# 國 立 交 通 大 學

電子工程學系 電子研究所

博 士 論 文

# 在實體設計階段改善設計品質/診斷能力之方法

# The Methods on Improving Design Quality/Diagnosibility in Physical Design Stage

研 究 生：吳孟臻

指導教授：周景揚 教授

中 華 民 國 一００ 年 七 月

# 在實體設計階段改善設計品質/診斷能力之方法

# The Methods on Improving Design Quality/Diagnosibility in Physical Design Stage

研 究 生：吳孟臻　　Student：Meng-Chen Wu

指導教授：周景揚　　Advisor：Jing-Yang Jou

國 立 交 通 大 學

電子工程學系 電子研究所

博 士 論 文

A Dissertation
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
Doctor of Philosophy
in
Electronics Engineering

July 2011
Hsinchu, Taiwan, Republic of China

中華民國一００年七月

# 在實體設計階段改進設計品質/診斷能力的方法

學生：吳孟臻　　　　　　　　　　指導教授：周景揚 博士

國立交通大學 電子工程學系 電子工程研究所

## 摘要

在今日晶片設計流程的每個設計階段中，實體設計階段（physical design）將電路設計從閘層級電路（gate-level netlist）實做成為幾何佈局（geometric layout）的關鍵性階段。平面規劃（floorplanning）是當中的第一個步驟。根據不同的設計限制，平面規劃安排每個區塊的位置並且對於評估一些關鍵性的設計度量非常地有幫助，像是預測晶片面積、功率消耗、關鍵路徑的邊界以及總導線長度等等。

導線的連線長度對於時脈最佳化是一個很重要的指標。為了確保晶片效能，我們必須限制住關鍵導線的長度以滿足時脈要求。針對非關鍵連線的最佳化處理，非曼哈頓結構（non-Manhattan structure），像是 X 或 Y 結構，已被提出討論且可大量的降低實體資源的使用。然而，擺置與繞線步驟是在平面規劃之後執行，設計的佈局又由平面規劃所決定，因此，我們需要提早考量導線最佳化的問題，尤其是在實體設計初期的平面規劃階段。此外，在系統單晶片

（system-on-a-chip，SoC）設計中，使用電壓島（voltage island）技術以降低功率消耗最近開始受到歡迎。目前這項技術僅被考慮在系統層級架構或在擺置流程完成之後的階段。在階層式設計與可重複使用之半導體智慧產權（intellectual propoerty，IP）廣泛的使用之下，為了得到較佳的初步結果，必須提早在平面規劃/擺置階段考慮電壓島的產生以降低功率消耗問題。

在本論文的第一部份，在平面規劃階段我們提出兩個考量不同設計限制的演算法。第一個演算法在平面規劃時，針對 X 結構繞線，加入了等腰直角三角形以及等腰梯形的新區塊形狀。本方法可更進一步的處理任意區塊，只要該區塊可被分割成矩形以及等腰直角三角形。第二個演算法在平面規劃階段考量電壓島的產生以及滿足效能限制，此限制是限制住關鍵連線的範圍。實驗結果說明本方法非常的有效率，在滿足時脈要求下，可同時的考慮功率繞線代價以及功率消耗的取捨。

在另一方面，晶片設計的第一次矽晶片的成功率越來越低，而利用失敗晶片以改善下次矽晶片的良率是很重要的一件事情。測試性設計（design-for-test，DFT）以及除錯化設計（design-for-debug，DFD）這兩種電路通常都在邏輯合成階段被加入設計電路之中。利用這些電路，我們可在製作矽晶片之後對設計的功能性進行驗證及確認正確性。由於資源的限制，DFT 以及 DFD 電路僅能提供內部電路訊號的一小部分，我們需要其他方式來得到更多的訊息以便進行錯誤分析。聚焦離子束（Focus-Ion-Beam, FIB）是可在矽晶片製造完成之後，獲取內部導線

信號技術中的一種技術。而隨著技術的逐漸進步、製程的縮小，FIB 技術的解析度卻沒有隨著製程的快速縮小。因此，對於先進技術而言，經由 FIB 所能觀察到的導線以及能被修改的電路，在比例上明顯地降低許多，並且限制了很多在除錯過程中可藉由 FIB 技術所檢查的訊號。

本論文的第二部份介紹一除錯化設計的架構，藉由在後繞現階段（post-routing step）調整實體佈局以增加可被 FIB 觀察及可被 FIB 修改之訊號比例。根據設計規則及時脈要求，本實體佈局調整方法使用預先定義好的一些簡單操作方法，因此，本設計架構不需要複雜的繞線器（router）做為核心並且可被整合入目前任一的 APR 軟體。根據使用 90 奈米函示庫所獲得的實驗數據，本除錯化設計架構有效地增進可被 FIB 觀察到或是修改的訊號比例，即便在不同的 FIB 參數設定下，仍可維持相同的使用面積以及晶片效能。

# THE METHODS ON IMPROVING DESIGN QUALITY/DIAGNOSIBILITY IN PHYSICAL DESIGN STAGE

**Student: Meng-Chen Wu Advisor: Dr. Jing-Yang Jou**

**Department of Electronics Engineering**
**Istitute of Engineering**
**National Chiao Tung University**

## Abstract

*In the IC-design flow nowadays, the physical design is the critical stage which implements the design from gate-level netlist into geometric layout. Floorplanning is the first step in physical design. According to different constraints, floorplanning step arranges the location of each block and is useful to estimate some critical design metrics, such as area, power consumption, boundaries of critical paths, and total wirelength.*

*The wirelength of nets is an important issue for timing optimization. To maintain the chip performance, we have to bound the wirelength of critical nets for meeting timing issues. To minimize the total wirelength, non-Manhattan structures, such as the X and Y architectures, propose different flavors in reducing the use of physical resources. However, the placement and routing steps are after floorplanning step and the layout of design is decided by floorplanning step. Thus, we need to consider wirelength optimization at floorplanning step for reducing wirelength. Besides,* *using voltage island methodology to reduce power consumption for System-on-a-Chip (SoC) designs has become popular recently. Currently this approach has been considered either in system-level architecture or post-placement step. Since hierarchical design and reusable intellectual property (IP) are widely used, it is necessary to optimize floorplanning/placement methodology considering voltage islands generation to solve power consumption problems.*

*In the first part of this dissertation, we present two algorithms considering design constraints at floorplanning step. The first algorithm deals with isosceles right triangular and trapezoidal blocks for X-architecture routing strategy. This approach can be further applied to pack with any block, which can be divided into rectangles and isosceles right triangles. The second one handles floorplanning considering voltage islands generation and performance constraints, which restrict the boundaries of critical nets. This method is flexible and can be extended to hierarchical design. The experimental results show that our method is not only effective in meeting performance constraints but also simultaneously takes the consideration about the balance between power routing cost and total power dissipation.*

*On the other hand, the first silicon of design is often failed and using failed chips to improve the yield for next silicon is very important. The design-for-test (DFT) and design-for-debug (DFD) circuits are often generated and integrated during logic synthesis stage. With these circuits, we can verify the functionality of design after manufacturing. Because of limitation of resource, the DFT and DFD circuits only deliver a small portion of internal signals and we need other approaches to get more information for failure analysis. The focused ion beam (FIB) is one of techniques to obtain the signals after manufacturing chips. While the technology* *node continually and aggressively scales, the resolution of FIB techniques does not scale as fast. Thus, the percentage of signals which can be observed through FIB probing is significantly decreased for advanced process technologies, which limits the candidates that can be physically examined through the FIB techniques during the debugging process.*

*The last part of the dissertation introduces a methodology to modify the layout and increase the signals for FIB observation. In the post-routing step, the layout modification is made through pre-defined simple operations subject to design rules and timing constraints. Hence, the proposed methodology does not require a complicated router as its core and can be applied to the layout generated with any commercial APR tool. The experimental result based on an 90nm technology has demonstrated that the proposed methodology can effectively increase the number of signals for FIB observation while the overall area and circuit performance remain the same.*

# Acknowledgements

I would like to express my gratitude to all those who gave me the possibility to complete the dissertation. My sincerest appreciation goes to my advisor, Dr. Jing-Yang Jou, for his guidance in completing this dissertation. A special acknowledgement is forward to professor Hung-Ming Chen and professor Mango Chia-Tso Chao, for their kindly help to complete this dissertation.

Besides, special thanks for Ming-Ching Lu, Meng-Jai Tasi, Tsung-Wei Chang, Kuo-An Chen, and all EDA members for these wonderful years.

Last but not least, I wish to thank my wife, Li-Chi Chuang, my parents, my family and my friends for their invaluable support, advise and encouragement throughout my study years.

Meng-Chen Wu

*National Chiao Tung University*

*July 2011*

# Table of Contents

# List of Tables

# List of Figures

# Chapter 1

# Introduction

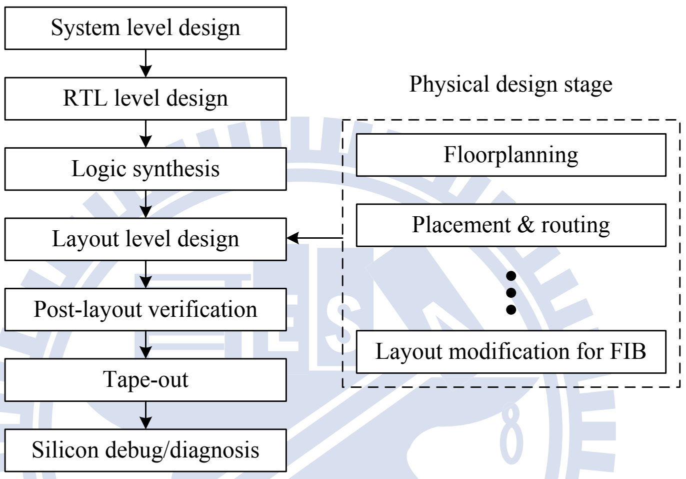


Figure 1.1: The IC design flow and the related fileds of our works.

In the IC-design flow nowadays as illustrated in Figure 1.1, electronic design automation (EDA) tools are involved in all design stages and the development of tools in each stage is necessary for designers to overcome the dramatically increased circuit complexity. The physical design stage is the critical stage which implements the design from gate-level netlist into geometric layout. This stage consists of several steps and each step targets on different issues for optimization. The floorplanning is the first step in physical design. Based on the structure of design, this step decides the initial layout and a bad floorplan will give the large unused deadspace of chips and routing congestion problems. According to different constraints, the floorplanning step arranges the location of each block and is useful to estimate some critical design metrics, such as area, power consumption, boundaries of critical paths, and total wirelength.

After getting the floorplanning of design, the step of place and route determines wirelength of the design. Generally speaking, the nets can be divied into critical and non-critical nets, depending on whether they are on critical paths or not. To maintain the chip performance, the wirelength of critical nets have to be bounded for meeting timing issues. Further, to minimize the total wirelength, non-Manhattan structures, such as the X and Y architectures, propose different flavors in reducing the use of physical resources. Taking the X architecture as an example, it has routing layers for 45- and 135-degree angles at upper layers [45] and the wirelength reduction is significant with proper routing system [10]. However, the layout of the design is decided by floorplanning step, we have to consider the wirelength optimization at floorplanning step if possible.

With the process of today's technologies, more and more devices can be integrated into a signal chip. To cope with the increasing design complexity, hierarchical design and reusable intellectual property (IP) modules are widely used within the System-on-a-Chip (SoC) design [12][50]. Meanwhile, increased circuit density and performance compel the need to reduce power consumption that increases significantly as designers strive to utilize the advancing silicon capabilities [20][29]. Using voltage island methodology to reduce power consumption for SoC designs has become popular recently. Currently this approach has been considered either in system-level architecture or post-placement step. Since the power planning is significantly impacted with the layout of designs, it is necessary to optimize floorplanning/placement methodology considering voltage islands generation to solve power consumption problems.

In the first half of this dissertation, we present two algorithms considering design constraints at floorplanning step. The first algorithm introduces non-rectangular blocks, such as isosceles right triangular and trapezoidal blocks, into normal floorplanner. This algorithm not only reduces total wirelength with X-architecture routing strategy but also has the same performance when performing rectangular and non-rectangular block packing. The second algorithm targets on reducing power consumption with voltage island techniques and meeting performance constraints, which restrict the boundaries of critical nets. The proposed algorithm is effective in meeting performance constraints and can simultaneously consider the tradeoff between power routing cost and total power dissipation. With proposed algorithms, we can consider several constraints in the early step and have the better initial solutions for the following steps of physical design.

On the other hand, today's modern designs are often failed with their first silicon because of the high complexity, high variation, and incomplete characterization of advanced process technologies. To extract the internal signals, the design-for-test (DFT) and design-for-debug (DFD) circuits are often generated during logic synthesis stage. With these circuits, such as scan chains [5] or trace buffers [2][3][4][57], we can verify the functionality of design after manufacturing. Although pre-silicon verifications are performed before tape-out, the verification tools are bounded by performance and capacity and cannot guarantee there is no escaped errors [49]. It is emergency to debug/diagnose these failed chips and improve the quality of next design or next silicon. Because of the limitation of resource, the DFT and DFD circuits only deliver a small portion of internal signals and we need other approaches to get more information for failure analysis with silicon chips [19].

Besides DFT and DFD circuits, there are several techniques to obtain the signals from silicon, such as electron beam (E-beam) [41], laser voltage probe (LVP) [58]

Table 1.1: FIB observable rates for 0.18$\mu$m and 90nm technologies.

| circuit | technology | | difference |
|---|---|---|---|
| | 0.18um(a) | 90nm(b) | (a)-(b) |
| s38417 | 72.66 | 36.57 | 36.09 |
| s38584 | 60.72 | 28.62 | 32.11 |
| s35932 | 85.06 | 50.84 | 34.21 |
| b17 | 41.95 | 17.86 | 24.09 |
| b20 | 50.87 | 25.62 | 25.23 |
| b21 | 46.57 | 23.47 | 23.10 |
| b22 | 46.91 | 23.56 | 23.36 |
| avg. | 57.82 | 29.50 | 28.32 |

and *focused ion beam* (FIB) techniques [1][40][43]. Among these techniques, the FIB requires no area overhead within the design and has the shorter process time compared to others. The utilization of observing signals with FIB usually targets on silicons and is not taken into consideration at physical design stage. As shown in Table 1.1, the average rate of observing signals through FIB techniques is under 30% with 90nm technology, and it will be worse with 65nm or 40nm technology in the future. However, the automatic place and route (APR) tools only focus on some design constraints, such as routability or timing issues, and we need an approach to take the advantage of FIB techniques.

In the last part of the dissertation, we propose a methodology with layout modification for FIB observation. This methodology uses some pre-defined operations to change the routing layers of signals and do not need any complicated routing system. It can be used in the post-routing step of physical design stage and applied with the layout generated by any APR tool. With the proposed methodology, we greatly increase the number of observable signals with FIB techniques while the overall area and circuit performance are the same to the original design.

