# 國立交通大學

## 電機學院IC設計產業研發碩士班

## 碩士論文

# 覆晶封裝在封裝與機板共同設計階段一個線長驅策
# 及被範圍條件限制的訊號區塊擺放方法

# Wire Length Driven Flip-Chip Pin-Out Designation by Range Constrained Pin-Block Floorplanning in Package-Board Codesign

研究生: 翁嘉倫

指導教授: 陳宏明 教授

中華民國九十六年九月

# 覆晶封裝在封裝與機板共同設計階段一個線長驅策
# 及被範圍條件限制的訊號區塊擺放方法

# Wire Length Driven Flip-Chip Pin-Out Designation by Range Constrained Pin-Block Floorplanning in Package-Board Codesign


研 究 生: 翁嘉倫　　Student: Chia-Lun Weng

指導教授: 陳宏明 教授　　Advisor: Professor Hung-Ming Chen

國 立 交 通 大 學

電機學院 IC 設計產業研發碩士班


碩 士 論 文

A Thesis

Submitted to College of Electrical and Computer Engineering

National Chiao Tung University

in partial Fulfillment of the Requirements

for the Degree of

Master

in

Industrial Technology R & D Master Program on

IC Design

September 2007

Hsinchu, Taiwan, Republic of China

中華民國九十六年九月

# 覆晶封裝在封裝與機板共同設計階段一個線長驅策

# 及被範圍條件限制的訊號區塊擺放方法

學生: 翁嘉倫　　　　　　　　　　　　　指導教授: 陳宏明 教授

國立交通大學電機學院產業研發碩士班

## 摘　　　要

隨著先進製程技術快速發展之下，越來越多的電路可以被整合在單一晶片裡，這樣的趨勢造成封裝設計與訊號之間的連接變得更爲複雜。然而，傳統打線封裝技術在一些特殊設計上已不敷實用，取而代之的是目前已被廣泛使用的覆晶封裝技術。覆晶封裝設計裡在封裝與機板共同設計階段中，一般排放錫球的方式都是以資深工程師依經驗手動擺放，這是一個很費時又反覆的過程，影響了產品上市的時間 (TTM)。因此,[1]以自動化程式產生面陣列引腳錫球圖 (Ball Grid Array, BGA)，以利工程師做事後微調的動作，大幅地減少這排放過程花費的時間。在這篇論文裡，我們展現了自動化產生面陣列引腳錫球圖的過程，並且在封裝與機板共同設計階段提出一個被範圍條件限制的訊號區塊擺放改進方法並且使用模擬退火演算法 (Simulated Annealing Algorithm, SA) 來執行。同時我們也對此特殊擺放的需求發展一個表示方法。我們提出的改進方法在訊號組態資訊參數設定上較有彈性，以及可以確保最小的封裝尺寸，實驗數據結果顯示我們提出的改進方法優於論文 [1]的擺放方法。

Wire Length Driven Flip-Chip Pin-Out Designation by Range Constrained Pin-Block Floorplanning in Package-Board Codesign

Student: Chia-Lun Weng Advisor: Prof. Hung-Ming Chen

Industrial Technology R & D Master Program of
Electrical and Computer Engineering College
National Chiao Tung University

ABSTRACT

With the advanced fabrication technique developing rapidly, more and more circuits could be integrated in a single chip. This trend will cause the complication in package designs and signal interconnection. However, the typical peripheral wire-bond design may not be proper for use in some particular designs, flip-chip becomes a better choice. In flip-chip design, engineers generally arrange the ball chart in the manual manner on experience in package-board codesign. This process is iterative, time-consuming and it will lengthen the time-to-market (TTM) of products. [1] proposed a method of generating the BGA ball chart automatically by pin-block design and floorplanning, thus helped engineers respin the ball chart slightly and saved the arranging time dramatically. In this thesis, we exhibit the procedure of accomplishing the method of [1] and improve the flooplanner in [1]. The proposed pin-block floorplanner designates pin-out for flip-chip BGA package by using the range constraints, and it is based on simulated annealing algorithm. We also develop a representation for this special floorplanning requirement. It not only has flexibility on specifying critical parameters of the pin configuration, but also guarantees the minimum package size. Experimental results show that improved pin-block floorplanner can perform a better pin assignment than that in [1].

# 誌　　謝

還記得剛進交大這個新環境時, 讓我感到有些不適應, 老師教學認眞每天像度日如年, 時光匆匆兩年多的產碩班將告一個段落。在這段寶貴的時間裡, 我學習到另外一個領域的知識學問, 在這學習過程中也有遇到許多挫折和困難, 但我都必須一一克服, 所以, 在這種嚴格的訓練之下, 我學習到如何發現問題, 研究方法, 進而解決問題。當然有苦也會有樂, 在交大我也有享受到豐富收穫的喜悅, 感謝交大人陪我走過這段精華的時光, 感謝交大。

首先, 我要感謝我的指導教授陳宏明老師, 老師總是耐心地給予我指導告訴我正確的方向並且細心地提醒我一些小地方, 不僅在學業上給我提攜, 在爲人處世上更是讓我獲益匪淺。再者, 我要感謝博士班學長李仁傑, 啓發我的想法並在一次又一次的討論之後得到了答案。最後特別感謝麥偉基教授與江蕙如教授在百忙之中, 撥冗參與論文口試, 並對學生的研究給予寶貴的意見, 讓此篇論文更加完備充實。

另外要感謝的, 就是在實驗室的日子裡, 一起共同努力的學長, 學弟妹及夥伴們, 佳毅、佳正、禎徽、宇倫、信華、綸君、維廷、柏州以及產專班的同學, 華鼎、黃俊。在學業上互相切磋交換意見, 也都適時地給予我莫大的幫助, 使我的研究生活走得更順遂, 感謝你們。

最後, 我要感謝我的家人, 爸爸、媽媽、哥哥、哥哥的女友。因爲有你們的支持, 我才能順利地在這兩年多的時光內完成學業, 感謝你們給我的幫助。

要感謝的人實在太多了, 無法一一致謝, 在此, 獻上最眞誠的心, 感謝所有關心過我的人, 謝謝你們。

翁嘉倫 謹誌

2007年9月,VDA lab, 交大, 新竹, 台灣

# Contents

# List of Tables

# List of Figures

# Chapter 1

# Introduction

Moore's Law has been announced in 1965, it has gone by almost a half of century. Moore's Law says that the amounts of transistors on the single chip would be double per two years. In fact, the fabrication technique has improved over double in current age of deep submicron (DSM), that is, the amounts of transistors on the single chip would be double per eighteen months. As silicon technology scales, more and more circuits could be integrated in a single chip. The amounts of input/output (I/O) signals increase dramatically every unit area. Then, the I/O densities become higher and the space of finite routing resource becomes smaller. This trend will cause the complication in package designs and interaction between package and board. There were several works [17], [18], [20] which were related to package and board physical designs. [17] presented a simulated annealing technique to find a pin assignment solution which considered the routability issue on PGA package and board, but no other DSM effects were considered. [18] presented some efficient patterns to assign the solar balls on board in order to meet the high I/O count for certain products, but still existed a limit to the number of solar balls that were used for power delivery, and routability between package and board. [20] proposed an efficient algorithm which assigned and routed the solder balls of BGA package. But, its fanout routing was only in a single layer and only the routability issue on board was considered.

Comparing with the typical peripheral wire-bond design, flip-chip design can accommodate more input/output (I/O) signals, and can obtain the smaller area of package size. The ball grid array (BGA) solder balls are arranged between package and board. BGA form fully utilizes the area beneath the substrate to assign signal pins and makes up for the routability that the typical peripheral wire-bond design is restricted. Besides, the power and ground pins are located at the center of package and the die is located upon these power and ground pins. The heat generated from the die can be diffused efficiently through these pins in order to avoid the Hot Spot, and thus reduce the defected rate of chips. Flip-chip design has been used widely at present. For package-board codesign, engineers generally arrange the ball chart in the manual manner on experience. This process is iterative, time-consuming and it will lengthen the time-to-market (TTM) of products. [1] proposed a method of generating the BGA ball chart automatically by floorplanning pin-block, thus helped engineers respin the ball chart slightly and saved the arranging time dramatically.

Figure 1.1 shows the typical interface design flow for IC-package-board codesign and a new design flow based on proposed automation design process, respectively. In Figure 1.1(a), we can observe that it spends about one week designating pin-out manually and estimating package size. Then, it spends another one week locating I/O buffer and pad manually and estimating die size. In order to tradeoff signal performance and package cost, engineers always take few weeks to rework package substrate and PCB layout, and rearrange pin-out until the all requirements are satisfied.

On the contrary, Figure 1.1(b) shows the proposed process in [1]. Comparing with manual works shown in Figure 1.1 (a), it completes the works more efficient. The run time of designating pin-out automatically and acquiring the minimum package size is less than five seconds and it just needs about half or one day to fine-tune pin-out. Similarly, if the idea of locating I/O buffer and pad automatically is available, the expected run time of optimizing die size is less than ten seconds. Furthermore, the run time of fine-tuning pad location is expectably half or one day. Obviously, it can shorten the runtime throughout the automation process and obtain the minimum package size and die size.

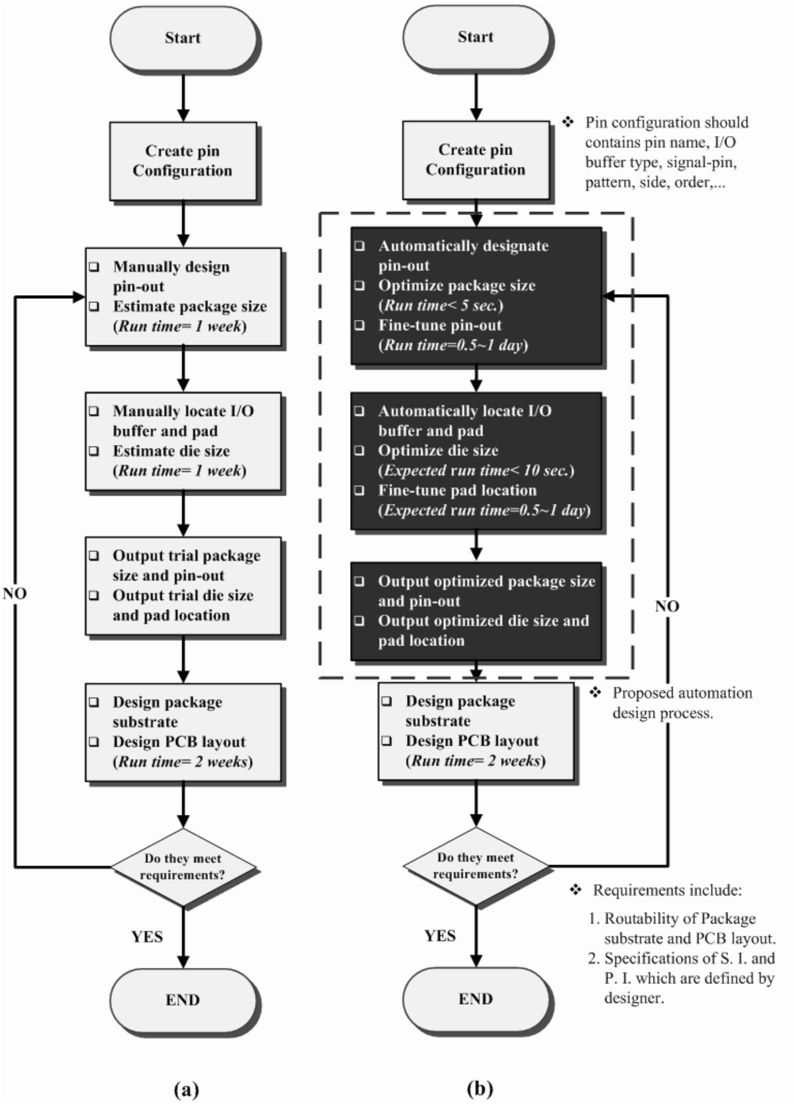


Figure 1.1: The typical flow and proposed approach in interface design for IC-package-board codesign. The focus of [1] is to automate pin-out designation and to minimize package size during design stage [1].

The focus of [1] is to automate pin-out designation and to minimize package size during package-board codesign stage.

Research [1] is the first attempt in solving flip-chip pin-out placement problem in package-board codesign. It evaluates the minimum package size which can accommodate all pins. Then, it will construct pin blocks, which consist of signal-pin blocks and power-pin blocks, and fill them into the minimum package to create the rough pin-out. Finally, the pin-block floorplan will be made into a square one by using an iterative algorithm. The square floorplan can accommodate all pin blocks and has no excess pin blocks in each side. We will accomplish the aforementioned issue in [1] and describe the detail of the automation process implementation in Chapter 3.

However, the method in [1] is less flexible in determining pin configuration chart. The reason is that, we should specify the order for all pin blocks in each side. Because of this order restriction, pin blocks with no freedom have no chance to be placed in better positions in each side of the chip and the results of this floorplan will incur a big cost on the wire length. While empty spaces appear on this floorplan, the worse positions (e.g. the pin blocks which are specified in side1 are assigned in side4 and have empty spaces between the pin blocks which belong to the same group, as shown in Figure 1.2 (b)) will cause another cost as well. Consequently, we have surveyed some methods about floorplanning and placement and try to improve the floorplanner in [1]. The target is to improve the design process and obtain the optimal pin-out.

We have studied some researches about floorplanning and placement which are as follows. Floorplans can be divided into two categories, the slicing floorplanning and non-slicing floorplanning. Many floorplanners are based on slicing floorplans [5], [6], [7], [8]. There are two major advantages of using slicing floorplans. Firstly, the one to one correspondence between skewed slicing trees and normalized Polish expression reduces the search space and leads to a faster runtime. By using simulated annealing method, the efficiency of this benefit is more significant. Secondly, the shape flexibility, that are the modules which may assume any shape permitted by its shape constraints (e.g. a range

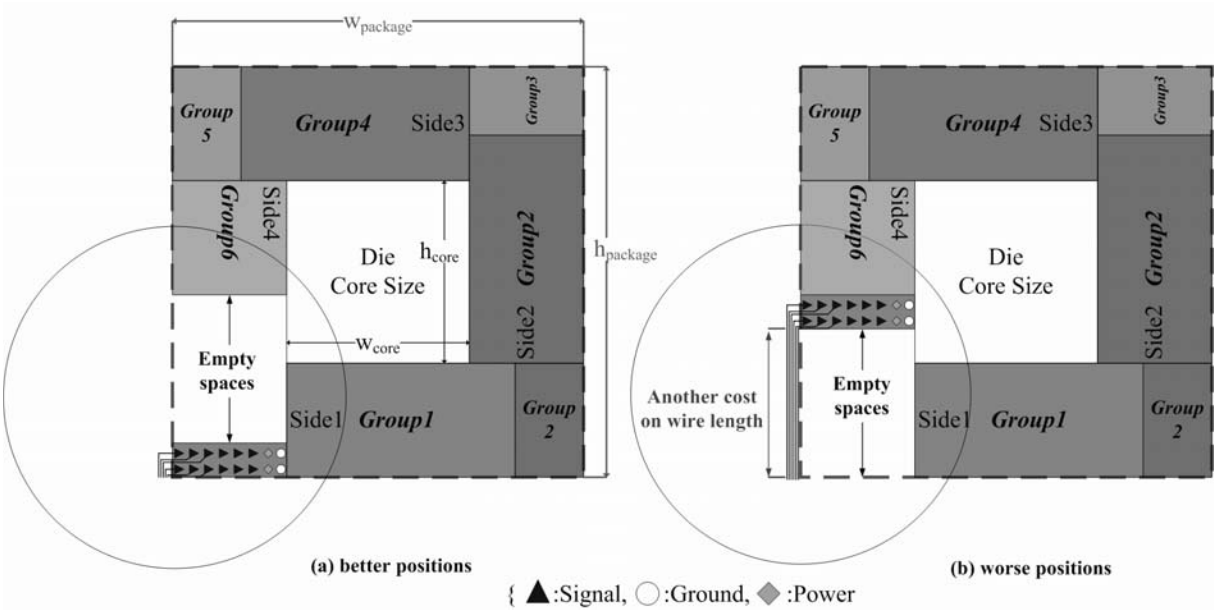


Figure 1.2: While empty spaces appear on floorplan, the worse positions will cause another cost as well.

of possible aspect ratios), can be fully exploited to give a close to optimal final packing based on an efficient shape curve computational technique.

