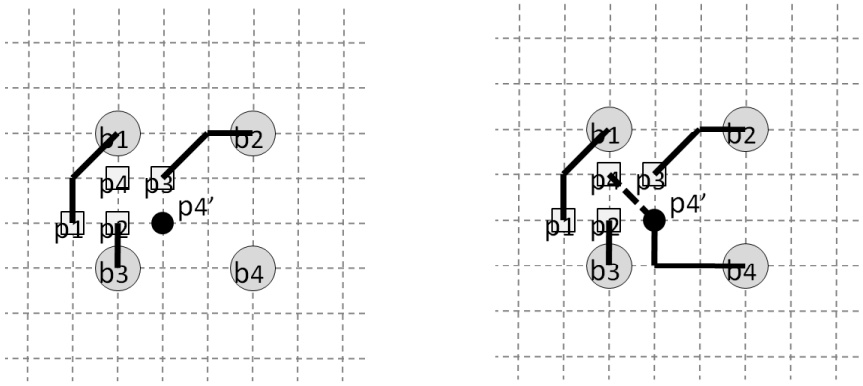


(a) (b) (c)

Figure 4.10 Wire direction decision



(a) I/O connection assignment (b) Un-routable connection



(a) Escape point p_4' (b) Route by p_4'

Figure 4.11 Configuration of escape points

4.11 (a) shows an example where the I/O connections for *P* and *B* have been assigned. According to the routing order, p_1, p_2 and p_3 are first routed as shown in Figure 4.11(b). Then we can find that it is un-routable for p_4 as the V-H routing should be initially carried out. But we can apply the non-Manhattan wires, so we do a pretreatment before routing. For one I/O pad or

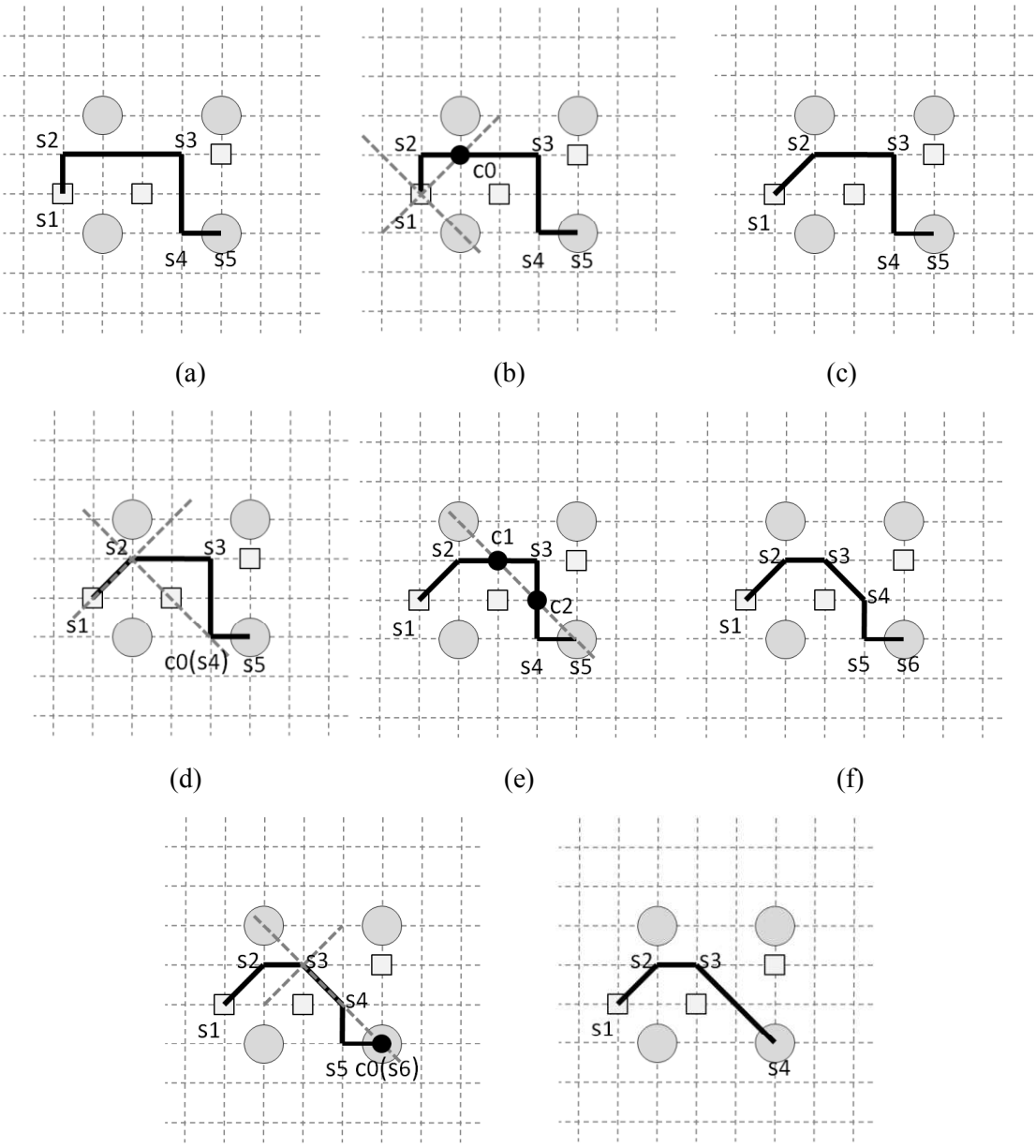
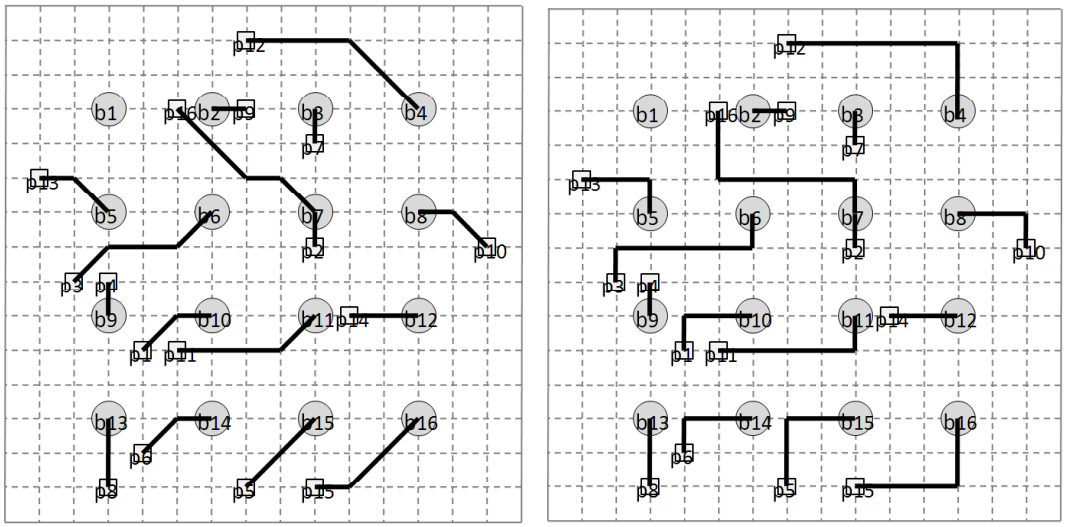


Figure 4.12 Non-Manhattan modification

a bump ball, if its 0 degree and 90 degrees directions are blocked by other pins, we set an escape point in its 45 degrees or 135 degrees direction to make the wire routable. As shown in Figure 4.11 (c), for p_4, its 0 degree and 90 degrees directions have been blocked, so we set an escape point p_4' in its 135 degrees direction, which is in the direction of target b_4. Then we can obtain the connection path by the help of p_4' as shown in Figure 4.11 (d).



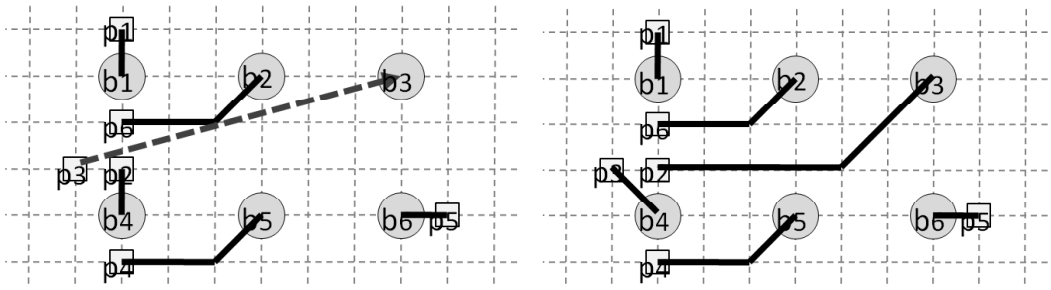
(a) Non-Manhattan routing (b) Manhattan routing

Figure 4.13 RDL routing result

[Step 3] In this step, we generate the initial connection path by V-H routing, according to the wire direction obtained by the first step. As the scale for each pair of I/O pad and bump ball is not too large, maze routing algorithm is adopted for this initial routing. After V-H routing from *P* to *B*, the bending points in connection path are stored in *S*.

[Step 4] Finally, we utilize non-Manhattan modification to shorten the wire-length. We also use an example to illustrate this modification. As shown in Figure 4.12 (a), the initial V-H routing path is generated by the last step. There are five bending points of this path, marked as s_1 to s_5. Then in Figure 4.12 (b), we draw 45 and 135 degrees lines from s_1. As there is a crossing point c_0 on the existing path, we add the 45 degrees line segment between s_1 and c_0 into connection path. Then delete previous 0 and 90 degrees line segments between s_1 and c_0 and update the set of *S*, as shown in Figure 4.12 (c). Next, we make the modification for s_2 by the same method. From Figure 4.12 (d) we can find that this time there is an obstacle on the line segment between s_2 and c_0, so the crossing 135 degrees line is shifted in parallel to find whether there exist other crossing points. As a result, the new crossing points, c_1 and c_2 can be obtained, as shown in Figure 4.12 (e). Then this new 135 degrees line segment between c_1 and c_2 is added, the previous 0 and 90 degrees line segments between them are deleted and the set of *S* is updated (Figure 4.12 (f)). Next, s_3 are modified as shown in Figure 4.12 (g) and the result is shown in Figure 4.12 (h). Finally we make the modification for the last s_4 but there is no crossing point on the connection path, so we do no process on it. Until here, this step is over.

Refer to the I/O pad assignment result in Figure 4.9, by Algorithm 5, the RDL routing



(a) (b)

Figure 4.14 Ripping up and rerouting

result is shown in Figure 4.13 (a). From this figure we can obtain the total wire-length is 45 by non-Manhattan routing. Compared with Manhattan routing in Figure 4.13 (b), whose total wire-length is 54, the non-Manhattan routing is effective for shortening wires.

