國立交通大學

電子工程學系 電子研究所

碩士論文

針對已出線信號之排線規劃器

A Bus Planner for Escaped Signal on Printed Circuit Board

研究生：管長毅

指導教授：陳宏明 教授

中華民國一00年七月

針對已出線信號之排線規劃器

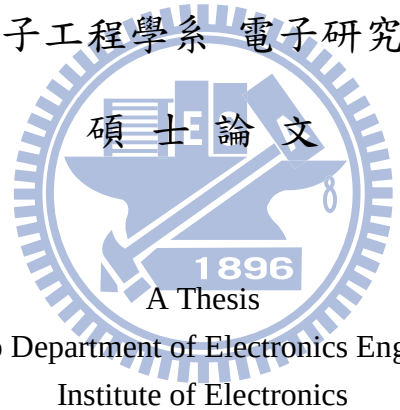
A Bus Planner for Escaped Signal on Printed Circuit Board

研 究 生：管長毅　　Student：Chung-Yi Kuan

指導教授：陳宏明　　Advisor：Hung-Ming Chen

國 立 交 通 大 學

電子工程學系 電子研究所



碩 士 論 文

A Thesis

Submitted to Department of Electronics Engineering and

Institute of Electronics

College of Electrical and Computer Engineering

National Chiao Tung University

in partial Fulfillment of the Requirements

for the Degree of

Master of Science

in

Electronics Engineering

July 2011

Hsinchu, Taiwan, Republic of China

中華民國一００年七月

針對已出線信號之排線規劃器

學生: 管長毅　　指導教授: 陳宏明 教授

國立交通大學　電子工程學系　電子研究所　碩士班

摘　　要

隨著數位電路設計以及製程的蓬勃發展，逐年增加的信號數量並不僅止於晶片系統的內部，各個系統之間的信號傳輸也多於以往，這劇烈的增加了封裝以及印刷電路板上的設計難度並導致傳統的手動設計成為瓶頸。設計師們在印刷電路板繞線問題上所遭遇的困難和晶片上相當得不同，在晶片設計中自動繞線器的挑戰主要在於總體的線長以及設計面積，另一方面，電路板設計者則需要考量排線信號之間的偏移以保障系統的可靠度。有鑒於這項事實，這篇論文旨在提供一個自動化演算法以有效率得協助電路板設計者完成排線的繞線。

由於近年來這項議題逐漸被重視，近年也有不少相關的自動化作品被提出，雖然都能夠在繞線時維持排線之間線長的匹配以達到控制偏移的效果，然而多數抑或問題在定義上仍然不夠實際又或只能施行於特定的繞線模式，這些限制導致他們難以被實際應用於真實的設計。

本文中考慮到在實務設計上覆晶(flip-chip)脫逸繞線(escape routing)及封裝的設計往往和電路板同步進行，這樣的情況下信號的腳位無法由電路板設計者自行決定而必須經由所有設計團隊的討論所獲得，我們的演算法能夠依照給定的固定腳位完成排線設計。藉著避免使用常見的最短路徑演算法，在能夠完成繞線之餘我們的結果亦保留了相對完整的區塊，以供設計者之後自由運用。

A Bus Planner for Escaped Signals on Printed Circuit Board

Student: Chung-Yi Kuan Advisor: Prof. Hung-Ming Chen

Department of Electronics Engineering
Institute of Electronics
National Chiao Tung University

ABSTRACT

Though the routing for high speed boards stays still a manual task by now, people have realized the necessity of its automation in recent years. Generally, such routing problems are classified into two stages: escaping route and PCB area routing. Plenty of works focusing on the former have been published, while the latter, a quite practical problem, is not yet well addressed.

Sometimes, the packages/components vendors have to start their design without the specifications of board designers, and the boundary pins are therefore fixed or advised to follow. This truth make previous works in escape routing can barely used in practice. We will describe this inflexible boundary pin escaping problem in this work, and propose an improved approach over one recent research [20]. Not only can we have a way to address, but we also further plan the wires in a better way to preserve the precious routing resources in limited number of layers on the board, and to effectively deal with obstacles. Our approach has different feature compared with conventional shortest-path based routing paradigm. In addition, we consider length-matching requirement and wire shape resemblance for high speed signal routes on board. Most existing matching algorithms are highly constrained on routing directions, making them usable only for certain design patterns. Solution

provided in this work aims to be free from such limitations. Our results show that we can utilize routing resource very wisely, and can efficiently resemble nets in the presence of the obstacles.

誌　　謝

本論文的完成有賴於許多人的幫助。首先我要特別感謝指導教授陳宏明老師給予我許多的機會以及指引我研究方向，讓我在這碩士短短的兩年間有許多收穫。同時感謝梶谷洋司(Yoji Kajitani)老師給予我機會參予 Fetch Router 的研究，帶給了我相當多研究上的靈感。對於張耀文教授、李毅郎教授及黃俊銘組長在口試時的指導也萬分感謝。

感謝 VDA 實驗室的每位同學。特別是謝謝宗穎學長的指導以及留下的研究資料，成為了這篇論文的基礎。謝謝敬雨、RJ 及欣吳在 PCB 研究上給我莫大的幫助以及鼓勵。謝謝 Benbean 從碩一開始帶著我熟悉工作站的管理，抱歉的是始終沒幫上甚麼忙。感謝九八級的大家在課業上相互分憂解勞。再次謝謝各位。

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The routing problem has been one of the major topic in the automation of IC design for long time and is getting serious for PCBs when working frequencies of chips are getting higher. We have increasing pin count on very dense boards, while current CAD tools for board routing are incapable of providing acceptable solutions automatically. Explicitly, rather than the overall wire length, the major PCB routing issues lie in IR drop, crosstalk and impedance matching. All of these effects are scaled up greatly than that in IC. That is why both the shape and length of nets must be carefully controlled by certain criteria in PCB routing. When the length of routes are not matched, the impedance could result in skew on the arrival time of signals. Detour of shorter nets is most common solution to such condition, while it is not always plausible for nets which are completely enclosed by either obstacles or other nets. Consequently, a key to board routing is the management of spaces, the routing resources. Though PCB designs are not limited to single layer, due to manufacturing cost, we should keep the number of layers used minimum, which generates the problem of layer planning.

1.1 Contributions

In this thesis, we want to address a practical problem: how we can route the wires or buses on the board with inflexible pins at package boundaries. To reduce time-to-market, it is not uncommon for industry that boards are designed alongside with packages by different groups. Since works are ran in parallel, after the negotiation with package designers, it is impossible to make change on pin assignment anymore. Another occasion is that components of area-I/Os like flip-chips may go through escape routing ([16, 15]) before the board design. Under both situations, board designers are advised to follow the given order of the boundary pins, and have no privilege to rearrange the pins in their design. It is worth mentioning that the topic in work [6] looks similar to this but is not really the same problem since its mission is to solve the given order of the boundary pins for area routing on board, which is on the opposite of the direction of wire planning.