Because of some constraints by using slicing floorplans, there are also some interesting researches in non-slicing floorplans recently. Here are three methods, sequence-pair [3], B*-trees [9], O-Tree [10], which have been proposed for placement of hard modules. However, the B*-trees [9] method can also handle soft modules. The sequence-pair method has later been extended to handle pre-placed modules [2] and soft modules [4]. Recently, there is a research that can handle all of the placement constraint simultaneously in floorplan design [13], including pre-placed constraint, range constraint, boundary constraint, alignment, abutment, and clustering, etc. In order to handle soft modules, it has to solve a mathematical programming problem to determine the exact shape of each module numerous times in the floorplanning process. It wastes a lots of system's resources and results in long runtime. By the way, the B*-trees method has been also extended to handle pre-placed modules [11] and boundary constraint [12]. Comparing with B*-trees [9] method, [11] provides more complete and stronger structure to solve pre-placed modules problems. In floorplanning, it is important to allow users to specify placement constraints in order to get improvement of system's performance, chip area or wire length, etc.

Three common types of placement constraints are pre-placed constraint, boundary constraint, and range constraint. For pre-placed constraint, we require a module to place exactly at a certain position in the final packing. In fact, the problem of floorplanning with obstacles can be solved by treating the obstacles as pre-placed modules. This problem has been considered in both slicing and non-slicing floorplans [2], [4], [6], [11], [13]. For boundary constraint, we require a module to be placed along one particular side of the final floorplan: on the left, on the right, at the bottom, or at the top. This is useful when users want to place some specific modules along the boundary for getting good input-output connections. This problem is considered recently in both slicing and non-slicing floorplans [7], [12], [13]. For range constraint, we require a module to be placed within a given rectangular region in the final packing. This is indeed a more general formulation of the placement constraint problem and any pre-placed constraint can be written as a range constraint by specifying the rectangular region such that it has the same size as the module itself.

## 1.1 Our Contributions

In our proposed methodology, we can consider the core region as a pre-placed module which must be placed in the center of the final packing and pin-blocks as range constraint modules which must be placed within given rectangular regions such that there are no two rectangular blocks overlapping.

In this thesis, we exhibit the procedure of accomplishing this BGA package [1] automatically. We also develop a new representation for this special floorplanning requirement and improve the floorplanner in [1] by using range constraint and simulated annealing algorithm to place pin-blocks. The experimental results show that our improved pin-block floorplanner has more flexibility on specifying critical parameters of the pin configuration and can obtain a better pin-out which has lower cost than that in [1]. Finally, we will show the experimental results of the method in [1] and our improved pin-block floorplanner, and the improvement of our improved pin-block floorplanner.

## 1.2 Organization of this Thesis

The organization of this thesis is as follows. Chapter 2 describes a key previous work and the detail of the automation process implementation in [1]. Chapter 3 describes our improved pin-block floorplanner which is implemented by simulated annealing algorithm with range constraint. Chapter 4 shows the experimental results. We draw conclusions in Chapter 5.

# Chapter 2

# Pin-Out Designation and Package Size Optimization by Pin-Block Floorplanning

In this chapter, we introduce a novel and efficient approach [1] to designate pin-out automatically for flip-chip BGA package when designing chipsets. This approach has taken practical experiences and techniques into account, such as signal integrity, power delivery and routability. It can not only reduce the runtime by means of automation process, but also guarantee the minimum package size and the minimum core size during design stage.

## 2.1 Overview of Fast Flip-Chip Pin-Out Designation Respin by Pin-Block Design and Floorplanning

### 2.1.1 Package-Board Codesign by Considering Signal Integrity and Power Delivery

Figure 2.1 depicts a sketch of PCB layout. Generally the length of signal net from package pin to component or connector on PCB is the primary contributor to parasitic inductance which will make package pins exacerbate simultaneous switching noise (SSN). In order to minimize the physical length of the package pins thus reduce the total parasitic induc-

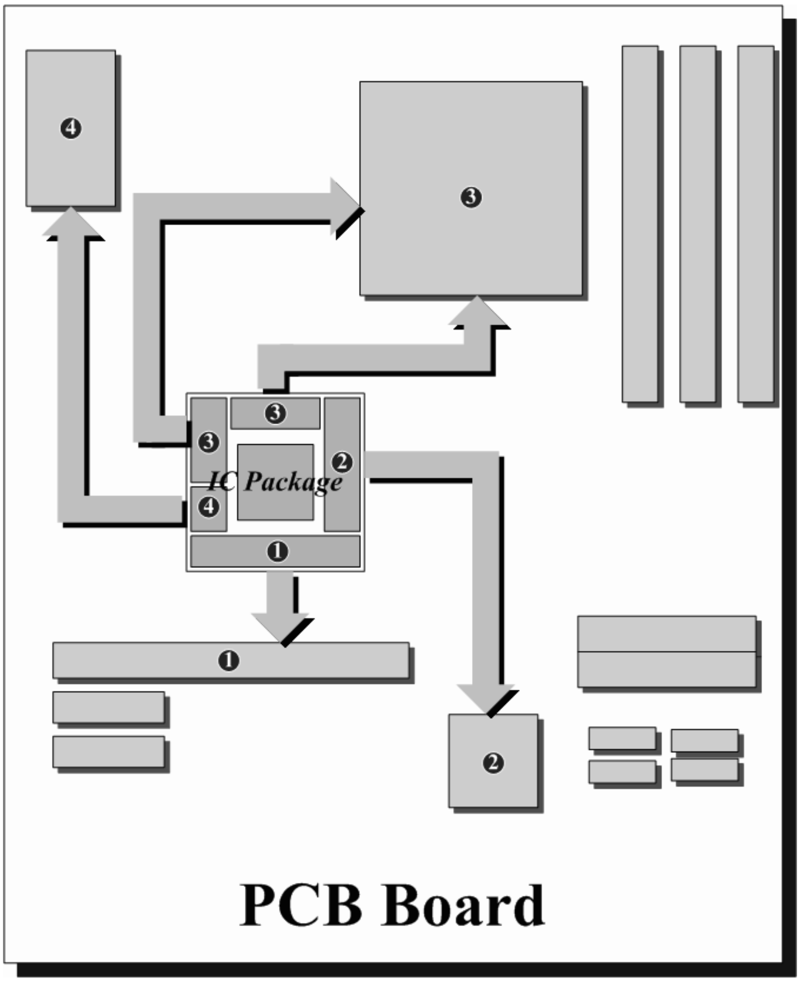


Figure 2.1: A general layout of PCB board. The location of pins on IC package should be restricted in specific regions to meet minimum net-length [1].

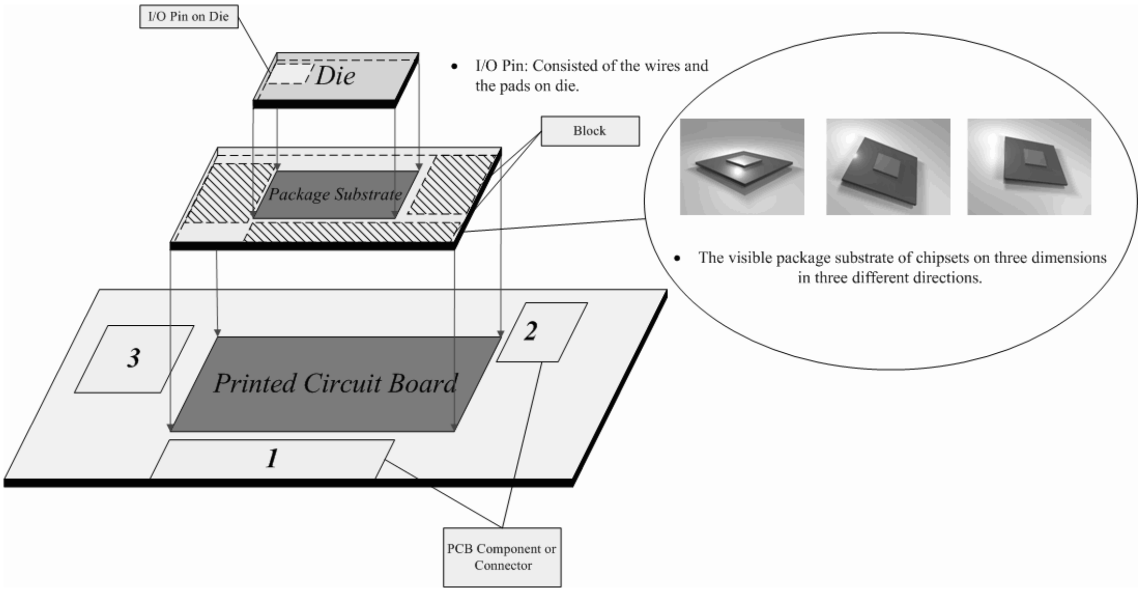


Figure 2.2: The visible package substrate of chipsets.

tance, the signal pins should be allocated and restricted in the particular region according to the certain location of corresponding components or connectors.

Figure 2.2 represents a diagram of a complete flip-chip package. The top layer denotes the die which consists of some logic circuits, I/O pins and I/O buffers. The middle layer denotes the package substrate. Then, the bottom layer denotes the printed circuit board. There are some important connecting devices including PCB components or connectors, etc. on the board. By the way, this diagram shows the visible package substrate of the real industrial case.

Figure 2.3 shows the simplified cross-section of a flip-chip package which is mounted on PCB board. The solder bumps are considered as a bridge that is used to connect the die and the substrate, and the solder balls are exploited to connect the substrate and the PCB board. Based on experienced method, the bumps which are beneath the die and located close to die edge will be routed signal nets through package top layer. On the contrary, the bumps which are located around the core of die will be routed signals through vias and fanned out nets on package bottom layer. For package pins, the solder balls are routed signal nets in the same rule to share finite routing resource.

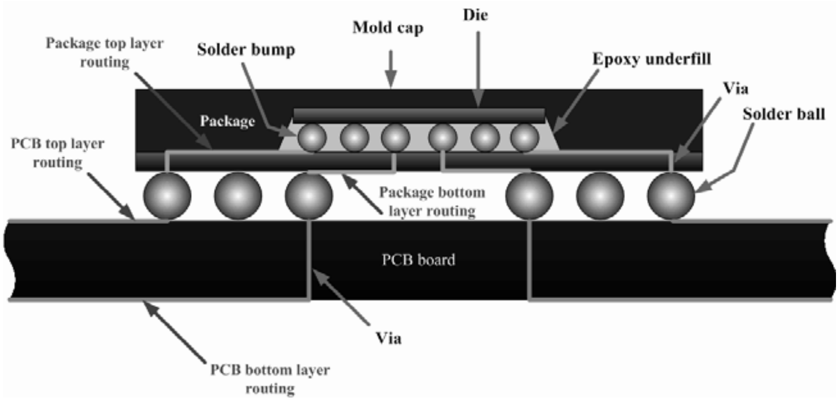


Figure 2.3: The cross-section of a flip-chip package which is mounted on PCB board [1].

For routability, the routing issues should be considered including the net width, spacing, the distance between the pad pitches and so on. Thus, these factors will influent the excess row number used for placing signal pins. By demonstrating in [1], the space between two pads can only be penetrated by two nets. That means only 3 signal nets can be fanned out on PCB top layer. Another 3 signal nets on the same column can be routed through vias and fanned out on PCB bottom layer. In addition, the outer pins include power and ground pins which are routed through vias to the second and third layer of PCB board, besides the signal pins. So, the average row number of outer pins is 8.

As for signal integrity, return path inductance should be considered as well. The unfavorable placement and number of return path pin, power or ground, will maximize current return loops and increase return path inductance. This will dramatically degrade signal integrity and exacerbate radiated emissions. The optimal pin designation is to place signal pin and power or ground pin proximally close to each other, so that each signal pin can be tightly coupled to a return path pin. This will minimize the effect of the return path inductance. However, this optimal design will create such signal-pin blocks which have fewer signal pins within a large block area. [1] addresses six proposed signal-pin patterns for user to make decision that which pattern is most properly exploited for placing signal pins. There exists tradeoff between signal performance and package cost.

Considering power delivery issue, designers can freely define the demand of power

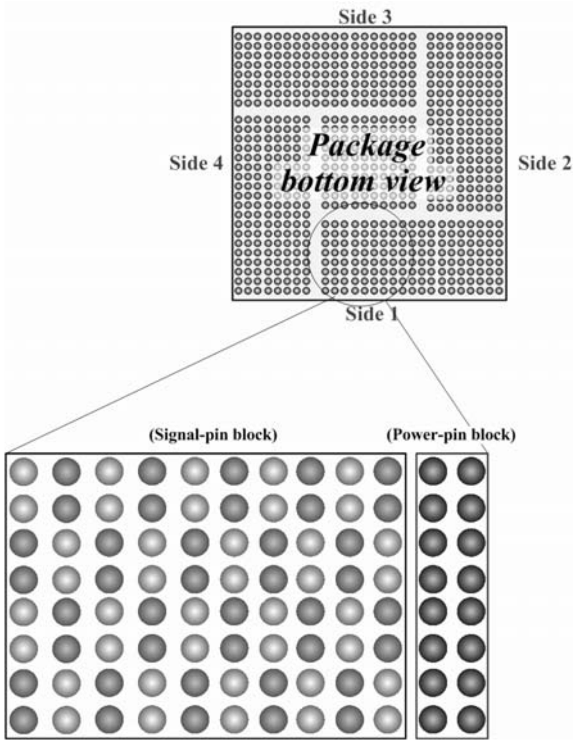


Figure 2.4: A complete pin block includes signal-pin block and its related power-pin block. It is located on the region close to corresponding component on PCB [1].

pins for individual signal configuration. While the signal-pin block is constructed, the proposed automation process will create the power-pin block then place it adjacent to the related signal-pin block. The signal-pin block and the power-pin block will be formed into a complete pin block, as shown in Figure 2.4.

### 2.1.2 Pin-Out Designation Automation by Pin-Block Construction and Floorplanning

In the stage of pin-block construction, designers should determine pin configuration chart, shown in Figure 2.5. The pin configuration chart include all critical parameters defined for placing signal pins, including signal-pin name, selected signal-pin pattern (pattern), the own specific bus (group), the distribution region (side), placement sequence (order), power-pin name and the number of power pins.

After finishing the implementation and placement of all blocks, a rough pin designa-

| Signal-pin name | I/O buffer type | I/O width (um) | I/O height (um) | Selected signal-pin pattern | Group | Side | Order | Power-pin name | Power-pin NO. |
|---|---|---|---|---|---|---|---|---|---|
| AD_P[0:7] | AIO1XH0J | 40 | 500 | 1 | 1 | 1 | 1 | VDDA | 10 |
| AD_N[0:7] | AIO1XH0J | 40 | 500 | 1 | 1 | 1 | 1 | VDDA | 10 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| AD[0:15] | BIO1XH0J | 30 | 350 | 3 | 2 | 1 | 2 | VDDB | 8 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| TEST_IN[0:6] | CIO1XH0J | 25 | 400 | 4 | 3 | 2 | 1 | VDDC | 5 |
| TEST_OUT[0:6] | CIO1XH0J | 25 | 400 | 4 | 3 | 2 | 1 | VDDC | 5 |
| TRAP[0:6] | CIO1XH1J | 25 | 400 | 4 | 3 | 2 | 1 | VDDC | 5 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |

Figure 2.5: An example of pin configuration chart. In this pin configuration we can define specific information as inputs of our proposed automated approach [1].