# Chapter 2

# Preliminary

## 2.1 Review of B*-trees

The B*-tree is an ordered binary tree for modeling a non-slicing floorplan. Given an *admissible placement* [16] (in which no module can move left or down), we can construct a unique B*-tree in linear time. Further, given a B*-tree, we can also obtain a placement by packing the blocks in linear time with a contour structure [16]. The construction of a B*-tree has three properties:

**Root node** presents the block at the bottom-left corner, whose $x$- and $y$-coordinates is associated $(x_{root}, y_{root}) = (0, 0)$.

**Left child of a node** is placed on the right-hand side and adjacent to its parent block in the placement; i.e., $x_j = x_i + w_i$.

**Right child of a node** is placed above its parent block and has the same $x$-coordinate; i.e., $x_j = x_i$. With the contour structure, we can compute the $y$-coordinate of a block in constant time.

Figure 2.1 illustrates an admissible placement and its corresponding B*-tree. To construct this placement from the B*-tree, we first pick $n_0$, the root of the tree, and place $b_0$ on the bottom-left corner. Then we reach the left child of $n_0$, $n_1$. Block $b_1$ is then placed on the right of $b_0$. The process repeats recursively in the depth-first search (DFS) order. Therefore, since $n_1$ does not have a left child, we traverse $n_3$,

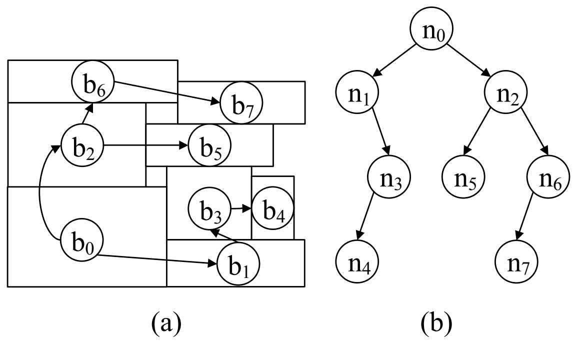


Figure 2.1: (a) An admissible placement. (b) The B*-tree representing the placement.

the right child of $n_1$. The process continues until all nodes are traversed, and finally we will have the corresponding placement.

## 2.2 Non-Manhattan Routing Architectures

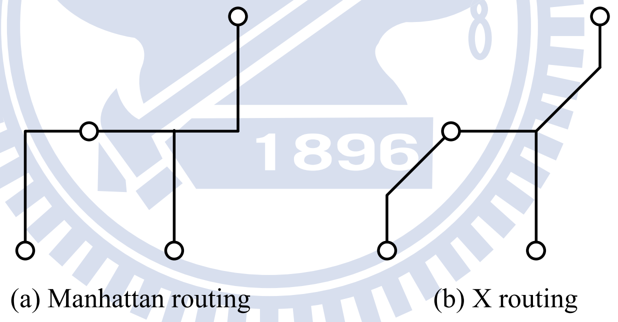


Figure 2.2: Examples of two routing architectures.

With today's advanced integrated circuits (ICs) manufacturing technology in deep submicron (DSM) environment, there are billions of transistors on a single chip. In interconnect-dominated IC design regime, due to the demand of high performance and low power consumption, we need more merits in basic wiring architectures to pave the road. Recently the X architecture [45] and the Y architecture [8], featuring non-Manhattan structures [23], propose more powerful approaches in reducing the use of physical resources, such as total wirelength and number of vias. Especially by the X architecture shown in Figrue 2.2, the wirelength reduction could reach 20% and the via reduction could reach 30% [45]. However, in order to take the advantage of the X or Y architectures, it is essential to develop new tools for these architectures, especially for early stages in physical design.

From triangular blocks packed in the Y architecture [27], they place right triangular blocks in hexagonal cells on a rectangular chip to fit the routing track of Y-architecture. Although their area usage is 94% in average, the runtime of their approach is much longer (3 and 5 hours for 33 and 49 blocks) than normal floorplanner and they do not reveal the wirelength optimization between their approach and Y architecture.

Thus, we believe that it is necessary to explore the situation in which rectangular blocks are packed with right isosceles triangular blocks. The right isosceles right triangular blocks have 45-degree angle and are the same with X architecture. Moreover, by using these special/flexible blocks, we can obtain more choices for pin assignment and more shapes can be used in floorplans. B*-trees [7] have reported good packing results in the Manhattan architecture, so we attempt to use this data structure to achieve good packing with the X architecture.

## 2.3 Voltage Island Techniques

The dynamic and static power dissipation in CMOS digital circuits both have direct relationship with supply voltage $Vdd$: dynamic power is proportional to $Vdd^2$ and static power is proportional to $Vdd$. Applying lower $Vdd$ under the performance requirements is obviously one of the effective ways to reduce power consumption. One of the techniques to reduce power consumption is voltage island methodology, which is proposed by IBM [25]. A voltage island is a group of on-chip cores powered

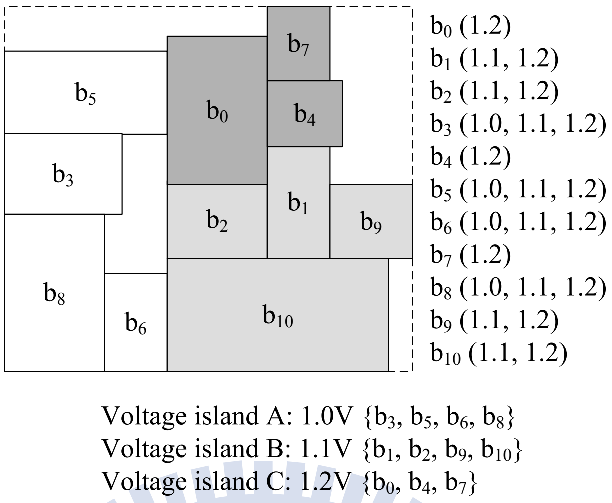


Figure 2.3: An example of a design with three voltage islands.

by the same voltage source, independently from the chip-level voltage supply. This concept (in use of voltage islands) permits operating different portions of the design at different supply voltage levels. As illustrated in Figure 2.3, the design has 10 blocks which have their own operating voltages. It has three voltage islands and each block is set to its lowest operating voltage to minimize the power consumption.

## 2.4 Floorplanning with Performance Constraints

Traditional floorplanners/placers minimize total wirelength but they can not guarantee critical nets to meet bounded delay. This problem becomes more important because timing convergence is a big issue in DSM design. In order to meet critical delay constraint, there are methods proposed in [44, 55] during floorplanning. Since actual interconnect delay after appropriate buffer insertions will be close to linear in terms of distance, they assume linear function in terms of distance to estimate delay is used. In their assumption, there are a source at $(x_s,y_s)$ and a sink at $(x_t,y_t)$, and the delay of the net $D_{s,t} = \delta Dist_{s,t} = \delta(\mid x_t - x_s \mid + \mid y_t - y_s \mid)$, where $\delta$ is a constant to scale the distance to timing, and $Dist_{s,t}$ is the maximum distance between source and sink, equal to the half perimeter of the bounding box of the two points. Thus, the delay could be represented with distance as: $Dist_{s,t} = (\mid x_t - x_s \mid + \mid y_t - y_s \mid) = D_{s,t}/\delta \leq D_{max}/\delta$, where $D_{max}$ is the given maximum delay bound.

In [44], they use a method that adjusts the width and height for the bonding box of performance-constrained blocks dynamically into the rectilinear super blocks. The width and height for the bounding box of these constrained blocks are denoted with $W_{perf}$ and $H_{perf}$. For their simulated-annealing based method, they restrict constrained blocks in the bounding box, where the $W_{perf} = H_{perf} \leq D_{max}/\delta$ at the beginning. Simulated annealing is characterized as chaotic process where a square range-box is appropriate to use for approximate guidance. And at lower temperature, a specific range box is almost fixed and cannot be changed easily to exactly capture delay bound.

In [55], it uses the linear delay model to do sub-placement (to place a set of feasible sub-placements for the performance-constrained blocks) by restricting the longest distance of performance-constrained blocks. First, they handle the set of performance-constrained blocks into several kinds of rectilinear blocks. For these rectilinear blocks, they have to satisfy the performance constraint: $W_{perf} + H_{perf} \leq D_{max}/\delta$. Besides, among these rectilinear blocks, they choose the one with minimum deadspace and fix the shape of these blocks (thus fix the delay) for further processing with other blocks. By using the pre-clustered shape-fixed appropriate rectilinear block, they guarantee that the performance constraints will be satisfied throughout the remaining processing.

However, these approaches focus on single operating voltage and are not suitable for today's multiple voltages design. In this dissertation, we improve these techniques and propose the method to meet performance constraints in multiple voltages design.

## 2.5 Background of Focus Ion Beam

Because of the high complexity, high variation, and incomplete characterization of advanced process technologies, the first silicon of today's design is often failed. Besides, the current design verification flow are bounded by performance and capacity and cannot guarantee there is no escaped errors [49]. Thus, *Failure analysis*, or *post-silicon debug*, takes the critical roles to identify the root of errors with the information from failed chips and further fix them in next design or next silicon.

Post-silicon debug has few ways to access the internal signals, such as scan chains [5] or trace buffers [2][3][4][57]. In order to have more internal signals from silicon chips, the physical probing techniques have been used over years. One of these techniques is Focus Ion Beam (FIB). The operations of FIB system are similar with scanning electron microscope (SEM) or a transmission electron microscope (TEM). The FIB system uses a focused beam of ions (Ga+ in most today's FIB system) instead of electrons in SEM and TEM. Because of using large, heavy, and positive ion beams, there are some features about FIB system. The FIB can image the sample surface with high resolution during operating with a low beam current and, further, it can mill the sample surface with a high beam current [60].

One of the most important features of FIB is quickly observing signals on silicon chips without re-fabrication. Figure 2.4 demonstrates the concept to observe signals with FIB techniques. First, the FIB system is applying with a high beam current (Ga+) to remove the inter-layer dielectric (ILD) and form a hole until reach-

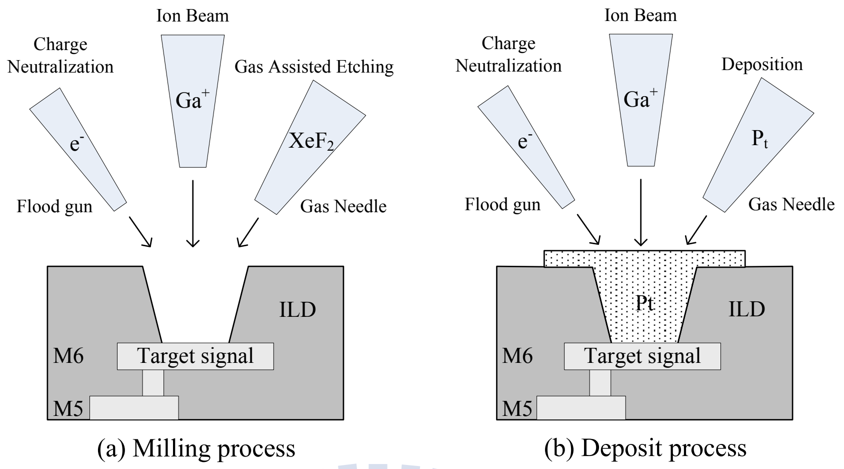


Figure 2.4: An example of FIB probing using (a) FIB surface mill and (b) FIB deposition.

ing the target signal. The optional gas ($XeF_2$ in the case) is used for preventing the re-deposition of sputtered surface material. After successful milling process, the FIB deposits conduct (Pt in the case) onto the dug hole and forms a probe pad [60] to access the target signal.

With the two features, milling and deposition, the FIB is very suitable with circuit editing, such as cutting existing metal and reconnecting it to a desired location (other metal lines or pre-placed space cells). The process of circuit-editing can be quickly applied to modify silicon circuits without another tape-out, and it need no area overhead in design when performing milling or deposition. Thus, FIB techniques are useful for physical probing or circuit editing and popular for observing signals at post-silicon debug.

## 2.6 Organization

The remainder of this dissertation is organized as follows. In Chapter 3, we present a packing method which takes the advantage of X architecture and uses non-rectangular blocks to reduce total wirelength. The proposed method is very efficient and effective in area/wirelength optimization, and its performance is compatible with normal floorplanners. In Chapter 4, we propose the another method to simultaneously generate voltage islands and meet performance constraints, which restrict the boundaries of critical nets. This method not only get the good balance in the trade-off between power consumption and power routing cost, but also achieve very good area usage while meeting performance constraints, even if these blocks are at different voltage islands. In Chapter 5, we develop a layout-adjustment framework to increase the observable signals with FIB probing techniques. We propose several simple operations to adjust the metal layers of routed signals. Hence, it could be quickly integrated into today's design flow and applied to the layout generated with any APR tool. Finally, we give conclusion and discuss the future works in Chapter 6.

# Chapter 3

# Floorplanning with Non-rectangular Blocks

In this chapter, we propose the packing scheme for floorplanning with isosceles right triangular blocks for X-architecture routing structures. First, we propose the motivation and our problem formulation. Then, we give the solution for packing with rectangular and isosceles right triangular blocks. Further, to demonstrate that our method could handle any combination with these two block shapes, the packing scheme with trapezoidal blocks is presented. Finally, the experimental results show that our method has almost the same performance with normal floorplanner. Besides, we also show the wirelength estimation for the comparison between our method and one normal floorplanner.

## 3.1 Motivation

The non-Manhattan routing structures, such as X architecture and Y architectures, propose more choices in routing directions besides horizontal and vertical routing tracks and there are several researches working with these architectures in the routing step [8][9][45]. In floorplanning/placement steps, Li *at al* [27] present the packing algorithm for non-rectangular blocks with Y architecture. In order to take the advantage of Y architectures, they use Hexagon/Triangle blocks for placement. However, these layouts are very different with today's design and hard to realize.

Although they have 94% area usage in average, which is closed to 95% - 96% area usage with Manhattan placement, their run time is quite long (few hours). Besides, their experiments only target on area with two benchmarks and have no discussion about wirelength with Y architectures.

In this chapter, we develop the first packing algorithm with non-rectangular blocks for X architectures. Taking the advantage of special angles, such as 45- and 135-degree, we add the isosceles right triangular blocks and trapezoidal blocks into normal floorplanner. With these special blocks, we have more choices for block shapes in floorplanning step and more locations for pin assignments for wirelength optimization. This approach can be quickly implemented into the floorplanning step and much suitable for current designs. Even with non-rectangular blocks, the average area usage is above 95%, which is comparable to normal floorplanner. We also report the wirelength reduction with Manhattan-half-perimeter wirelength (MHPWL) and X-half-perimeter wirelength (XHPWL) [10] to demonstrate the better results of proposed algorithm.

## 3.2 Problem Formulation

In this section, we define the problem about packing with rectangular blocks, isosceles right triangular blocks, and trapezoidal blocks.