Considering the fixed boundary pin planar routing, our approach will try to mitigate the aforementioned effects. Our work is a two-stage planner: the first stage is to obtain a planar topology for all the connections with more routing spaces available in subsequent pin pair routing; the second stage is to obtain a refined routing with awareness of length-matching constraint and wire shape resemblance. In order to reserve the routing space for the rest of the nets and to avoid the net crossing during board wire planning, our routing in first stage will optimize the routing order to ensure nets are free from conflicts and consume least routing resource, at the same time. During the second stage of refinement, we propose length-constraint-aware heuristics utilizing the routing resources reserved before to achieve min-max length bounds of nets. Our approach features a complete obstacle-avoiding area routing on board design.

1.2 Organization of This Thesis

The remainder of this work is organized as follows. Chapter 2 describes the board routing issues. Chapter 3 describes the first stage of our routing methodology to obtain planar topology. Chapter 4 reveals a post-processing routing that targets length constraints and wire shape resemblance. Chapter 5 shows our results followed by the conclusion in Chapter 6.

Chapter 2

Preliminaries

In this chapter, a list of surveyed works are briefed, and the main problem formulation in this thesis is stated. There has been quite a few works addressed routing problems on PCB, such as [3, 4, 7, 9, 10, 11, 12, 13, 14, 15, 6, 8]. Most of board-level routings focus on escape routing including the escape from inner pins to the boundary and from the boundary to other components. The first stage of escape routing problem is to route the wires from the bump balls located close the center of the component to the slots on the component boundary [9, 12, 13, 6]. [6]proposed a problem that has a constraint on pin ordering from the board designers. In the second stage, which is also the focus in this thesis, is the connection between the escaped slots to other components or packages on the board [3, 11, 14]. Some refer this step as PCB routing or area routing.

BSG-router [14] proposed an matching algorithm that applies to general designs and is no longer limited to face-to-face buses. Through mathematical programming, this work matches the net length precisely as well. However, its routes occupy vast routing area and are in unnatural shapes composed of numerous short jogged nets (net-jogging) which can cause serious signal integrity issues. Not all the works focus on net-by-net routing, an automatic bus planner is proposed in [3] providing good results in bus planning under fair pin assignment. But the fact that buses

are planned in bunches making it can easily be blocked by scattering obstacles. In addition, those matching issues are not addressed and solved in the work. Many of these routing algorithms are based on the concept of shortest path, which cannot take the whole set of nets into account; only the target net is taken into consideration in each iteration. As a result, the decision of routing order gets to affect the quality of results tremendously, and a bad one can result in wasting time on rip up and reroute. Several articles worked on more general routing problems are also inspiring in giving a rough planning of routes. Work [1] provides a good framework for topological routing, however the use of partitioning and Delaunay triangulation increases the complexity, and it still suffers from net ordering problem. By solving these two stages altogether, called "simultaneous escape routing", some of the latest work acquired even better solutions for enlarged solution spaces [7, 8]. [8] uses negotiated congestion based router to achieve the routing, this technique is widely used in modern academic global routers. On the other hand, [7] is featuring a new concept "boundary routing" to solve simultaneous escape routing problem, but it may be limited to certain escape patterns.

2.1 Problem Formulation

The input of our routing problem is a PCB design in arbitrary format where the peripheral packages as well as border of the board are represented as simple geometrical rectangles to which pins of signals attached. More generally, since chips are not necessary to be peripheral, the component could be area I/O chips with their signals escaped using methods proposed in [16] [15] before performing our algorithm. Here, we further require that the net-list contains only one-to-one connection. Our goal is to establish valid routes for each signal with proper layer assignment. Besides, the mismatch in wire-length among signals of a bus should be minimized.

Chapter 3

Against-Wall Topology Routing

An algorithm is presented here to tackle the problem addressed in the previous chapter. Since considering strict matching constraint in routing complicates the problem, in this step, we will not strongly confine routes with that constraint. As the name of this algorithm implies, we will try to route the nets against the other one so that wires share similar lengths. Against wall router, however, requires the source and target to have a certain block aside them, and the block must be consecutive. This means that some earlier nets can not success with this approach. To ensure this, our flow of this chapter is to connect separated components in design together and then we start against wall routing to nets. One more good thing about going against walls is that it is less likely to block potential paths in routing future wires.

3.1 Dynamic Pin Sequences

When determining routing topology, we do not really care about physical positions of the pins and packages, only the relation between pins is concerned. This implies that we may somehow model the input data into some simpler expressions. A dynamic pin sequence(DPS) is defined to be a cyclic sequence composed of all the pins in clockwise order along the boundary of a block. Here a block refer to an area in random continuous shape that can not be used for routing. To be concrete, it

can be either a single component or multiple components connected by nets. In the Fig.3.1, the DPS S_1 is an expression of component 1 by listing every pins following the direction of the dashed arrow.

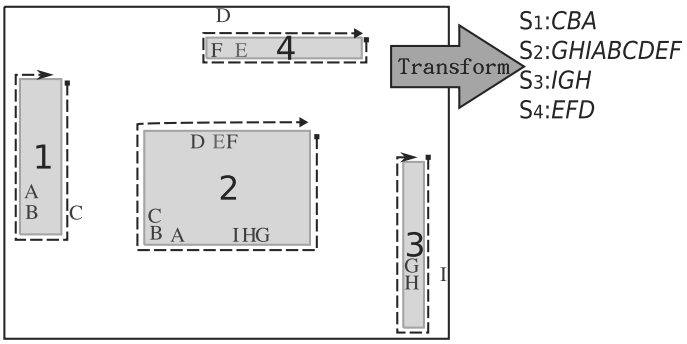


Figure 3.1: An illustration of pin-sequence generation of a component for PCB routing.

At very beginning of our algorithm, each component is converted into a DPS, and this is for two important purposes. First, based on our observation, the Lemma 3.1.1 which suggests DPS can be used to examine the compatibility of nets. Second, the routing order for against-wall router can be determined efficiently with a polynomial-time algorithm from DPS which will be detailed in Section 3.3. Both of the properties exist only if pins are all on a single block. Consequently, before we can benefit from these characters, we must figure out another mechanism to combine all the DPSes into one; this is to connect of all the components together.

Lemma 3.1.1. *Arbitrary two nets having their tokens interleaving each other on a DPS are incompatible on a single layer.*

Before we introduce to our algorithm, we describe some operations of DPSes. A DPS is a cyclic sequence which can be shifted, an operation removing the first element and appending it to the end, illustrated at lower right of Fig. 3.2. The shifted DPS are still considered to be the same as the original DPS. The other operation, combination, is described as follows. There is at least one token in common between the two DPSes participating this operation. One of the common tokens is selected

to be the Component Connection Pin(CCP). A DPS is then shifted so that the CCP is at the front of the DPS and inserted before the CCP token found at the other DPS. In Fig. 3.2, where S_1 and $S2$ are combined with token F as the CCP. Once the pin is connected, its two components will be bonded into one super component whose DPS is exactly the final DPS(pins following the order of the dashed arrows in Fig. 3.2).