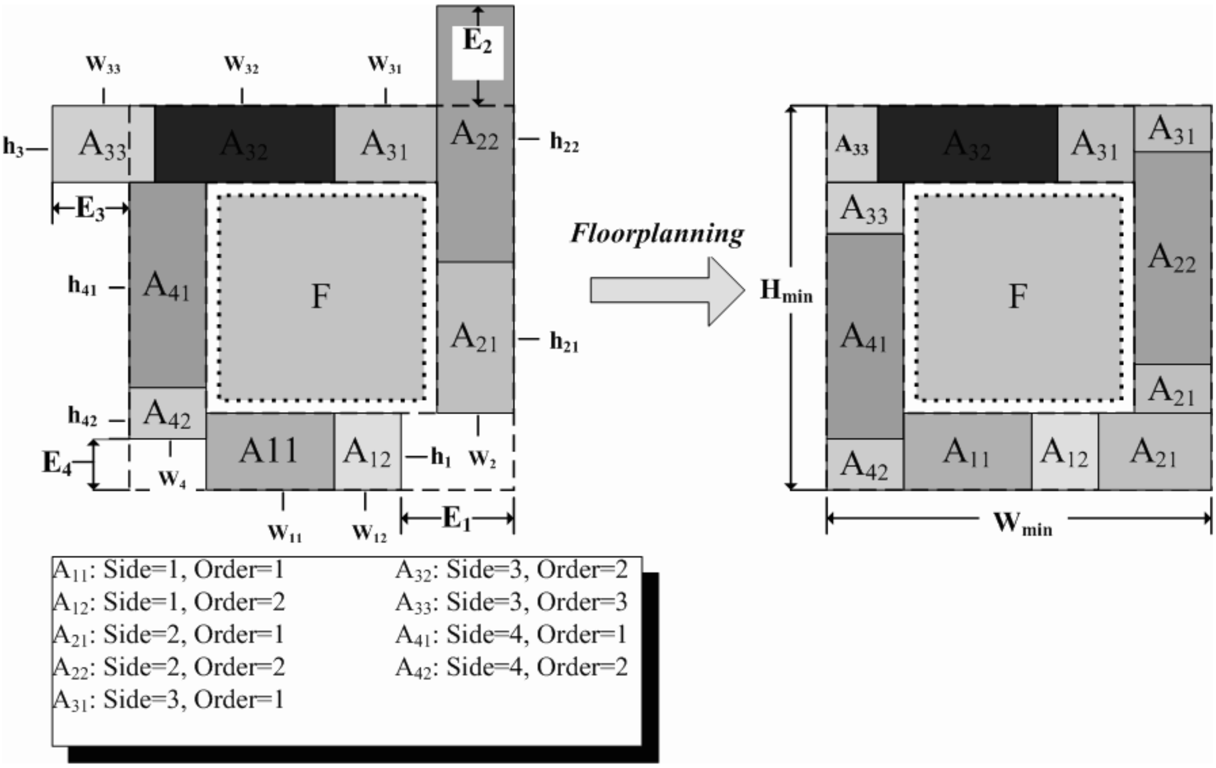


Figure 2.6: A minimum package size can be obtained after we designate and floorplan all pin blocks [1].

tion will be obtained, shown in Figure 2.6. $A_{ij}$ denote the specific bus, where i and j are to represent side and order that blocks are located on and defined in pin configuration by designer. Furthermore designer can acquire parameters $w_{ij}$ and $h_{ij}$ ($w$ and $h$ represent the width and height of each block respectively). At the same time, parameters $E_1$ to $E_4$ can be evaluated from this rough pin designation ($E_1$ to $E_4$ represent the width or height of the empty or excess area in each side of minimum package model). The next step is to minimize package size by mathematical programming and acquire a feasible pin designation. The problem is formulated as follows:

$$Minimize \qquad f = \sum_{j=1,3}(\sum_i w_{ji} + E_j)h_j + \sum_{j=2,4}(\sum_i h_{ji} + E_j)w_j + F$$

*subject to*

$$W = w_4 + \sum_i w_{1i} + E_1 = w_2 + \sum_i w_{3i} + E_3 \tag{2.1}$$

$$H = h_1 + \sum_i h_{2i} + E_2 = h_3 + \sum_i h_{4i} + E_4 \tag{2.2}$$

$$W = H \tag{2.3}$$

$$E_1 + E_2 + E_3 + E_4 \geqq 0 \tag{2.4}$$

where $w_{1i}$, $h_1$, $w_2$, $w_{3i}$, $h_3$, $h_{4i}$, $w_4$ can be evaluated in the previous step, $E_1$ to $E_4$ represent the width or height of the empty or excess area in each side of minimum package model, and $F$ represent the area of core region, shown in Figure 2.6. Equation (2.1) to (2.3) will restrict the shape of package to be square. The purpose of Equation (2.4) is to insure that the minimum package size can accommodate all pin blocks with almost no void pin positions.

The final step of proposed methodology is to floorplan pin blocks, that is, to fine-tune the location of pins in the excess area and fill them into the adjacent empty area. Figure 2.6 shows an example to represent how the rough floorplan can be translated to a square one.

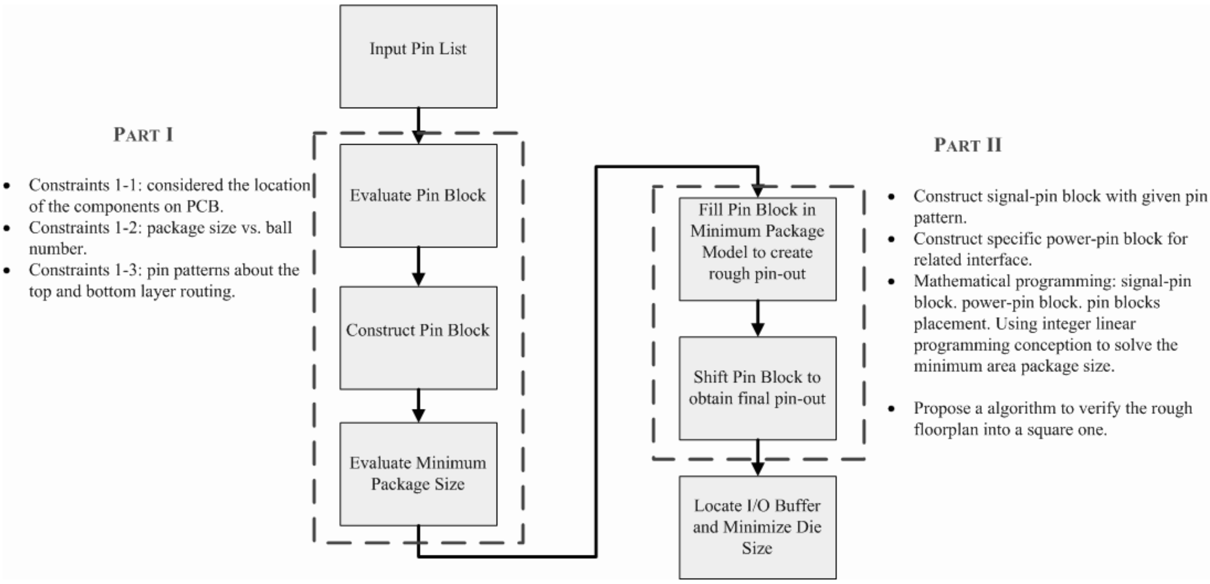


Figure 2.7: The design flow of automation process.

## 2.2 Designation Automation Flow

At the beginning of this procedure, we show the design flow of automation process below (as in Figure 3.1).There are two critical parts in this design flow, Part I and Part II, respectively. There are three steps in Part I, *evaluate pin block*, *construct pin block* and *evaluate minimum package size*, respectively. We should consider several constraints among Part I. Constraints 1-1: considered the location of the components on PCB. The longer wire length of signal net has been pulled, the larger parasitic instance is created. The parasitic instance will cause package pins to exacerbate simultaneous switching noise (SSN). Thus, the signal pins should be placed according to the position of relative components and connectors on print circuit board. Constraints 1-2: package size vs. ball number. Because of the routability on PCB top layer, the space between two pads can only be penetrated by two nets. That means the locations of PCB pad near PCB edge can only allow three signal pins on one column. Other symmetrical four signal pins closed to the core of die can place solder balls through vias. As a result, the average of row number of outer-pin (power-pin, ground-pin and signal-pin) will be predefined as eight. Constraints 1-3: pin patterns about the top and bottom layer routing. Making a decision between these six patterns is based on experience of engineers. According to the characteristics of signal-pin patterns, engineers can select an adaptable pattern for designating pins.

There are two steps in Part II, *fill pin block in minimum package model to create rough pin-out* and *shift pin block to obtain final pin-out*. User defines the signal pins in which side and in which group. Then, our program constructs the signal-pin blocks with chosen patterns and also builds the specific power-pin blocks for related interface. After previous procedure, it will group the signal-pin blocks and the power-pin blocks into modules such as pin blocks and fill pin blocks in the minimum package model to create the rough pin-out floorplan. In next step, we propose an algorithm to regulate the exceeding region of the rough pin-out flooplan into a square one which is accommodated all pin blocks.

## 2.3 Pin-Block Floorplanning

At present, we just get a rough floorplan which owns the property described previously with the minimum core size and the minimum package size. However, this rough floorplan still has pin block in excess area and is not consistent with the size of minimum package floorplan. So, we proposed an simple algorithm to make the rough floorplan into a square one. This algorithm does not destroy the position of pin block enormously and make the location of pin block as near to the corresponding side of the chip as possible. The algorithm is shown as follows:

Algorithm:

1. $i \leftarrow 1$ //start from side 1

2. $i-1 \leftarrow 4$, iff $i=1$ ; $i+1 \leftarrow 1$, iff $i=4$

3. while ($E_i < 0$) do, for $i \epsilon \{1,2,3,4\}$

4. if ($E_i \neq 0 \&\& E_i < 0$)

5. if $E_{i-1} > E_{i+1}$

6. shift pins clockwise //fill the pin block into empty area in last side until the current $E_i$ value is zero

7. $E_i \leftarrow 0$, $E_{i-1} \leftarrow E_{i-1} + E_i$

8. else

9. shift pins anticlockwise //split the pin block in excess area then group it into next side

10. $E_i \leftarrow 0$, $E_{i+1} \leftarrow E_{i+1} + E_i$

11. $i \leftarrow i + 1$ //check next side

12. until all $E_i$ values are large than or equal to zero

The algorithm starts form the side1 of the rough floorplan. It will check whether this side have excess pin block. If this side has excess blocks, the algorithm will check the empty space of the previous side and the next side. If the space of the previous side is greater than the other, the algorithm will shift pins in the clockwise direction and fill the pin blocks which are in the excess area into the empty area in the previous side until the current value of $E_i$ is zero. Otherwise, it will shift pins in the anticlockwise direction and split the pin blocks in excess area into one module, then it will group it into the next side. Every iteration of this procedure will check the value of $E_i$ in each side whether it is greater than zero. While all values of $E_i$ in every side are greater than zero, then the algorithm will stop and represent the final square floorplan. Next, We will implement two real cases in practice, that will be more clearly about describing this automation process.

The program will gather statistics about the columns of pins in each side by means of feeding the program with the pin-lists. Afterwards, the program solves the linear problem to computer the minimum core size and the minimum package size. Figure 2.8 describes the columns of pins in each side, the minimum core size, the minimum package size, the columns in excess region of the package and the sum of all values of $E_i$.

The rough floorplans of this two cases are showed as in Figure 2.9 and Figure 2.10 respectively. In Figure 2.9, we can observe that the side1 has columns in the excess area ($E_1 = -3$), the side2 has empty space ($E_2 = 2$), the side3 has excess space ($E_3 = -6$)

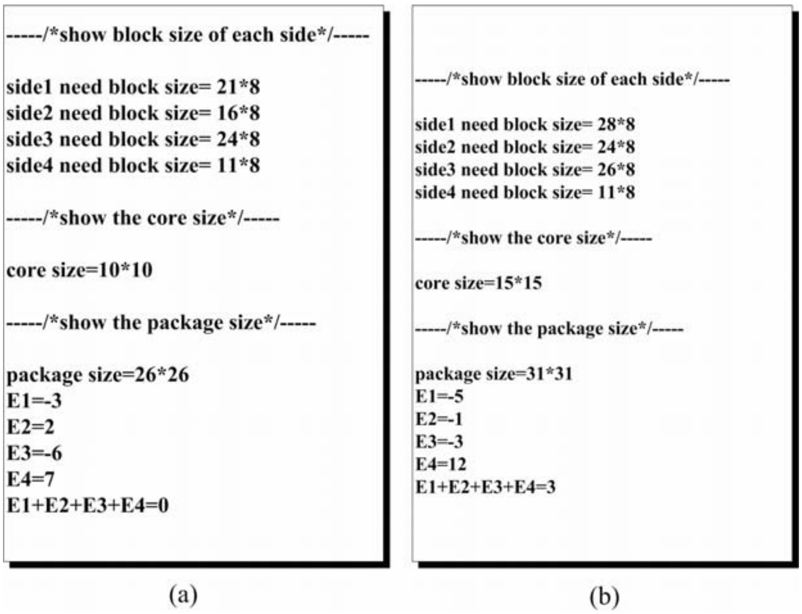


Figure 2.8: (a) Result in Case I. (b) Result in Case II.

and the side4 has columns in empty area ($E_4 = 7$) in case I. In Figure 2.10, we find that the side1 has excess space ($E_1 = -5$), the side2 has columns in excess area ($E_2 = -1$), the side3 has columns in excess area ($E_3 = -3$) and the side4 has empty space ($E_4 = 12$) in case II. Next, our program will implement the simple algorithm described above. The rough floorplan will be shifted pins and will be made as a square one.

Figure 2.11 and Figure 2.12 describe the proceedings of carrying out the algorithm which makes the rough floorplan as a square one. In Figure 2.11 (a), the side1 has three columns of pin blocks in excess area ($E_1 = -3$) and, thus, the algorithm will check the empty space of the previous side ($E_4$) and the next side ($E_2$). The space of the previous side is greater than the other (i.e. $E_4 = 7 > E_2 = 2$), the algorithm will shift pins in the clockwise direction and fill the pin blocks which are in the excess area into the empty area in the previous side until the current value of $E_i$ ($E_1$) is zero. The $E_1$ will be set to 0, $E_4$ is equal to 4 (i.e. $7 - 3$) and $E_2$ $E_3$ are the same, as shown in Figure 2.11 (b). Next step, we check the side2 (in Figure 2.11 (b)). Because the values of $E_2$ is greater than zero ($E_2 = 2$), so it is kept in the original status. Then, the side3 (in Figure 2.11 (b)) has six columns of pin blocks in excess area ($E_3 = -6$). The space of the next side is

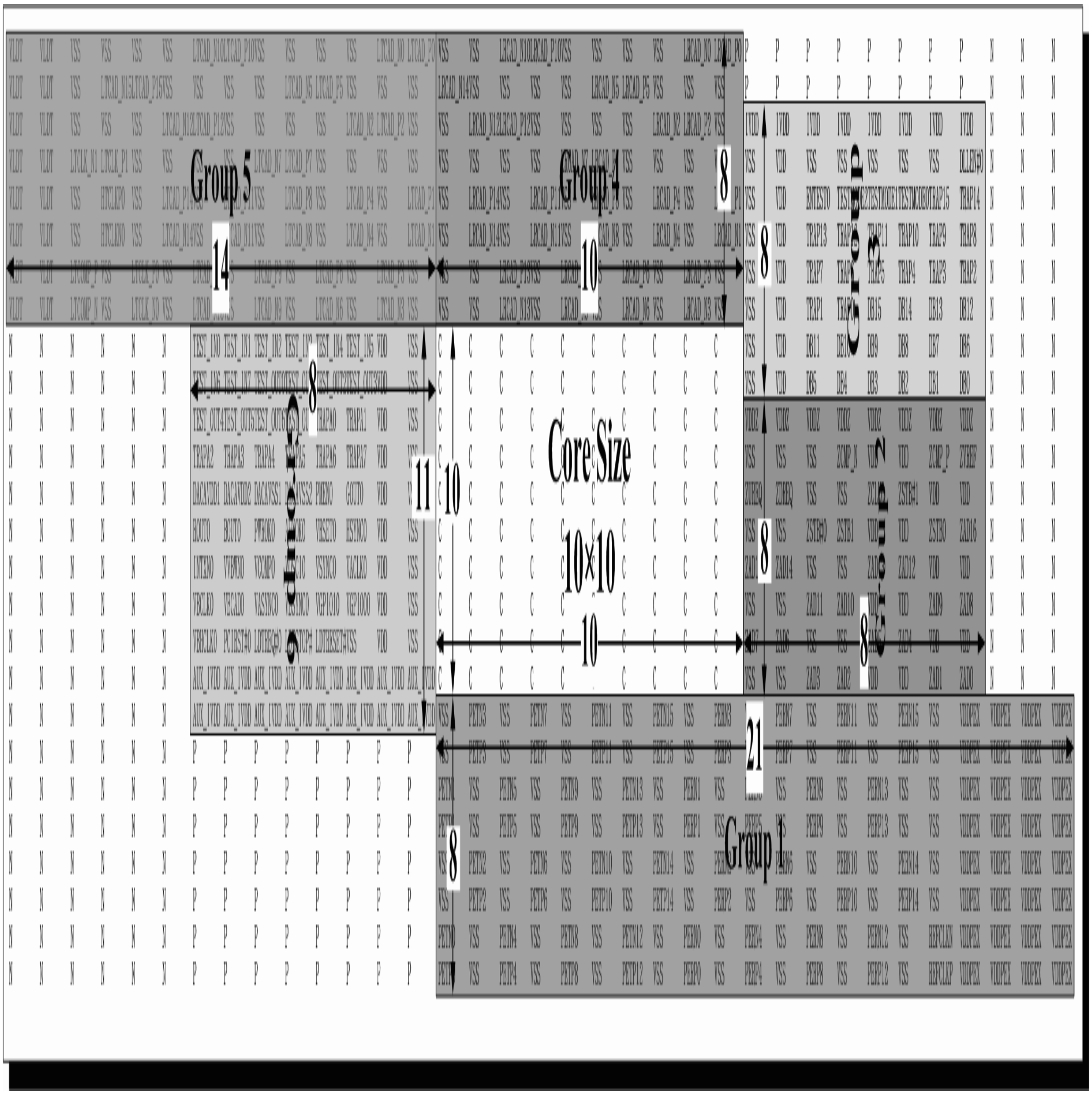


Figure 2.9: Rough floorplan in Case I. (uppercase P denotes the minimum package size, uppercase C denotes the minimum core size, uppercase N denotes nothing and pin-block groups are denoted in shaded blocks.)