4.4.4 Rip-up and Reroute

In this study, since single RDL routing is considered, if the routing paths of some connections are intersected by other routed wires, these connections may fail to route. Therefore, finally we rip-up and reroute these un-routable connections to improve the routability, if any.

The rip-up and reroute is realized by exchanging the un-routable connection and its adjacent connection, which are introduced in [55]. The exchange order depends on the Manhattan Distance between the un-routable I/O pad and other bump balls in ascending order. In other word, the bump ball which is nearest to the un-routable I/O pad is first to be exchanged. As shown in Figure 4.14 (a), the connection of p_3 cannot be routed. Hence, we rip-up the nearest connection b_4, exchange it with p_3, and reroute them respectively. The routing result is shown in Figure 4.14 (b). For one un-routed connection, the rip-up and reroute may be executed several times until it is routable or all the exchanging cases are tried. However, there may remain some un-routable connections in the end. By this rip-up and reroute procedure, we improve the routability as far as possible.

4.4.5 Discussion on Time Complexity

As mentioned above, the proposed method is divided into four phases: initial I/O connection assignment, pair-exchange modification, non-Manhattan RDL routing, rip-up and reroute. For initial I/O connection assignment, as we assign the I/O pads to bump balls by sorting the

Manhattan distance between them, the time complexity of this phase is O(mn), where m, n are the number of bump balls and I/O pads, respectively. Since the pair-exchange modification is carried out between any pair, the time complexity of this phase is O(n^2). In non-Manhattan RDL routing, for one connection the wire-length is O(u) in the worst case, where u is the number of grids. So the time complexity of routing all connections is O(nu). For rip-up and reroute, in the worst case, all the connections should be ripped-up and exchanged with each other, so the time complexity of this phase is O(n^2). Since u is much larger than m and n, the total time complexity of the proposed method is about O(nu).

4.5 Experimental Results and Analysis

We implemented our proposed sorting-based I/O pad assignment and non-Manhattan RDL routing method in C language, which is executed on a PC with 3.40GHz Intel Core 2 CPU and 8GB RAM. The partition-based method in [55] and Delaunay-triangulation method in [56] for I/O pad assignment with V-H routing are also implemented for comparison. Moreover, for the proposed method, both Manhattan routing and non-Manhattan routing are executed respectively. Three experiments are carried out. Experiment 1 is on comparison with method [55] and [56], which executed on the same package size but different pad location. Experiment 2 is also on comparison with method [55] and [56], but on different package sizes from small to large. Experiment 3 is executed on obstacles for our method.

Experiment 1

For Experiment 1, there are five test data named Data01 to Data05 with the same package size 5*5 and fixed location of bump balls. For each data, the location of I/O pads is randomly set, and the capacity constraint is set as 2. The experimental results are listed in Table 4.3, where "#*P*" denotes the number of I/O pads, and "#*B*" denotes the number of bump balls. Wire-length is given in accordance with the number of unit grid, and the routability and execution time of these four methods are also listed in this table. Take the wire-length of the proposed method followed by Manhattan routing as basal value 100%, the wire-length percentage of all methods is illustrated in Figure 4.15. From the experimental results we can see that, all methods achieve 100% routability for all test data. Compared with method [55], our proposed method, no matter followed by Manhattan routing or non-Manhattan routing, obtains shorter wire-length within a small CPU time. And relative to Manhattan routing, non-Manhattan routing can get much

Table 4.3 Experimental results on the same package size but different location

	#P	#B	Method [55] + Manhattan Routing			Method [56] + Manhattan Routing			Proposed Method + Manhattan Routing			Proposed Method + Non-Manhattan Routing		
			Routability	*Wire Length*	*CPU Time*	*Routability*	*Wire Length*	*CPU Time*	*Routability*	*Wire Length*	*CPU Time*	*Routability*	*Wire Length*	*CPU Time*
Data01	25	25	100%	114	0.015 s	100%	88	0.024 s	100%	90	0.014 s	100%	71.2	0.015 s
Data02			100%	91	0.014 s	100%	93	0.028 s	100%	83	0.015 s	100%	68.4	0.016 s
Data03			100%	89	0.015 s	100%	65	0.024 s	100%	63	0.014 s	100%	53.6	0.015 s
Data04			100%	88	0.014 s	100%	82	0.026 s	100%	82	0.014 s	100%	67.4	0.016 s
Data05			100%	112	0.014 s	100%	82	0.026 s	100%	80	0.015 s	100%	65.9	0.016 s
Average			100%	98.8	0.014 s	100%	82	0.026 s	100%	79.6	0.014 s	100%	65.3	0.016 s

Table 4.4 Experimental results on different package sizes

	#P	#B	Method [55] + Manhattan Routing			Method [56] + Manhattan Routing			Proposed Method + Manhattan Routing			Proposed Method + Non-Manhattan Routing		
			Routability	*Wire Length*	*CPU Time*	*Routability*	*Wire Length*	*CPU Time*	*Routability*	*Wire Length*	*CPU Time*	*Routability*	*Wire Length*	*CPU Time*
Data06	100	100	100%	430	0.060 s	100%	373	0.086 s	100%	369	0.068 s	100%	310.6	0.078 s
Data07	225	225	100%	883	0.251 s	100%	731	0.298 s	100%	703	0.261 s	100%	592	0.299 s
Data08	400	400	100%	1527	0.733 s	100%	1214	0.822 s	100%	1190	0.759 s	100%	1023	0.880 s
Data09	625	625	100%	2606	1.753 s	100%	2330	1.927 s	100%	2113	1.810 s	100%	1775	2.084 s
Data10	900	900	100%	3386	3.554 s	100%	2969	3.870 s	100%	2825	3.669 s	100%	2532	4.244 s

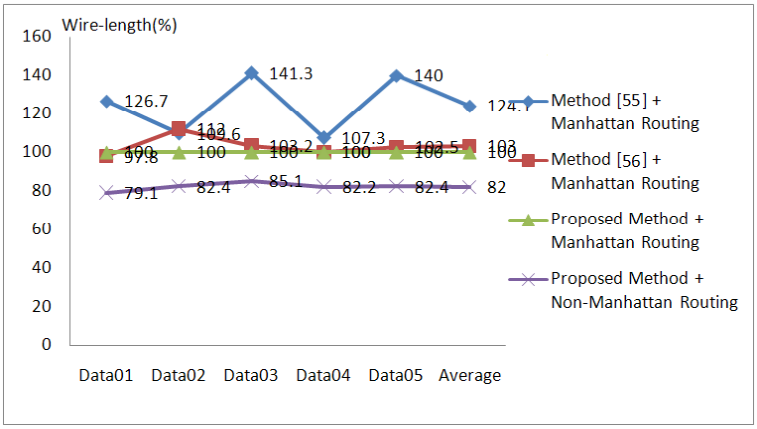


Figure 4.15 Wire length comparison in Experiment 1

shorter length. This mainly owes to the our adoption of pair-exchange modification, since method [55] takes no measure to reduce the wire-length, and the average wire-length reduction is 24.1% by Manhattan routing, and 42.1% by non-Manhattan routing as shown in Figure 4.15.

Then compared with method [56], our proposed method followed by Manhattan routing obtains a little shorter wire-length, except for Data01. Then by non-Manhattan routing, the wire-length is much reduced for all five test data. In addition, the execution time of method [56] is a little larger than ours, which is mainly because of its constructing graph. The average wire-length reduction in our method is also shown in Figure 4.15, which is 3.0% by Manhattan routing, and 21.0% by non-Manhattan routing.

Experiment 2

For Experiment 2, there are five test data Data06 to Data10 with different package sizes from small (10*10) to large (30*30), the location of I/O pads is also randomly set, and the capacity constraint is set as 2. The experimental results are listed in Table 4.4. Same as Experiment 1, we take the wire-length of the proposed method followed by Manhattan routing as basal value 100%, the wire-length percentage of all methods is illustrated in Figure 4.16. The experimental results show that, all the design methods achieve 100% routability for all test data. The comparison results with method [55] and [56] are similar with Experiment 1. The proposed method can generate shorter wire-length not only in small scale but also in large scale packages in reasonable CPU time. Although the execution time of our method by non-Manhattan routing is larger, it is acceptable for practical use. Compared with method [55] and [56], the wire-length reduction in our method are 22.7% and 4.5% on the average by Manhattan routing, and 37.1%

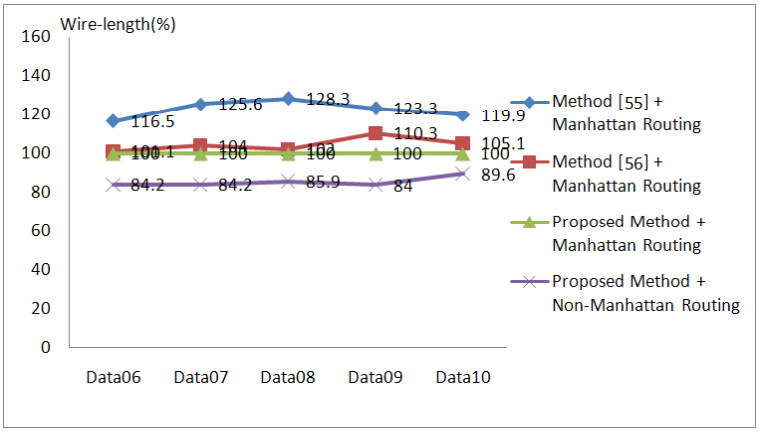


Figure 4.16 Wire length comparison in Experiment 2

and 18.9% on the average by non-Manhattan routing, respectively.

Then, combing the results of Experiment 1 and Experiment 2 for ten test data, we can calculate the overall average wire-length reductions compared with methods [55] and [56] are 23.4% and 3.8% by Manhattan routing, and 39.6% and 20.0% by non-Manhattan routing, respectively. Obviously, the experimental results show that our proposed sorting-based I/O pad assignment is effective on reducing wire-length for flip-chip designs in the reasonable CPU time.