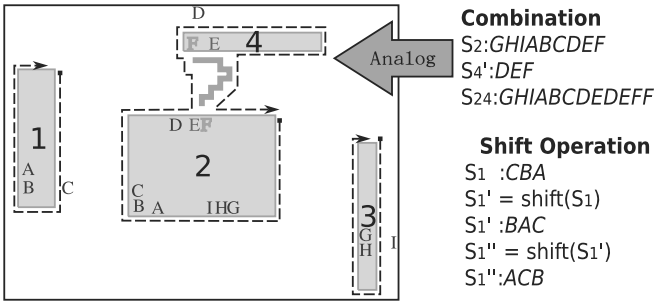


Figure 3.2: Display two operations in DPS: shifting and combination.

Here comes two questions: in what order should DPSes be combined, and which element should be the CCP in each combination? A random selection can not be good since these decisions will directly influence how many layers are used in the result.

About the first question, there are a few things to know. To combine a design composed of m components, combination operations will be performed $m-1$ times, so all the components must be chosen at least for one time. By considering all the components as nodes of a complete graph, the properties reminds us the algorithms for minimum spanning tree(MST) which select $m-1$ edges over a graph to span its m elements with minimum cost. Here we let the weight of an edge to be the number of connection between its two components. It is because we expect two components sharing most nets having the greatest chance to generate pairs that could be eliminated. For simplicity of implementation, Prim's algorithm is used here.

As in Algo. 1, Prim's algorithm maintains its node in two sets, black and white. The connected nodes are put into black set while unconnected nodes are in white, and the edge of the largest weight from black set to white set are chosen every iteration until the $N-1$ iterations are done. Every time an edge is chosen, the DPS of component belonging to white set are combined with the *base* DPS. Therefore, the *base* which stores our final result will get longer and longer as the process going on. It will eventually cover all the pins of the design before the process ends. This heuristic does not guarantee the best combining order, but is generally good enough in our approach.

Algorithm 1 Adapted-Prim-algorithm for DPS combination

Require: List of sequences S
Ensure: Combined DPS sequence *base*
  $d \Leftarrow$ ARRAY(SIZE(S), -1) {The maximum weight from visited nodes}
  $u \Leftarrow$ ARRAY(SIZE(S), *white*) {Nodes not yet been visited as white}
  $n \Leftarrow 0$ { First node }
  $base \Leftarrow S[n]$
  for i from 0 to SIZE(n) $-$ 1 **do** {Combines N - 1 times}
    $u[n] \Leftarrow black$
    $max \Leftarrow -1$
    for j from 0 to SIZE(S) $-$ 1 **do**
      $d[j] \Leftarrow$ MAX($d[j]$, COMMON_PINS($base, S[j]$)) { Weight of edge in Prim's algo.}
      if $max < d[j]$ and $u[j] = white$ **then**
        $c \Leftarrow j$
    $n \Leftarrow c$
    $base \Leftarrow$ COMBINE_DPS($S[n], base$) { Node n Selected}

One thing to note is that obstacles can be considered as special components having no pins on them; these components are converted into null DPSes. They have nothing to connect with and should be dropped before our algorithm starts.

Next is the second question: how the CCP is selected? Our goal is to make the final DPS containing as less interleaving nets as possible, and we have found some special palindrome-like patterns in our manual trails. These palindromes occur when

some components of bus are connected with pins stand face-to-face(sometimes side-by-side), as shown in Fig. 3.3, which is most desirable condition. In our example

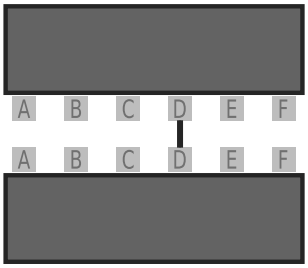


Figure 3.3: An illustration of a component connecting pin(CCP), is chosen from the pin-sequences, is the first connection among all nets in components. Pin$D$ is set to be the CCP and connected to the corresponding component.

in Fig. 3.2, a good case could be DPS2:$GHIABCDEF$ merging DPS1:BAC. They can be combined perfectly and generate a palindrome subsequence $ABCCBA$[1] as long as C is selected as CCP. The palindrome can be generated when opposite subsequences exist, like ABC in DPS2 versus CBA in DPS1. Sometimes the best solution is not obvious; shifts must be applied to find them. From this example, we realize that finding a good CCP seems to be a sort of string comparison problem, if we reverse one DPS. We therefore manage it with a classic algorithm for longest common sequence(LCS) problem [18]. This problem is proved to be NP-hard in general form, while it is polynomial-time solvable for fixed number of sequences. To fit the problem here, LCS algorithm need some modification. Another slightly changed version of LCS problem called longest common cyclic subsequence(LCCS) is already stated in [19] which still returns the longest common subsequence, only that the inputs sequences are allowed to rotate. Algorithm for this problem can be implemented similarly to the traditional LCS with one of the sequences has to be duplicated twice before comparing [2]. The duplication does scale up the problem

[1]According to the definition of combination, for certain CCP, the resultant compatible pins are the same whether you decide to shift first sequence or not.

[2]The complete implementation of LCCS is not as easy as described here. We are allowed to do this way without loss of correctness is because our sequences can have at most two instances for each unique symbol. DPSes are not general strings as defined in LCCS problem.

size of LCS, while by only a constant factor 2, which does not increase the time complexity dramatically. This process is shown in Fig. 3.4. Note that it is a heuristic since the optimality in Lemma 3.1.2 is ruined in the presence of more than two components. After the common cyclic sequence is selected, all of these signals are valid candidate for CCP. In our experiments, we select the net having physically closest pins as our CCP.

Lemma 3.1.2. *For two DPSes, after one of them inverted, the subsequence found by performing LCS algorithm contains maximum number of non-interleaving pins between these two sequences.*

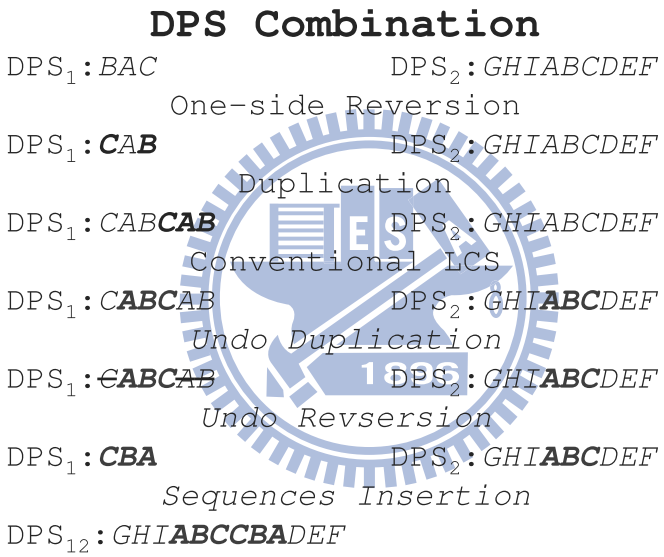


Figure 3.4: LCCS algorithm is used here which find out most compatible pins between two DPSes.