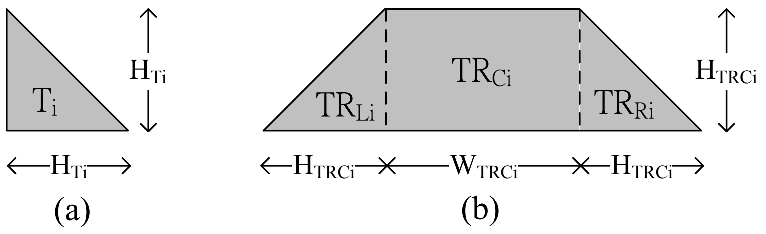


Figure 3.1: (a) An isosceles right triangular blocks. (b) A trapezoidal blocks and its sub-blocks.

**INPUT**

- The set of rectangular blocks, $B = \{b_1, b_2, ..., b_n\}$, and their corresponding widths and heights are denoted as $W_i$ and $H_i$, $1 \leq i \leq n$.

- The set of isosceles right triangular blocks, $T = \{t_1, ..., t_m\}$. As illustrated in Figure 3.1(a), these blocks have the same widths as heights, and we only need the heights to present these blocks, denoted by $H_{ti}$, $1 \leq i \leq m$. In the following discussion, we present these blocks as *triangular blocks* for shortly.

- The set of trapezoidal blocks. To handle these blocks, we divide them into 3 sub-blocks, the left-triangular block, the center-rectangular block, and the right-triangular block, as shown in Figure 3.1(b). Thus, trapezoidal blocks are denoted with $TR = \{TR_{L1}, TR_{C1}, TR_{R1}, ..., TR_{Lp}, TR_{Cp}, TR_{Rp}\}$. The width and height for $TR_{Ci}$ are $W_{TR_{Ci}}$ and $H_{TR_{Ci}}$, $1 \leq i \leq p$. Because the two triangular blocks have the same height as the rectangular block, we can use the height to present both the height and width of their triangular blocks, which is $H_{TR_{Li}} = H_{TR_{Ri}} = H_{TR_{Ci}}$, $1 \leq i \leq p$.

**OUTPUT**

- The floorplanning with mixed non-rectangular blocks.

**OBJECTIVE**

- Optimize a pre-defined cost metric, such as maximizing area usage or minimizing total wirelength.

## 3.3 Floorplanning with Triangular Blocks

In this section, we first explain the feasibility condition for the floorplanning with triangular blocks. Then, we present the calculation to explain how to pack with these blocks.

### 3.3.1 Feasibility Condition for Mixed Rectangular and Triangular Blocks

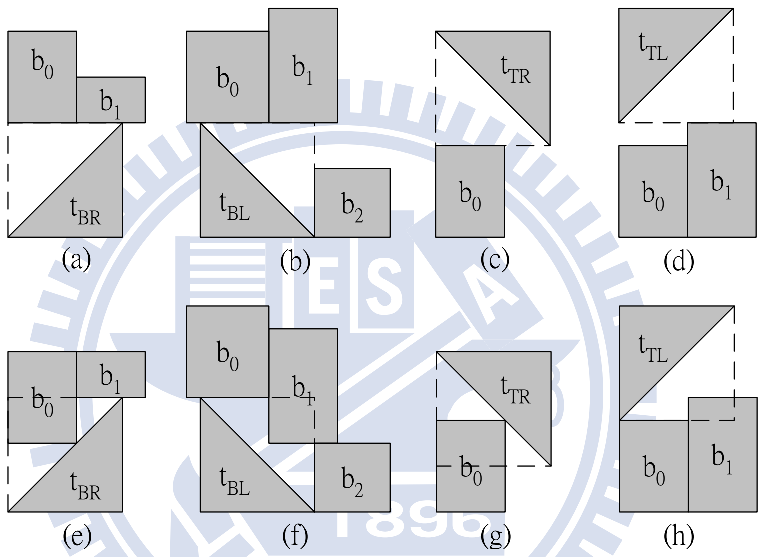


Figure 3.2: (a)(b)(c)(d) The triangular blocks of type BR, BL, TR, and TL. (e)(f)(g)(h) After adjustment of (a)(b)(c)(d).

Figure 3.2 show the floorplans with different types of triangular blocks. If we treat these triangular blocks as rectangular blocks, the deadspace will be quite large and unaccepted. In actually, we could minimize deadspace by moving down these triangular blocks or rectangular blocks. According to the property of B*-trees, the right child of the node $N$ will be placed at the same $x$-coordinate and just right upon the block, $b_N$. So, the falling down procedure described below can be achieved easily by B*-trees.

### 3.3.2 The Packing with Triangular Blocks

According to the packing scheme of B*-trees, it constructs the floorplan by the DFS ordering of nodes in a B*-tree and the blocks will first obtain $x$-coordinate, then $y$-coordinate. We take the advantage of this scheme, and enhance it to handle triangular blocks. The phase needed to be adjusted is the calculation of $y$-coordinate.

The triangular blocks are classified into four kinds according to the positions of right angles as shown in Figure 3.2 and we define some notations used in the following sections.

- *BR, BL, TR, TL*: it indicates the location of right angle. For example, $BL$ means the right angle locates at bottom-left corner and $TR$ is the right angle locating at top-right corner.
- $W_b$, $H_b$, $x_b$, and $y_b$: the width, height, $x$- , and $y$-coordinate of the bottom-left corner for block $b$.
- $W_{t_i}$, $H_{t_i}$, $x_{t_i}$, and $y_{t_i}$: the width and height for the triangular block, and $x$- , and $y$-coordinate of the bottom-left corner for the bounding box of the triangular block. Although, the value of $W_{t_i}$ is equal to $H_{t_i}$, we use two notations for explaining easily.

#### Case I: BR and BL

From Figure 3.3, we can see the triangular blocks with type BR and BL. To move down the blocks above the two triangular blocks, we have to modify the original calculation of their $y$-coordinate. According to the packing scheme of B*-trees, we have the $x$- and $y$-coordinate of the blocks whose DFS ordering are earlier than the block $b$ and the $x$-coordinate of block $b$ is already known before we calculate

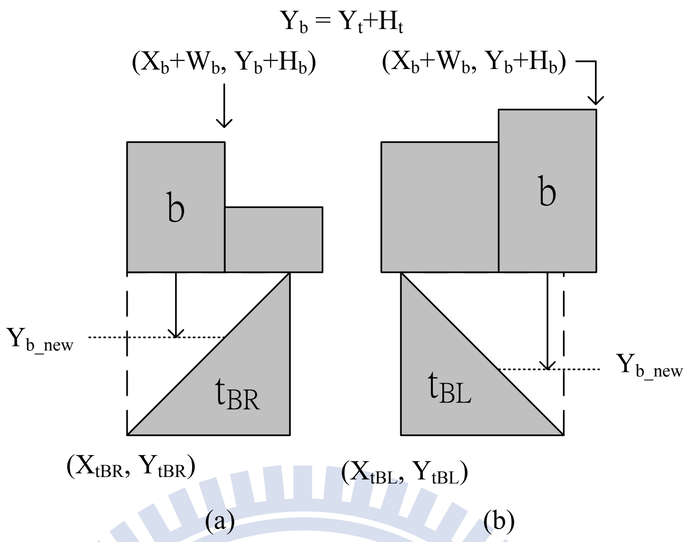


Figure 3.3: Illustrations for adjusting blocks $t_{BR}$ and $t_{BL}$.

its $y$-coordinate. Once we need to place the block $b$ upon the triangular block of type BR, $t_{BR}$, the $y$-coordinate of $b$ can be represented as following equation:

$$y_b = \begin{cases} y_{t_{BR}} + (x_b + W_b - x_{t_{BR}}) & \text{if } x_{t_{BR}} \leq x_b + W_b \leq x_{t_{BR}} + W_{t_{BR}} \\ \text{origianl packing scheme of B*-tree} & \text{otherwise} \end{cases}$$

If we need to place the block $b$ upon the type BL triangular block, $t_{BL}$, the $y$-coordinate of $b$ can be represented as following:

$$y_b = \begin{cases} y_{t_{BL}} + (x_{t_{BL}} + W_{t_{BL}} - x_b) & \text{if } x_{t_{BL}} \leq x_b \leq x_{t_{BL}} + W_{t_{BL}} \\ \text{origianl packing scheme of B*-tree} & \text{otherwise} \end{cases}$$

The two adjustments are shown in Figure 3.2(e) and Figure 3.2(f), which are corresponding to Figure 3.2(a) and Figure 3.2(b). If the width of block $b$ covers a set of $r$ blocks in the projections of $x$-coordinate, the $y$-coordinate of $b$ can be determined by:

$$y_{b,max} = max\{y_i | \text{the } y\text{-coordinate of placing block } b \text{ on the block } i, \text{ where}$$
$$i = 1, 2, ..., r.\}$$

This condition is shown by the block $b_1$ in Figure 3.2(f). With the above equation, we pick up $y_{b,max}$, the the maximum $y$-coordinate of block $b$, so we can guarantee there is no overlapping blocks.

**Case II: TR and TL**

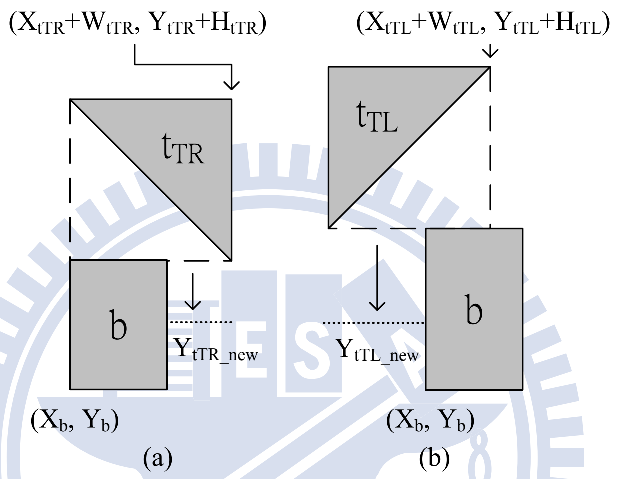


Figure 3.4: Illustrations for adjusting blocks $t_{TR}$ and $t_{TL}$.

The other two types, TR and TL, are shown in Figure 3.4. Different from the first two types (BR and BL), we move down these triangular blocks. Thus, we have to calculate the $y$-coordinate of these triangular blocks when we place them above other rectangular blocks. We show the $y$-coordinate of triangular block, $y_{t_{TR}}$, as following equation:

$$y_{t_{TR}} = \begin{cases} (y_b + H_b) - [(x_{t_{TR}} + W_{t_{TR}}) - (x_b + W_b)] & \text{if } x_{t_{TR}} \leq x_b + W_b \leq x_{t_{TR}} + W_{t_{TR}} \\ \text{origianl packing scheme of B*-tree} & \text{otherwise} \end{cases}$$

The $y$-coordinate, $y_{t_{TL}}$, is presented as:

$$y_{t_{TL}} = \begin{cases} (y_b + H_b) - (x_b - x_{t_{TL}}) & \text{if } x_{t_{TL}} \leq x_b \leq x_{t_{TL}} + W_{t_{TL}} \\ \text{origianl packing scheme of B*-tree} & \text{otherwise} \end{cases}$$

For these two cases, we should keep $y_{t_{new}} = 0$ if $y_{t_{new}} < 0$ after adjustment. The Figure 3.2(g) and Figure 3.2(h) show the floorplans after applying the two equations above to Figure 3.2(c) and Figure 3.2(d). Besides, we also need to guarantee the triangular blocks, $t_{TR}$ or $t_{TL}$, to be placed without overlapping the set of $r$ blocks covered by them. The equation is the same as we used in Case I, and the result is like the block $t_{TL}$ shown in Figure 3.2(h).

**Case III: TR vs BL, TL vs BR**

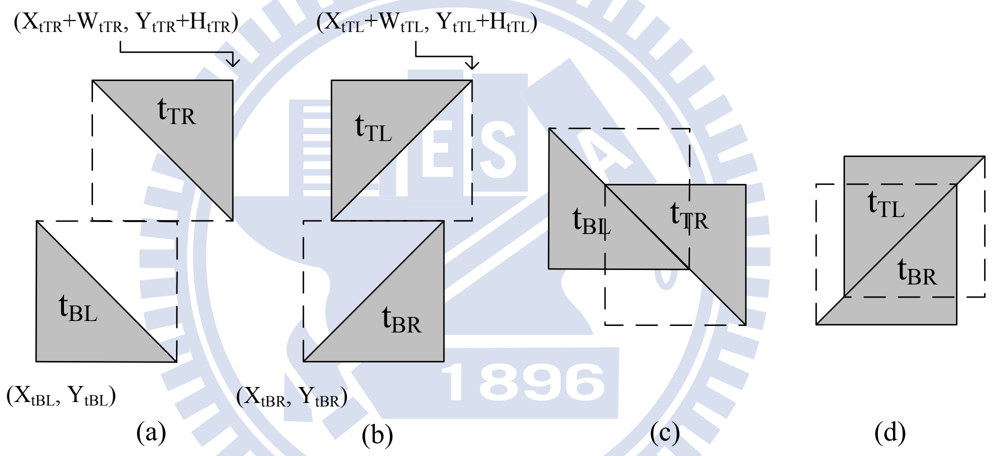


Figure 3.5: (a)(b) The triangular blocks match for each other. (c)(d) After adjustment of (a)(b).

There are two special cases from case I and case II. We can see the illustrations in Figure 3.5. When type TR matches type BL, or type TL matches type BR, these cases cause the exceptions from all equations discussing above. We need new equations to handle these two cases.

$$y_{t_{TR}} = \; y_{t_{BL}} - [x_{t_{TR}} + W_{t_{TR}} - (x_{t_{BL}} + W_{t_{BL}})]$$

for a type TR triangular block matching a type BL triangular block and

$$y_{t_{TL}} = y_{t_{BR}} + (x_{t_{TL}} - x_{t_{BR}})$$

for a type TL triangular block matching a type BR triangular block. For these two cases, we should also keep $y_{t_{TR}} = 0$ or $y_{t_{TL}} = 0$ if they are less than zero after adjustment.

Notice that, the $y$-coordinates of theses blocks related with triangular blocks have to be the maximum values of their covered blocks to avoid overlapping other blocks, so we must take care about choosing $y_{t_{TR}}$, $y_{t_{TL}}$, and $y_{t,max}$ for the correct calculation.

## 3.4 Floorplanning with Trapezoidal Blocks

By the discussions and equations of previous section, we handle floorplans with triangular blocks. In this section, we first present the feasibility conditions for trapezoidal blocks, then we show how to solve this problem with B*-trees.