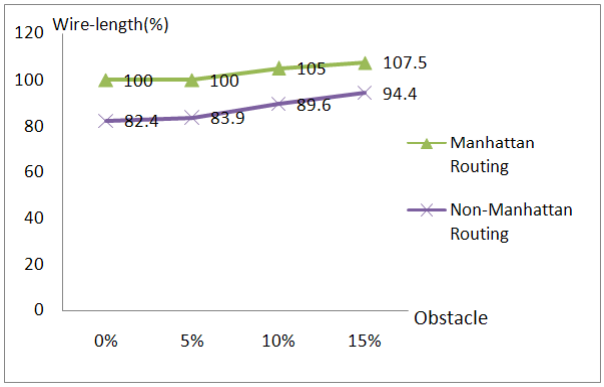
Experiment 3

Finally we test our proposed method for dense routing area in Experiment 3. A small package size Data05 (5*5) and a large package size Data08 (20*20) are tested. For each data, 5%, 10% and 15% of the routing area is randomly set with obstacles respectively, and the experimental results are listed in Table 4.5. In this table, the experimental result without obstacle is also listed. Respective for two package size data, we take the wire-length without obstacle of the proposed method followed by Manhattan routing as basal value 100%, the length percentage of other wires are illustrated in Figure 4.17 (a) and (b). And the routability of all data is illustrated in Figure 4.18. Figure 4.19 and Figure 4.20 give the non-Manhattan routing results for Data 05 and Data08 without obstacles and with 15% obstacles, respectively.

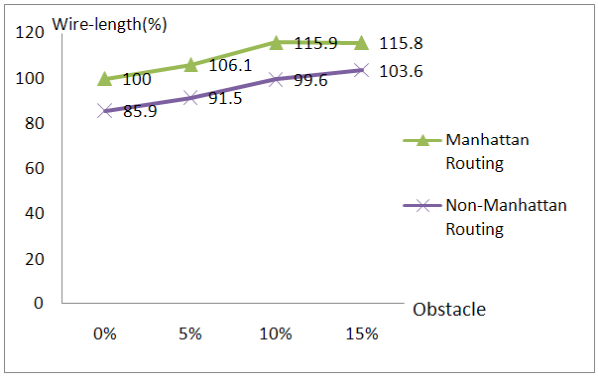
The results in Table 4.5 and Figure 4.17 show that, either the small package size or the large one, the wire-length increases with the increasing of obstacles both by Manhattan routing and non-Manhattan routing, apart from Data08 with 15% obstacles. This exception dues to some un-routable connections. When the obstacles are added, more connections should detour wires

Table 4.5 Experimental results on setting obstacles

	#P	*#B*	*obstacles*	Proposed Method + Manhattan Routing			Proposed Method + Non-Manhattan Routing		
				Routability	*Wire Length*	*CPU Time*	*Routability*	*Wire Length*	*CPU Time*
Data05	25	25	0%	100%	80	0.015 s	100%	65.9	0.016 s
			5%	100%	80	0.015 s	100%	67.1	0.016 s
			10%	100%	84	0.015 s	100%	71.7	0.016 s
			15%	100%	86	0.015 s	100%	75.5	0.016 s
Data08	400	400	0%	100%	1190	0.759 s	100%	1022.5	0.880 s
			5%	99.75%	1263	0.797 s	99.75%	1089.3	0.900 s
			10%	99.50%	1379	0.865 s	99.50%	1185.1	0.982 s
			15%	98.75%	1378	0.965 s	98.75%	1232.4	1.104 s



(a) Data05



(b) Data08

Figure 4.17 Wire length comparison in Experiment 3

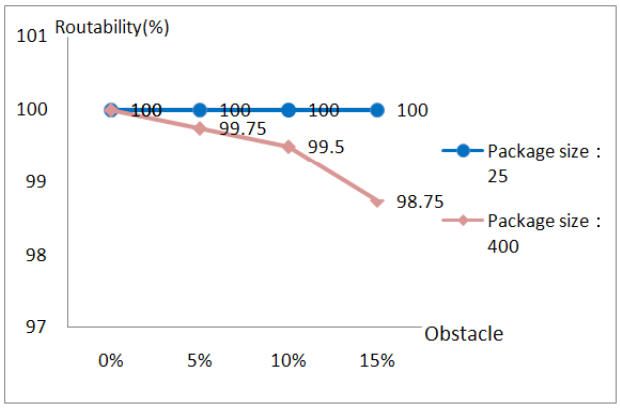
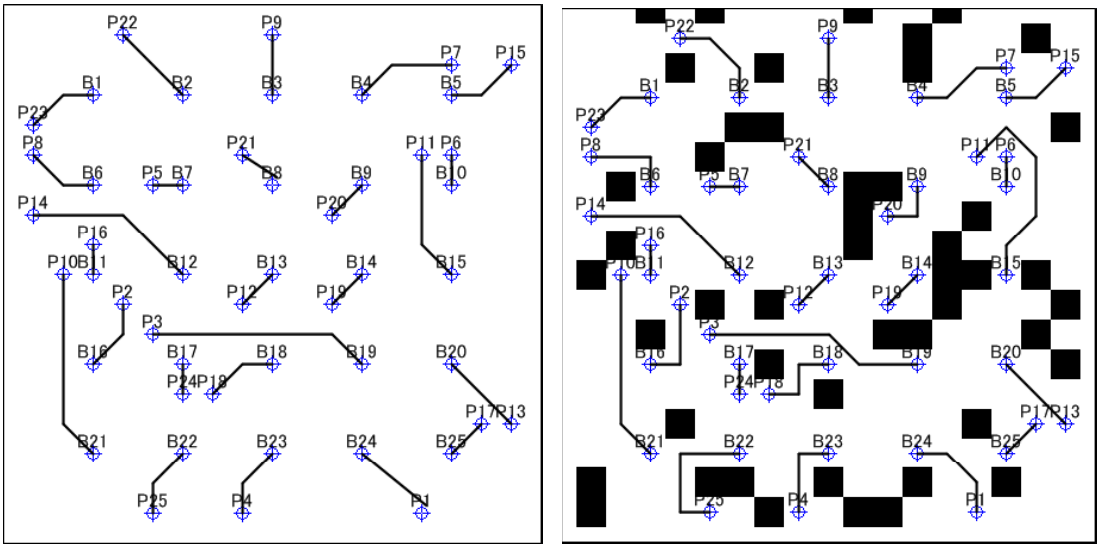


Figure 4.18 Routability comparison in Experiment 3

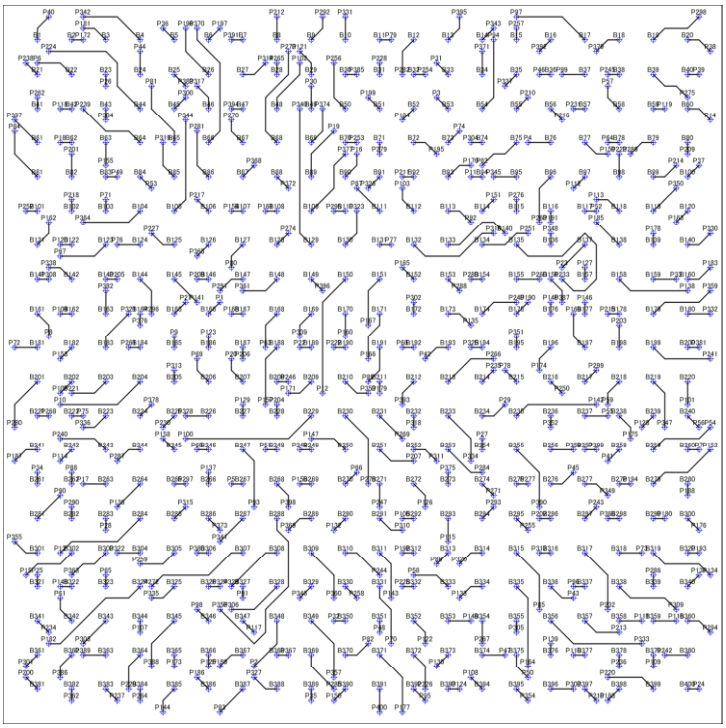


(a) Without obstacles (b) With 15% obstacles

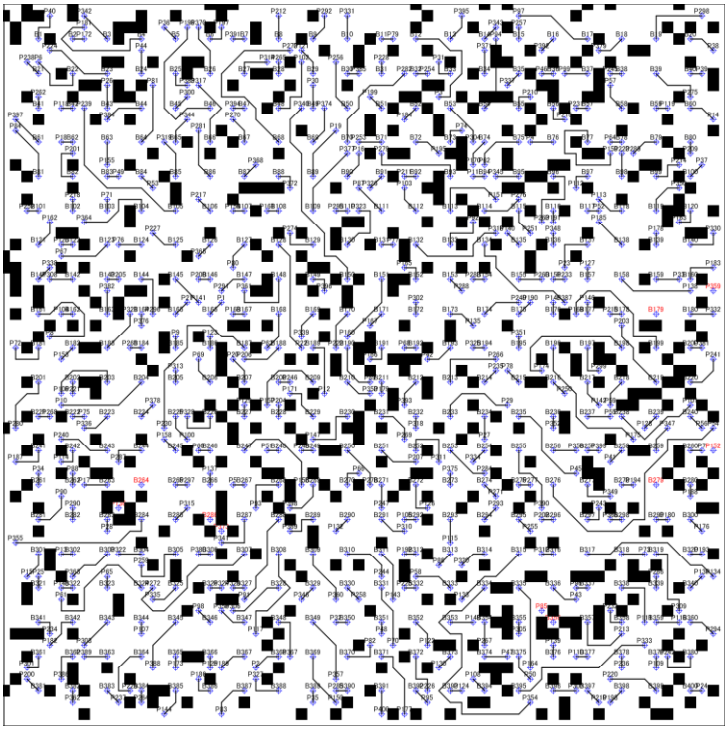
Figure 4.19 Non-Manhattan routing results for Data05 in Experiment 3

to complete the routing, and this detouring results in the increasing of wire-length.

For the routability, combining Figure 4.18, Figure 4.19 and Figure 4.20, we can see that, for small package size, the routability retains 100%. However, if package size is large, because of the large quantity of connections and less available routing area, the routability decreases. When 15% obstacles are set, the routability is reduced to 98.75%. Hence, how to further optimize the I/O pad assignment and improve the routability in the dense routing area remains for our future works.



(a) Without obstacles



(b) With 15% obstacles

Figure 4.20 Non-Manhattan routing results for Data08 in Experiment 3

4.6 Conclusions

In this chapter, we proposed a sorting-based I/O pad assignment for area I/O flip-chip design instead of traditional methods such as partition-based one, and non-Manhattan RDL routing method was applied to reduce the total wire-length. The approach initially assigns the I/O pads to bump balls by sorting the Manhattan distance between them. Three kinds of pair-exchange procedures are then carried out to improve the initial assignment. Then to shorten the wire-length, non-Manhattan RDL routing is adopted to connect the I/O pads and bump balls. Finally some un-routed connections are ripped-up and rerouted. The proposed method is effective on reducing wire-length no matter of the I/O pad locations and package sizes. Compared with a partition-based method, the proposed method can reduce the total wire-length by 23.4% using Manhattan routing, and 39.6% using non-Manhattan routing. Compared with another Delaunay-triangulation method, the proposed method can reduce the total wire-length by 3.8% using Manhattan routing, and 20.0% using non-Manhattan routing in the reasonable CPU time.