3.2 CCP Routing

After we locate the CCPs, we can start to connect our components, since the priority among them is as the order they are picked. CCPs suffer little from routing order because their route span a tree and a tree on plan will not form any closed region and hence will not block each other. A special router is designed for CCPs

which is very different from the router used for other nets. This is because the routing of these nets have a great influence on solution qualities. If some of them are too close to each other or to obstacles, the remaining nets will easily failed due to the lack of capacities at narrow channels when performing against-wall routing. As a result, their routing policy is right opposite to other nets: they tend to travel away from the obstacles.

Through some trials, a general A^* router [2] guided by the bus planner proposed in [3] is chosen for this task. On the Hanan grid, points are allowed to move when new points are inserted, and we make the space This making all the nodes evenly occupy the bin in Hanan grid so that generated routes are guaranteed to have some spaces with obstacles around. Then, a monotonic router is in charge of connecting these nodes and finishing the entire CCP route. With these spared margin, existing routes will less likely to trap those non-CCP nets. This, however, may not eliminate all the hazards, especially in regions with a large amount of obstacles. Updating congestion cost and rerouting CCPs may still needed in some sceneries. Fig. 3.5 shows the result of the connections.

3.3 Layer Assignment and Routing Order

The final result of combination leave us a long DPS containing all the nets in design; all the components are turned into a super component by CCPs. A notable fact is that the DPS should still hold the character of being a clockwise-ordered pin sequence of the design. To assign the layer of nets, we must select the maximum pin set from the DPS that can be routed without crossing. As the shapes of the blockage do not really matter in finding the routing topology, we can treat it as a circle, and the blank area outside this circle is our routing resource, as shown in Fig. 3.5. Applying a rough against wall route on the circle generate those stroked arcs around. We can also found D causes a conflict with E in Fig. 3.5, as suggested

by Lemma 3.1.1. From the figure, it occurs to us that an algorithm for maximum independent set(MIS) problem of string compatibility could be employed to solve the layer assignment and routing ordering. A work [17] on channel routing happens to satisfy our needs. The MIS problem on a general graph is an NP-complete problem, while utilize special structure of arcs, [17] can acquire optimal solution with an elegant dynamic programming(DP) algorithm. This algorithm was proposed to solve the problem in channel routing by modeling the pins on both side of channel as a closed ring; it finds maximum number of nets which can be routed inside single-layer channel without crossing. In [17], MIS of an arc from element i to j is the maximum number of cuts which can reside between i and j. It derived a recursive relationship between MIS of the arcs[3] based on the property above where k is the element such that j and k forms a cut.

$$\text{MIS}(i,j) = \begin{cases} 0, \text{ if } i \geq j \\ \text{MAX}[\text{MIS}(i,j-1), \text{MIS}(i,k-1) + \text{MIS}(k+1,j-1) + 1], \text{ if } i \leq k < j \\ \text{MIS}(i,j-1), \text{ otherwise} \end{cases}$$

Further, it can be solved with a dynamic programming in time complexity of $O(n^2)$, where n is the number of elements in given sequence. Though our routing, opposite to this work, is routing outside the circle, they are conceptually identical problem. Inspect Fig.3.5 carefully, we can see that if signal H is in the MIS of signal I, H must be routed before I, or it will be blocked; we say H is a prerequisite of I and this relation forms a partial order. This observation is also consistent with the fact that CCPs all have MIS of 0, they must be routed. Modifying the algorithm for channel routing slightly by adding an extra field in the DP table storing the MIS, we can simultaneously get the MIS of nets and serve them as our routing orders.

After order and layer assignment are figured out, we can finally apply the against wall routing on the design. The router is in fact a shortest-path router, say, maze

[3]The first case is the boundary condition, the second stands cut_{jk} has a chance to be selected into $\text{MIS}(i,j)$, and the last case is when the cut_{jk} is outside the range of i to j.

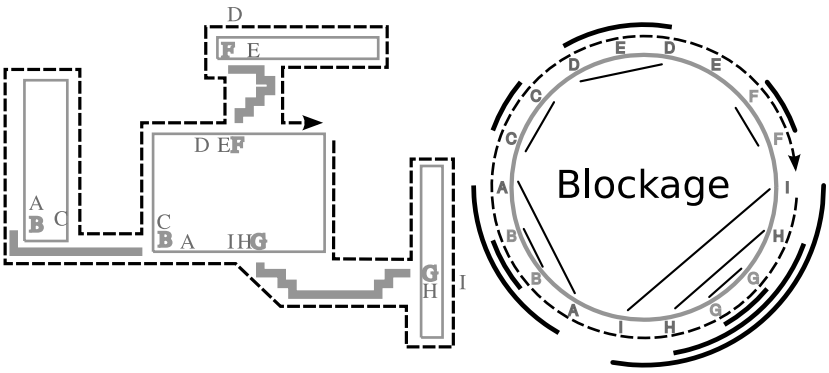


Figure 3.5: The left side is the design with all components connected. The right side is the combined DPS of example design arranged as a ring. Following the dashed arrow in both side, you will see they exactly express each other.

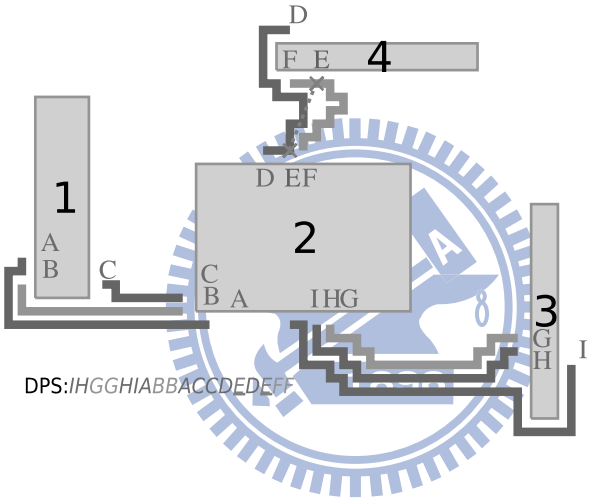


Figure 3.6: The routing result of against-the-wall routing. All nets are connected in order and routed against the "walls". The results show that the nets will be close to the wires of the connection of the CCPs and have resembled wire shape for bus requirement.

router, which assigns strong costs on those grids not beside an obstacle. Despite being a grid based router, this router has only two directions to go, it either goes clockwise along border or inversely, and is hence very efficient in run time. In the long run, all the nets have been connected, and the topology route is finished. The example result is shown in Fig. 3.6.

Chapter 4

Peripheral Matching

The results could be still far from desirable, even if we have tried to prevent large skew by planing nets altogether in topology routing. As been discussed in introduction, the presence of skews among signals may lead to undesired degrading on signal integrity or, in extreme cases, cause malfunction of chips. Here we propose a length-matching method that could enhance solution in terms of wire-length variation by reassembling routes by a linear programming(LP) framework.