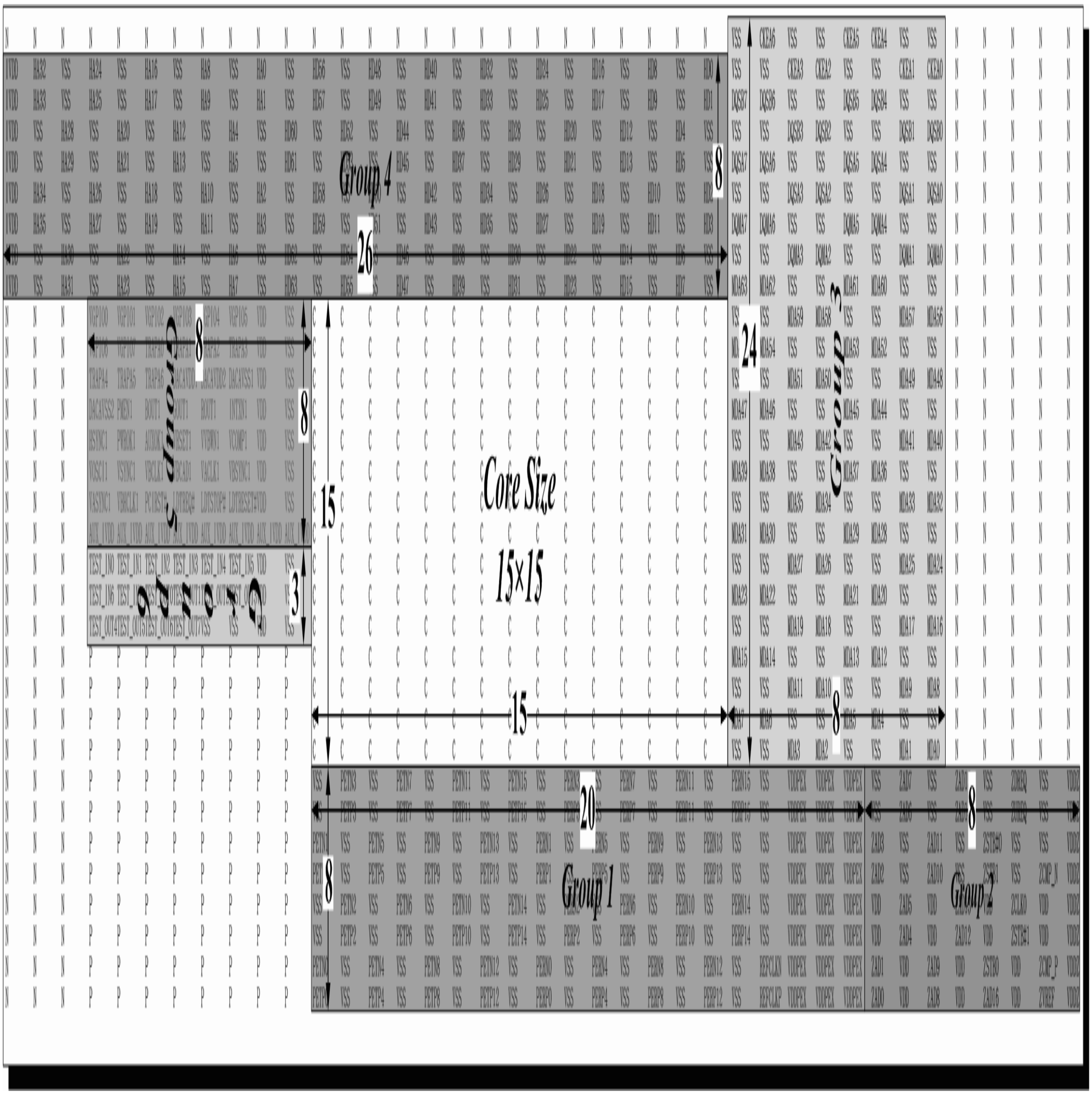


Figure 2.10: Rough floorplan in Case II. (uppercase P denotes the minimum package size, uppercase C denotes the minimum core size, uppercase N denotes nothing and pin-block groups are denoted in shaded blocks.)

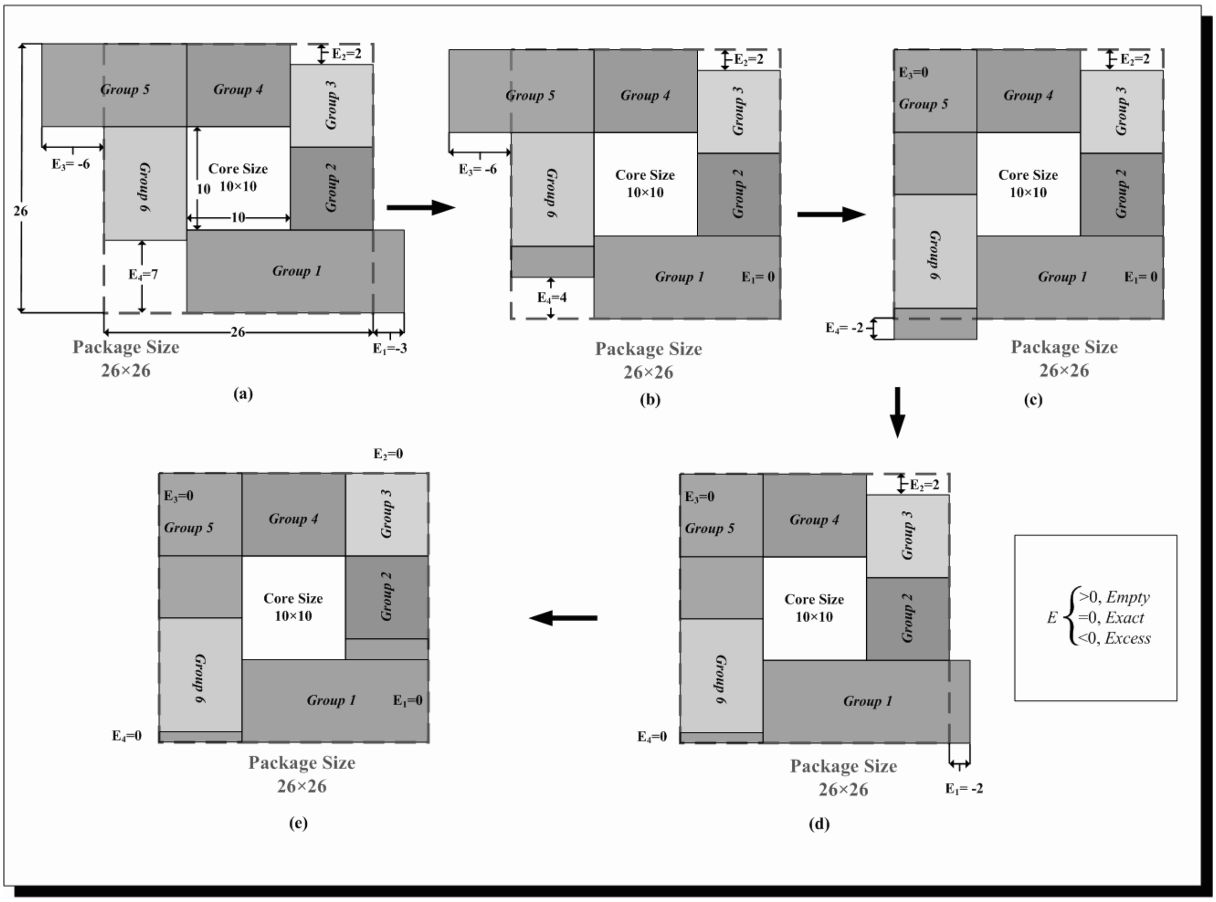


Figure 2.11: Diagrams for interpretation of carrying out the algorithm from the rough floorplan to the final floorplan in Case I.

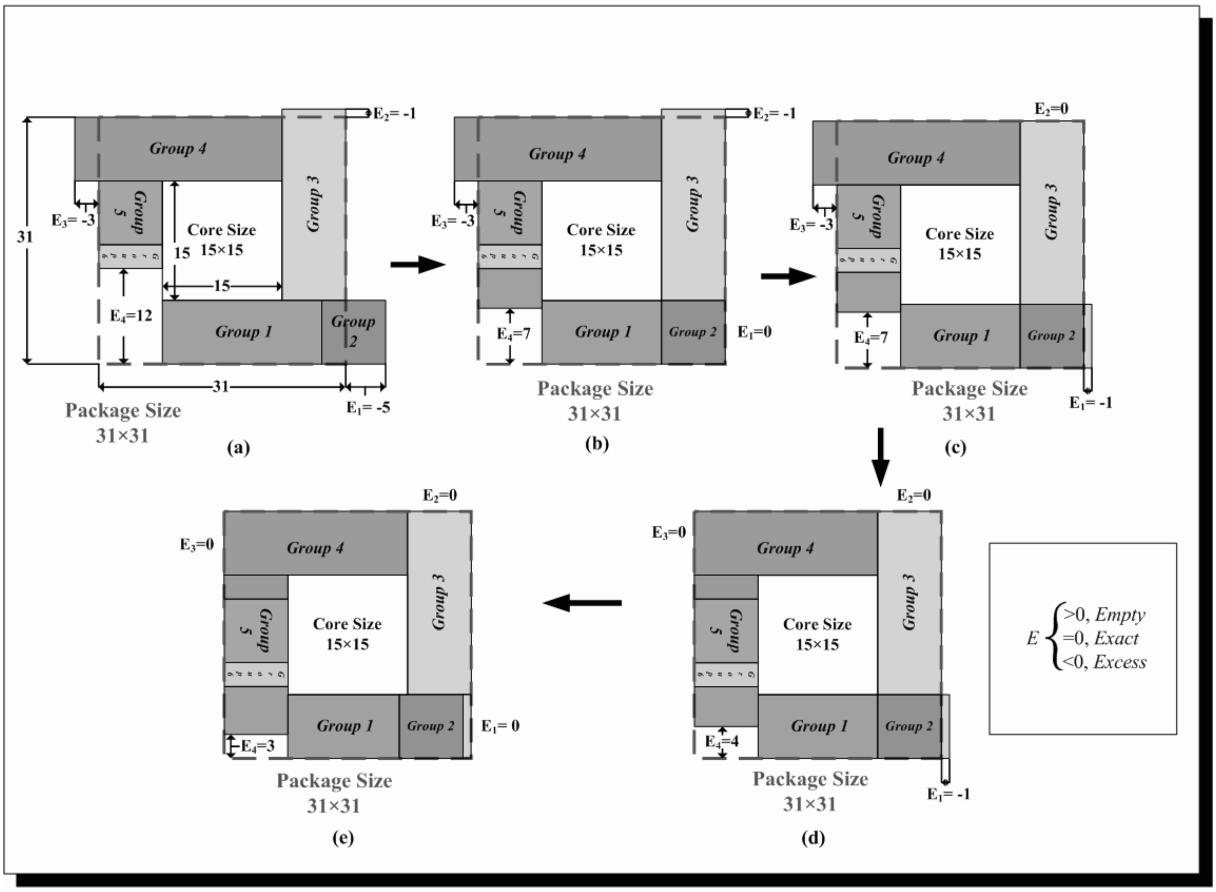

Figure 2.12: Diagrams for interpretation of carrying out the algorithm from the rough floorplan to the final floorplan in Case II.

greater than the other ($E_4 = 4 > E_2 = 2$) and, thus, the algorithm will fill the pin blocks which are in excess area into empty area in next side ($E_4$). The $E_3$ will be set to 0, $E_4$ is equal to $-2$ (i.e. $4 - 6$) and $E_1$ $E_2$ are the same, as shown in Figure 2.11 (c). Next step, we check the side4 (in Figure 2.11 (c)) and side4 has two columns of pin blocks in excess area ($E_4 = -2$). The space of the previous side is equal to the other ($E_3 = E_1 = 0$) and, thus, the algorithm will fill the pin blocks which are in excess area into the next side ($E_1$). Then ($E_4$) will be set to 0, ($E_1$) is equal to $-2$ (i.e. $0 - 2$), as showed in Figure 2.11 (d). At present, it is just implemented in one cycle. In fact, the algorithm is iterative and will implement repeatedly until the value of $E_i$ in each side is greater than zero. Otherwise, it will never stop. Figure 2.13 shows the final floorplan of case I and is accurate at the corresponding Figure 2.11 (e). Case II is a more general case about this algorithm, as shown similarly in Figure 2.12 and Figure 2.14.

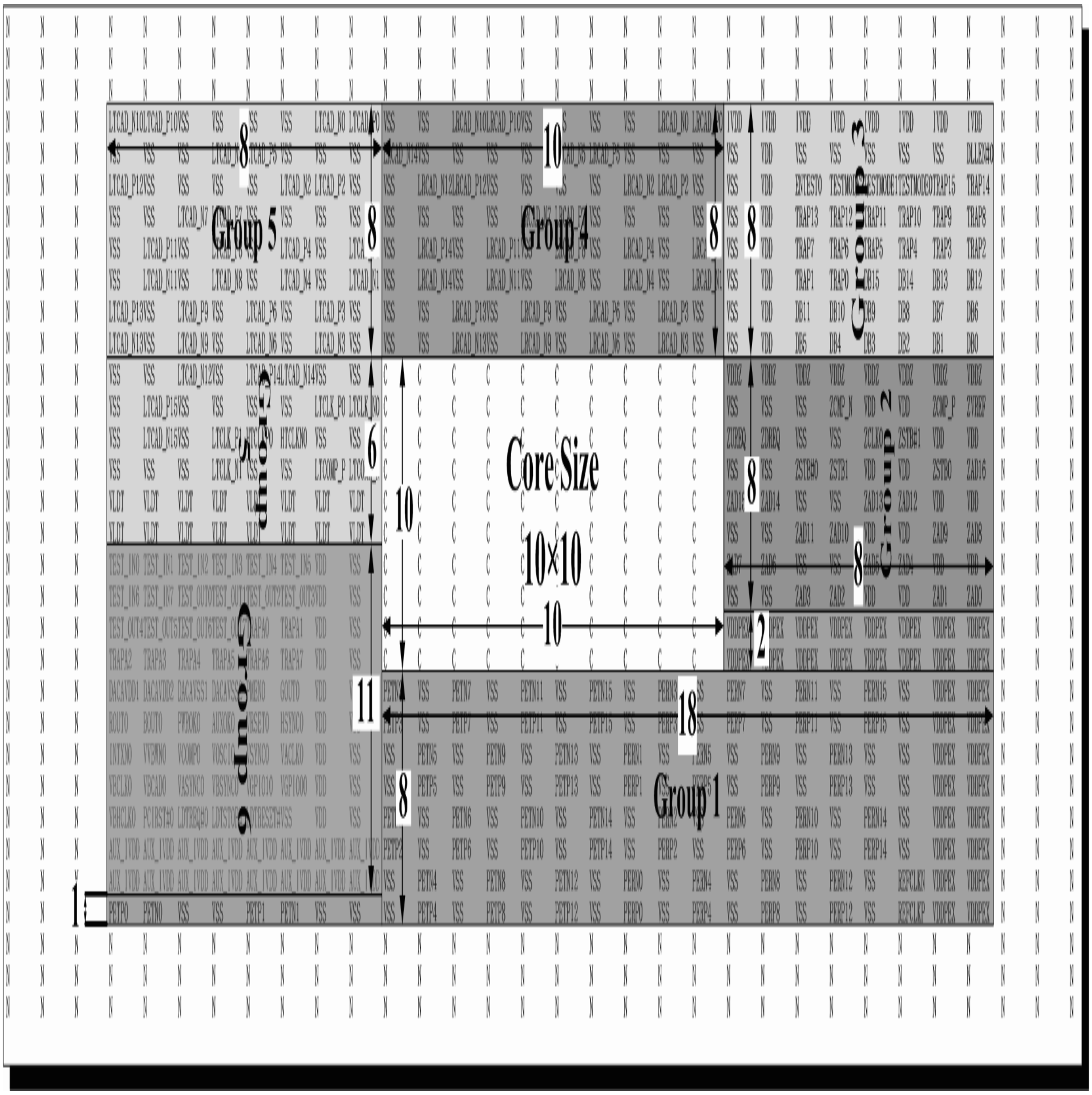


Figure 2.13: Final floorplan in Case I. (uppercase P denotes the minimum package size, uppercase C denotes the minimum core size, uppercase N denotes nothing and pin-block groups are denoted in shaded blocks.)