### 3.4.1 Feasibility Condition for Mixed Rectangular and Trapezoidal Blocks

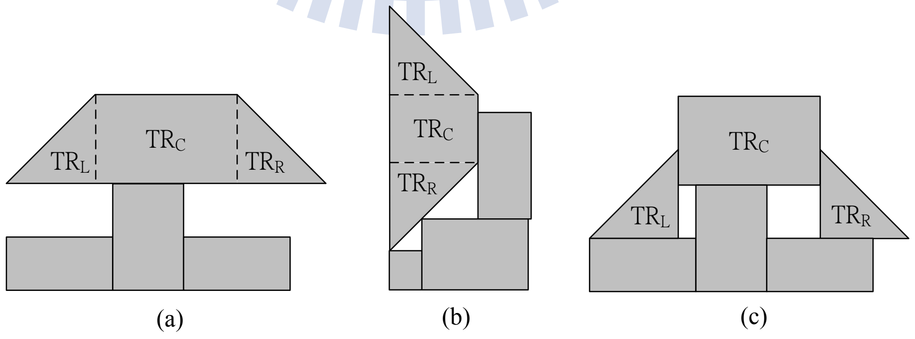


Figure 3.6: (a)(b) The floorplan with trapezoidal blocks. (c) Packing with original B*-tree scheme, block $TR_L$ and $TR_R$ have falling down problems.

The trapezoidal blocks can be placed in horizontal or vertical direction, and we can see the two directions in Figure 3.6(a) and Figure 3.6(b) respectively. We divide the trapezoidal block into three sub-blocks: a left-triangular block $TR_L$, a center-rectangular block $TR_C$, and a right-triangular block $TR_R$. The two triangular blocks should be set to correct types for maintaining the shape of the trapezoidal block. If we use the standard packing scheme of B*-trees, we may get the falling down problem as shown in Figure 3.6(c). To fix this problem, we use the dummy blocks to shift the falling blocks upward their correct positions. This method is very similar to handle the alignment constraints presented in [55], so we modify the method to solve the falling down problem in the following discussion.

### 3.4.2 The Packing with Trapezoidal Blocks

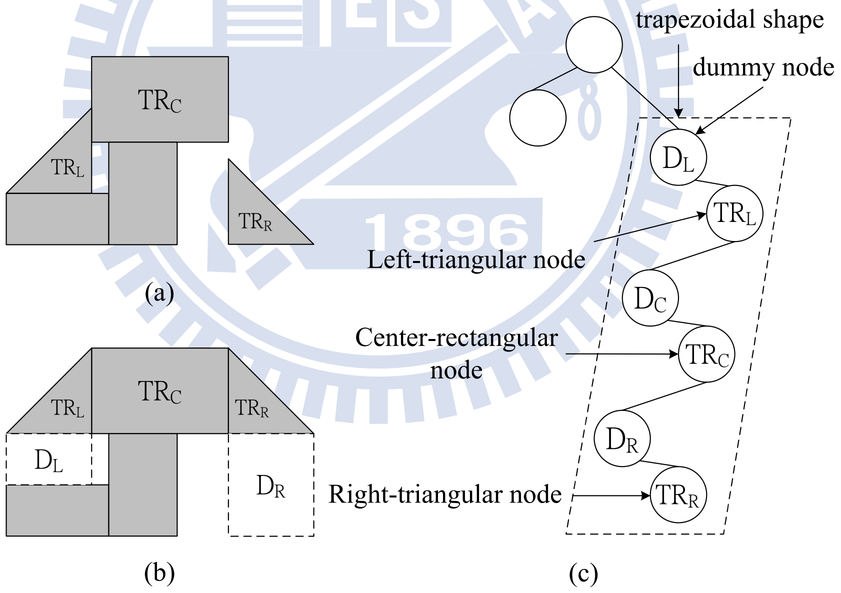


Figure 3.7: (a) Falling down problems occur at block $TR_L$ and $TR_R$. (b) The final floorplan of (a) with proper dummy blocks. (c) The B*-tree with the trapezoidal shape of (b).

For the horizontally trapezoidal blocks, according to the comparison of Figure 3.7 and the alignment constraints in [55], we can find the similarity between them. The trapezoidal block can be divided into three sub-blocks and the sub-blocks form a special case in alignment constraints where the three sub-blocks need to have the same $y$-coordinate and abut one by one. Based on this observation, we can model the trapezoidal block to an alignment constraint and apply the method of solving alignment constraints in [55]. For each sub-block, we add a dummy block right below it and their widths are set to be zero. These blocks are only used for pulling up the target blocks to their correct positions and do not disturb the position of other blocks. In the B*-tree, we also add the dummy nodes for dummy blocks. The nodes with corresponding sub-blocks are set to be the right children of these dummy nodes. Further, the dummy node is the left child of the node of previous sub-block. This shape, formed by three dummy nodes and three sub-nodes of sub-blocks, is a special case in alignment shapes and we call it *trapezoidal shape* in this chapter and shown in Figure 3.7(c).

First, the heights of dummy blocks will be set to zero. We pack all blocks by the packing scheme with triangular adjustment and get $y$-coordinates of all blocks including sub-blocks of the trapezoidal block. If their $y$-coordinates are the same, they already form the horizontally trapezoidal block. Otherwise, there must be falling problems. So, we need to calculate the heights of their corresponding dummy blocks to shift them to the right positions by the following equation:

$$\Delta_b = \begin{cases} y_{max} - y_b & \text{if } y_{max} > y_b \\ 0 & \text{otherwise} \end{cases}$$

where

$$y_{max} = max\{y_{TR_L}, y_{TR_C}, y_{TR_R}\}, \quad \text{and} \quad b = \{TR_L, TR_C, TR_B\}.$$

After calculation, the movement of each sub-blocks can be obtained and we set the height of the corresponding dummy block to each $\Delta_b$, $b = TR_L, TR_C, TR_R$.

Then, blocks are repacked and the sub-blocks will be placed at the right positions. To summarize the discussion of horizontally trapezoidal blocks above, we use Figure 3.7 for illustration. The B*-tree we obtained is shown in Figure 3.7(c), and we can see there is a trapezoidal shape. After first-packing, the result floorplan is in Figure 3.7(a) and there are falling problems for block $TR_L$ and $TR_R$. Then we calculate the heights of these dummy blocks, $\Delta_{TR_L}$ and $\Delta_{TR_R}$, and repack the floorplan with them. Figure 3.7(b) shows the correct floorplan after repacking with the proper dummy blocks. Note that, the widths of dummy blocks are all zero, we draw them in dotted line with the same widths of their corresponding blocks for explaining more easily.

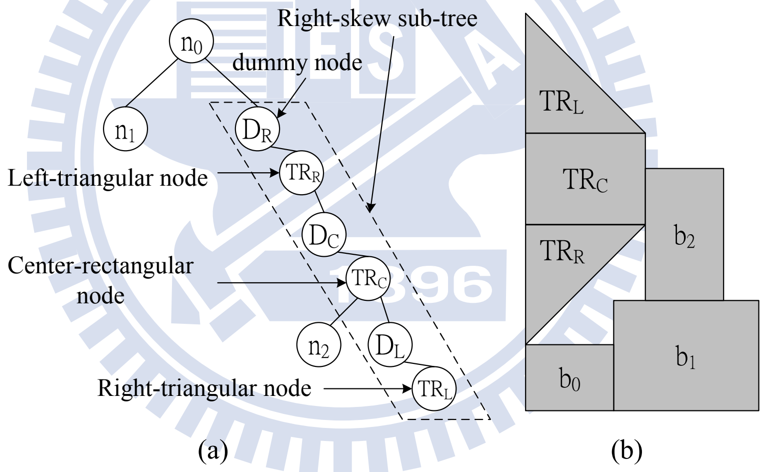


Figure 3.8: (a) The B*-tree with a right skew sub-tree for the vertically trapezoidal blocks. (b) The final floorplan of (a).

If we have one vertically trapezoidal block, the shape of this trapezoidal block could be maintained more easily than horizontal one. Based on the property of B*-trees, the block, which is corresponding to the right child node of a parent node, is placed at the same $x$-coordinate and just upon its parent block. We could keep dummy nodes and sub-nodes of vertically trapezoidal block into a right-skew sub-tree as shown in Figure 3.8(a) and set the heights of dummy blocks to be zero.

Then we use the triangular packing scheme discussing at the pre-section and the final floorplan shows in Figure 3.8(b). Note that placing vertically trapezoidal blocks could be done without repacking.

## 3.5 The Algorithm of Packing with Non-rectangular Blocks

```
Algorithm: Packing scheme with isosceles right triangular
           and trapezoidal blocks
Input: A set of rectangular blocks, isosceles right triangular blocks,
       and trapezoidal blocks.
Output: A floorplan with minimum deadspace.
1.  Initialize a B*-tree for the input blocks;
2.  Simulated annealing process:
3.  do
4.    perturb();
5.    first-packing();
6.    if the $y$-coordinates for sub-blocks of the horizontally
        trapezoidal block are not equal
7.      then adjust heights of dummy blocks
              to maintain the shape of trapezoid block;
8.            re-packing();
9.    evaluate the B*-tree cost;
10. until converged or cooling down;
11. return the best solution;
```

Figure 3.9: The packing scheme with isosceles right triangular blocks and trapezoidal blocks.

The flow of our algorithm is summarized in Figure 3.9. We use simulated annealing to search the best solution. The B*-tree is perturbed to another by the following operations:

- **Op1:** Rotate a node.
- **Op2:** Flip a node.
- **Op3:** Move a node to another place.

- **Op4:** Swap two nodes.
- **Op5:** Rotate a set of trapezoidal nodes.
- **Op6:** Flip a set of trapezoidal nodes.
- **Op7:** Move trapezoidal nodes to another place.

The first four operations are used in [7] and the others are designed for trapezoidal blocks. In $Op1$, we rotate a node and this action can be applied to rectangular and triangular nodes. For triangular node, $Op1$ changes its type to another. In $Op2$, we flip a node. Same as $Op1$, we need to maintain the correct types for triangular nodes. $Op3$ and $Op4$ change the relations of nodes to get a different floorplan. We do not apply these two operations to trapezoidal nodes. For trapezoidal nodes, we design $Op5$ - $Op7$ to perturb these blocks. First, we use the center node to denote the entire trapezoidal shape. In $Op5$, we rotate the center node and keep the shape of the trapezoidal block by maintaining the correct trapezoidal shape in the B*-trees. The $Op6$ is almost the same as $Op5$ but it flips these blocks. Finally, we use $Op7$ to move all sub-nodes of a trapezoid block to another place and maintain its shape at the new position.

## 3.6 Experimental Results

We demonstrate the results of our algorithm with the modified MCNC benchmarks so that they can be used for floorplanning with triangular and trapezoidal blocks. We call the modified benchmarks with trapezoidal and triangular blocks in the following format: tami33-$x$-$y$ and tami49-$x$-$y$, which means we translate $x$ blocks from rectangles to trapezoids and add $y$ triangular blocks. For example, the benchmark tami33-3-5 has 38 blocks, which are 30 rectangular blocks, 3 trapezoidal blocks,

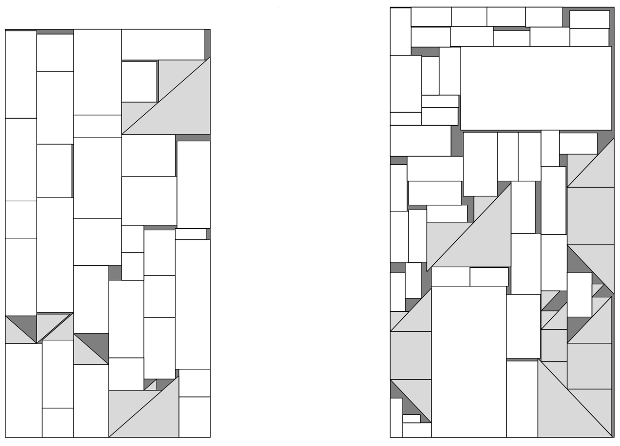


Figure 3.10: The result floorplans of (a) tami33-0-10 and (b) tami49-4-10.

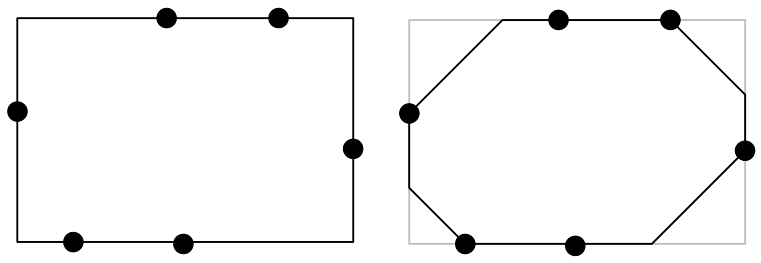
and 5 extra triangular blocks. To compare with rectangular floorplans, we also use ami33-0-$y$ and ami49-0-$y$ with additional $y$ rectangular blocks and implement the B*-tree [7] on the same environment to get the final results. In these benchmarks, the areas are the same for those extra triangular and rectangular blocks.

Table 3.1 shows the experimental results of our algorithm and it has the comparable efficiency to the rectangular floorplan. For area usage, the floorplans with triangular or trapezoidal blocks achieve the same performance as rectangular floorplanner. Figure 3.10 shows the packing results for modified tami33-0-10 and tami49-4-10. For every benchmark only with additional triangular blocks, the differences of area usage are all less then 1%. The difference between ami49-0-5 and tami49-0-5 is only 0.29%. This means we can have the same performance if we only have triangular blocks in our design. Even for the benchmark with many special blocks, tami49-4-10, the area usage is 93.70% and the difference with ami49-0-10 is less than 3%. By using these triangular blocks, we can obtain more choices for pin assignment and more special shapes in floorplan. In fact, any shapes combined by rectangles and triangles can all be handled by our algorithm.

Table 3.1: The area and runtime comparison between rectangular floorplanner and our approach.

| circuit | blocks | chip ($mm^2$) | triangles | trapezoids | result ($mm^2$) | usage(%) | Time(secs) |
|---|---|---|---|---|---|---|---|
| ami33-0-5 | 38 | 1.19 | 0 | 0 | 1.21 | 98.35 | 11.9 |
| tami33-0-5 | | | 5 | 0 | 1.22 | 97.54 | 26.6 |
| tami33-3-5 | | | 5 | 3 | 1.23 | 96.75 | 37.0 |
| ami33-0-10 | 43 | 1.33 | 0 | 0 | 1.36 | 97.79 | 16.0 |
| tami33-0-10 | | | 10 | 0 | 1.37 | 97.08 | 30.9 |
| tami33-3-10 | | | 10 | 3 | 1.38 | 96.37 | 69.8 |
| ami49-0-5 | 54 | 39.08 | 0 | 0 | 40.36 | 96.83 | 14.4 |
| tami49-0-5 | | | 5 | 0 | 40.48 | 96.54 | 42.68 |
| tami49-4-5 | | | 5 | 4 | 41.56 | 94.03 | 132.2 |
| ami49-0-10 | 59 | 39.43 | 0 | 0 | 40.98 | 96.22 | 17.9 |
| tami49-0-10 | | | 10 | 0 | 41.14 | 95.84 | 48.1 |
| tami49-4-10 | | | 10 | 4 | 42.08 | 93.70 | 107.0 |

For wirelength results, we add pins into test cases and create connectivity between original and extra blocks. The pin locations are normalized to be distributed on the boundary of each block. We use MHPWL and XHPWL [10] to estimate the wirelength of a net by the bounding box enclosing the net. The MHPWL means Manhattan-half-perimeter wirelengths estimation and the XHPWL means the X-half-perimeter wirelength estimation as shown in Figure 3.11. The total wirelength is the summation of all nets and the results are shown in Table 3.2. We can see com-



(a) MHPWL (b) XHPWL

Figure 3.11: Two kinds of wirelength estimation approach

parable wirelength for both floorplans and we have the better wirelength reduction with XHPWL and our proposed packing scheme.