Since we have found that dominating mismatches in bus often occur on those nets at a long distance from their groups members, our next algorithm focuses on matching the routing around components. That is, we assume the route we get from last step are virtually matched away from components. Usually, even if they are not exactly matched, the experiences suggest the responsible skews will be small relative to those caused by ill-placed pins. It is because our bus have been guided by its CCP and had similar shapes and distances in topology routing. Nevertheless, owing to the existence of obstacles, this assumption may fall impractical in some cases. We will relax this assumption later in Section. 4.3.2.

4.1 Preliminaries

Before we describe the algorithm, terms and notations that will be referred in problem formulations and explanations of our algorithm are defined.

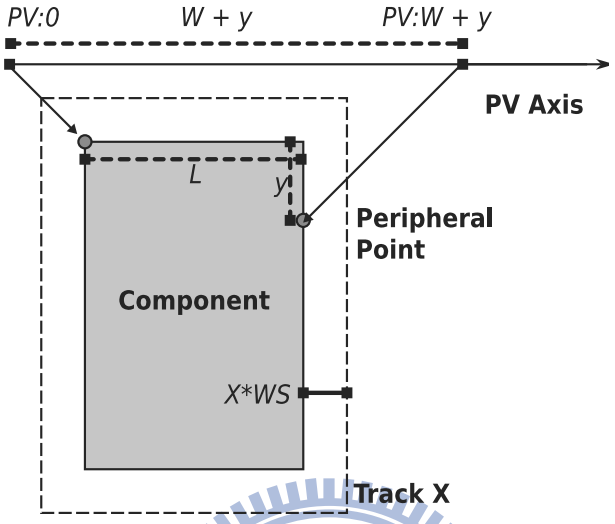


Figure 4.1: Map the points around the component into our custom coordinate system.

Given a rectangle R, we say the rectangle having the same center with both width and height larger than R by $X \times 2WS$ to be the box$X$ relative to R, where WS is the specified wire space of nets.

For an arbitrary rectangle R, we can define a coordinate system called peripheral value(PV). PVs and peripheral points are one-to-one mapped, that is, there exists exactly one point on R for every unique PV and vice versa.

4.2 Problem Formulation

Now we define our matching sub-problem as follows. Given one rectangle representing the component, there are some fixed pins on it. The $\hat{N}$th box of the component is given a special name "matching window", where $\hat{N} = 2^{\lceil log_2 N \rceil}$ and N is number of nets in the bus. The $\hat{N}$ can be depicted as the minimum value greater

than N while is a power of 2. That is, if the number of nets is exactly a power of 2, then $\hat{N} = N$, as the case in illustrating example, it is two times N's most significant bit in two's complement otherwise. Margins are reserved between sides of the component and the matching window as shown in Fig. 4.2; our algorithm will clean up all the existing routes inside and reroute them. The margin size is determined for reasons explained in next section. On the matching window, there is a point referred to as bus cut at which the bus generated from last step intersects the matching window (the arrow labeled "Bus"). The goal is to reconnect the path between the pins and the bus cut while keeping least length variations. Even though

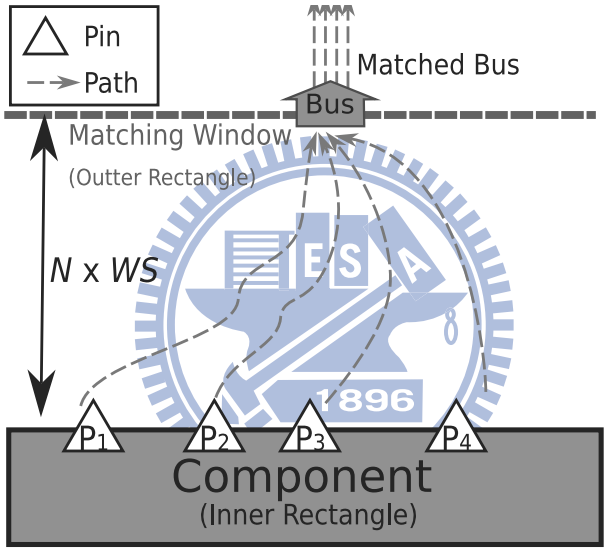


Figure 4.2: Outside the dashed horizontal line is planed bus and the curved paths are part of traces to be rerouted.

not shown in Fig. 4.2, the pins may not all sit on the same edge of the component so the routing region might be L shape or U shape as well. Although it is intuitive and naive for the extension, for simplicity, we will not consider these cases here. To make the whole idea more concrete, a simple example composed of only four nets in Fig. 4.4 are illustrated.

4.3 Linear Programming in Matching Length

The idea of this approach is mainly inspired by clock synthesis, another well-developed topic in IC design which also takes matching problem seriously, only that rather than buses, a super huge multi-terminal net, clock, is routed. Most of this collection of works treat their clock routing as a branched tree structure with clock source as the root, and clock sinks as leaves of the tree. Their main objective for the problem is hence balancing the delays for paths from root to all the leaves. With this in mind, we may also regard the cut point and pins in our problem as the clock source and clock sinks respectively.

Another feature of classic works in clock synthesis is that, due to the problem scale, they tend to solve it in a bottom-up manner so that the problem size is reduced. Neighbor leaves are first grouped as a sub-tree that matches all the members locally, and these minor trees are balanced when grouped into larger ones. Level by level, the clock tree is built and balanced; we also borrow this idea to our solution.

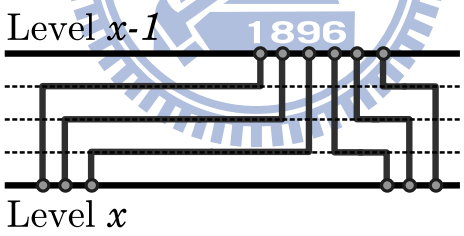


Figure 4.3: The Z-shape routing are used in connection between levels.

Instead of matching all net to the bus cut at once, the nets are required to match another adjacent net and form new groups before they reach the second level and the groups must be routed together thereafter. The router will route a group together in Z shape from one level to next, therefore the number of box used between levels equal to the group size of lower level as in Fig. 4.3. Before the group reaches next level, it must match another adjacent group[1]. In this way, the bus can be

[1]Unless it is left unpaired at the level, in such situation, it can go straight to next one.

represented as a tree whose height is $\lceil \log_2 N \rceil + 2$; The added two represents the layers of leaves and the root so that for a bus composed of only one net, there is still a tree composed a root and a leave. In this manner, we can see the number of boxes needed is at most 1 for first and second levels, 2 for level 3, 4 for level 4, ..., and 2^{x-2} for level x, till level $\lceil log_2 \hat{N} \rceil + 1$. The last level is not considered because it stands for the position of bus cut. The overall boxes must be used are therefore can be obtained, which determines our window size.

$$1 + \sum_{i=2}^{\lceil log_2 N \rceil + 1} 2^{i-2} = 1 + 1 + 2 + \cdots + 2^{\lceil log_2 N \rceil - 1} = 2^{\lceil log_2 N \rceil} = \hat{N}$$

Each node in the tree is turned into a PV and put into LP, and the overall problem is to generate the constraints with these $\hat{N}+1$ constants (root and leaves) and $2\hat{N}-2$(all other nodes) variables. By feeding these equations to solvers, desired position for nodes in PV can be figured out; we can reshape the nets based on these solutions. The constraints can be classified into three categories: tree-structure inequality, path equality, and blockage constraints.