Chapter 5

Application of I/O Pad Assignment and RDL Routing to 3D IC

In this chapter, an application of I/O pad assignment and RDL routing method to 3D IC is proposed on the basis of the sorting method of Chapter 4. The approach initially assigns the same numbered I/O pads in two RDLs to micro-bumps by sorting the sum Manhattan distance between them. A pair- exchange modification is then carried out to improve the initial assignment. Then single layer routing in two RDLs are carried out respectively. Finally some un-routed connections are ripped-up and rerouted.

5.1 Introduction

In recent circuit design, because of the increasing of circuit complexity, the propagation delay and energy consumption have become important problems in the interconnection. However, it is difficult to solve these problems by traditional 2D IC. Therefore, 3D IC has become the choice for high-performance circuits because of its high signal processing speed and low power consumption [58]-[60].

In 3D IC, between two adjacent dies, through silicon vias (TSV) are widely used to connect two I/O pads that belong to the same signal [61]-[62]. However, the two I/O pads need to be aligned in the same vertical position, which is difficult to achieve. As a result, micro-bumps and RDLs are often adopted [63]. Micro-bumps are used to connect two adjacent die, and RDL is used to redistribute the I/O pads to the micro-bumps. As shown in Figure 5.1, by attaching RDLs on the adjacent dies, the I/O pads can be distributed on RDLs. Then two I/O pads can be assigned

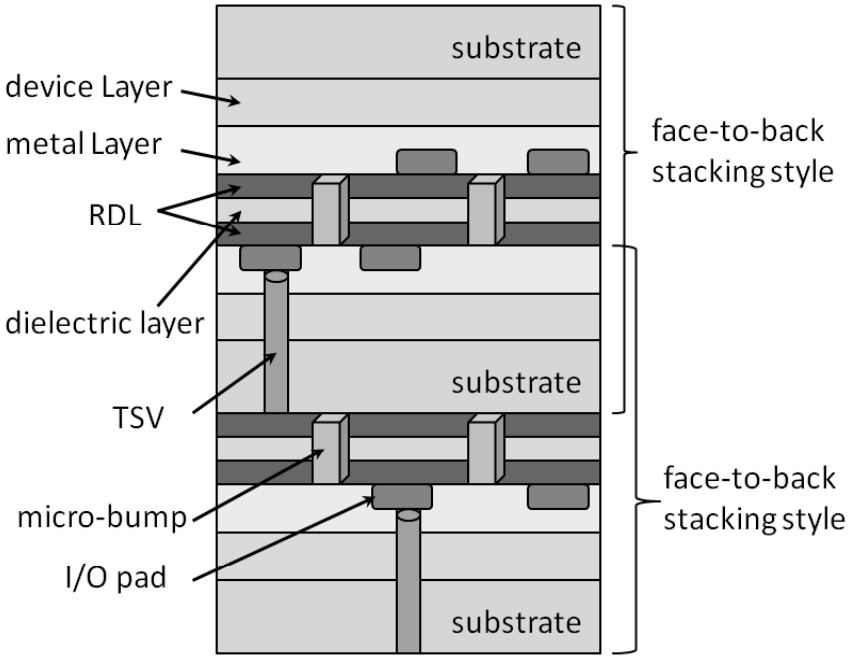
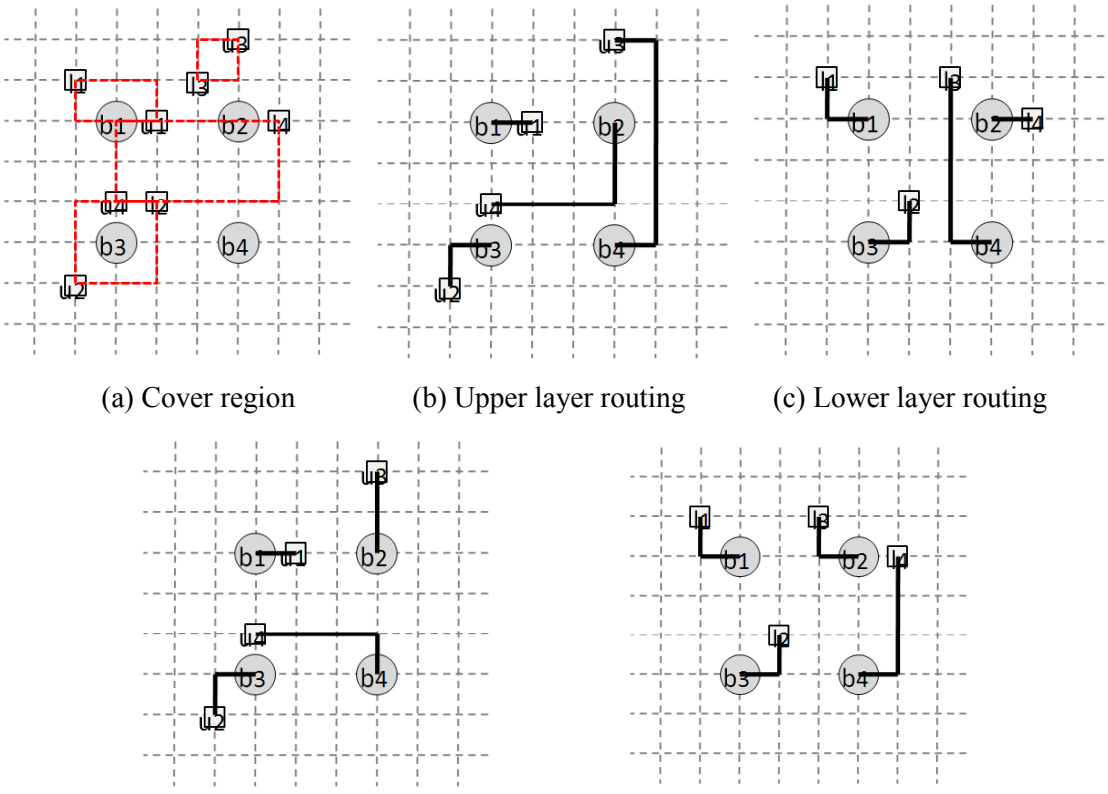


Figure 5.1 RDL and micro-bump structures in 3D IC [63]

to a feasible micro-bump and connected in upper layer and lower layer, respectively. Thus, the same signal between two dies can be connected.

In this study, given a set of I/O pads in two RDLs and micro-bump for 3D IC design, we propose a sorting-based I/O pad assignment and RDL routing method by extending the method described in Chapter 4. The objective of this study is to reduce the total wire-length, meanwhile improve the routability as far as possible. The whole design process is composed of four phases. In the first phase, we generate the I/O pad assignment by sorting the sum Manhattan distance between the same numbered I/O pads in two RDLs and their nearest bump balls. In the second phase, pair-exchange is used to shorten the total wire-length. In the third phase, connect the I/O pads and micro-bumps by maze routing algorithm. Finally some un-routed connections are ripped-up and rerouted in the last phase. The experimental results show that the proposed method is able to obtain the routes with shorter total wire-length in reasonable CPU time.

The remainder of this chapter is organized as follows: Section 5.2 describes some previous works about this study. Section 5.3 gives the problem definition of this work. Section 5.4 details the four phases of design algorithm. Section 5.5 illustrates the experimental results and analysis. Finally, Section 5.6 concludes this chapter.



(a) Cover region (b) Upper layer routing (c) Lower layer routing

(d) Upper layer routing by our method (e) Upper layer routing by our method

Figure 5.2 Comparison with matching-based method

5.2 Related Works

As mentioned earlier, micro-bumps and RDLs are often used in recent circuit designs. Several works have been published to solve the I/O pad assignment and RDL routing problem for 3D IC design.

In [63], an inter-die RDL routing method is proposed based on the integer linear programming. An I/O pad assignment method using order relation was proposed in [64]. However, these works with non-crossing constraint of two nets don't maintain routability for single RDL routing.

In [65], a matching-based I/O pad assignment method was presented. However, it just considers the matching for micro-bumps that are inside the cover region of two I/O pads, which may cause long wires of some connections. For example, in Figure 5.2 (a), as the cover region of u_3 and l_3 (same signal I/O pads in two RDLs) does not include any micro-bump, the

assignment for them will be decided at the last. It causes long wires of connection between u_3 and the assigned micro-bump b_4, and connection between l_3 and b_4, as shown in Figure 5.2 (b) and Figure 5.2 (c).

In this chapter, for the I/O pad assignment, we consider the sum Manhattan distance between same numbered I/O pads in two RDLs and micro-bumps. Then pair-exchange procedure is carried out to improve the initial assignment. Moreover, the exchange order is also considered. Compared with the matching-based method in [65], our method takes advantage of generating shorter total wire-length, as shown in Figure 5.2 (d) and Figure 5.2 (e). The total wire-length of two layers can be reduced from 25 to 19. The following sections discuss the proposed method in detail.

5.3 Problem Definition

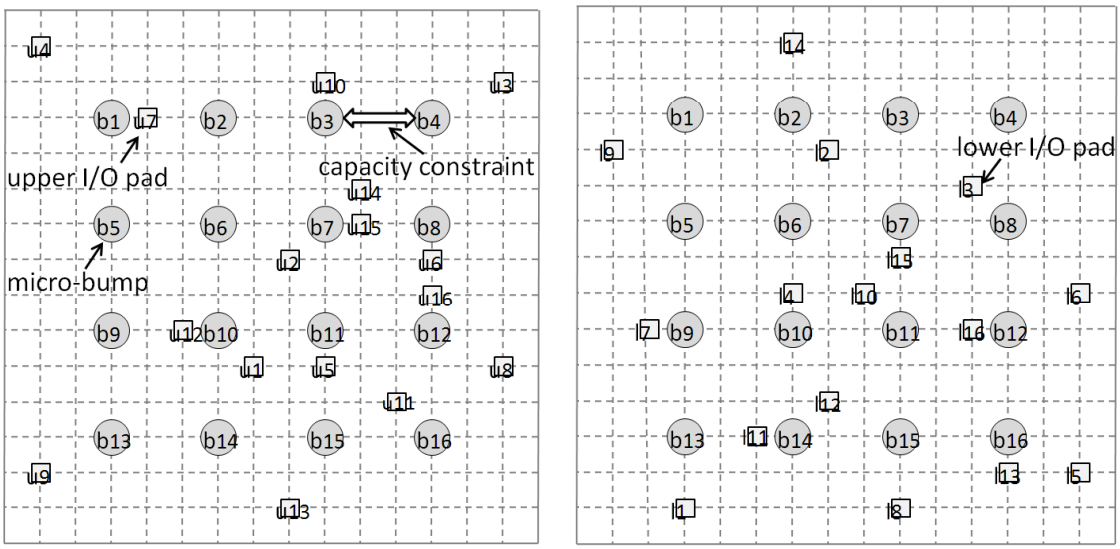
In this study, the micro-bump assignment problem is defined as follows: the input includes the same numbered I/O pads in upper layer and lower layer, and micro-bumps; it outputs the routes of assigned connections in two RDLs. The objective is to reduce the total wire-length, meanwhile improve the routability as far as possible.