Table 3.2: Experimental results for wirelength estimation.

| circuit | MHPWL($mm$)(a) | XHPWL($mm$)(b) | (a)-(b) |
|---|---|---|---|
| ami33-0-5 | 46.0 | 44.7 | 1.3 |
| tami33-0-5 | 46.1 | 44.5 | 1.6 |
| ami33-0-10 | 48.9 | 47.7 | 1.2 |
| tami33-0-10 | 50.6 | 48.1 | 2.5 |
| ami49-0-5 | 667.8 | 639.1 | 28.7 |
| tami49-0-5 | 671.5 | 640.9 | 30.6 |
| ami49-0-10 | 639.6 | 613.4 | 26.2 |
| tami49-0-10 | 637.7 | 609.6 | 28.1 |

## 3.7 Summary

We have presented an efficient yet effective algorithm to handle the floorplanning with isosceles right triangular blocks based on the the B*-tree representation. Better than only with rectangular blocks, the floorplan with isosceles triangular blocks has more choices for block shapes and pin locations on the boundary of a block. We reveal the solutions to handle isosceles right triangular and trapezoidal blocks in rectangular floorplans, and guarantee a feasible floorplan with these special blocks. Further, our proposed algorithm can deal with all shapes which are the combination of rectangle and isosceles right triangle. The experimental results show floorplanning with isosceles right triangular blocks by our method can achieve the same performance with normal floorplanners in both area usage and wirelength optimization.

# Chapter 4

# Floorplanning with Voltage Island Generation and Performance Constraints

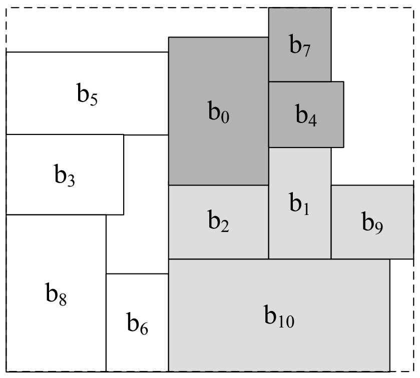
In this chapter, we propose the heuristic method to generate voltage island and simultaneously meet performance constraints. In the beginning, we reveal the motivation and problem formation of this chapter. Next, the method for generating voltage island is proposed. Then, we show the approach to meet performance constraints for blocks with different supplying voltages. Finally, we demonstrate our method with MCNC benchmarks. Comparing with two previous works, our method has the best performance in reducing deadspace and the cost of level shifters. Besides, we also present the results in meeting performance constraints with the less cost of level shifters.

## 4.1 Motivation

Voltage island architecture [25] can achieve power saving and has become more and more popular [6][11][17][18][25][26][54]. In [17][18], they partition IP cores into several sub voltage islands, floorplan each sub voltage island independently, and floorplan all voltage islands to form the final result. This approach somewhat restricts the exploration of solution space. In [54], a post-placement approach of generating voltage islands is proposed. However, chip floorplanning level has more flexibilities and leads the initial solution of power planning. Thus, the better way to maximize the power saving is to consider voltage island generation during the floorplanning/placement stage.

Performance is one of the major concerns since the interconnect delay dominates the circuit performance for DSM VLSI design. Minimizing total wire length, as traditional floorplanners/placers did, can not guarantee bounded delay for critical nets. It is desirable to minimize the critical net delay to optimize performance or to meet the delay constraints by placing these blocks/cores with critical nets close enough to each other. In [44], the maximum delay of performance-constrained blocks is bounded by the summation of its height and width of the bounding box enclosing those blocks. However it is not trivial to bound the maximum delay for those performance-constrained blocks in voltage island architecture, especially for those which are not in the same voltage island. Thus, we need an approach to solve the problem about placing blocks in different voltage islands while meeting their performance constraints.

The methodology we proposed targets on generating voltage islands in chip floorplanning stage instead of post-placement one [54], in order to have more flexibilities in design. In this chapter, we still adopt B*-tree [7] as our floorplan representation and underlying implementation since B*-tree can provide very good quality of non-slicing floorplans in area and wirelength costs. Our methodology can save power consumption and routing cost by location constraint [7], and solve the critical delay problems by performance-constrained consideration [55], even these blocks are in different voltage islands. Besides, we present heuristics to obtain voltage islands more easily and efficiently with the original B*-tree.



$b_0$ 1 (1.2, 0.8)
$b_1$ 2 (1.1, 18.2) (1.2, 23.4)
$b_2$ 2 (1.1, 7.8) (1.2, 9.2)
$b_3$ 3 (1.0, 1.3) (1.1, 1.5) (1.2, 2.6)
$b_4$ 1 (1.2, 1.8)
$b_5$ 3 (1.0, 0.9) (1.1, 1.5) (1.2, 2.1)
$b_6$ 3 (1.0, 10.2) (1.1, 12.6) (1.2, 15.8)
$b_7$ 1 (1.2, 0.7)
$b_8$ 3 (1.0, 9.8) (1.1, 11.2) (1.2, 15.5)
$b_9$ 2 (1.1, 15.8) (1.2, 22.2)
$b_{10}$ 2 (1.1, 6.1) (1.2, 10.0)

Voltage island A: 1.0V $\{b_3, b_5, b_6, b_8\}$
Voltage island B: 1.1V $\{b_1, b_2, b_9, b_{10}\}$
Voltage island C: 1.2V $\{b_0, b_4, b_7\}$

Figure 4.1: An example of a design with three voltage islands and the corresponding power table.

## 4.2 Problem Formulation

In this section, we define the problem about floorplanning with voltage islands generation while meeting performance constraints.

**INPUT**

- The set of rectangular blocks, $B = \{b_1, b_2, ..., b_n\}$, and their corresponding widths and heights are denoted as $W_i$ and $H_i$, $1 \leq i \leq n$.
- The power table of blocks. In this table, we have the information about the number of supplying voltages, $SV_i$, and the pair of supplying voltage and power consumptions for the voltage, $(V_i, P_i)$, $1 \leq i \leq n$.
- The list of performance constraints. This list has the information about the blocks connecting with some critical nets.

**OUTPUT**

- The floorplanning with proper number of voltage islands while meeting performance constraints.

**OBJECTIVE**

- Optimize a pre-defined cost metric, such as minimizing dead space or being trade-off between power consumption and the cost of level shifters.

## 4.3 The Method of Generating Voltage Islands

In this section, we propose the method for voltage islands generation with B*-tree representation, discuss the strategy to merge different voltage islands in the B*-trees before packing.

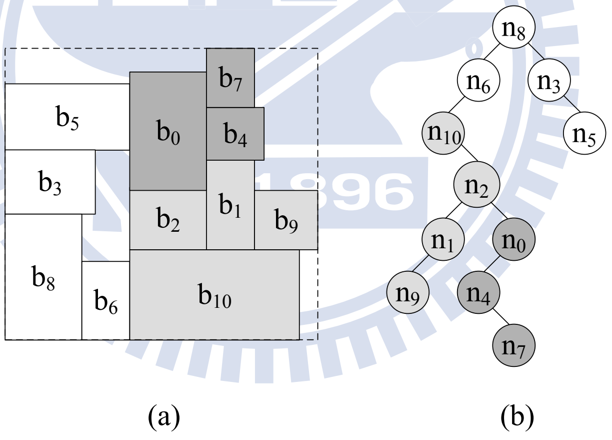


Figure 4.2: An illustration of generating voltage islands with the lowest voltage of each block to maximize power saving.

First, we use the similar methods with [18] to generate voltage islands. The way to minimize the power consumptions is to operate each block at its lowest voltage, floorplan each voltage island, and floorplan all voltage islands to form final results. Figure 4.2 demonstrates an example of a design with three voltage islands:

one for $\{b_0, b_4, b_7\}$, one for $\{b_1, b_2, b_9, b_{10}\}$, and one for $\{b_3, b_5, b_6, b_8\}$. This solution could be quickly implement with B*-trees. First, we assign the lowest supplying voltage to each block, construct the sub-trees for corresponding voltage islands, combine all sub-trees and pack to obtain the resultant floorpaln. However, this method is obviously not optimal since the exploration of solution space is limited and the deadspace/wirelength could be unacceptable. It needs to develop a method to pack all blocks at the same time and maximize the solution space. With only the parent-child relationship of each node in the B*-tree and, we build some heuristic rules for increasing the opportunities to enlarge or merge voltage islands before packing the floorplanns.

According to the properties of B*-trees at Section 2.1 and the discussion in the previous chapter, there is an important information that the blocks with parent-child relationships in the B*-trees have high opportunities to be placed abutted each other. For example, if $n_j$ is the left child (or right child) of $n_i$ in the B*-tree representation, then the block $b_j$ right abuts to the block $b_i$ (or is visible and above to the block $b_i$). Thus the probability that a node adds to a *compatible* subtree, in which the nodes are all the same voltage, and the subtree grows and maps to the same voltage island shape will be increased.

For heuristic idea, we build the swapping rules for generating voltage islands with B*-trees before obtaining geometric information. First, we randomly choose two nodes $n$, $p$ in the tree ( $V(n)$ and $V(p)$ denote the adopted voltages of node $n$ and node $p$), and if the following conditions appear, we swap the positions of these two nodes in a B*-tree:

- $V(p) = V(n)$: Node $p$ and node $n$ are compatible, no new voltage islands will be created, as shown in Figure 4.3(a).

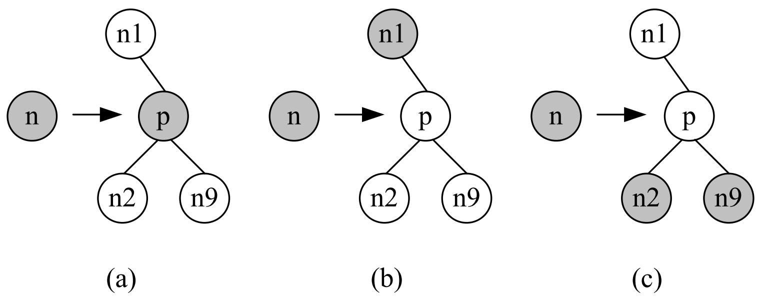


Figure 4.3: Three conditions to increase the probability of merging the same voltage islands.

- $V(p.parent) = V(n)$: Node $p$'s parent and node $n$ are compatible. We swap node $p$ and node $n$, and node $n$ becomes the leaf of the subtree, or connects two or three compatible subtrees to form a larger subtree (if one or both $p$'s children have the same voltage with node $p$'s parent), as shown in Figure 4.3(b).

- $V(p.leftchild) = V(n)$ and $V(p.rightchild) = V(n)$: Node $p$'s children both have the same voltage with node $n$. We swap node $n$ and node $p$, and node $n$ becomes the root of the subtree, and connects two compatible subtrees to form a larger subtree, as shown in Figure 4.3(c).

Besides the three swapping rules, we also modify $Delete\_Node$ and $Insert\_Node$ for perturbing B*-trees:

- Delete_Node: If we want to delete node $n$, we have to keep the connection of nodes with the same voltage. If the supply voltage of one child nodes ($n.leftchild$ or $n.rightchild$) is compatible with node $n$'s parent, we choose it to be new child of node $n$'s parent. If the children are in the same situation (both compatible or both not compatible), we randomly choose one of them.

- Insert_Node: If node $n$ has to be inserted into one subtree and the subtree exists compatible nodes, the node $n$ will be placed to join the cluster of the compatible nodes. If there does not exist any node compatible, we randomly choose one place to insert.

In [53], the *location constraint* (LC for short) was proposed to maintain the relation of blocks with packing of B*-trees. The blocks with LC relation will be placed with the desired shape (L-shape or T-shape) in the final floorplan. In [55], they extend the LC relation to *alignment shape* to guarantee the alignment constraints will be satisfied at all times. In the previous chapter, we enhance the *alignment shape* to *trapezoidal shape* for trapzoidal blocks.

From observation of these works, we know that if the nodes in the B*-tree are closer to each other, the chance that blocks are next to each other will be higher in the final floorplan. Thus, we define the term *diameter* for our heuristic method to handle blocks to be placed in a neighborhood:

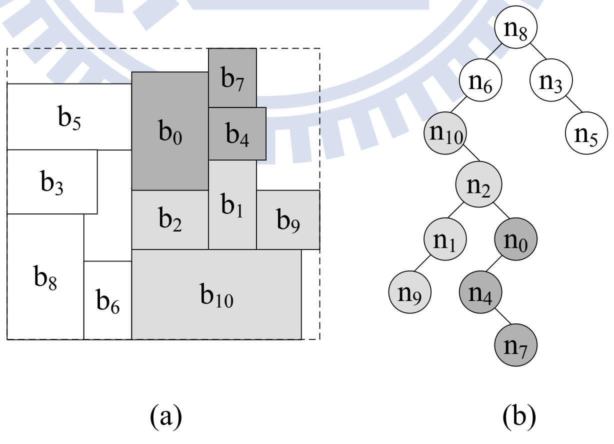


Figure 4.4: The *diameter* of two nodes is the number of nodes between them.

**Diameter,** $Dia(n_i, n_j)$: the number of nodes between nodes $n_i$ and $n_j$ in the B*-tree.

We use Figure 4.4 (from Figure 4.2) to illustrate this idea. In Figure 4.4, two nodes with diameter $Dia(n_i, n_j) = 0$ are in the direct parent-child relationship and almost abutting to each other. We can see that the blocks whose $Dia(n_i, n_j) \leq 2$ are almost adjacent or near to each other. There exists an exception that blocks $b_0$ and $b_5$ are placed next to each other but their corresponding nodes have the diameter larger than 2. The reason is that these constraints for locations could only handle nodes in the same sub-trees or close to others. For different sub-trees, it is very difficult to predict their locations after packing and do not taken into consideration in this chapter. Though the *diameter* does not guarantee the final relation of blocks, we could use it to guide the desired blocks being placed near each other without restricting their shapes (as in [55]).

The original B*-tree [7] focuses on normal constraints for floorplanning, such as deadspace/wirelength minimization or some pre-placed blocks, etc. With these proposed features, new operations and diameters between nodes, we could have more opportunities to generate larger voltage islands before obtaining geometric information. Since the subtree construction is just a method to increase the possibility to place blocks/cores together, we need a property checking function to check if there exists a suitable voltage island. We do it after the contour updated to make sure the number of voltage islands is acceptable. Our checking is efficient since the block is only checked one time while it is placed at the segment of contour line, instead of checking all placed blocks.