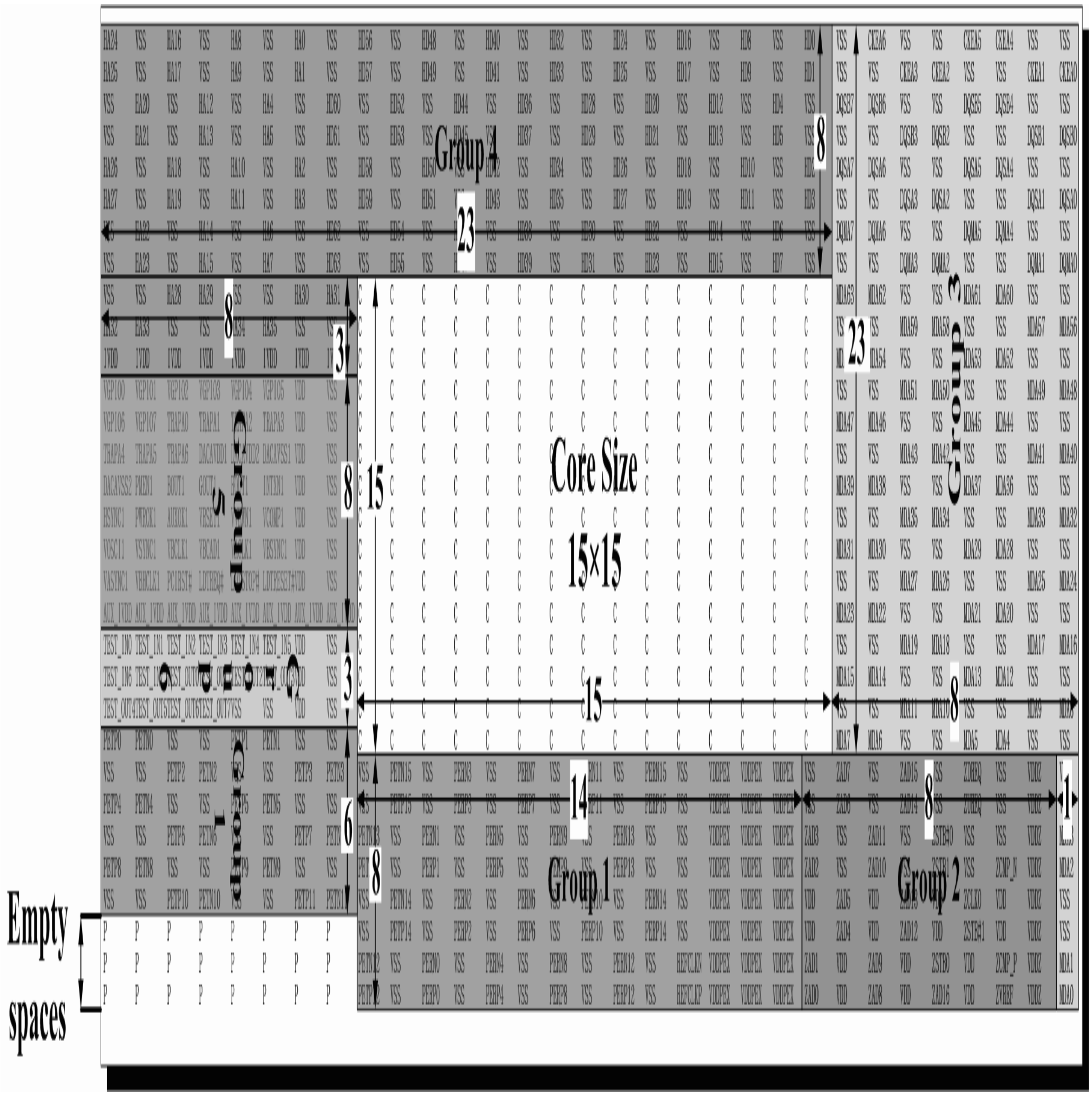


Figure 2.14: Final floorplan in Case II. (uppercase P denotes the minimum package size, uppercase C denotes the minimum core size, uppercase N denotes nothing and pin-block groups are denoted in shaded blocks.)

# Chapter 3

# Improved Pin-Block Floorplanner

Chapter 2 shows the detail of implementing the automation process in [1]. The floorplanner in [1] exists some problems which have been described before. That is, the method in [1] is less flexible in determining pin configuration chart and the non-well positions generated from [1] will cause another cost as well. In this chapter, our target is trying to solve those problems and we improve the floopplanner in [1] by range constraint to place pin-blocks. Meanwhile, improved pin-block floorplanner is implemented by using the simulated annealing algorithm.

## 3.1 Problem Definition

The minimum package size has been evaluated from Chapter 2 and we use the minimum package size directly as a graphic frame of our problem. In our problem, the final packing is consisted of core block and several pin blocks around core block. The core block of the packing is just denoted as a module which is restricted by pre-placed constraint and the pin blocks around the core block are considered as modules that are restricted by range constraint [8]. Figure 3.1 briefly describes the placement of the core block and shows how the pin blocks without range constraint are placed around the core block.

In Figure 3.1, the whole floorplan is in the first quadrant with its lower left corner at the origin $(0, 0)$. The core block (C) which is restricted by pre-placed constraint can be

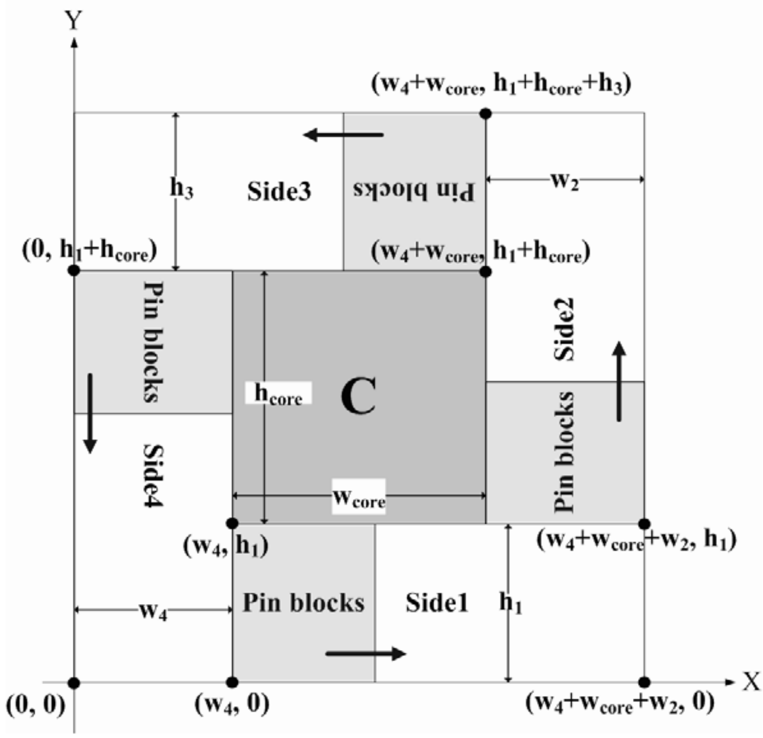


Figure 3.1: Final packing is consisted of core block and pin blocks without range constraint around the core block.

considered as a hard module with fixed orientation and the height of module C is denoted as $h_{core}$ and the width of it is denoted as $w_{core}$. The core block must be placed with its lower left corner at $(w_4, h_1)$ in the final packing. Because the pre-placed constraint is a special case of range constraint in which the module has no freedom of movement, so module C can be specified by a range constraint requiring the module to be placed within the rectangular region $R_{core} = \{(x, y)|w_4 \leqq x \leqq w_4+w_{core}, h_1 \leqq y \leqq h_1+h_{core}\}$. Then, pin blocks without range constraint are placed around the core block. For side1, the height of pin blocks is defined as $h_1$. Pin blocks are placed with fixed orientation and are assigned with its lower left corner at $(w_4, 0)$. Pin blocks are also placed toward the increasing direction of the x coordinate. Analogically, the height of pin blocks placed in side2 is defined as $w_2$ and pin blocks are assigned with its lower left corner at $(w_4+w_{core}+w_2, h_1)$ toward the increasing direction of the y coordinate. And pin blocks placed in side3 with its height as $h_3$ are assigned with its lower left corner at $(w_4+w_{core}, h_1+h_{core}+h_3)$ toward

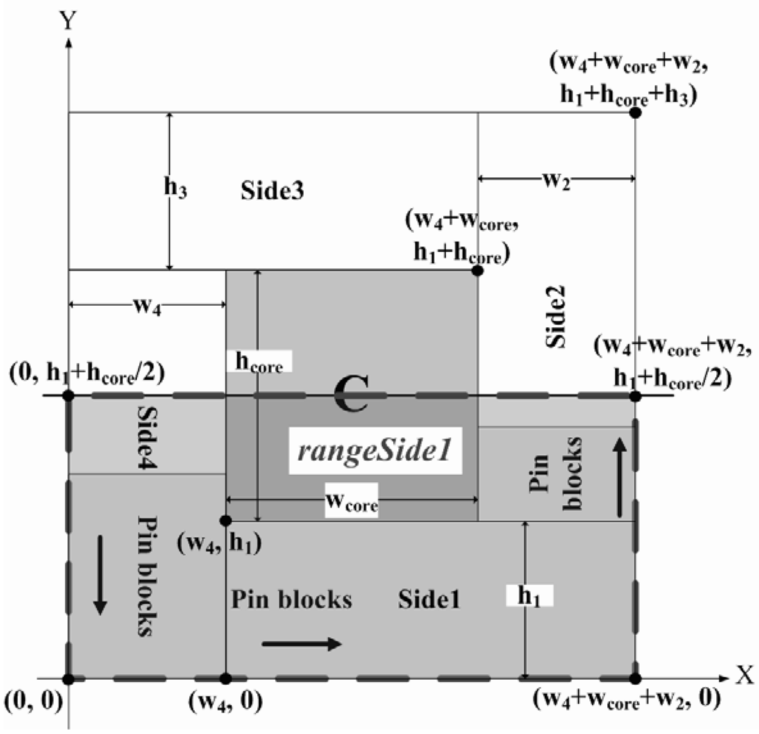


Figure 3.2: Pin blocks with the range constraint which must be placed within the given rectangular region.

the decreasing direction of the x coordinate. At last, pin blocks placed in side4 with its height as $w_4$ are assigned with its lower left corner at $(0, h_1 + h_{core})$ toward the decreasing direction of the y coordinate. This manner of placing pin blocks is equally like the rough floorplanning which is described before in Chapter 2. However, our problem defines the range constraint to floorplan pin blocks which is placed around the core block.

Figure 3.2 briefly shows an example about the definition of the range constraint for side1. We specify the rectangular region called $rangeSide1$ and pin blocks with range constraint must be placed within $rangeSide1$. Range constraints are listed below and shown in Figure 3.3:

- $rangeSide1 = \{(x, y)|0 \leqq x \leqq w_4 + w_{core} + w_2, 0 \leqq y \leqq h_1 + h_{core}/2\}$.
- $rangeSide2 = \{(x, y)|w_4 + w_{core}/2 \leqq x \leqq w_4 + w_{core} + w_2, 0 \leqq y \leqq h_1 + h_{core} + h_3\}$.
- $rangeSide3 = \{(x, y)|0 \leqq x \leqq w_4 + w_{core} + w_2, h_1 + h_{core}/2 \leqq y \leqq h_1 + h_{core} + h_3\}$.

- $rangeSide4 = \{(x, y)|0 \leqq x \leqq w_4 + w_{core}/2, 0 \leqq y \leqq h_1 + h_{core} + h_3\}$.

A feasible packing must satisfy all placement constraints (including the pre-placed constraint and the range constraint). A feasible packing is a nonoverlap placement of all pin blocks in it, and all pin blocks can be accommodated within the feasible packing. Our target is to improve the floorplanner in [1] by using the range constraint and place pin blocks into better positions on package. And our objective function is formulated as W+$\rho$P, where W is the sum total wire length which is calculated the Manhattan distance between pins and the edge of each side which the pins belong to, P is a penalty term which is an estimation of the square distance between pins and their desired positions in each side, and $\rho$ is a user-defined balance constant which controls the relative importance of wire length and penalty term in the objective function. Meanwhile, improved pin-block floorplanner is implemented by using simulated annealing algorithm.

## 3.2 Range Constrained Pin-Block Sequence Pair (RCPBSP)

We propose a novel sequence-pair representation to construct a scheme of our problem. This representation describes all varieties of perturbations in placing the pin blocks around the core block. For case I, the representation is resulted in $s$ = (123456, I1 II1 II2 III1 III2 IV1). The first sequence denotes groups (i.e. group1, group2,. . ., group6). The second sequence means the range constraint. They are I, II, III and IV, that is, $rangeSide1$, $rangeSide2$, $rangeSide3$ and $rangeSide4$, respectively. The number behind the rangeSide means the order of groups which are placed in the same range constraint. Figure 3.4 shows some examples of the representation in our problem.

In Figure 3.4, all floorplans are feasible packings which satisfy the range constraint in each side and the representation is showed under each graph. In Figure 3.4 (a), the placement of pin blocks is started from group1 with $rangeSide1$ and the sequence of groups is showed as (123456) in the anticlockwise direction. In Figure 3.4 (b), the placement is started from group3 with $rangeSide2$ and the sequence is (324561). Group2 and group3

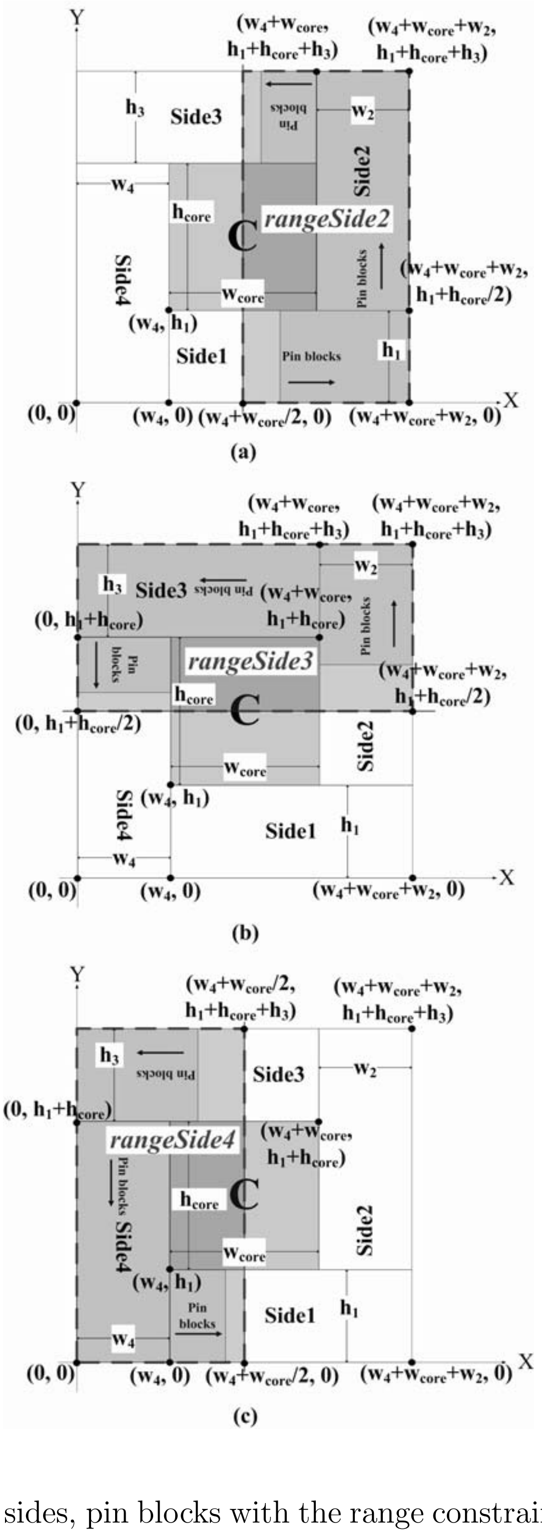


Figure 3.3: For the other sides, pin blocks with the range constraint must be placed within *rangeSide*2, *rangeSide*3 and *rangeSide*4.