Let $U = \{u_1, u_2 \dots u_n\}$ and $L = \{l_1, l_2 \dots l_n\}$ be the set of n I/O pads in upper layer and lower layer respectively, and $B = \{b_1, b_2 \dots b_m\}$ be the set of m micro-bumps (n<=m). The same numbered I/O pads, which are called I/O pad pair and represented as n_i, should be assigned to the same micro-bump. Besides, there is capacity constraint in the space between adjacent micro-bumps. In this work, we formulate that it is only allowed to route wires in a single RDL.

As illustrated in Figure 5.3, there are 16 I/O pads in upper layer, 16 I/O pads in lower layer, and 16 micro-bumps for connection. It is assumed that the capacity constraint is 2. In this study, all the distance and wire-length are defined as the number of grids.

5.4 Design Algorithms

The proposed method includes four sequential phases: initial assignment, pair-exchange modification, RDL routing, rip-up and reroute. The flow chart of the design process is shown in Figure 5.4. The basic idea for I/O pad assignment and pair exchange is similar with the methods adopted in Chapter 4



(a) Upper layer (b) Lower layer

Figure 5.3 Problem definition in two RDLs

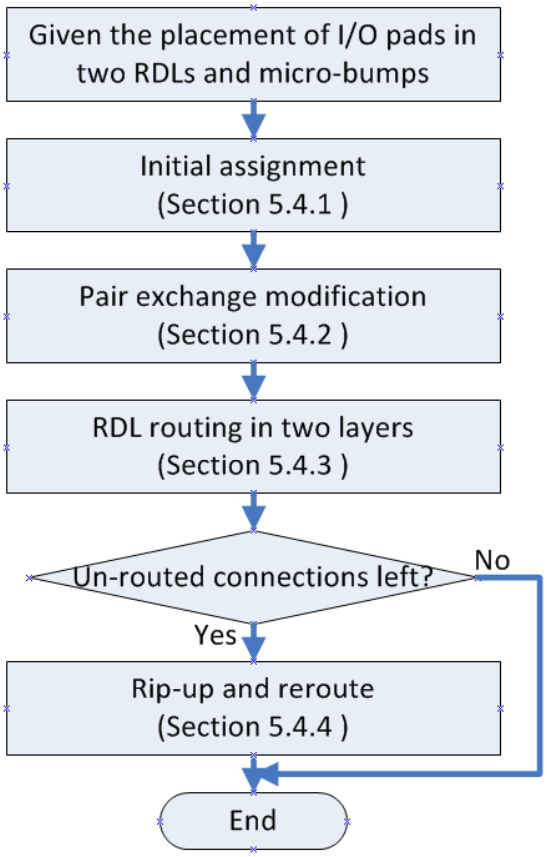
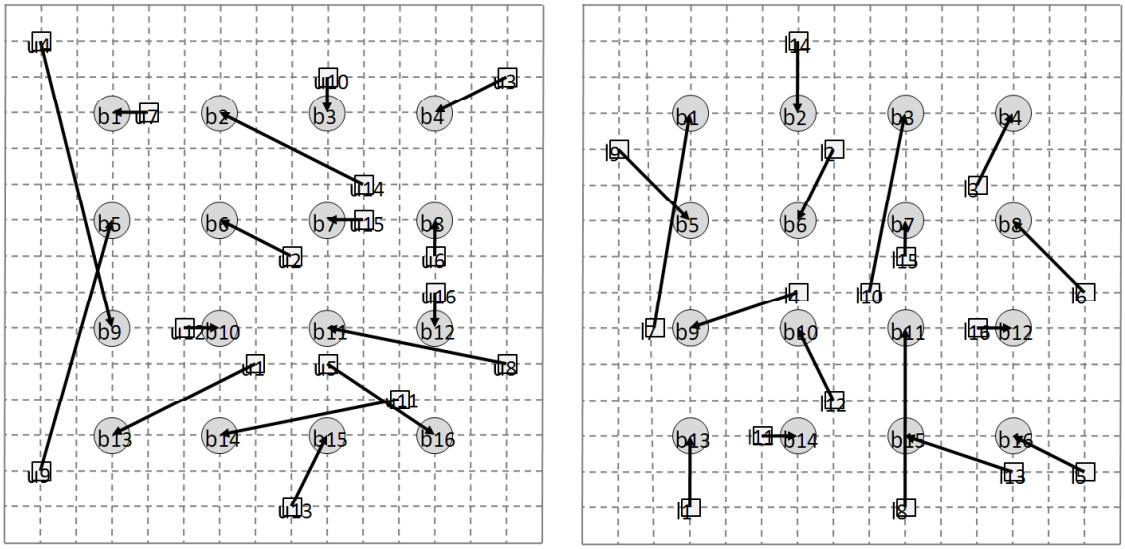


Figure 5.4 Flow chart of I/O pad assignment and RDL routing for 3D IC



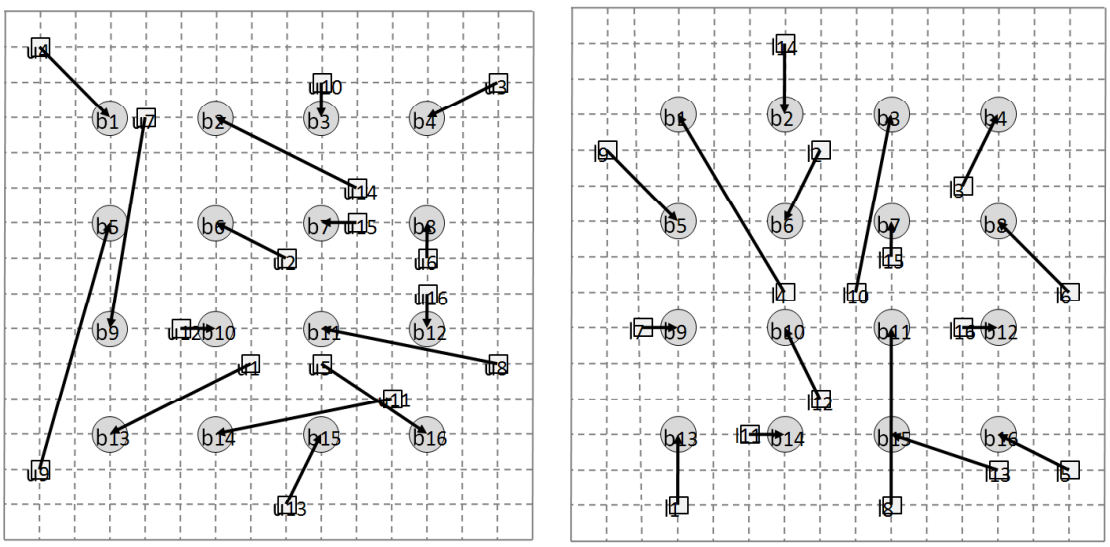
(a) Upper layer (b) Lower layer

Figure 5.5 Initial assignment result

5.4.1 Initial Assignment

Given the placement of I/O pads in two RDLs and micro-bumps, according to the location of them, the I/O pads with the same number are assigned to one micro-bump. The Manhattan distance between I/O pad and bump ball is taken into consideration for the assignment. We define $MD(u_i, b_j)$, $MD(l_i, b_j)$ as the Manhattan distance between u_i and b_j, l_i and b_j, respectively. Similar with Algorithm 1 of Chapter 4, initially for each I/O pad pair $n_i = (u_i, l_i)$, we can get one micro-bump with the smallest sum of Manhattan distance by calculating and comparing $MD(u_i, b_j) + MD(l_i, b_j)$. If there is just one I/O pad pair assigned to the unique micro-bump, the connection can be determined; if there is more than one I/O pad pairs assigned to the same micro-bump, we should sort these I/O pad pairs in ascending order according to the calculated sum of Manhattan distances. Then, choose the first I/O pad pair in the sequence since it is the nearest one to the same micro-bump, and put others into the next loop until all the I/O pad pairs are assigned to non-connected micro-bumps.

Take the case in Figure 5.3 to explain this procedure. In the first loop of micro-bump assignment, after the process of calculating and comparing the sum Manhattan distance $MD(u_i, b_j) + MD(l_i, b_j)$, I/O pad pairs n_1, n_2, n_3, n_6, n_9, n_{10}, n_{11}, n_{12}, n_{14}, n_{15} and n_{16} are firstly assigned to their unique bump ball. Then the sorting for the set of n_4 and n_7 with the same b_1, and the set of n_5, n_8 and n_{13} with the same b_{15} are carried out. Then we can determine the connection of n_7 and



(a) Upper layer (b) Lower layer

Figure 5.6 Pair-exchange modification result

n_{13} that are first in sequence. After the first loop, three I/O pad pairs n_4, n_5 and n_8 have not been assigned, which are put into next loop. Finally after three loops, all the I/O pads are assigned to different bump balls, as shown in Figure 5.5.