## 4.4 The Heuristic for Performance Constraints

Traditional floorplanners/placers minimize total wirelength but they can not guarantee critical nets to meet bounded delay. This problem becomes more important because timing convergence is a big issue in DSM design. In order to meet critical delay constraint, there are methods proposed in [44, 55] during floorplanning. Since actual interconnect delay after appropriate buffer insertions will be close to linear in terms of distance, linear function in terms of distance to estimate delay is used. Assume there are a source at $(x_s,y_s)$ and a sink at $(x_t,y_t)$, and the delay of the net $D_{s,t} = \delta Dist_{s,t} = \delta(\mid x_t - x_s \mid + \mid y_t - y_s \mid)$, where $\delta$ is a constant to scale the distance to timing, and $Dist_{s,t}$ is the maximum distance between source and sink, equal to the half perimeter of the bounding box of the two points.

There is a major problem in this performance model with voltage island architecture. The performance of each block is different according to its supplying voltage and we need to find the suitable bounding box for these blocks even if they are in different voltage islands. The legal supply voltage has big impact on the driving strength, thus the bounding box size. If signals are communicated by high supply voltage, the bounding box for performance-constrained blocks will be larger than one with lower voltage. In addition, the allowable box size should be a function of the supply voltage.

Our approach combines the advantages of these methods in [53, 55, 44], keeping the flexibility of the sub-placement for the performance constraints. In a B*-tree, we set the diameter, which is defined in previous section, of performance-constrained nodes and let these nodes be handled with other nodes as if they are not under restriction. And then, after floorplanning, we check whether these constrained blocks are placed in the desired bounding box or not. If these constrained blocks are in the desired bounding box, we accept this floorplan. The bounding box will be shrunk at every time we accept a new floorplan until meeting the performance constraints, like mentioned in [44]. Thus, the total area and wirelength can be better optimized. This is further verified in the condition that supply voltages of the performance-constrained blocks are possibly different.

To meet performance constraints, the initial bounding box of each constrained group is the bounding box of first floorplan we got after first packing. We use its half perimeter to calculate first $D_{bound}$, where $D_{bound}$ presents the delay of the shrinking bounding box, and we can also get $D_{s,t}$ of performance-constrained blocks. There is no doubt we have a first valid floorplan whose $D_{s,t} < D_{bound}$. Every time we accept a valid floorplan, we decrease the $D_{bound}$ and repeat it until $D_{bound} <= D_{max}$, the maximum delay of this constraint. Once the condition $D_{s,t} <= D_{bound} <= D_{max}$ is achieved, we get a floorplan meeting performance constraints and can use $D_{max}$ to constrain performance blocks in further process. By reducing $D_{bound}$ step by step, we can get more solution space than [55] and guarantee the final floorplan meets the performance constraints by $D_{bound} <= D_{max}$ as [44]. As mentioned before, in order to handle the blocks in different voltage islands, the $D_{bound}$ is adjusted, $D_{bound} \times FUNC(vol_{max})$, with the maximum voltage of corresponding islands. The function of $FUNC(vol_{max})$ is related different voltages and could be set by designer according to the used voltages. We suggest to set $FUNC(vol_{max}) = 1$ with the maximum supplying voltage and reduce the value with other lower voltages.

## 4.5 The Details of Proposed Algorithm

To floorplan with voltage island generation and meeting performance constraints, our algorithm is based on the simulated annealing method and we only consider hard modules in this chapter. We perturb a B*-tree to another by the following operations:

- $Op1$: Change the supply voltage of a node.
- $Op2$: Rotate a node.
- $Op3$: Flip a node.
- $Op4$: Swap two node.
- $Op5$: Move a node to another place.

The first operation $Op1$ is only applied to the blocks with more than two supplying voltages. In this operation, we change the voltage of target node to be the same with its parent or child, if possible. If the operation is successfully applied, we could expect that the larger voltage islands is formed after packing. The next two operations, $Op2$ and $Op3$, are applied to all nodes and they only change the shape of voltage islands without disturbing the number of islands. For performance constraints, these two operations do not change any diameter and we only need to check the bounding box of performance constraints with rorating a node after packing.

$Op4$ and $Op5$ change the relations of blocks to get a different placement and B*-tree structure based on our heuristics. Like the $Op1$, these two operations expect to extend the voltage islands. To get better results with $Op4$, we give the higher probability to the two nodes with the same voltage, and lower probability to the two nodes with different voltages. In the original B*-tree, it gives the random move for $Op5$. In our algorithm, we use the new *Delete_Node* and *Insert_Node* here to generate voltage islands as large as possible. Moreover, if we tighten the shape of performance-constrained blocks at the beginning, we may be forced to raise the

```
Algorithm: Floorplanning with Performance Constraints
                for Voltage Islands(blocks, power_table, constraints)
Input: A set of blocks, the power table and performance constraints.
Output: A floorplan with voltage islands and meeting the given constraints.
1.  Initialize a B*-tree for the input blocks and constraints;
2.  Simulated annealing process;
3.  do
4.     perturb();
5.     performance constraint check();
6.     if diameters of performance-constrained blocks fail to meet constraints
7.        then adjust the corresponding nodes in the B*-tree
8.     packing();
9.     voltage island check();
10.    if the property is not good
11.       then add the penalty to the cost function
12.    evaluate the cost of this floorplanning;
13. until converged or cooling down;
14. return the best solution;
```

Figure 4.5: The algorithm of floorplanning with performance constraints for voltage island generation.

supply voltages of some of them to the higher one to match the voltage of the island which these blocks belong to; or we will get a malicious B*-tree structure that the voltage property is withered. When the temperature becomes lower in annealing process, we do not accept a solution that violates performance constraints even if it has better cost, the best solution will be kept until next feasible solution with better cost.

Figure 4.5 shows our algorithm in detail. First, we initialize the B*-tree and set up the power table and constraints (see line 1). We construct the initial B*-tree according to the order which the blocks are presented and set all blocks to their lowest voltage. Then, we start the simulated annealing (SA) process. After each perturbation (see line 4), we check the performance-constrained blocks (see line 5).

Because there are no exact coordinates of performance-constrained blocks in this stage (before packing), we use the *diameter* to maintain their relation in the B*-tree (close to each other). If there exists any diameter between each pair of nodes of performance-constrained blocks larger than 2, we use the probability to decide whether accepts this perturbation or not. If we do not accept this perturbation, we will reorder their relations and make sure there is no diameter larger than 2 to expect the more adjacency of performance-constrained blocks (see line 7). The adjusting function consists two steps: remove the node with the largest diameter and then insert it into the original group with diameter less then 2. Then we do the packing, check the performance-constrained blocks, accept a placement without violating performance constraints (see line 8), and pass it to the next step. If the performance constraints check fails, we treat it as a failing step in the simulated annealing process. We check the property of voltage islands for this placement (see line 9), if the property is not good for this placement (ex. too many voltage islands or too many level shifters), we penalized the solution so that it will be rejected by SA. After checking the property of voltage island, the placement is evaluated by its area, wire length, and the cost of level shifters (see line 12). This cost function, $cost = \alpha$area $+\beta$power $+\gamma$level shifters where $\alpha + \beta + \gamma = 1$, takes care of area of the floorplan, the total power consumption and the cost of level shifters. In our experimental setting, we set $\alpha = 0.9$, $\beta = 0.05$, and $\gamma = 0.05$. The iteration continues until the end of SA scheme (see lines 3-13) and the best solution is reported(see line 14). According to this flow, we can guarantee the performance constraints are met during the perturbation and we will try to get the placement with good property of voltage islands at each perturbation.

## 4.6 Experimental Results

We implemented our algorithm and the benchmarks we used are MCNC benchmarks with the power table created by us, which is shown in Figure 4.1. Table 4.1 shows the comparisons between [17], [18] and our approach on deadspace, power consumption and cost of level shifters. Because we need the level shifters for changing voltages between different voltage islands, for these costs, we use a simple evaluation for these benchmarks. We assume all level shifters are placed on the boundary of every voltage island, except for the boundary of chips. There are two reasons for supporting this assumption. First, the designers could reserve the thin area on the boundary of each island, and these area are used for level shifters and the wire connections between voltage islands. Second, the power pins are placed outside the boundary of chips and we can ignore these level shifter in our design (it should be already level shifters at the boundaries between chip and blocks). Thus, based on the reasons, we evaluate the cost of level shifters with the total boundary length between different voltage islands except outmost of the floorplan.

Table 4.1: The experimental results on some MCNC benchmarks.

| Circuit | Table | [Hu et al.2004][17] | | | | [Hung et al.2005][18] | | | | Ours | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Dead | P | C | CPU | Dead | P | C | CPU | Dead | P | C | CPU |
| hp | pt2 | 3.10% | 86.4 | 1 | 23 | 3.10% | 86.4 | 1 | 18 | 3.10% | 86.4 | 1 | 15 |
| | pt3 | 2.98% | 78.3 | 1 | 25 | 2.98% | 78.3 | 1 | 22 | 2.98% | 78.3 | 1 | 18 |
| ami33 | pt3 | 4.25% | 115.8 | 1.46 | 82 | 5.06% | 114.7 | 3.22 | 109 | 2.07% | 123.2 | 1 | 89 |
| | pt3-1 | 3.75% | 136.6 | 2.46 | 86 | 4.26% | 125.6 | 5.56 | 145 | 2.23% | 136.3 | 1 | 89 |
| ami49 | pt2 | 4.08% | 157.9 | 1.49 | 263 | 6.21% | 148.3 | 3.77 | 358 | 3.34% | 151.5 | 1 | 243 |
| | pt3 | 4.42% | 151.1 | 1.53 | 246 | 6.92% | 146.5 | 3.87 | 398 | 3.38% | 156.2 | 1 | 234 |
| | pt3-1 | 5.24% | 200.1 | 1.12 | 275 | 7.32% | 193.5 | 2.63 | 366 | 3.52% | 196.4 | 1 | 234 |
| | pt3-2 | 4.21% | 223.7 | 1.20 | 255 | 6.54% | 218.9 | 2.89 | 387 | 3.64% | 222.9 | 1 | 240 |

Table 4.1 demonstrates the experimental results of the three methods, [17], [18] and our approach. In Table 4.1, columns 1 and 2 are the name of circuits and their

corresponding power tables. In the power tables, $pt2$ means these blocks having at most two operating voltages, and $pt3$, $pt3-1$, and $pt3-2$ are with at most three operating voltages. In the following columns, $Dead$, $P$, $C$, and $CPU$ are the deadspace, power consumption, the cost of level shifters which is normalized to us, and run time of each benchmarks, respectively. For power consumption, we sum up the power consumption of each block with its corresponding operating voltage. We implement the work of [17] and [18] based on the B*-tree representation, which is comparable in experimental setup and assumptions to our work. For these experimental results, the proposed algorithm shows the best results at the trade-off between power consumption and the cost of level shifters with the minimum deadspace at each benchmark. The mehod of [18] gets the best results of power consumptions, however, their deadspace and the cost of level shifters are the largest in all benchmarks. Because their method merges voltage islands first and floorplans these voltage islands, their method has the best results in power consumption but also gets the larger deadspace and costs for level shifters.

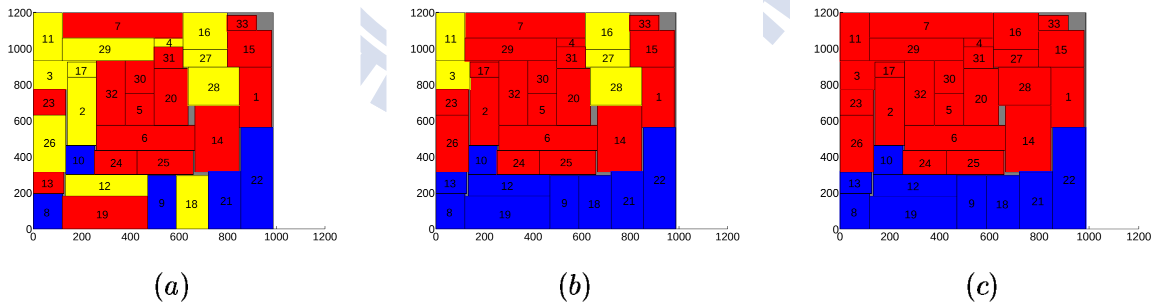


Figure 4.6: An illustration of $ami33$ with different voltage assignment methodologies

We propose a simple way to generate voltage island with the original B*-trees. In this experiment, we fix each block at its lowest power and try to minimize the area and its dead space is 2.12% and power is $113.6mW$. As illustrated in Figure 4.6(a), the floorplans of circuit $ami33$ with 3 usable supply voltage demonstrates the final result from the original B*-tree. Next, we raise supply voltages of blocks to form the larger voltage islands. In Figure 4.6(b), there are four voltage islands and its power consumption is $136.8mW$. Further, we reduce the voltage islands again and the power consumption is raised to $146.3mW$. For this heuristic method, we can reduce much cost of level shifters with no extra area overhead.

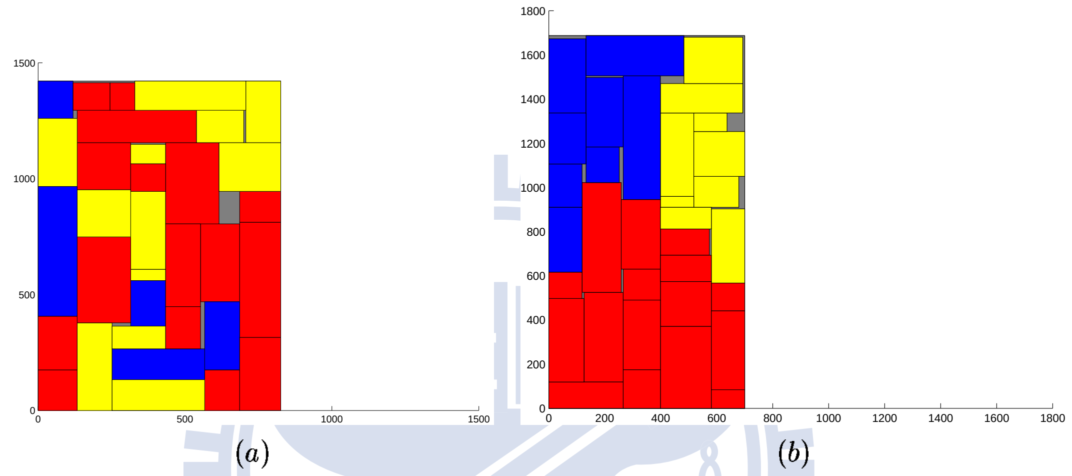


Figure 4.7: Two floorplans of circuit *ami*33 with 3 usable supply voltage. (a) The result of original B*-tree. (b) The result of proposed method.

However, this method depends on the initial floorplan. Figure 4.7 demonstrates the results of *ami*33 with 3 usable supply voltage. In Figure 4.7(a), the results has the best results in deadspace (deadspace=1.47%, power=113.6mW) but it is very difficult to form suitable voltage islands with the heuristic method. With our proposed method, the cost of level shifters is much smaller than it with a little deadspace overhead (deadspace=2.07%, power=123.2mW).

Table 4.2 shows the comparison of our results with [55] which considers only performance constraints. Column 3 gives the number of performance-constrained blocks, there are one group in ami33 and ami49-2, two groups in ami49-3, and each group has 3 blocks. Both methods meet performance constraints but our approach could get much lower cost of level shifters with slightly increased power consumption.