4.3.1 Tree Structure Inequalities

To reduce problem complexity, we forbid any route to go back toward the component. More clearly, the route can only move toward the outer rectangle or stay at the box it belong. Along with the definition of PV, we are able to transform the matching problem from two-dimension to one-dimension. Because the length from inner rectangle to outer rectangle is constant, the LP solver just needs to decide the horizontal movement of routes.

However, to model our problem as a pure linear programming, one more challenge to resolve is that taking absolute value in constraints is not allowed [2]. For this reason, we put some extra inequality constraints to pre-assign the order of variables, this can

[2]Absolute values between variables implies conditional operations which can be realized in constraints only if extra binary variables are introduced.

possibly shrink the solution spaces while considering the lost in efficiency from LP to ILP, we take it as a fair trade-off. Plus, according to our experience, this compromise seldom shrinks feasibility tremendously. The rules we made for variables are stated as follow. The nodes in tree are given an index in the style of a traditional complete tree. Namely, the root is indexed 1, a left child of node$x$ is node$2x$, and node$2x+1$ is the right child. Nodes between root and leaves are variables. Constant location of pins are at the leaves of the tree. We know there is $2N$ leaves for a complete tree of $\lceil \log_2 N \rceil + 2$ in height, this mean half of the leaves would be left unused. To prevent the nodes touch their siblings, they must be at least a wire space apart from each other. The final rule is that, for all the nodes, constants or variables, they must be greater than their left child and lesser than their right one. For instance, V_4 are restrict to left side of C_9 by this constraint, in Fig.4.4. We also assign the location constant of the pin$k$ to be at the jth node, where the relation of j and k is: $j = 2\hat{N} - 2 + 2k + (k \mod 2)$. This is intended to make the range of variables

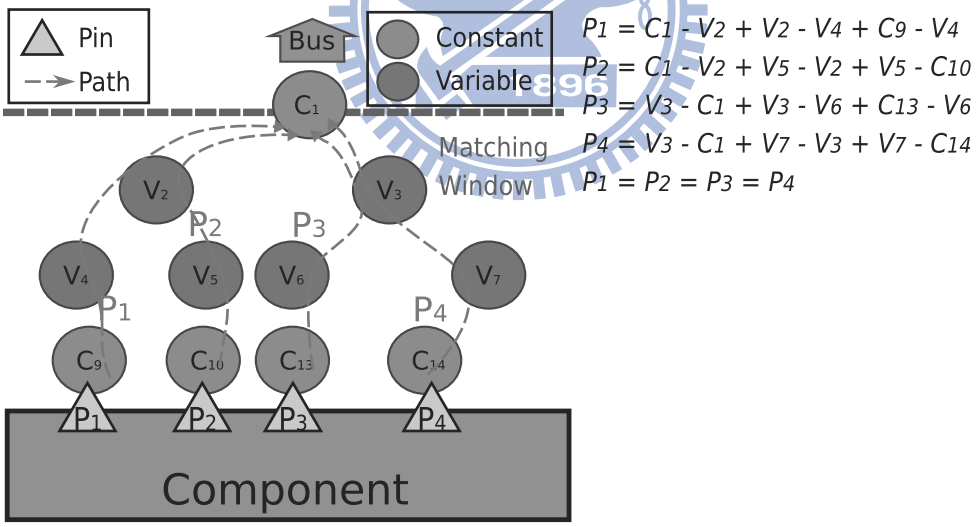


Figure 4.4: The equality constraints are described by the formulas on the right side which contain only the horizontal distance of each segments, since the vertical parts are identical to all paths.

not so limited by the final rule.

4.3.2 Path Length Equalities

Under the tree structure, you can know the distance from a certain pin point to the bus cut. There is only one path from the leaf node to the root; summing up the distances of all the variables you traveled plus the vertical distance is the matching length. As can be seen at right side of Fig. 4.4, path length equivalent constraint composed of $N-1$ equations (by erasing variable Ps with substitution), and this is our major objective. Because the vertical length is a constant for all the paths, they canceled each other in the equations.

Though this LP could be a feasible problem, to make sure the solver will not generate redundant detour, it is recommended to have one of the paths, say P_1, to be minimized in the objective, along with other equation the overall length is also minimized. In addition, when this bus has nets that have been assigned to multiple layers by our topology router, an objective function is needed. In this situation, rather than one LP, this bus will be solved by a number of LPs depending on the number of layers this bus are assigned to. Designers may have to give a desirable bounds for the bus, so that each LP confines their paths to fit in the given range.

Previously, we have claimed that the routes outside the matching window are almost matched while against wall routing might encounter obstacles and introduce significant variations to bus that broke the assertion. However, this assumption was actually made to describe our problem and can be relaxed by adding those mismatches found in bus into the constant term of path equalities. More straightforward, if one net P_1 is longer than the other net P_2 by x units in the bus routing, the equality between the two nets can be put like this.

$$P_1 = P_2 + x/2$$

Where P_1 and P_2 is the overall path length of corresponding nets as in Fig. 4.4; the $x/2$ means that nets have two terminals, and the mismatch can hence evenly

distributed to both the terminals.

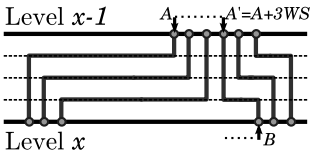


Figure 4.5: For the precision, the length calculation must consider space occupations of wires as shown in figure.

The equations listed in Fig. 4.4 are for demonstration purpose only; they are not complete. In reality, wires themselves do occupy some spaces, and our model has neglected this fact by now, and considering these spaces brings out some more constants in the equalities. An example in Fig. 4.5 reveals this, if we always use the left most positions for our nodes, rather than node value A, A' is actually used in calculation of the distance for the right branch in this level. The computation detail of these constants is intuitive and trivial, and is hence omitted here.

4.3.3 Obstacle Constraints

The matching window is not necessary to be a free space. From time to time, the region can be occupied with obstacles, and plan our routes on them are illegal. Consequently, some conditions are added to ensure valid routes in our solution. By scanning through each box, we can identify a range in which our nodes can move around. Suppose all the obstacles are in rectangle shapes, we may then determine the position relative to the tree by examining their edges. According to the position, we know which nodes are supposed to be bounded. Only the node closest to the obstacle will be involved in such a constraint as tree structure inequalities are going to take effect for other relevant nodes.