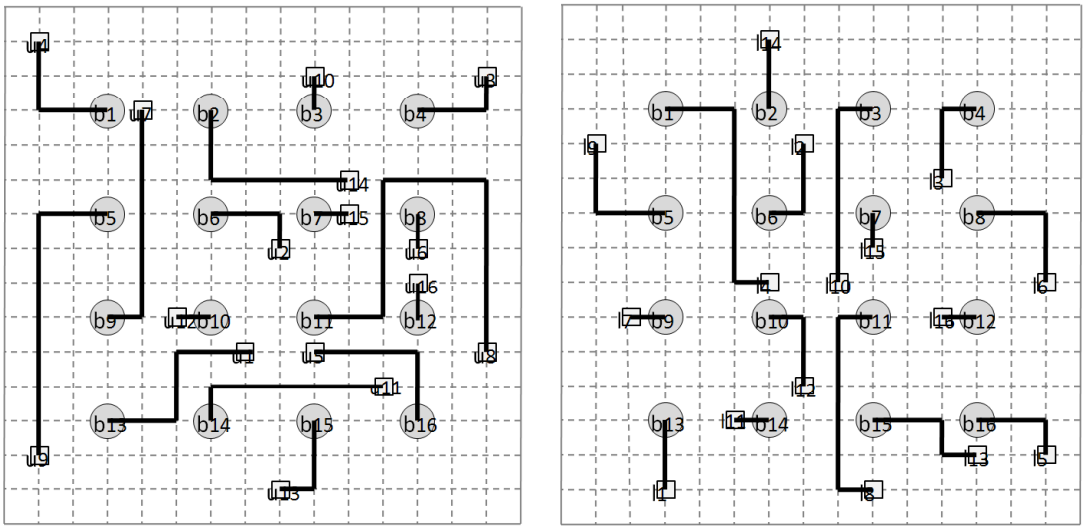
5.4.2 Pair-exchange Modification

To make the routing result better, the initial assignment result should be further modified by pair-exchange. This modification aims to shorten the total wire-length of two RDLs. Similar with Algorithm 2 of Chapter 4, for every two connection pairs, we compare the total Manhattan distance before and after pair-exchange. If the total Manhattan distance after pair-exchange is less than the previous one, this pair-exchange is accepted.

Refer to the initial assignment result in Figure 5.5, after the modification, the pairs of n_4 and n_7 are exchanged, and the modified result is shown in Figure 5.6.

5.4.3 RDL Routing

In this study, the connection path is determined by maze routing algorithm. The routing order in each layer is determined by calculating the Manhattan distances between I/O pads and their assigned micro-bumps in ascending order. Initially we assign the wire direction for each



(a) Upper layer (b) Lower layer

Figure 5.7 RDL routing result in two layers

connection to avoid the congestion area using the method mentioned in Section 4.3. Then the single layer routing in each of two RDLs is carried out respectively. Refer to the pair-exchange modification result in Figure 5.6, the RDL routing result in two layers is shown in Figure 5.7.

5.4.4 Rip-up and Reroute

Finally, if the paths of some connections are intersected by other routed wires, these connections may fail to route. Therefore, we need to rip-up and reroute these un-routable connections to improve the routability, if any. The rip-up and reroute is also realized by exchanging the un-routable connection and its adjacent connection, which is mentioned in Section 4.4.

5.4.5 Discussion on Time Complexity

As mentioned above, the proposed method is divided into four phases: initial assignment, pair exchange modification, RDL routing, rip-up and reroute. For initial assignment, as we assign the I/O pad pairs to micro-bumps by sorting the sum Manhattan distance between them, the time complexity of this phase is O(mn), where m, n are the number of micro-bumps and I/O pad pairs, respectively. Since the pair-exchange modification is carried out between any pair, the time complexity of this phase is O(n^2). In RDL routing, for one connection the wire-length is O(u) in the worst case, where u is the number of grids. So the time complexity of routing all

connections is O(nu). For rip-up and reroute, in the worst case, all the connections should be ripped up and exchanged with each other, so the time complexity of this phase is O(n^2). Since u is much larger than m and n, the total time complexity of the proposed method is about O(nu).

5.5 Experimental Results and Analysis

We implemented our proposed I/O pad assignment and RDL routing for 3D IC in C language, which is executed on a PC with 3.40GHz Intel Core 2 CPU and 8GB RAM. The matching-based I/O pad assignment method in [65], which maintains routability for single RDL routing, is also implemented for fair comparison. For each test data, the locations of I/O pads are randomly set. Two experiments are carried out. Experiment 1 is on comparison with method [65], which executed on the same package size but different I/O pad locations. Experiment 2 is also on comparison with method [65], but on different package sizes from small to large.

Experiment 1

For Experiment 1, there are five test data named Data01 to Data05 with the same package size and fixed location of micro-bump. For each data, the location of I/O pad pairs is randomly set. The experimental results are listed in Table 5.1, where "*#G*" denotes the number of routing grids, "*#N*" denotes the number of same numbered I/O pad pairs, "*#B*" denotes the number of micro-bumps, and "*#C*" denotes the capacity constraint. The wire-length is given in accordance with the number of unit grid, and the routability and execution time of these two methods are also listed in this table. Taking the total wire-length of method [65] as basal value 100%, the wire-length percentage comparison is illustrated in Figure 5.8. From the experimental results we can get that, both methods achieve 100% routability for all test data. Compared with method [65], our proposed method obtains shorter total wire-length within a small CPU time, and the average wire-length reduction is 17.52%.

Experiment 2

For Experiment 2, there are five test data named Data06 to Data10 with different package sizes from small to large, and the location of I/O pad pairs is randomly set as well. The experimental results are listed in Table 5.2. Same as Experiment 1, the wire-length percentage comparison is illustrated in Figure 5.9. The experimental results show that, both methods achieve 100%

Table 5.1 Experimental results in two RDLs on the same package size but different location

	#G	*#N*	*#B*	*#C*	Method [65]					Proposed Method				
					Routability	*Wire Length*			*CPU Time*	*Routability*	*Wire Length*			*CPU Time*
						upper	lower	total			upper	lower	total	
Data01	23*23	25	25	3	100%	167	165	332	0.025 s	100%	139	113	252	0.024 s
Data02					100%	176	191	367	0.028 s	100%	192	155	347	0.027 s
Data03					100%	153	210	363	0.026 s	100%	192	104	296	0.025 s
Data04					100%	125	118	243	0.024 s	100%	105	94	199	0.023 s
Data05					100%	163	136	299	0.025 s	100%	123	106	229	0.024 s
Average					100%	156.8	164	320.8	0.026 s	100%	150.2	114.4	264.6	0.025 s

Table 5.2 Experimental results in two RDLs on different package sizes

	#G	*#N*	*#B*	*#C*	Method [65]					Proposed Method				
					Routability	*Wire Length*			*CPU Time*	*Routability*	*Wire Length*			*CPU Time*
						upper	lower	total			upper	lower	total	
Data06	34*34	36	36	4	100%	378	332	710	0.056 s	100%	284	258	542	0.051 s
Data07	47*47	49	49	5	100%	433	473	906	0.126 s	100%	355	391	746	0.109 s
Data08	62*62	64	64	6	100%	690	686	1376	0.333 s	100%	584	552	1136	0.290 s
Data09	79*79	81	81	7	100%	1093	1032	2125	0.896 s	100%	861	948	1809	0.768 s
Data10	98*98	100	100	8	100%	1237	1334	2571	1.866 s	100%	1087	1060	2147	1.513 s

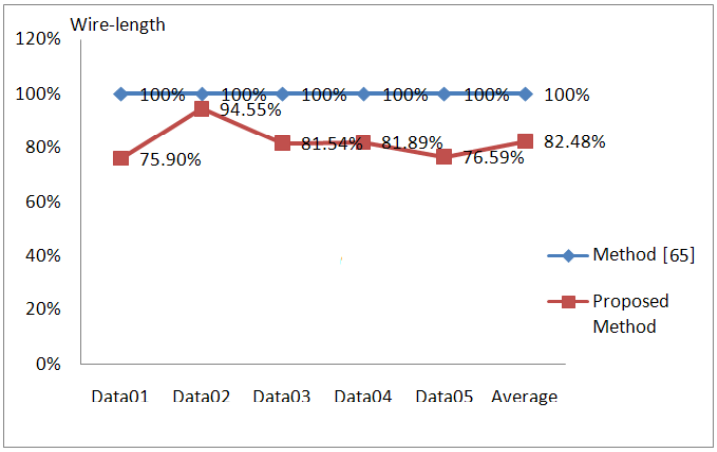


Figure 5.8 Total wire-length comparison in Experiment 1

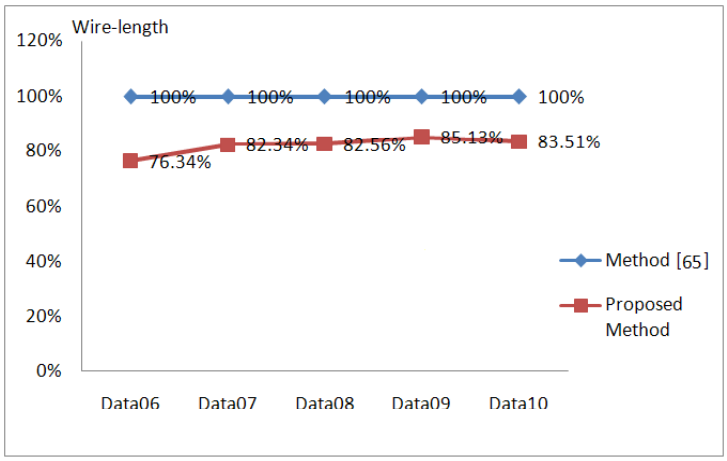
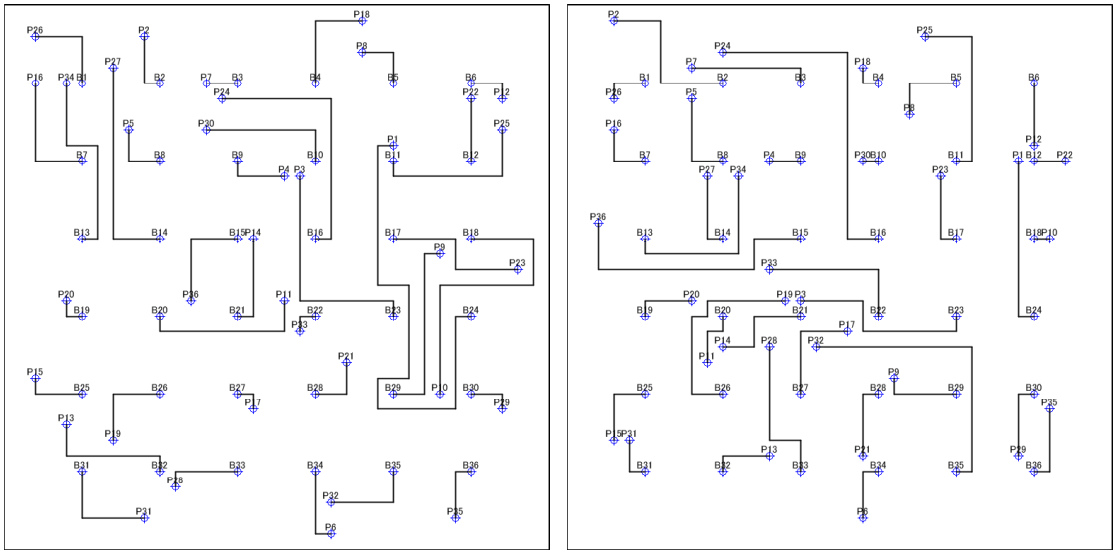


Figure 5.9 Total wire-length comparison in Experiment 2

routability for all test data. The comparison result with method [65] is similar with Experiment 1. The proposed method can generate shorter total wire-length not only in small scale but also in large scale packages in reasonable CPU time, and the maximum and minimum wire-lengths reductions are 23.66% and 14.87% respectively. Obviously, the experimental results show that our proposed method is effective on reducing total wire-length for 3D IC design in the reasonable CPU time.