Table 4.2: The experimental results with performance constraints.

| Circuit | Table | Perf. | Perf. Const. Only [55] | | | | Ours | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | Area | Dead | P(mw) | C | Area | Dead | P(mw) | C |
| ami33 | pt3 | 3 | 1.181 | 2.2% | 113.6 | 4.34 | 1.18 | 2.02% | 121 | 1 |
| | pt3-1 | | | | 131.1 | 4.93 | 1.181 | 2.2% | 145.1 | |
| ami49-2 | pt2 | 3 | 36.56 | 3.1% | 147.1 | 4.5 | 36.78 | 3.64% | 156 | |
| | pt3 | | | | 142 | 6.33 | 36.89 | 4.53% | 146.6 | |
| | pt3-1 | | | | 183.1 | 6.89 | 36.87 | 3.86% | 200.9 | |
| | pt3-2 | | | | 208 | 6.7 | 36.89 | 3.93% | 221.9 | |
| ami49-3 | pt2 | 6 | 36.64 | 3.3% | 147.1 | 4.48 | 36.8 | 3.68% | 156.8 | |
| | pt3 | | | | 142 | 6.43 | 36.98 | 4.14% | 149.7 | |
| | pt3-1 | | | | 183.1 | 6.6 | 37.1 | 4.46% | 215.9 | |
| | pt3-2 | | | | 208 | 6.25 | 37.07 | 4.38% | 223.3 | |

Figure 4.8 illustrates final floorplanning result of $ami49-2, pt3$ with performance-constrained blocks 5, 6, and 7, and they are not on the same voltage island. The comparison between two approaches on power consumption and the cost of level shifters. With both meeting performance constraints, our approach obtains much lower cost of level shifters with slightly more power consumption.

## 4.7 Summary

In this chapter, we propose an algorithm which can simultaneously handle floorplanning with voltage island generation and meet performance constraints. Given the power table of design, the proposed algorithm generates voltage islands with appropriate number of blocks according to the trade-off between power consumption and cost of level shifters. Although the proposed algorithm can handle more than three supplying voltages, the balance between number of voltage islands and cost of levle shifters needs to be taken into consideration and we suggest three voltage islands is the better solutions in our experiments. For meeting performance constraints, we present a heuristic method to have more opportunities placing blocks

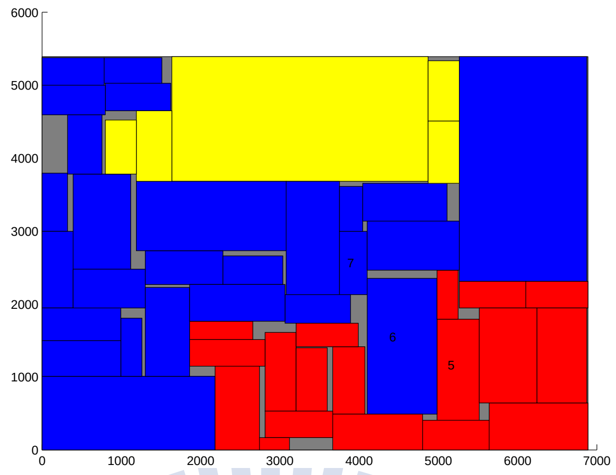


Figure 4.8: An example of *ami*49 with proposed methodology to generate voltage islands with performance constraints.

close to each other before packing of B*-trees. Further, we shrink the bounding box of performance-constrained blocks and guarantee the final solution satisfying the given constraints. With the experimental results with MCNC benchmarks, our algorithm is effective and efficient to generate voltage islands while meeting performance constraints, even the blocks are in different voltage islands.

# Chapter 5

# Layout Modification for FIB Techniques at Post-routing Stage

In this chapter, we first give the motivation about signal observation for FIB techniques. Then, we define the FIB observation and our problem formulation. Next, we propose the operations to modify layout to increase the number of signals to be observed with FIB techniques. The flow of our work is explained in the following. Finally, we demonstrate the experimental results and the timing impact between original and modified layout.

## 5.1 Motivation

The *failure analysis* takes the critical role in today's debugging flow. There are several physical probing techniques, such as electron beam (E-beam) [41] and laser voltage probe (LVP) [58]. However, the E-beam techniques are hard to observe the deeper signals and the LVP techniques need extra area overhead within design, such as additional cells [31] or larger cells instead the original cells [32]. Compared to these techniques, the *Focused Ion Beam* (FIB) techniques utilize the milling and deposition processes to create additional access points for internal signals on the surface of chips, require no area overhead within the design, and have shooter process time in current industry.

Table 5.1: FIB observable rates for 0.18$\mu$m and 90nm technologies.

| circuit | technology | | difference |
|---|---|---|---|
| | 0.18um(a) | 90nm(b) | (a)-(b) |
| s38417 | 72.66 | 36.57 | 36.09 |
| s38584 | 60.72 | 28.62 | 32.11 |
| s35932 | 85.06 | 50.84 | 34.21 |
| b17 | 41.95 | 17.86 | 24.09 |
| b20 | 50.87 | 25.62 | 25.23 |
| b21 | 46.57 | 23.47 | 23.10 |
| b22 | 46.91 | 23.56 | 23.36 |
| avg. | 57.82 | 29.50 | 28.32 |

To observe signals with FIB techniques, we need the *clean space* above the target signal, which means there is only inter-layer dielectric (ILD) above the target signal and no other metal lines cover it. However, with the high metal density and advanced technologies, the design is manufacturing with high metal stack technologies (more than 6 layers). The circuit layout is more denser than before, and the signals at lower layers are hard to be observed. Besides, the resolution of FIB techniques is limited with physical constraints and cannot scale as fast as technology node. As shown in Figure 5.1 (the same as Figure 1.1), the FIB observation rate is about 30% for the large circuits with 90nm technology and it will be worse with 65nm or 40nm technology in the future. However, in the current design flow, the automatic place and route (APR) tools are responsible for the detailed cell placement and signal routing and focus on some design constraints, such as routability or timing issues. The routes of design are almost determined at this step and also limit the observation rate of signals. In order to utilize the FIB techniques for post-silicon debugging, we have to develop some approaches with minimum impacts to routed circuits.

In this chapter, we propose the methodology for layout modification at post-routing step and greatly increase the number of signals accessed with FIB techniques. We introduce pre-defined operations which change routing layers of existing metal lines without performing complicated routing. Therefore, the proposed methodology is effective and efficient and could be applied to the layout generated with any APR tool. The experimental results with UMC 90nm technology files demonstrate the proposed methodology significantly increases the number of signals which can be observed or edited with FIB techniques while keeping the timing of critical paths without any area overhead.

## 5.2 Definition of FIB Observation

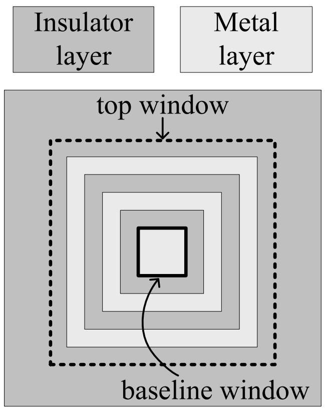
In the section, we give the definitions about FIB probing holes and the FIB observable nets.

### 5.2.1 Definition of an FIB Probing Hole

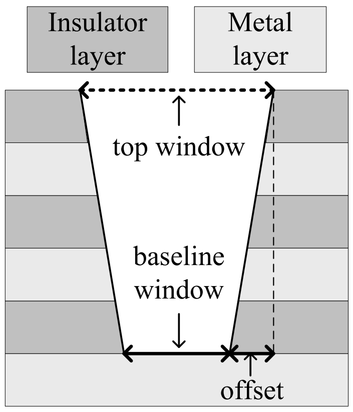
Since the first step of observing signals with FIB is surface milling, which forms the holes, we give the definition about the FIB probing holes. As shown in Figure 5.1, the FIB probing holes have several features:

- **Baseline windows:** the area for the bottom of the FIB probing hole (usually square). For higher successful rate about FIB observation, this area needs to be larger than the square of given minimal width, which is respect to FIB techniques we used.

- **Aspect ratio:** because the focused ion beam or its reflection from the surface may also hit the edge of the dub hole, the edge of the hole is not directly orthogonal and have a few angles and some offset from bottom to top. The angles are determined with the power level of the current with FIB. Although



(a) Top view of the FIB hole



(b) Cross-section view of the FIB hole

Figure 5.1: Illustration of an FIB hole.

the more power of current can accelerate the process of surface milling, it creates larger angles and needs more clean space above the target signal.

- **Top windows:** the area for the top of the FIB probing hole. Due to the aspect ratio, the size of top windows is always larger than the baseline window, and we need larger size at the top if we have to observe the signal in the deeper layers. However, it is difficult to have a signal at deeper layer without any metal above it. Even if we use the lower power of FIB current, the probability to observe signals at deeper layers is much lower than they are on higher layers.

### 5.2.2 Definition of an FIB Observable Signal

In the previous discussion, we briefly describe the FIB holes for successful milling to target signals. A signal in the design means a net in the netlist, and it may correspond to metal lines in different routing layers with via connections. We give the precisely definition about observable signals as following.

The signal can be observed with FIB techniques if it has the following features:

1. The FIB hole can be formed and reach the surface of target line with the given parameters, such as baseline window, aspect ratio, and top window.

2. The FIB hole cannot cut any existing metal line.

3. The overlap between the surface of target line and baseline window is larger than given minimal width according to FIB techniques (800-1000nm for current technology).

4. There is only target metal in the middle of baseline window.

### 5.2.3 Problem Formulation

According to the discussion and the definition about FIB techniques, we have the idea if we could reserve more signals in the higher layers, we could have more observable signals. Thus, the problem of our layout modification for FIB observation can be defined as following:

**INPUT**

- The layout files after placement and routing
- The library files for target technology
- The parameter of FIB techniques

**OUTPUT**

- The modified layout files
- The signals and their feasible locations for FIB observation

**OBJECTIVE**

- Maximize the number of signals for FIB observation

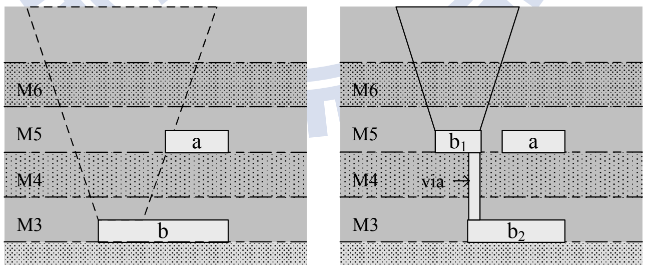
## 5.3 Layout Modification for FIB Techniques

In this section, we first describe several pre-defined operations for layout modification. Then, we present the ranking method for potentially observable signals. Finally, we detail flow about our layout modification for FIB observation.

### 5.3.1 Basic Operations

To make signal be observable, we propose three operations and describe each operation in the following. Moreover, the operations we proposed only focus on changing routing layers of signals to minimize the impact to routed signals.

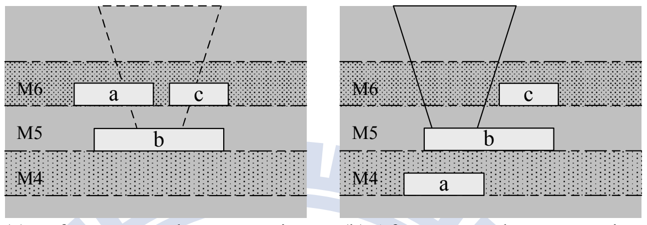
- **Move-up operation:** moving up the segment of target line.
- **Move-down operation:** moving down the segment of target line.
- **Swapping operation:** the combination of Move-up and Move-down operations.



(a) Before a move-up operation (b) After a move-up operation

Figure 5.2: (a) Signal $b$ cannot be observed. (b) Signal $b$ can be observed through segment $b_1$.

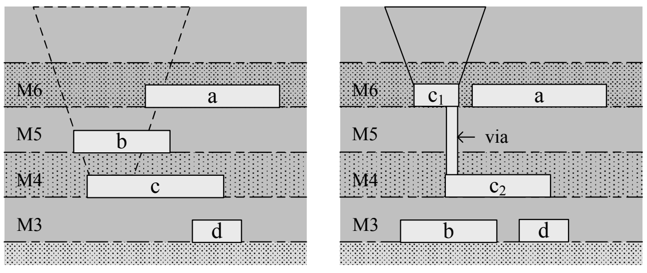
The first operation, Move-up operation, is moving up the target signal to higher routing layer. To minimize the impact to original routing, we only move a part of line segment for target signal. For higher successful FIB rate, the length of the segment we moved needs to exceed the minimum width of baseline window. We use Figure 5.2 for illustration. First, we check whether there is enough space for via connections and the new segment on the higher layer. If there is enough space, we divide the target signal, move one segment of target signal to the higher layer for observation, and maintain the connection with additional vias.



(a) Before a move-down operation (b) After a move-down operation

Figure 5.3: (a) Signal $b$ cannot be observed. (b) Signal $a$ have been moved down and the space is released for signal $b$.

Due to timing constraints or no enough space, some signals cannot be moved to upper layers. The second operation, Move-down operation, tries to move down the blockage line to release the space for others. We use Figure 5.3 for illustration. In Figure 5.3(a), the necessary space for observing signal $b$ is blocked by signal $a$ and $c$. We check the enough space below the two signals and, then, move signal $a$ down to release the space for signal $b$. With successful move-down operation, signal $b$ is observable in Figure 5.3(b).



(a) Before swapping operation (b) After swapping operation

Figure 5.4: (a) Signal $c$ cannot be observed. (b) After move-down and move-up operations, signal $c$ is observable.

The last operation is swapping two signals at different layers. This operation combines the first two operations in order to create more space for the lower signals. Figure 5.4 demonstrates the idea of swapping operations. In Figure 5.4(a), the signal $c$ is blocked by signal $a$ and $b$, but there is no space for directly observing signal $c$ after moving down signal $b$. Thus, we divide signal $c$ into $c_1$ and $c_2$. First, we move down the signal $b$ to release the space and move up the segment of $c_1$ to upper layer. With these operations, signal $c$ is observable through segment $c_1$.

With these operations, some of unobservable signals in the original layout become observable. Thus, we use *potentially observable signals* (POSs) for these signals, and the *potentially observable locations* (POLs) for these line segments, which can be observed with FIB after proper movements. However, there are more than thousands of POLs and the priority of these POLs affects the final signal observation rate. For example, POL $l_a$ of signal $a$ and POL $l_b$ of signal $b$ need the same space. If signal $a$ has another POL for observation, the space should be reserved for signal $b$ for maximizing the number of observable signals. As the results, we propose a greedy method for ranking these POLs.

### 5.3.2 Ranking Method for Potentially Observable Signals

First, we define the cost about observing signals from their POLs. The cost of a POL is the number of operations to make it be observable. For example, if the POL only needs one operation to be observable, such as one move-up or move-down operation, its cost is 1. With the cost of each POL, we set the cost for POSs with two criterias and rank POSs with them. The first is the number of *POLs* and the second is the minimum *moving cost* of POLs.