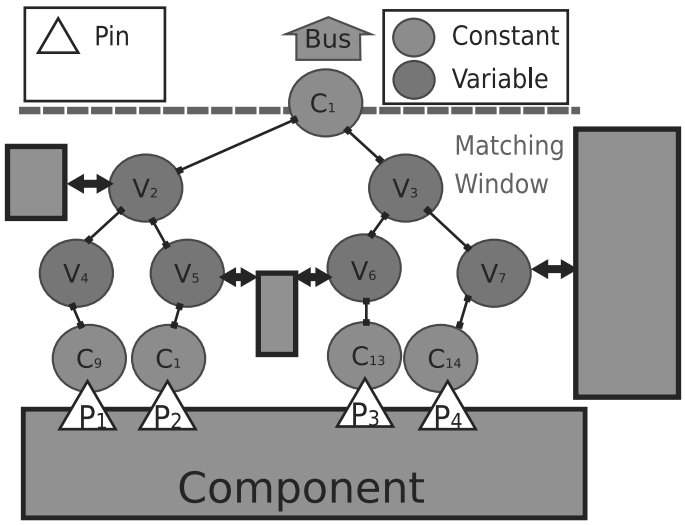


Figure 4.6: The arrows between obstacles and nodes indicates the extra constraints caused by the blocks.

4.4 Final Refinement

After the LP has been solved, the routes are retrieved by reversely converting the solution from values to points. Regardless of all the efforts, some final tunes might need to get rid of those variations. These adjustments are performed locally by adding bumps around those shorter wires. The routing priority acquired in topology routing can be reused here. By negating routing order, the nets will be tuned from the inner most to outer most. This is reasonable and intuitive; those inner nets have lesser routing around them, serving them first could prevent other nets from occupying these critical regions.

Chapter 5

Experiment Results

We have implemented our methodology in C++ on a machine of quad Intel Xeon CPUs cores with 32GB memory. Also, several cases shown in references (Test Cases I and II resemble the cases shown in [14] since the cases in [14] are not available; Test Cases III to V are from [**?**], provided from design houses) are served as our benchmark. Table 5.1 exhibits the test cases and run time for our router. *Nets* and *Component* indicate the number of nets and components in test cases. *Grid size* shows the size (the number of grid cell on PCB board)[1] of the routing problem, *Total Time* includes the time for building the routing grid and the run time spent by our router.

Fig. 5.1 shows the Test Case I which has five components. The routing result

[1]The grid sizes of the test cases in this work are larger than the BSG grid sizes in [14], which shows the granularity of our approach.

Table 5.1: The summary of test cases and experimental results.

	#Nets	#Components	Grid size	Total Time
Test Case I	17	5	250 × 250	< 1 *sec*
Test Case II	18	4	300 × 230	< 1 *sec*
Test Case III	57	5	300 × 230	< 1 *sec*
Test Case IV	81	8	640 × 560	2 *sec*
Test Case V	104	9	900 × 900	2 *sec*

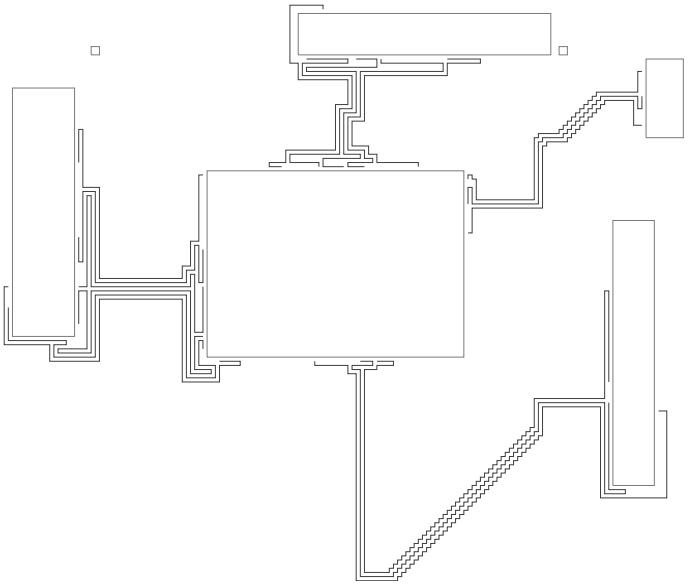


Figure 5.1: The routing result of Test Case I. The result shows that the routing nets are centralized as buses and the routing area is better utilized. The numbers beside pins are the routing order automatically figured out by our topology router.

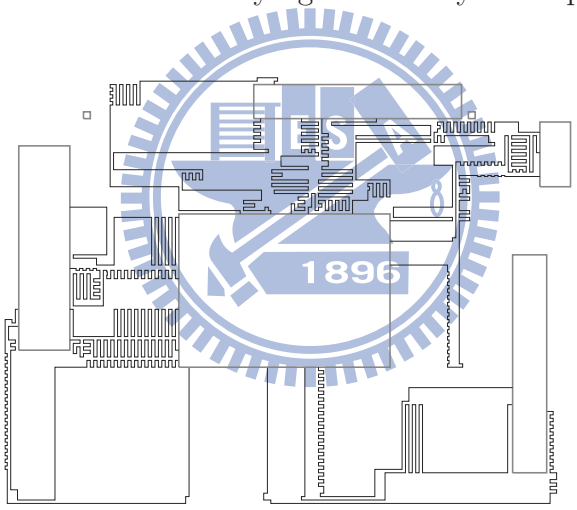


Figure 5.2: The routing result of Test Case I with our own implementation of work [14].

is quite different from that of [14] as can be seen in Fig.5.1 and Fig.5.2 [2]. The run times can not be compared fairly for some efficiency considerations, e.g. variables reduction, proposed in that work are not fully implemented in our reimplement version. Comparison of the results show that almost all nets are routed in certain region

[2]They compete in same wire space with identical LP solver. BSG grid size is determined dynamically according to topology as described in this work.