Figure 5.10 shows the routing result of Data6. Since the assignment and routing in two layers are considered in this study and they are related to each other, to guarantee the routability, the capacity constraint is not strict in this study. How to further improve the routability in the dense routing area remains for our future works.



(a) Upper layer (b) Lower layer

Figure 5.10 Routing results for Data06 in Experiment 2

5.6 Conclusions

In this study, an application of I/O pad assignment and RDL routing method to 3D IC is proposed on the basis of the sorting method of Chapter 4. The approach initially assigns the I/O pad pairs to micro-bumps by sorting the Manhattan distance between them. A pair-exchange modification is then carried out to improve the initial assignment. Then single layer routing in two RDLs are carried out respectively. Finally some un-routed connections are ripped-up and rerouted. Compared with the traditional matching-based method, the proposed method is able to obtain the routes with shorter total wire-length in reasonable CPU time. For small scale package, the average wire-length reduction is 17.52%. Then for large scale packages, the maximum and minimum wire-length reductions are 23.66% and 14.87%, respectively.

Chapter 6

Conclusions

This thesis focuses on the equal-length routing for disordered pins and RDL routing for flip-chip in PCB design. Some optimizations, such as better wire-length balance, smaller worst length error or shorter wire-length, can be realized by the proposed routing algorithms.

Firstly, a region-aware routing 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. Our proposed method could be applied in both no-obstacle routing and obstacle-ware routing problems. Compared with another greedy method for disordered pins, our method gets a smaller standard deviation, in other words, a better wire-length balance among the nets, by adopting coefficient α. Besides, our method is effective in reducing worst length error, and the average reduction is 36.69%.

Secondly, we proposed a sorting-based I/O pad assignment and non-Manhattan RDL routing method for area I/O flip-chip design. The approach initially assigns the I/O pads to bump balls by sorting the Manhattan distance between them. Three kinds of pair-exchange procedures are then carried out to improve the initial assignment. Then to shorten the wire-length, non-Manhattan RDL routing is adopted to connect the I/O pads and bump balls. Finally some un-routed connections are ripped-up and rerouted. The experimental results show that our proposed method is effective on reducing wire-length no matter of the I/O pad location and package size. Compared with a partition-based method, the proposed method can reduce the wire-length by 23.4% using Manhattan routing, and 39.6% using non-Manhattan routing. Compared with another Delaunay-triangulation method, the proposed method can reduce the wire-length by 3.8% using Manhattan routing, and 20.0% using non-Manhattan routing in the

reasonable CPU time.

Finally, based on the sorting method, we extend the I/O pad assignment and RDL routing method in 3D IC design. The approach initially assigns the I/O pad pairs to micro-bumps by sorting the Manhattan distance between them. A pair-exchange modification is then carried out to improve the initial assignment. Then single layer routing in two RDLs are carried out respectively. Finally some un-routed connections are ripped-up and rerouted. Compared with a matching-based method, the proposed method is able to obtain the routes with shorter total wire-length in reasonable CPU time. For small scale package, the average wire-length reduction is 17.52%. For large scale packages, and the maximum and minimum wire-lengths reductions are 23.66% and 14.87% respectively.

In the equal-length routing research for disordered pins, the more intensive the pins are, the more difficult it is to get the ideal standard deviation of all the wires. Besides, in the RDL routing research for flip-chip, the large quantity of obstacles may make routability decrease. We hope we could do future research in much denser routing area.

Publication List

Journal Paper

1. R. Zhang and T. Watanabe, "Sorting-Based IO Connection Assignment and non-Manhattan RDL Routing for Flip-Chip Designs," *IEEJ Trans. on Electronics, Information and Systems*, Vol. 135, No.12, pp.1535-1544, 2015.

2. R. Zhang, T. Pan, L. Zhu and T. Watanabe, "Layer Assignment and Equal-length Routing for Disordered Pins in PCB Design," *IPSJ Trans. on System LSI Design Methodology*, Vol.8, pp.75-84, 2015.

3. X. Jiang, R. Zhang and T. Watanabe, "An Efficient Algorithm for 3D NoC Architecture Optimization," *IPSJ Trans. on System LSI Design Methodology*, Vol.6, pp.34-41, 2013.

International Conference Paper

4. R. Zhang, T. Pan and T. Watanabe, "A Sorting-Based Micro-Bump Assignment for 3D ICs," *12th International SoC Design Conference (ISOCC)*, pp.139-140, 2015.

5. R. Zhang, T. Pan, L. Zhu and T. Watanabe, "A Length Matching Routing Method for Disordered Pins in PCB Design," *2015 20th Asia and South Pacific Design Automation Conference (ASP-DAC)*, pp.402-407, 2015.

6. T. Pan, R. Zhang, Y. Takashima and T. Watanabe, "A Randomized Algorithm for the Fixed-Length Routing Problem," *2014 IEEE Asia Pacific Conference on Circuits and Systems (APCCAS)*, pp.711-714, 2014.

7. Y. Tian, R. Zhang and T. Watanabe, "Efficient Delay-matching Bus Routing by using

Multi-layers," *2014 International Conference on Electronics Packaging (ICEP)*, pp.728-731, 2014.

8. R. Zhang, X. Wei and T. Watanabe, "A Sorting-Based IO Connection Assignment for Flip-Chip Designs," *2013 IEEE 10th International Conference on ASIC (ASICON)*, pp.1-4, 2013.

9. R. Zhang and T. Watanabe, "A Parallel Routing Method for Fixed Pins using Virtual Boundary," *2013 IEEE TENCON Spring Conference*, pp.99-103, 2013.

10. X. Jiang, R. Zhang and T. Watanabe, "An Efficient Design Algorithm for Exploring Flexible Topologies in Custom Adaptive 3D NoCs for High Performance and Low Power," *2011 IEEE 9th International Conference on ASIC (ASICON)*, pp.535-538, 2011.

Domestic Conference Paper

11. Q. Xu, T. Pan, R. Zhang, Y. Tian and T. Watanabe, "A High Density Escape Routing Method for Staggered-Pin-Array Based Mixed-Pattern Signal Model," IPSJ 第 14 回情報科学技術フォーラム, pp.249-250, 2015.

12. R. Zhang, T. Pan, L. Zhu and T. Watanabe, "A Length Matching Routing Method for Disordered Pins in PCB Design," 電子情報通信学会技術研究報告, Vol.114, No.476, pp.103-108, 2015. (Invited)

13. X. He, R. Zhang and T. Watanabe, "Line-sweeping Based IO Assignment and Diagonal Routing," 電子情報通信学会ソサイエティ大会 2014, A-3-5, 2014.

14. Y. Tian, R. Zhang and T. Watanabe, "Efficient Length-matching Bus Routing by using Multi-layers," 電気関係学会九州支部連合大会 2013, pp.485-486, 2013.

15. R. Zhang and T. Watanabe, "A Parallel Routing Method using Virtual Boundary," 電子情報通信学会総合大会講演論文集 2012 年_基礎・境界, pp.86, 2012.

16. R. Zhang and T. Watanabe, "A Multi-layer Routing Method for Parallel Isometric Wires," 電気関係学会九州支部連合大会 2010, pp.426-427, 2010.

17. X. Xu, Y. Hu, R. Zhang and T. Watanabe, "A Study of Isometric-Wire Routing Problem using Multi-Layers," 火の国情報シンポジウム 2010, A-4-2, 2010.

Bibliography

[1] T. Yan, Q. Ma and M. D. F. Wong, “Advances in PCB Routing,” *IPSJ Trans. on System LSI Design Methodology*, Vol.5, pp.14-22, 2012.

[2] M. M. Ozdal, “Routing Algorithms for High-Performance VLSI Packaging,” University of Illinois at Urbana-Champaign, 2005.

[3] 高木 清, “よくわかるプリント配線板のできるまで,” 日刊工業新聞社, 2007.

[4] P. Elenius and L. Levine, “Comparing Flip-Chip and Wire-bond Interconnection Technologies,” *Chip Scale Review*, Vol.4, pp.81–87, 2000.

[5] J. H. Lau, D. Rice and C. G. Harkins, “Thermal Stress Analysis of Tape Automated Bonding Packages and Interconnections,” *IEEE Trans. on Components, Hybrids, and Manufacturing Technology*, Vol.13, Issue.1, pp.182-187, 1990.

[6] J. H. Lau, S. J. Erasmus and D. W. Rice, “Overview of Tape Automated Bonding Technology,” *Circuit World*, Vol.16, No.2, pp.5-24, 1990.

[7] C. P. Wong, S. Lou and Z. Zhang, “Flip the Chip,” *Science*, Vol.290, No.5500, pp.2269-2270, 2000.

[8] J. W. Fang and Y. W. Chang, “Area-I/O Flip-Chip Routing for Chip-Package Co-Design,” *2008 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pp.518-522, 2008.

[9] Z. Zhang and C. P. Wong, “Recent Advances in Flip-Chip Underfill: Materials, Process, and Reliability,” *IEEE Trans. on Advanced Packaging*, Vol.27, Issue.3, pp.515-524, 2004.

[10] J. T. Yan and Z. W. Chen, “Pre-Assignment RDL Routing via Extraction of Maximal Net

Sequence," *2011 IEEE 29th International Conference on Computer Design (ICCD)*, pp.65-70, 2011.

[11] J. W. Fang, I. J. Lin, P. H. Yuh, Y. W. Chang and J. H. Wang, "A Routing Algorithm for Flip-chip Design," *2005 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pp.753-758, 2005.

[12] J. W. Fang and Y. W. Chang: "Area-I/O Flip-Chip Routing for Chip-Package Co-Design Considering Signal Skews", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol.29, Issue.5, pp.711-721, 2010.

[13] A. B. Kahng, J. Lienig, I. L. Markov and J. Hu, "VLSI Physical Design: From Graph Partitioning to Timing Closure," Springer, 2011.

[14] M. J. S. Smith, "Application-Specific Integrated Circuits," Addison-Wesley Professional, 2008.

[15] J. Lienig, "Introduction to Electromigration-Aware Physical Design," *2006 International Symposium on Physical Design*, pp.39-46, 2006.