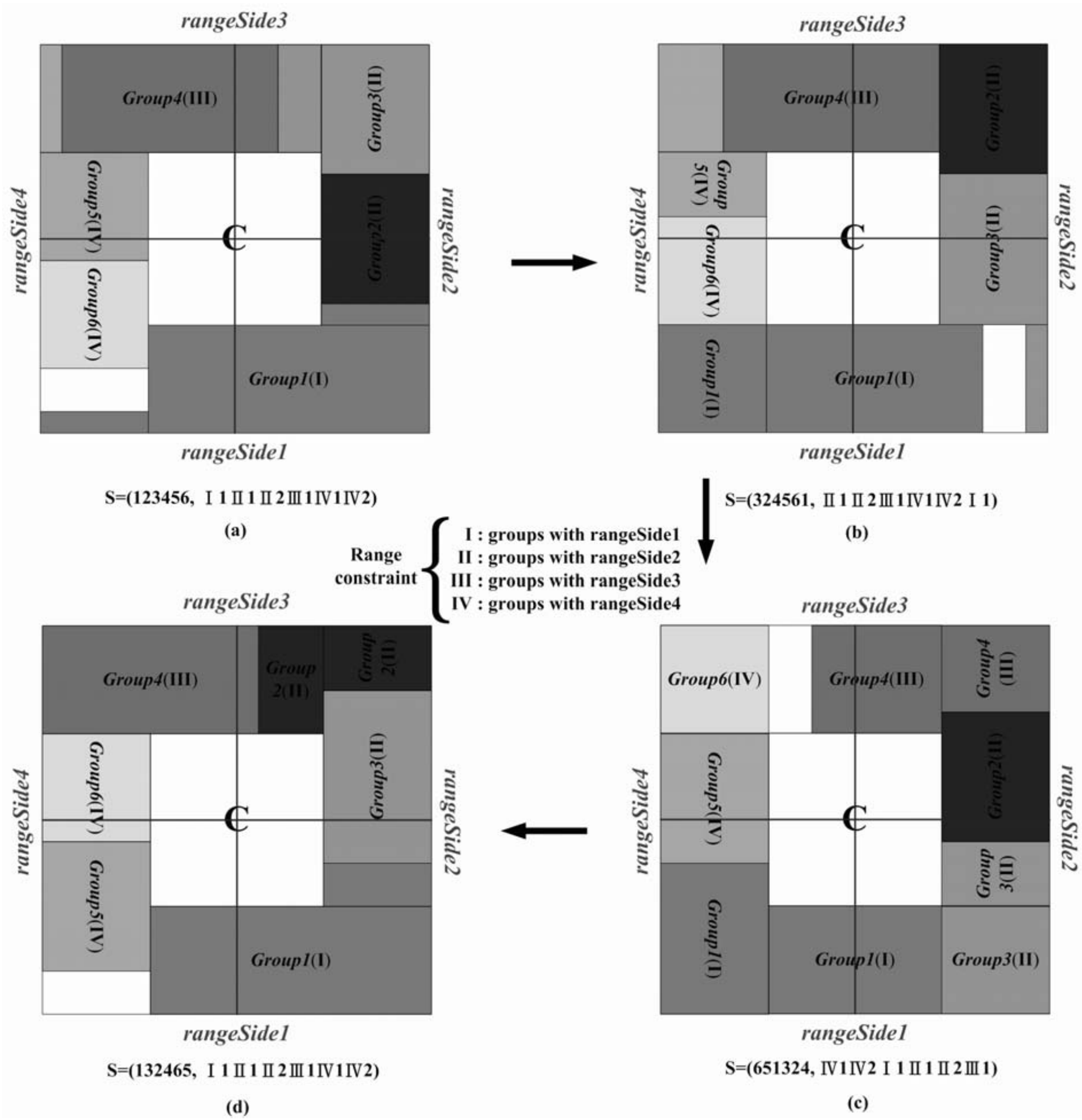


Figure 3.4: Some examples of the representation in our problem.

have done pair-interchanged exchanging in this stage. Similarly, the representations of Figure 3.4 (c) and Figure 3.4 (d) are showed in the same way. And later we will describe the perturbations of this representation in detail.

## 3.3 Floorplanning with Range Constraint

In the aforementioned description, we know that all groups must be placed within a given rectangular region. Figure 3.5 illustrates an example where five groups with range constraint are restricted in $rangeSide1$. (group a, b, c, d and e, respectively). Because of hard modules, each group has the same height as eight. The width of each group is depended on the amount of signal pins and the pattern selected by user. So, the only thing that we care about is the total width of groups which must be placed within $rangeSide1$, where $w_a$, $w_b$ , $w_c$, $w_d$ and $w_e$ denotes the width of group a, b, c, d and e, respectively. The total width of groups in $rangeSide1$ is $w_{s1} = w_a + w_b + w_c + w_d + w_e$. And the definition of $rangeSide1$ is shown as follows and illustrated in Figure 3.5 (b):

- $R_s$: The start point of $rangeSide1$, $R_s = 0$.
- $R_e$: The end point of $rangeSide1$, $R_e = R_s + h_{core}/2 + h_1 + w_{core} + w_2 + h_{core}/2$.
- Cs: The check start point of groups restricted by $rangeSide1$, $C_s = R_s + rd_a$.
- Ce: The check end point of groups restricted by $rangeSide1$, $C_e = C_s + w_{s1}$.

The minimum core size (C) of the chip is evaluated as $h_{core} * w_{core}$. And $rd_a$ means random selected start-point of group a where group a should be placed from. Firstly, we consider the allowed range of all groups. For group a, the allowed range of module a can be denoted as $ar_a = R_e - w_a$, as shown in Figure 3.5 (c). The random selected start point (i.e. $rd_a$) should satisfy the restricted range $R_s \leqq rd_a \leqq ar_a$. Similarly, other groups are all obeying this rule of randomly selecting start point. A feasible solution of the floorplan will satisfy the range constraints as following rules:

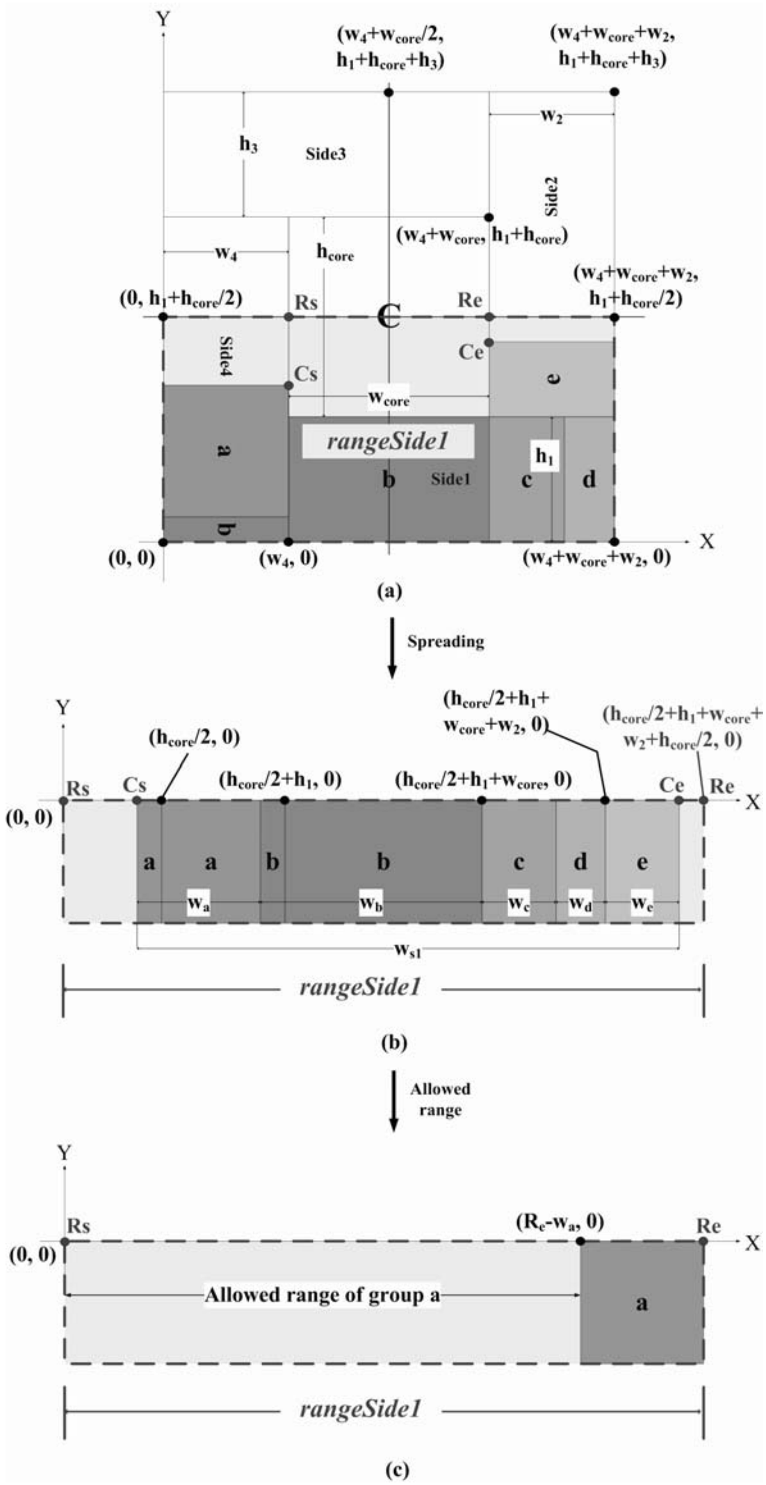


Figure 3.5: An example illustrates groups with range constraint (*rangeSide*1).

- Random selected start-point should be within the restricted range: $R_s \leqq rd_a \leqq ar_a$.

- Total width of groups restricted by $rangeSide1$ should be within the spreading length of the given rectangular region: $C_s \geqq R_s$ and $C_e \leqq R_e$.

In addition, the start point of groups in $rangeSide2$ is carried on the checked end-point ($C_s$) of groups in $rangeSide1$. Groups in $rangeSide2$ are placed from the start point at $(C_s + 1)$ and the order of groups is following the representation mentioned before.

Analogously, the aforementioned rules are proper for other range constraints in each side. Consequently, we will acquire a feasible solution as long as the range constraints of the four sides are all satisfied and non-overlapping modules appear in the final packing.

## 3.4 RCPBSP Packing by Simulated Annealing

### 3.4.1 Solution Perturbation and Cost Function

Firstly, our problem is basically implemented on simulated annealing algorithm. We briefly describe perturbations in our representation.

- In the first step, we randomly select which $rangeSide$ should be perturbed.

- In the second step, we randomly select any two groups of $rangeSide$ selected in first step and do pair-interchanged exchanging. If selected $rangeSide$ has more than two groups, then the two groups selected in first step have done pair-interchanged exchanging. Otherwise, the sequence of our representation is still the same.

- In the final step, we randomly select the start point of the group with the first order in selected $rangeSide$ and rearrange other groups by following the sequence of the representation until all $rangeSides$ have been considered.

However, the cost function is defined as W+ $\rho$ P where W is an estimation of total wire length which is Manhattan distance between signal pins and the edge of each side which

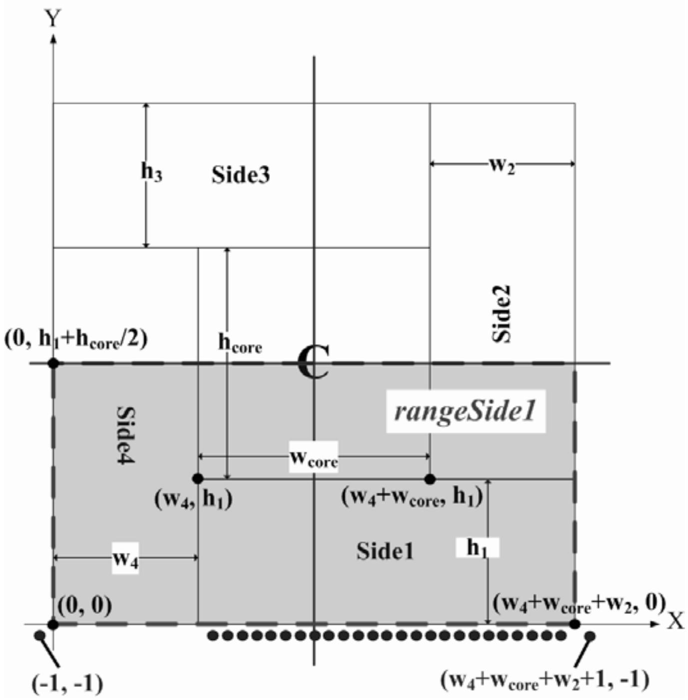


Figure 3.6: An estimation of wire length between signal pins belonged to *rangeSide*1 and the edge of side1.

signal pins belong to. P is a penalty term which is an estimation of the square distance between pins and their desired positions in each side, and $\rho$ is a user-defined balance constant which controls the relative importance of wire length and penalty term in the objective function.

For *rangeSide*1, Figure 3.6 describes an estimation of wire length between signal pins belonged to *rangeSide*1 and the edge of side1. There are three parts of the estimation of the wire length. They are signal pins with *rangSide*1 placed in side4, signal pins with *rangSide*1 placed in side1 and signal pins with *rangSide*1 placed in side2. And the computation is translated as following form:

- $Length = |x - (-1)| + |y - (-1)|$,

  $0 \leqq x \leqq w_4$, $0 \leqq y \leqq (h_1 + h_{core}/2)$

- $Length = |y|$,

  $w_4 \leqq x \leqq (w_4 + w_{core} + w_2)$, $0 \leqq y \leqq h_1$

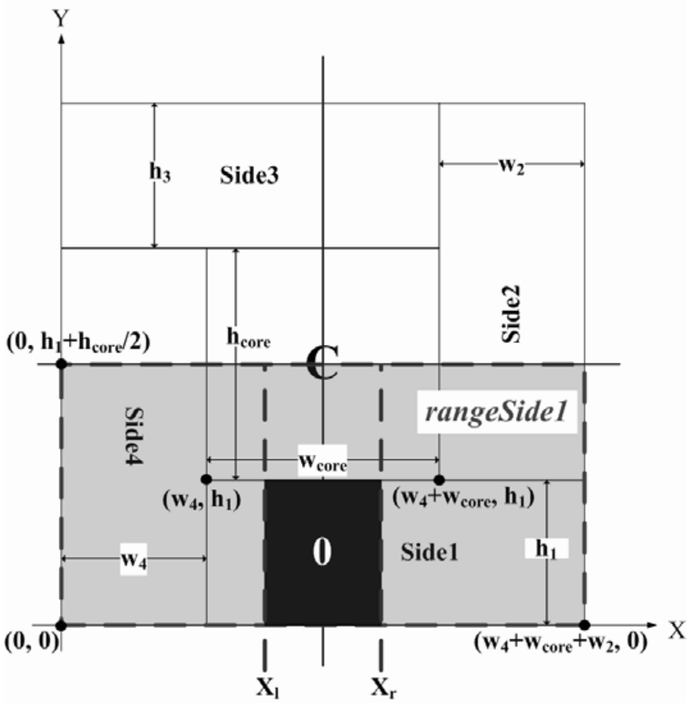


Figure 3.7: An estimation of the penalty term which is the distance between pins and their desired positions in $rangeSide1$.

- $Length = |x - (w_4 + w_{core} + w_2 + 1)| + |y - (-1)|$,

  $(x_4 + w_{core}) \leqq x \leqq (w_4 + w_{core} + w_2)$, $h_1 \leqq y \leqq (h_1 + h_{core}/2)$

And other rangeSides are also following the manner of estimating the wire length.

Another estimation term of cost function is the penalty term and the evaluation of penalty term is also formulated as following form. But just for $rangeSide1$, as shown in Figure 3.7.

- $Penalty = [|y| + |(w_4 - X_l)|]^2$,

  $0 \leqq x \leqq w_4$, $0 \leqq y \leqq (h_1 + h_{core}/2)$

- $Penalty = [|(x - X_l)|]^2$, $w_4 \leqq x \leqq X_l$, $0 \leqq y \leqq h_1$

- $Penalty = 0$, $X_l \leqq x \leqq X_r$, $0 \leqq y \leqq h_1$

- $Penalty = [|(x - X_r)|]^2$,

  $X_r \leqq x \leqq (w_4 + w_{core} + w_2)$, $0 \leqq y \leqq h_1$

- $Penalty = [|X_r - (w_4 + w_{core} + w_2)| + |(y - (h_1))|]^2$,

  $(w_4 + w_{core}) \leqq x \leqq (w_4 + w_{core} + w_2)$, $h_1 \leqq y \leqq (h_1 + h_{core}/2)$

Where $X_l$ denotes the left edge of the given desired position in the side1 and $X_r$ represents the right edge of the given desired position in the side1. $RangeSide2$, $rangeSide3$ and $rangeSide4$ are also following this manner of estimating the penalty term.

Figure 3.8 reveals the desired positions in other three $rangeSides$ and the purpose of this specification is hopefully floorplanning the pins to the center of each side approximately. Besides the desired position assigned weight centrally, as shown in Figure 3.9 (a). There are two weight behaviors including left-position and right-position in order to make the final floorplan have rotating ability. Figure 3.9 (b) and Figure 3.9 (c) show the rotating ability of the final floorplan in clockwise direction and in anticlockwise direction, respectively.