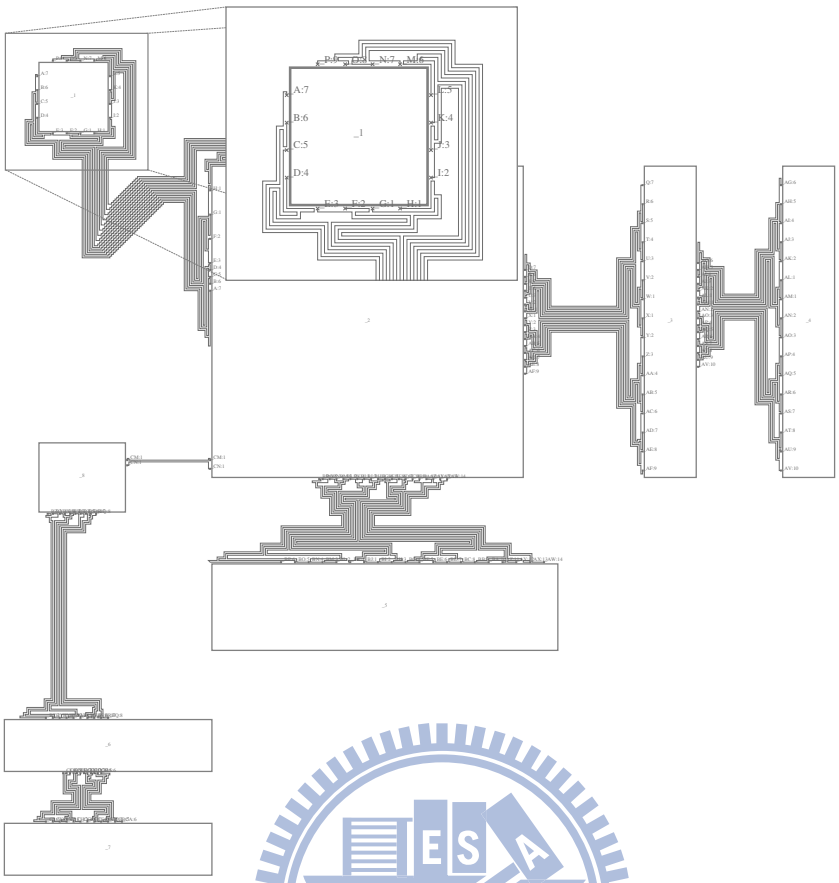


Figure 5.3: The routing result of Test Case IV. This case has eight components and the length constraints are conformed.

and the routing area is better utilized in our work. On the other hand, controlled by the solver, there is no neat workaround to prevent those unnecessary snaking in Fig. 5.2, and those irregular routing shape might result in certain undesired effects under electromagnetic verification.

Fig. 5.3 shows Test Case IV which has more components than the case in Fig. 5.1. The net distribution between each component are similar to the result of bus routing. Our router can also detour the nets to avoid crossing and conform with the length constraints. The top-left enlarged view shows our router can shift the nets in a certain region to meet the length constraints. Moreover, many net-joggings are avoided due to non-floorplan-based approach (area sizing). Even the components

are not face to face, our router can still find a good solution.

Chapter 6

Conclusion

In this work, we have proposed a board routing solution for a practical preassigned boundary pins problem. Instead of applying the shortest path algorithms like conventional routers, we have discovered some special properties of component and pins from cases and successfully utilized them into our work. Our approach is good in space management, and has the ability to meet the wire length and shape requirements at the same time. The experimental results show our work can preserve routing space in sequential planar routing under given boundary pins and conform with the length constraints.

Bibliography

[1] W. M. Dai, T. Dayan, and D. Staepelaere. "Topological Routing in SURF: Generating a Rubber-Band Sketch". In *Proceedings of ACM/IEEE Design Automation Conference*, pages 39–44, 1991.

[2] Z. Dong and M. Li. "A Routing Method of Ad Hoc Networks Based on A-star Algorithm". In *International Conference on Networks Security, Wireless Communications and Trusted Computing*, pages 623–626, 2009.

[3] H. Kong, T. Yan, and D. F. Wong. "Automatic Bus Planner for Dense PCBs". In *Proceedings of ACM/IEEE Design Automation Conference*, pages 326–331, 2009.

[4] H. Kong, T. Yan, and D. F. Wong. "Optimal Simultaneous Pin Assignment and Escape Routing for Dense PCBs". In *Proceedings of IEEE Asia and South Pacific Design Automation Conference*, pages 275–280, 2010.

[5] Y. Kubo, H. Miyashita, Y. Kajitani, and K. Tateishi. "Equidistance Routing in High-Speed VLSI Layout Design". In *Proceedings of the Great Lakes Symposium on VLSI*, pages 439–449, 2005.

[6] L. Luo and M. Wong. "Ordered Escape Routing Based on Boolean Satisfiability". In *Proceedings of IEEE Asia and South Pacific Design Automation Conference*, pages 244–249, 2008.

[7] L. Luo, T. Yan, Q. Ma, D. F. Wong, and T. Shibuya. "B-Escape: A Simultaneous Escape Routing Algorithm Based on Boundary Routing". In *Proceedings of ACM International Symposium on Physical Design*, pages 19–25, 2010.

[8] Q. Ma, T. Yan, and M. Wong. "A Negotiated Congestion based Router for Simultaneous Escape Routing". In *Proceedings of International Symposium on Quality Electronic Design*, pages 606–610, 2010.

[9] M. M. Ozdal and D. F. Wong. "Simultaneous Escape Routing and Layer Assignment for Dense PCBs". In *Proceedings of ACM/IEEE Design Automation Conference*, pages 822–829, 2004.

[10] M. M. Ozdal and D. F. Wong. "Algorithmic Study of Single-Layer Bus Routing for High-Speed Boards ". In *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, volume 25, pages 490–503, Mar. 2006.

[11] M. M. Ozdal and D. F. Wong. "Two-Layer Bus Routing for High-Speed Printed Circuit Boards". In *ACM Transactions on Design Automation of Electronic Systems*, volume 11, pages 213–227, Jan. 2006.

[12] M. M. Ozdal, D. F. Wong, and P. S. Honsinger. "An Escape Routing Framework for Dense Boards with High-Speed Design Constraints". In *Proceedings of IEEE/ACM International Conference on Computer-Aided Design*, pages 759–766, 2005.

[13] M. M. Ozdal, D. F. Wong, and P. S. Honsinger. "Simultaneous Escape-Routing Algorithms for Via Minimization of High-Speed Boards". In *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, volume 27, pages 84–95, Jan. 2008.

[14] T. Yan and D. F. Wong. "BSG-Route: A Length-Matching Router for General Topology". In *Proceedings of IEEE/ACM International Conference on Computer-Aided Design*, pages 499–505, 2008.

[15] T. Yan and D. F. Wong. "A Correct Network Flow Model for Escape Routing". In *Proceedings of ACM/IEEE Design Automation Conference*, pages 332–335, 2009.

[16] M. F. Yu and W. M. Dai. "Single-Layer Fanout Routing and Routability Analysis for Ball Grid Arrays". In *Proceedings of IEEE/ACM International Conference on Computer-Aided Design*, pages 581–586, 1995.

[17] K. J. Supowit. "Finding a Maximum Planar Subset of a Set of Nets in A Channel". In *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, volume 6, pages 93–94, Jan. 1987.

[18] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. "Introduction to Algorithms". Second edition, Page 350–355.

[19] F. Nicolas and E. Rivals. "Longest Common Subsequence Problem for Unoriented and Cyclic Strings". In *Journal Theoretical Computer Science*, volume 370, Issue 1–3, Page 1–18, 2007.

[20] T. Y. Tsai, R. J. Lee, C. Y. Chin, C. Y. Kuan, H. M. Chen, and Y. Kajitani "On Routing Fixed Escaped Boundary Pins for High Speed Boards". In *Proceedings of the Design Automation and Test in Europe Conference and Exhibition*, Page 1–6, Mar. 14-18, 2011.