[16] G. Xu, L. D. Huang, D. Pan and M. Wong, "Redundant-Via Enhanced Maze Routing for Yield Improvement," *2005 Asia and South Pacific Design Automation Conference (ASP-DAC)*, pp.1148-1151, 2005

[17] A. B. Kahng, B. Liu and I. I. Mandoiu, "Non-Tree Routing for Reliability and Yield Improvement," *2002 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pp.260-266, 2002.

[18] E. W. Dijkstra, "A Note on Two Problems in Connexion with Graphs," *Numerische mathematik*, pp.269-271, 1959.

[19] R. C. Prim, "Shortest Connection Networks and Some Generalizations," *Bell system technical journal*, Vol.36, Issue.6, pp.1389-1401, 1957.

[20] C. Chu and Y. Wong, "FLUTE: Fast Lookup Table Based Rectilinear Steiner Minimal Tree Algorithm for VLSI Design," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, Vol.27, Issue.1, pp.70-83, 2008.

[21] P. P. Saha, S. Saha and T. Samanta, "Rectilinear Steiner Clock Tree Routing Technique with

Buffer Insertion in Presence of Obstacles," *2015 28th International Conference on VLSI Design (VLSID)*, pp.447-451, 2015.

[22] C. Y. Lee, "An Algorithm for Path Connection and Its Applications," *IRE Trans. on Electronic Computers*, Vol.EC-10, Issue.3, pp.346-365, 1961.

[23] T. Y. Cheung, "Graph Traversal Techniques and the Maximum Flow Problem in Distributed Computation," *IEEE Trans. on Software Engineering*, Vol.SE-9, Issue.4, pp.504-512, 1983.

[24] D. Eppstein, "Breadth-first Search and Depth-first Search," University of California-Irvine, 1996.

[25] R. Diestel, "Graph Theory," Springer, 2005.

[26] N. Alon, R Yuster and U. Zwick, "Color-coding," *Journal of the ACM (JACM)*, Vol.42, Issue.4, pp.844-856, 1995.

[27] N Alon, R Yuster and U. Zwick, "Color-coding: A New Method for Finding Simple Paths, Cycles and Other Small Subgraphs within Large Graphs," *Proceedings of 26th Annual ACM Symposium on Theory of Computing*, pp.326-335, 1994.

[28] X. Deng, Y. Yao, J. Chen and Y. Lin, "Combining Breadth-first with Depth-first Search Algorithms for VLSI Wire Routing," *2010 3rd International Conference on Advanced Computer Theory and Engineering (ICACTE)*, pp.482-486, 2010.

[29] D. K. Kole, H. Rahaman, D. K. Das and B. B. Bhattacharya, "Optimal Reversible Logic Circuit Synthesis Based on a Hybrid DFS-BFS Technique," *2010 International Symposium on Electronic System Design (ISED)*, pp.208-212, 2010.

[30] Y. Kohira, S.Suehiro and A. Takahashi, "A Fast Longer Path Algorithm for Routing Grid with Obstacles using Bi-connectivity based Length Upper Bound," *2009 Asia and South Pacific Design Automation Conference (ASP-DAC)*, pp.600-605, 2009.

[31] Y. Kohira, S.Suehiro and A. Takahashi, "A Fast Longer Path Algorithm for Routing Grid with Obstacles using Bi-connectivity based Length Upper Bound, " *IEICE Trans. Fundamentals*, Vol.92, No.12, pp.2971-2978, 2009.

[32] 小平行秀, 高橋篤司, "CAFE router: 障害物を含む領域における連結度を考慮した複線配線手法," 電子情報通信学会技術報告, Vol.108, No.298, pp.73-78, 2008.

[33] Y. Kohira and A. Takahashi, “CAFE Router: A Fast Connectivity Aware Multiple Nets Routing Algorithm for Routing Grid with Obstacles,” *2010 Asia and South Pacific Design Automation Conference (ASP-DAC)*, pp.281-286, 2010.

[34] Y. Kohira and A. Takahashi, “CAFE Router: A Fast Connectivity Aware Multiple Nets Routing Algorithm for Routing Grid with Obstacles,” *IEICE Trans. on Fundamentals of Electronics, Communications and Computer Sciences*, Vol.93, No.12, pp.2380-2388, 2010.

[35] H. Kong, T. Yan and M. D. F. Wong, “Automatic Bus Planner for Dense PCBs,” *2009 Design Automation Conference (DAC)*, pp.326-331, 2009.

[36] M. M. Ozdal and M. D. F. Wong, “Length-Matching Routing for High-Speed Printed Circuit Boards,” *2003 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pp.394-400, 2003.

[37] M. M. Ozdal and M. D. F. Wong, “A Length-Matching Routing Algorithm for High-Performance Printed Circuit Boards,” *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, Vol.25, Issue.12, pp.2784-2794, 2006.

[38] M. M. Ozdal and M. D. F. Wong, “A Provably Good Algorithm for High Performance Bus Routing,” *2004 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pp.830-837, 2004.

[39] M. M. Ozdal and M. D. F. Wong, “Algorithmic Study of Single-Layer Bus Routing for High-Speed Boards,” *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, Vol.25, Issue.3, pp.490-503, 2006.

[40] Y. Kubo, H. Miyashita, Y. Kajitani and K. Takeishi, “Equidistance Routing in High-Speed VLSI Layout Design,” *Integration, the VLSI Journal*, Vol.38, No.3, pp.439-449, 2005.

[41] T. Yan and M. D. F. Wong, “BSG-Route: A Length-Matching Router for General Topology,” *2008 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pp.499-505, 2008.

[42] S. Nakatake, K. Fujiyoshi, H. Murata and Y. Kajitani, “Module Packing Based on the BSG-Structure and IC Layout Applications,” *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, Vol.17, Issue.6, pp.519-530, 1998.

[43] J. T. Yan and Z. W. Chen, “Obstacle-Aware Length-Matching Bus Routing,” *2011*

International Symposium on Physical Design, pp.61-68, 2011.

[44] I. Watson, C. Kirkham and M. Lujan, "A Study of a Transactional Parallel Routing Algorithm," *16th International Conference on Parallel Architecture and Compilation Techniques*, pp.388-400, 2007.

[45] T. Yan and M. D. F. Wong, "BSG-Route: A Length-Constrained Routing Scheme for General Planar Topology," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, Vol.28, Issue.11, pp. 1679-1690, 2009.

[46] 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," *2011 Design, Automation and Test in Europe Conference and Exhibition (DATE)*, pp.1-6, 2011.

[47] C. Y. Chin, C. Y. Kuan, T. Y. Tsai, H. M. Chen and Y. Kajitani, "Escaped Boundary Pins Routing for High Speed Boards," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, Vol.32, Issue.3, pp.381-391, 2013.

[48] D. S. Hirschberg, "Algorithms for the Longest Common Subsequence Problem," *Journal of the ACM (JACM)*, Vol.24, Issue.4, pp.664-675, 1977.

[49] L. Bergroth, H. Hakonen and T. Raita, "A Survey of Longest Common Subsequence Algorithms," *7th International Symposium on String Processing and Information Retrieval*, pp.39-48, 2000.

[50] C. Tan, D. Bouldin and P. Dehkordi, "An Intrinsic Area-array Pad Router for ICs," *10th Annual IEEE International ASIC Conference and Exhibit*, pp.265-269, 1997.

[51] J. W. Fang, I. J. Lin, Y. W. Chang and J. H. Wang, "A Network-Flow-Based RDL Routing Algorithm," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, Vol.26, Issue.8, pp.1417-1429, 2007.

[52] J. W. Fang, C. H. Hsu and Y. W. Chang, "An Integer Linear Programming Based Routing Algorithm for Flip-chip Design," *2007 Design Automation Conference (DAC)*, pp.606-611, 2007.

[53] P. W. Lee, C. W. Lin and Y. W. Chang, "An Efficient Pre-assignment Routing Algorithm for Flip-chip Designs," *2009 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pp.239-244, 2009.

[54] K. S. Lin, H. W. Hsu, R. J. Lee and H. M. Chen, “Area-I/O RDL Routing for Chip-Package Co-design Considering Regional Assignment,” *2010 IEEE Electrical Design of Advanced Packaging and Systems Symposium (EDAPS)*, pp.1-4, 2010.

[55] J. T. Yan, K. P. Lu and Z. W. Chen, “Routability-driven Partitioning-based IO Assignment for Flip-chip Designs,” *2010 IEEE Asia Pacific Conference on Circuits and Systems (APCCAS)*, pp.1075-1078, 2010.

[56] J. T. Yan and Z. W. Chen, “IO Connection Assignment and RDL Routing for Flip-chip Designs,” *2009 Asia and South Pacific Design Automation Conference (ASP-DAC)*, pp.588-593, 2009.

[57] E. Hursey, N. Jayakumar and S. P. Khatri, “Non-Manhattan Routing using a Manhattan Router,” *18th International Conference on VLSI Design*, pp.445-450, 2005.

[58] W. R. Davis, J. Wilson, S. Mick, et al., “Demystifying 3D ICs: the Pros and Cons for Going Vertical,” *Design and Test of Computers*, Vol. 22, Issue.6, pp.498-510, 2005.

[59] K. Bernstein, P. Andry, J. Cann, et al., “Interconnects in the Third Dimension: Design Challenges for 3D ICs” *2007 Design Automation Conference (DAC)*, pp.562-567, 2007.

[60] M. Koyanagi, T. Fukushima and T. Tanaka, “Three-Dimensional Integration Technology and Integrated Systems,” *2009 Asia and South Pacific Design Automation Conference (ASP-DAC)*, pp. 409-415, 2009.

[61] J. C. Chiang and S. Sinha, “The Road to 3D EDA Tool Readiness,” *2009 Asia and South Pacific Design Automation Conference (ASP-DAC)*, pp.429-436, 2009.

[62] D. Kung and R. Puri, “CAD Challenges for 3D ICs,” *2009 Asia and South Pacific Design Automation Conference (ASP-DAC)*, pp. 421-422, 2009.

[63] C. J. Chang, P. J. Huang, T. C. Chen and C. N. Liu, “ILP-Based Inter-Die Routing for 3D ICs,” *2011 Asia and South Pacific Design Automation Conference (ASP-DAC)*, pp.330-335, 2011.

[64] T. Y. Kuan, Y. C. Chang and T. C. Chen, “Micro-bump Assignment for 3D ICs Using Order Relation,” *2012 Asia and South Pacific Design Automation Conference (ASP-DAC)*, pp.341-346, 2012.

[65] J. T. Yan, Y. J. Tseng and C. H. Yen, “Efficient Micro-bump Assignment for RDL Routing in 3D ICs,” *21st IEEE International Conference on Electronics, Circuits and Systems (ICECS)*, pp.195-198, 2014.