### 3.4.2 Annealing Schedule

Our problem is implemented by using range constraint based on simulated annealing algorithm [16] and our annealing schedule is shown as follows:

Simulated Annealing Schedule($S_0$, $T_0$, $\alpha$, $\beta$, $M$, $Maxtime$):

begin

  $T = T_0$;

  $S = S_0$;

  $Time = 0$;

    repeat

      repeat

        $NewS = range\_constraint(S)$;

        $H = (Cost(NewS) - Cost(S))$;

        If(($\triangle h < 0$) or ($random < e^{-\triangle/T}$)) then $S = NewS$;

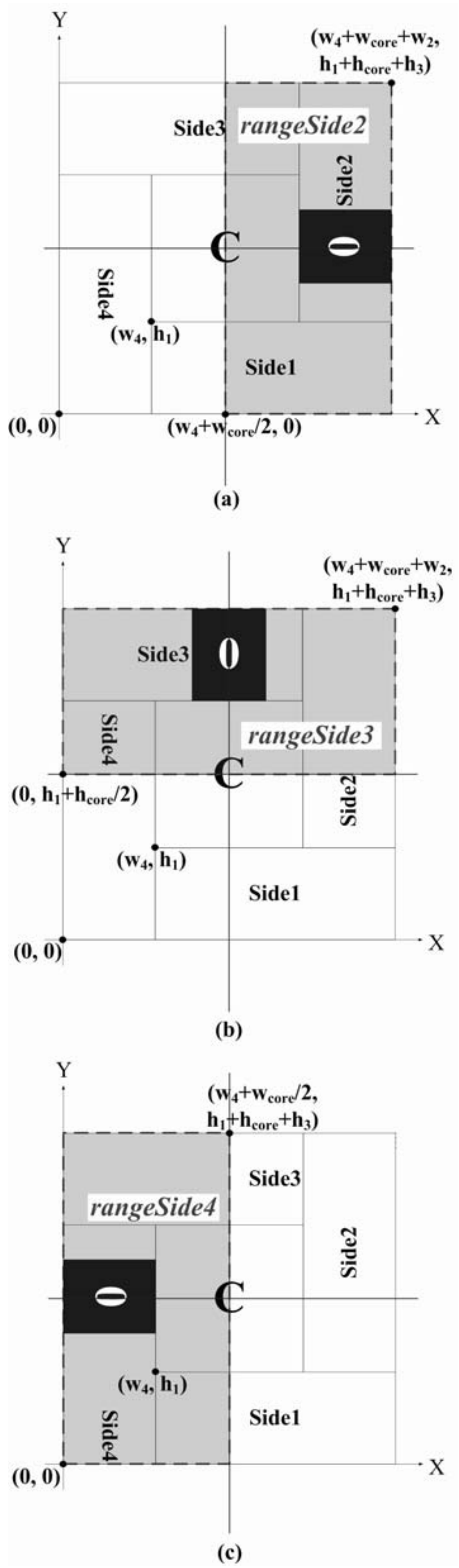


Figure 3.8: Reveals the desired positions in other three *rangeSides* and the purpose of this specification is hopefully floorplanning the pins to the center of each side approximately.

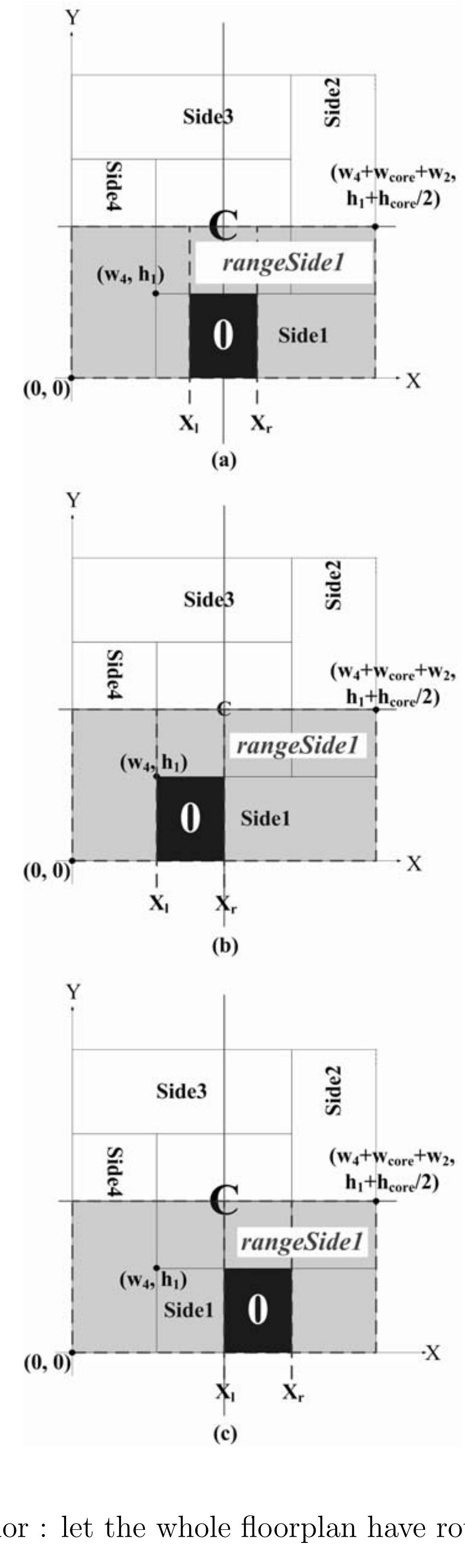


Figure 3.9: Weight behavior : let the whole floorplan have rotate ability. (a) Center. (b) Clockwise. (c) Anticlockwise.

```
            accept the solution;
            M = M - 1;
        until (M = 0)
        Time = Time + M;
        T = α * T;
        M = β * M;
    until (Time ≧ MaxTime);
  Output Best solution found;
End. (* of Simulated Annealing Schedule*)
```

Where $S_0$ is the initial solution, $T_0$ is the initial Temperature(defined as 100), $\alpha$ is the cooling rate(a typical value for $\alpha$ is 0.9), $\beta$ is a constant(defined as 2), $Maxtime$ is the total allowed time for the annealing process(defined as 500), $M$ represents the time until the next parameter update(defined as 5), $\triangle h$ is difference in costs, and $T$ is the temperature. *Range_constraint* function generates $NewS$ of any given solution $S$ and $Cost$ function evaluates the cost values of $NewS$ and $S$. In our annealing schedule, the initial solution is generated by several steps which are described as follows:

- In the first step, we randomly select a *rangeSide* and randomly select a group in that selected *rangeSide*.

- In the second step, we randomly select the start point of the selected group and place it, then we rearrange other groups which belong to the selected *rangeSide* in order.

- In final step, we rearrange other groups by following the original order which is specified by user until all *rangeSides* have been considered.

The annealing process is implemented iteratively to get a feasible solution with minimum cost and terminates when the total allowed time is less then the accumulative time.

# Chapter 4

# Experimental Results

We implemented our approach in the C++ Programming language and the platform is on intel(R) Pentium(R) M CPU 1.7GHz work station with 512 MB memory. We used four industrial cases as our benchmarks (case I, case II, case III, case IV). Table 4.1 shows the summary of pin configuration charts about these four cases.

We carried out two sets of experiments. For the first set, we compared SA-random floorplanner and our improved pin-block floorplanner with the method in [1] by considering wire length as cost function. The experiment results and improvement are shown in Table 4.2, Figure 4.1 and Figure 4.2. Here we propose another method called SA-random floorplanner to implement these four cases. The main difference between SA-random method and our method is the order of placing groups. In SA-random method, although the generation of initial solution is the same with our method, but the perturbation is still using the manner of generating the initial solution and it place groups by following the original order which is specified by user. However, our method places groups by following RCPBSP and has more flexibility than SA-random method. In Figure 4.1 and Figure 4.2, Case I and Case II have the same amount of groups and the improvement of our method is equally like that of SA-random method. But, there are obvious improvements of our method in Case III and Case IV while the amount of groups increases gradually. In Figure 4.2, we can see that our improved pin-block floorplanner has improved over [1] in wire

Table 4.1: The summary of these four industrial cases

| | Group NO. | Signal -pin | Power -pin NO. | Total pin NO. |
|---|---|---|---|---|
| CaseI | 6 | 254 | 80 | 334 |
| CaseII | 6 | 346 | 48 | 394 |
| CaseIII | 20 | 510 | 168 | 678 |
| CaseIV | 25 | 504 | 216 | 720 |

length and there is a incremental trend of the improvement in wire length while the size of case increases gradually.

In the second set of experiments, we consider wire length and the penalty term simultaneously in cost function. And the penalty term has three different desired positions, including Left, Center and Right. The purposes of Left, Center and Right are hopefully floorplanning the pin-blocks to the left, center and right of each side approximately, respectively. The results and the improvement are shown in Table 4.3, Table 4.4, Table 4.5, Figure 4.3 and Figure 4.4. Our improved pin-block floorplanner has improved over [1] in three different desired positions. In order to comparing with [1], pin-blocks are hopefully placed in the center of each side approximately and Center becomes the most important desired position that we care. For Center, there is the improvement at least more than a 32 percent over [1] and Table 4.5 clearly shows that wire length and penalty term are both improved in CaseII, CaseIII and CseIV. In addition, we also use the results of the method in [1] to be our initial solutions. Then, we get the same results by implementing our improved pin-block floorplanner. Because, our method is implemented by using simulated annealing algorithm to place pin-blocks. Our method will be implemented iteratively until the global solution is found.

Figure 4.5 is our result packing of CaseII , whose cost function considers wire

Table 4.2: Experimental results of the method in [1] and our improved pin-block floorplanner by considering wire length as cost function

| | | [1] | SA-random floorplanner | Our improved pin-block floorplanner | Improvement(%) | |
|---|---|---|---|---|---|---|
| Data | n | Wire length | | | SA | Our |
| CaseI | 6 | 1199 | 1149 | 1149 | +4.17 | +4.17 |
| CaseII | 6 | 1712 | 1640 | 1639 | +4.21 | +4.26 |
| CaseIII | 20 | 2406 | 2226 | 2225 | +7.48 | +7.52 |
| CaseIV | 25 | 2442 | 2240 | 2170 | +8.27 | +11.14 |

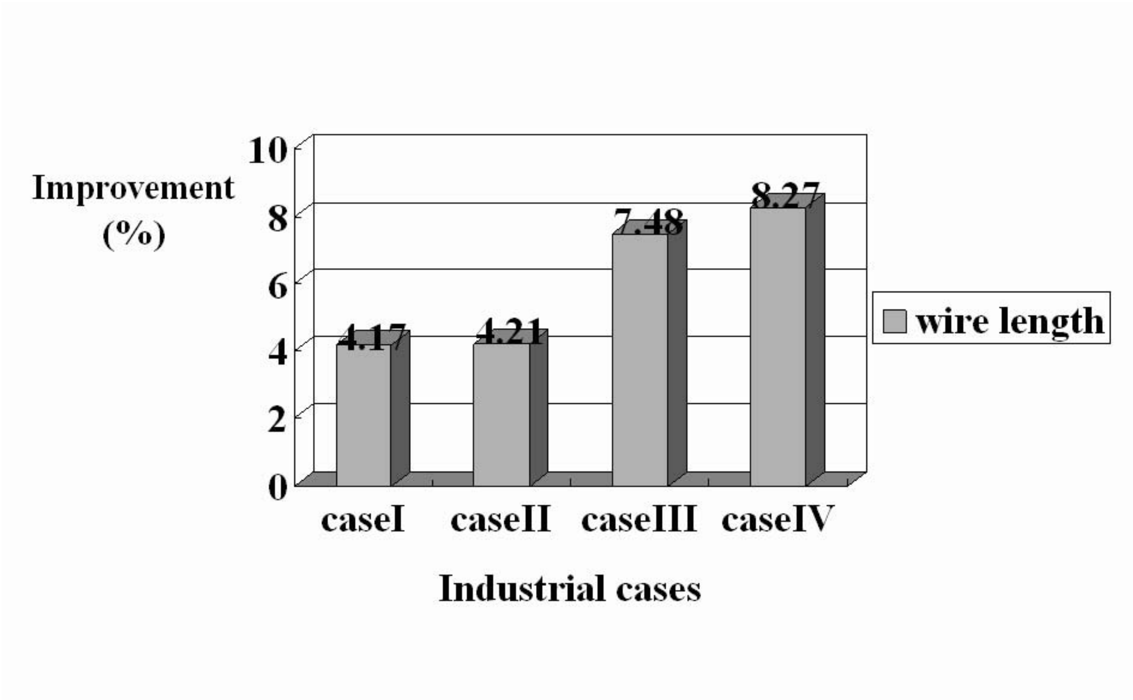


Figure 4.1: Shows the improvement of wire length in four industrial cases ([1] vs. SA-random floorplanner).

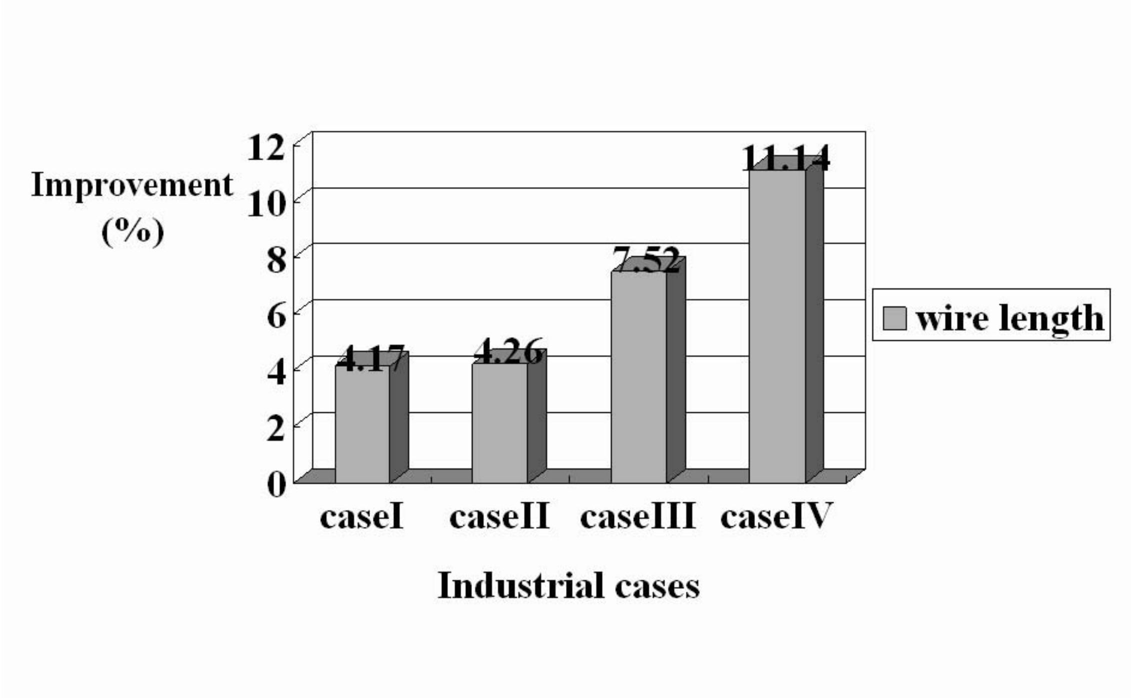


Figure 4.2: Shows the improvement of wire length in four industrial cases ([1] vs. our improved pin-block floorplanner).

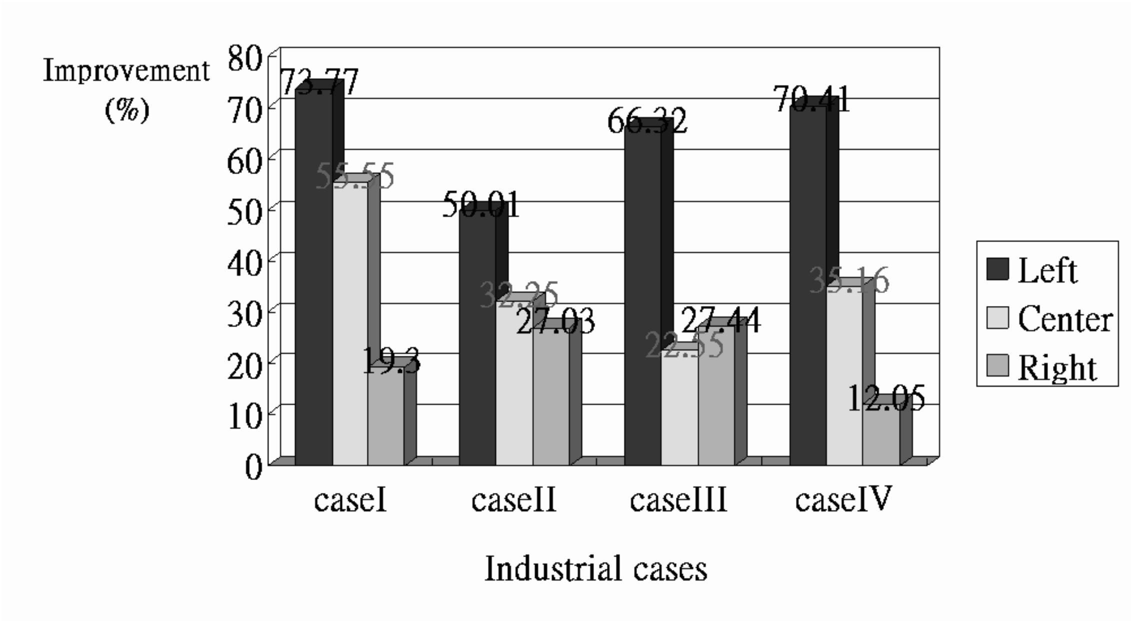


Figure 4.3: Illustrates the improvement of SA-random floorplanner by considering wire length and penalty term simultaneously in four industrial cases with $\rho$=5 ([1] vs. SA-random floorplanner).

Table 4.3: Shows experimental results of the method in [1] and SA-random floorplanner. Wire length and penalty term are considered simultaneously in cost function

| Data | n | | [1] | | | SA-random floorplanner | | |
|---|---|---|---|---|---|---|---|---|
| | | | Sum=wLength+ $\rho$ *penalty($\rho$ =5) | | | | | |
| | | | Left | Center | Right | Left | Center | Right |
| CaseI | 6 | Penalty | 12832 | 6200 | 3958 | 3156 | 2619 | 3156 |
| | | wL | 1199 | 1199 | 1199 | 1361 | 1216 | 1158 |
| | | Sum | 65359 | 32199 | 20989 | 17141 | 14311 | 16938 |
| CaseII | 6 | Penalty | 13668 | 8708 | 10838 | 6605 | 5802 | 7808 |
| | | wL | 1712 | 1712 | 1712 | 1994 | 1650 | 1750 |
| | | Sum | 70052 | 45252 | 55902 | 35019 | 30660 | 40790 |
| CaseIII | 20 | Penalty | 60665 | 27048 | 29744 | 19936 | 20853 | 21377 |
| | | wL | 2406 | 2406 | 2406 | 3294 | 2336 | 2765 |
| | | Sum | 305731 | 137646 | 151126 | 102974 | 106601 | 109650 |
| CaseIV | 25 | Penalty | 73239 | 31590 | 28750 | 21264 | 20348 | 25117 |
| | | wL | 2442 | 2442 | 2442 | 2761 | 2263 | 2998 |
| | | Sum | 368637 | 160392 | 146192 | 109081 | 104003 | 128583 |

Table 4.4: Shows experimental results of our improved pin-block floorplanner. Wire length and penalty term are considered simultaneously in cost function

| Data | n | | Our improved pin-block floorplanner Sum=W+ $\rho$ *P($\rho$ =5) | | |
|---|---|---|---|---|---|
| | | | Left | Center | Right |
| CaseI | 6 | Penalty | 3156 | 2619 | 3156 |
| | | wL | 1361 | 1216 | 1158 |
| | | Sum | 17141 | 14311 | 16938 |
| CaseII | 6 | Penalty | 6605 | 5802 | 7808 |
| | | wL | 1994 | 1650 | 1750 |
| | | Sum | 35019 | 30660 | 40790 |
| CaseIII | 20 | Penalty | 13786 | 15579 | 18331 |
| | | wL | 2890 | 2236 | 2723 |
| | | Sum | 71820 | 80131 | 94378 |
| CaseIV | 25 | Penalty | 18018 | 16976 | 18645 |
| | | wL | 2917 | 2201 | 2447 |
| | | Sum | 93007 | 87081 | 95672 |

Table 4.5: Shows the improvement of SA-random floorplanner and our improved pin-block floorplanner by considering wire length and penalty term simultaneously

| | | | Improvement(%) | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | | SA-random floorplanner | | | Our improved pin-block floorplanner | | |
| Data | n | | Left | Center | Right | Left | Center | Right |
| CaseI | 6 | Penalty | + | + | + | + | + | + |
| | | wL | - | - | + | - | - | + |
| | | Sum | +73.77 | +55.55 | +19.3 | +73.77 | +55.55 | +19.3 |
| CaseII | 6 | Penalty | + | + | + | + | + | + |
| | | wL | - | + | - | - | + | - |
| | | Sum | +50.01 | +32.25 | +27.03 | +50.01 | +32.25 | +27.03 |
| CaseIII | 20 | Penalty | + | + | + | + | + | + |
| | | wL | - | + | - | - | + | - |
| | | Sum | +66.32 | +22.55 | +27.44 | +76.51 | +41.78 | +37.55 |
| CaseIV | 25 | Penalty | + | + | + | + | + | + |
| | | wL | - | + | - | - | + | - |
| | | Sum | +70.41 | +35.16 | +12.05 | +74.77 | +45.71 | +34.56 |

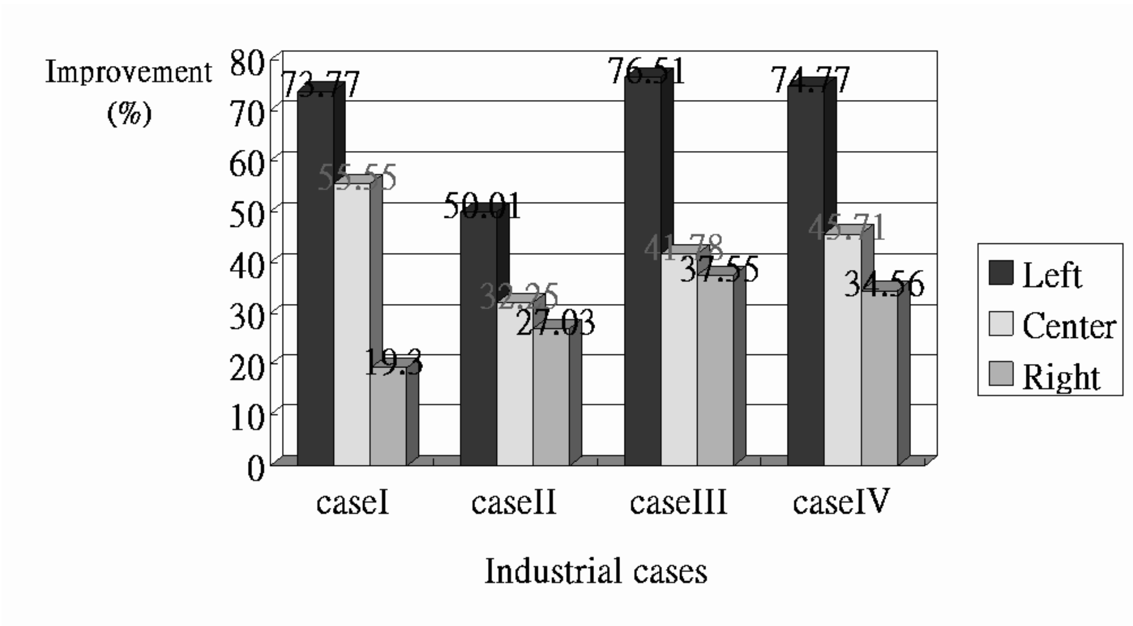


Figure 4.4: Illustrates the improvement of our improved pin-block floorplanner by considering wire length and penalty term simultaneously in four industrial cases with $\rho$=5 ([1] vs. our improved pin-block floorplanner).

length and penalty term in Center, where group 1 and 2 are constrained to be placed within $rangeSide1$ which is formed as a dotted line rectangle in Figure 4.5. Similarly, group 3 is within $rangeSide2$, group 4 is within $rangeSide3$, group 5 and 6 are within $rangeSide4$. By the way, the dotted line rectangles of other three $rangeSide$ are not shown in Figure 4.5. Comparing with the final floorplan of CaseII which is shown in Figure 2.14, our packing avoids the empty positions appearing in non-well positions which will cause extra cost on wire length. And our packing also owns flexibility on the order of placing groups.

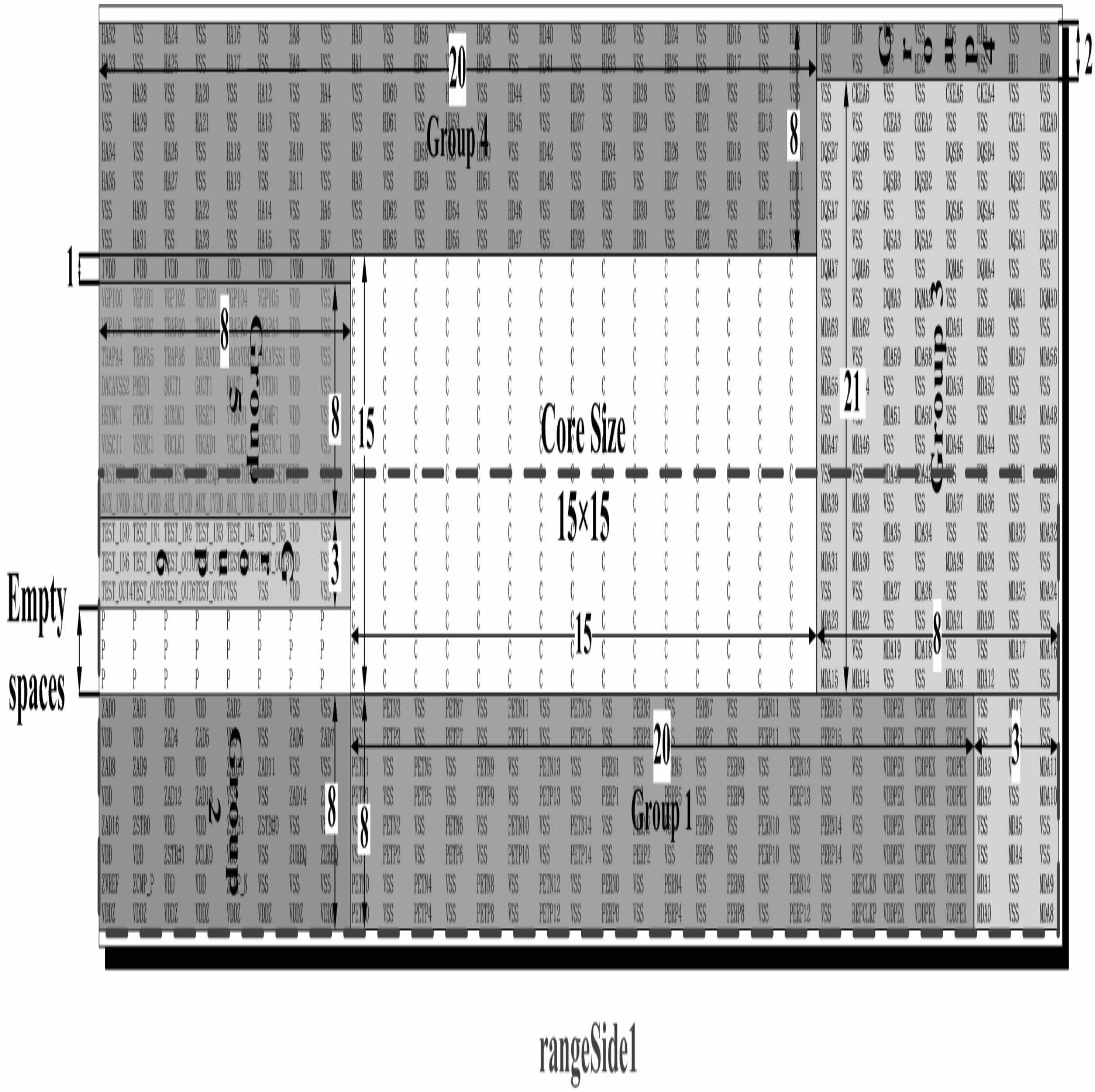


Figure 4.5: Our result packing of CaseII. The cost function considers wire length and penalty term in Center simultaneously.

# Chapter 5

# Conclusion and Future Work

We have proposed an improved pin-block floorplanner with range constraints in pin-out designation automation in flip-chip BGA package-board codesign. Our approach not only owns more flexible capability on placing groups, but also can avoid the non-well positions appearing in result packing. Furthermore, our approach can rotate the floorplan in final packing by exploiting the penalty term. It provides three different desired positions for user to select which floorplan is the most proper one in final packing. Experimental results reveal that our approach is better and more flexible than the method in [1].

About the future works, timing constraints is a critical issue of influencing the performance of our chip. The timing delay is accumulated outward form die to board and it could be considered in die-package-board codesign simultaneously. We should take the timing constraints into account to define our performance objectives. At the same time, we should extend our research inward in die-package codesign and try to place logic closer together so shorter routing resources can be used. In addition, the direction of placing pin blocks in four corners of result packing is pre-defined. Maybe, we can make the direction flexible and depend on which way can cause the lower cost.

# Bibliography


[1] R. J. Lee, M. F. Lai, and H. M. Chen, "Fast Flip-Chip Pin-Out Designation Respin by Pin-Block Design and Floorplanning for Package-Board Codesign," in *Proc. 12th Asia and South Pacific Design Automation Conf.*, Jan. 2007, pp.804-809.

[2] H. Murata, K. Fujiyoushi, and M. Kaneko, "VLSI/PCB placement with obstacles based on sequence-pair," in *Proc. Int. Symp. Physical Design*, 1997, pp.26-31.

[3] H. Murata, K. Fujiyoushi, S. Nakatake, and Y. Kajitani, "Rectangle-packing-based module placement," in *Proc. IEEE Int. Conf. Computer-Aided Design*, 1995, pp. 472-479.

[4] H. Murata, and E. S. Kuh, "Sequence-pair based placement method for hard/soft/pre-placed modules," in *Proc. Int. Symp. Physical Design*, 1998, pp.167-172.

[5] D. F. Wong, and C. L. Liu, "A new algorithm for floorplan design," in *Proc. 23rd ACM/IEEE Design Automation Conf.*, 1986, pp.101-107.

[6] F. Y. Young, and D. F. Wong, "Slicing floorplans with pre-placed modules," in *Proc. IEEE Int. Conf. Computer-Aided Design*, 1998, pp. 252-258.

[7] F. Y. Young, and D. F. Wong, "Slicing floorplans with boundary constraints," *IEEE Trans. Computer-Aided Design*, vol. 18, Sept. 1999, pp. 1385-1389.

[8] F. Y. Young, and D. F. Wong, "Slicing floorplans with range constraints," in *Proc. Int. Symp. Physical Design*, 1999, pp. 97-102.

[9] Y. C. Chang, Y. W. Chang, G. M. Wu, and S. W. Wu, "B*-trees: A new representation for non-slicing floorplans," in *Proc. 37th ACM/IEEE Design Automation Conf.*,

2000, pp. 458-463.

[10] P. N. Guo, C. K. Cheng, and T. Yoshimura, "An O-tree Representation of Non-Slicing Floorplan and Its Application," in *Proc. Design Automation Conf.*, 1999, pp.268-273.

[11] Y. H. Jiang, J. Lai, and T. C. Wang, "Module Placement with Pre-Placed Modules Using the B*-Tree Representation," *IEEE Int. Symp. Circuits and Systems*, vol. 5, May. 2001, pp. 347-350.

[12] J. M. Lin, H. E. Yi, and Y. W. Chang, "Module placement with boundary constraints using B*-trees," *Circuits, Devices and Systems, IEE Proceedings*, vol. 149, Aug. 2002, pp. 251-256.

[13] F. Y. Young, C. N. Chu, and M. L. Ho, "Placement Constraints in Floorplan Design," *IEEE Trans. Very Large Scale Integrated (VLSI) Systems*, vol. 12, July 2004, pp. 735-745.

[14] Robert J. Vanderbei, "LOQO User's Manual-Version 4.05," *Princeton University School of Engineering and Applied Science, Department of Operations Research and Financial Engineering, Princeton, New Jersey*, Oct. 2000. http://www.princeton.edu/ rvdb/.

[15] Frederick S. Hillier, and Gerald J. Lieberman, Introduction to Operation Research, *McGraw-Hill*, 2002.

[16] Sadiq M. Sait, and Habib Youssef, VLSI Physical Design Automation, *World Scientific*, 1999.

[17] S.-S. Chen, W.-D. Tseng, J.-T. Yan, and S.-J. Chen, "Printed Circuit Board Routing and Package Layout Codesign," In *Proceedings of IEEE Asia−Pacific Conference on Circuits and Systems*, 2002, pp. 155-158.

[18] T.-O. Chong, S.-H. Ong, T.-G. Yew, C.-Y. Chung, and R. Sankman, "Low Cost Flip Chip Package Design Concepts for High Density I/O," In *Proceedings of IEEE Electronic Components and Technology Conference*, 2001, pp. 1140-1143.

[19] J. Mcgrath, "Chip/Package Co-Design: The bridge between chips ans systems," In *Advanced Packaging*, June 2001.

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

# 作者簡歷

翁嘉倫, 民國六十九年十一月出生於新竹市。民國九十二年七月畢業於中原大學工業工程學系, 並於民國九十四年二月進入國立交通大學 IC 設計產業研發碩士班就讀, 從事 VLSI 實體設計自動化的相關研究。於民國九十六年九月取得碩士資格, 碩士論文題目爲『覆晶封裝在封裝與機板共同設計階段一個線長驅策及被範圍條件限制的訊號區塊擺放方法』。