First, the POS with less POLs has less chance to be observable. For example, the POS $a$ with one POL has only one location to become observable. If we adjust other signal first, the location could be blocked by other signal and POS $a$ cannot be observed with any operation. Thus, we first make these signal with less POLs to be observable to maximize the observation rate. Second, if there are signals with the same POLs, we choose the POL which has the minimum moving cost with high priority. With this criteria, the signal with minimum moving cost is adjusted with the minimum routing resource, like additional vias, upper routing metals. We could preserve more space and routing resource for other POSs and maximize the observation rate in the following steps.

```
Algorithm: Ranking for Potentially Observable Signals (POSs)
Input: The layout file, the set of unobservable signals,
       the aspect ratio, and the width of baseline window of FIB.
Output: The ranking list of POSs.
1.   for each unobservable signals
2.     check the free space for target signal.
3.       if target signal is completely blocked
4.         then target signal is unobservable.
5.       else
6.         for each POL of target signal
7.           evaluate the cost of observing the POL with basic operations.
8.           record the minimum cost of target POL.
9.           sort the cost of POLs from min to max.
10.        set the cost of target signal with the min POL
           and the number of POLs.
11.  sort the number of POLs for each POSs from min to max.
12.    if the number of POLs are equal
13.      then the POS with minimum cost first.
14.  return sorted ranking list of POSs.
```

Figure 5.5: The mehod of ranking unobservable signals.

We show the algorithm of ranking unobservable signals at Table 5.5. First, we check whether there is enough space above target signal. If the target signal is completely blocked by others, we mark it to be unobservable (Lines 3-4). If there is any POL for target signal, we evaluate the observing cost with proposed operations and record the minimum cost of this POL (Lines 6-8). After getting the cost of each POL, we sort these POLs according to the cost from minimum to maximum (Line 9). Then, we set the cost of target signal, a new POS, with the number of POLs and the minimum cost of all POLs (Line 10). After we obtain the cost of each POS, we sort these signals according to the number of POLs from minimum to maximum. If two or more POSs have the same POLs, we give the high priority to the POS with minimum cost (Lines 12-13). Finally, we obtain the ranking list of POSs (Line 14).

### 5.3.3 Layout Modification for Signal Observation with FIB Techniques

With the basic operations and the greedy ranking method, we detail the flow of layout modification for signal observation with FIB techniques.

```
Flow: Layout Modification for Signals Observation with FIB Techniques
Input: The original layout file: design.def,
       The FIB parameters file: FIB.para,
       The physical-design information file: tech.lef,
       The gate-level netlist file: netlist.v,
       The timing information file: tech.lib.
Output: A modified layout file: design_new.def,
        A list of observable signals: Obs.list.
1.  Analysis the original layout file to filter the observable signals.
2.  Rank the unobservable signals according to the proposed ranking method.
3.  For each POS in the ranking list
4.     Perform the basic operations to make it be observable if possible.
5.  Perform the the connection checking, design rule checking, and equivalent checking.
6.  Perform the timing analysis.
7.  If there is any path violating timing constraints
8.     Report the modified signals on the timing-violated path and recover them.
9.  Output the modified layout file and the list of observable signals.
```

Figure 5.6: The flow of layout modification for FIB observation.

There are several input files:

- **design.def:** the original layout file of design. It can be generated with any

APR tool.

- **FIB.para:** the parameters about current FIB techniques, including the baseline window and aspect ratio.

- **tech.lef:** the physical-design information for current cell library, including the size and space of each routing layer, the size of vias, and etc.

- **netlist.v:** the original netlist file of design. It describes the gate-level information for equivalence checking with modified layout file.

- **tech.lib:** the timing library for current cell library. It is used in timing analysis to check whether there is any timing violation or not with modified layout file.

There are two output files:

- **design_new.def:** the modified layout file of design.

- **Obs.list:** the list of FIB observable signals, and the POLs of each observable signal.

After getting the input files and parsing the metal lines of signals respect to FIB parameters, the first step is to filter the signals which are already observable in the original layout. Then, we apply the ranking method in previous section to extract the POSs and obtain the ranking list. With the ranking list, the layout modification operations are used to make these POSs become observable if possible. After checking all POSs in the ranking list, we perform several checking to guarantee the modified layout has the same relation as original layout, such connection checking, design rule checking, and equivalence checking. Then, the timing analysis is performed to make sure the modified layout meets the timing constraints. If there is any timing-violated paths, we report the modified signals on the paths and recover these signals for satisfying timing constraints. Finally, the modified layout file is generated, and we also report the list of observable signals and the POLs of each observable signal.

## 5.4 Experimental Results

We implement our layout modification and give the experimental results in this section. The benchmarks we used are the large circuits from ISCAS'89 and ITC'99. The library is set for UMC 90nm and we constrain metal 6 to be the top layer for routing signals. We first synthesis these benchmarks with Synopsys Design Compiler, and the layouts are obtained from the commercial APR tool, Cadence SoC encounter [61]. The SoC encounter outputs the original layout file in DEF format and timing information in SPEF format. We extract the timing information with PrimeTime [62]. The proposed flow is applied to the original layout file and we feedback the modified layout file to encounter for verifying the interconnection and other properties. Finally, we return the verified layout and a list of observable signals signals.

We set the width of baseline window to 1000nm and the aspect ratio of FIB holes to 1-to-10 in our experiments. Table 5.2 demonstrates the experimental results about FIB observation and the cell-utilized rate is 80%, which can present the cell density in the layout. In Table 5.2, Column 1 and 2 show the name of circuits and their total signals. Column 3 and 4 demonstrate the FIB observation rate before and after the proposed flow. Column 5 presents the difference of the Column 3 and 4. The upper bounds of observation rate list in Column 6. The runtime is presented at last column. As the results, our proposed flow for FIB observation significantly improves the rate of observable signals from 39.15% to 69.28% in average with the

Table 5.2: Result of applying the flow of FIB observation.

| circuit | total signals | FIB observable rate (%) initial (a) | modified (b) | imp. (b) - (a) | upper bound | runtime (sec) |
|---|---|---|---|---|---|---|
| s38417 | 9296 | 36.57 | 69.86 | 33.30 | 71.72 | 106 |
| s38584 | 5711 | 28.62 | 64.96 | 36.34 | 67.17 | 66 |
| s35932 | 5912 | 50.84 | 83.12 | 32.28 | 83.85 | 51 |
| b17 | 16826 | 17.86 | 46.71 | 28.85 | 48.90 | 476 |
| b20 | 7183 | 25.62 | 57.48 | 31.87 | 59.36 | 85 |
| b21 | 6448 | 23.47 | 54.12 | 30.65 | 56.54 | 81 |
| b22 | 9432 | 23.56 | 55.45 | 31.89 | 57.96 | 162 |
| avg. | - | 29.50 | 61.67 | 32.17 | 63.64 | - |

basic operations.

Table 5.3: The FIB observation rate with 800nm for width of baseline window.

| circuit | FIB observable rate (%) initial (a) | modified (b) | imp. (b) - (a) | upper bound | runtime (sec) |
|---|---|---|---|---|---|
| s38417 | 58.21 | 76.85 | 18.65 | 77.86 | 74 |
| s38584 | 47.25 | 69.63 | 22.38 | 70.97 | 50 |
| s35932 | 71.78 | 87.02 | 15.24 | 87.48 | 34 |
| b17 | 29.15 | 49.98 | 20.83 | 51.40 | 387 |
| b20 | 40.23 | 61.95 | 21.72 | 63.20 | 67 |
| b21 | 36.54 | 58.18 | 21.63 | 59.73 | 65 |
| b22 | 37.80 | 59.76 | 21.95 | 61.31 | 129 |
| ave | 45.85 | 66.20 | 20.34 | 67.42 | - |

Similar trend to Table 5.2, we demonstrate the efficient of proposed flow with different width of baseline window (800nm) in Table 5.3. It is obviously not only the initial observation rates but also the modified observation rates are higher than the experiments with larger width of baseline window. In the current industry, the two settings for width of baseline windows are all acceptable. However, the successful rate for larger width (1000nm) is higher than smaller one (800nm) and it also need longer process time. Hence, it is a trade-off problem and could be determined with failure-analysis engineers.

Table 5.4: FIB observable rates based on the initial layout with different cell-utilization rates.

| circuit | cell-utilization rates | | | | | |
|---|---|---|---|---|---|---|
| | 80 | | 85 | | 90 | |
| | initial | modified | initial | modified | initial | modified |
| s38417 | 36.57 | 69.86 | 34.04 | 67.76 | 32.95 | 67.55 |
| s38584 | 28.62 | 64.96 | 27.20 | 64.90 | 25.12 | 62.82 |
| s35932 | 50.84 | 83.12 | 47.42 | 80.13 | 43.15 | 77.87 |
| b17 | 17.86 | 46.71 | 16.64 | 43.80 | 16.11 | 42.51 |
| b20 | 25.62 | 57.48 | 23.49 | 53.00 | 22.67 | 50.54 |
| b21 | 23.47 | 54.12 | 23.34 | 52.40 | 19.61 | 50.93 |
| b22 | 23.56 | 55.45 | 21.03 | 51.91 | 20.45 | 52.05 |
| avg. | 29.50 | 61.67 | 27.60 | 59.13 | 25.72 | 57.75 |

In the current designs, the cell-utilization rates are usually set from 80% to 85% and we demonstrate the proposed flow on the initial layout with 3 kinds of cell-utilization rate in this experiment. Table 5.4 presents the experimental results with different rates from 80% to 90%. The initial observation rates become lower from 80% to 90% because it is difficult to observe signals with high-density design. With applying the proposed flow to each circuit, the modified observation rates are all increased in average. These experiments show the proposed flow is very efficient for improving FIB observation rates within different cell-utilization rates.

Table 5.5 lists the most critical path before and after applying our proposed flow. The RC extraction is obtained from encounter [61] and these information is reported from PrimeTime [62]. With the layout modification, there are extra vias added into the circuit and these vias affect the timing of the original circuit. Even with these impacts, the timing of modified circuit becomes a little faster than the initial one for the first two cases.

Table 5.5: The timing of most critical path with FIB observation.

| circuit | initial layout | modified layout |
|---|---|---|
| | critical path (ns) | critical path (ns) |
| s38417 | 1.09575 | 1.09486 |
| s38584 | 1.49767 | 1.49481 |
| s35932 | 1.79810 | 1.79595 |
| b17 | 6.72669 | 6.71904 |
| b20 | 4.26175 | 4.25394 |
| b21 | 4.32446 | 4.31786 |
| b22 | 5.39416 | 5.39174 |

Table 5.6: Timing comparison between MFOB with and without locking the top 50 longest paths.

| circuit | critical path (ns) | | avg. timing diff. for top 50 paths (%) | | FIB observable rate (%) | |
|---|---|---|---|---|---|---|
| | no lock | lock | no lock | lock | no lock | lock |
| s38417 | 1.09486 | 1.09485 | -0.07770 | -0.07887 | 69.86 | 67.22 |
| s38584 | 1.49481 | 1.49698 | -0.10887 | -0.10907 | 64.96 | 64.38 |
| s35932 | 1.79595 | 1.79602 | -0.07860 | -0.07907 | 83.12 | 82.78 |
| b17 | 6.71904 | 6.71894 | -0.10497 | -0.10731 | 46.71 | 45.97 |
| b20 | 4.25394 | 4.25793 | -0.12390 | -0.10775 | 57.48 | 54.40 |
| b21 | 4.31786 | 4.31997 | -0.10199 | -0.05618 | 54.12 | 51.74 |
| b22 | 5.39174 | 5.39194 | -0.12210 | -0.13058 | 55.45 | 53.55 |
| avg. | - | - | -0.10259 | -0.09555 | 61.67 | 60.01 |

To demonstrate the proposed flow with less impact for timing-critical paths, we lock the nets of 50 timing-critical paths and apply the same flow to maximize the signal observation rate. Table 5.6 shows the experimental results. The first column shows the name of circuits. The second and third columns show the timing of most critical path. The fourth and fifth columns demonstrate the average timing difference of these 50 paths in percentage. The last column give the observation rates. The timing difference of unlocking paths is 0.10259% and it decreases to 0.09555% with locking nets but the observation rate also decreases from 61.67% to 60.01%. As the results, the timing impact of proposed flow is very small and we can minimize the timing impact by locking the nets of some desired paths.

After comparing the initial and modified layouts, we conclude two reasons for the faster timing. First, in our flow, we perform more move-up operations than move-down operations. According to the physical-design information file (.lef), the metals at upper layers have smaller capacitance per unit-length, compared to the lower layers. Thus, the overall metal capacitance of modified signals are often smaller than initial one. Second, we often break a long metal line and move a portion of it to upper layers. This action indeed reduces the coupling capacitance from some long paralleled lines, which affect the timing of critical paths. With these conclusions, the overall metal capacitance of paths with modified signals often decrease and, thus, the timing of critical paths is also decrease with it.

## 5.5 Summary

In this chapter, we present the methodology for layout modification at post-routing step in physical design stage. First, we propose the basic operations to modify layout for observing potentially observable signals (POSs) and a ranking method with greedy selection is used for these POSs. Then, we detail the flow for FIB observation. The experimental results demonstrate the effective and efficient about our proposed flow. Within the same size and slightly better timing, the signals for FIB observation are significantly increased than original layout. Besides, the basic operations are simple and do not need any complicated routing system. Thus, the proposed flow can be easily integrated with today's design flow and applied to the layout generated by any commercial APR tool.

# Chapter 6

# Conclusion and Future Work

## 6.1 Conclusion

In this dissertation, we have presented several efficient and effective algorithms to consider different design constraints in physical design stage.

First, the packing scheme with isosceles right triangular blocks within normal floorplanner is developed. With new equations for calculating $y$-coordinates, the proposed scheme could handle floorplan with any block combined with rectangles and isosceles right triangles. Thus, the scheme gives more choices of block shapes and more locations for pin assignment. Further, this scheme reduces more wirelength under the same area usage compared to normal floorplanner and can be quickly integrated with today's design flow. Then, we propose the method to generate the voltage island and reduce the power consumption while meeting performance constraints, which limits the delay of critical nets. Considering the trade-off between power consumption and power routing cost, the proposed method gives the best solutions within the experimental results. Even the performance-constrained blocks are in different voltage islands, our method still has the best solutions and it is flexible and can be extended to different level. Finally, to obtain more signals after chip manufacturing, the framework of layout adjustment for FIB observation and FIB circuit-editing is proposed. This framework uses the pre-defined functions to change the routing layer of signals and greatly increases the signal observation rate. Without complicated routing operations, it can be applied to layout from any commercial APR tools. The framework not only increases the number of observable signals but also remains the same size and give the slight impact to the timing of original designs.

## 6.2 Future Work

There are some improvements could be done in the future:

**The methodology for pin assignment:** The pin assignment is an important issue to reduce the wirelength. In this dissertation, we use normal distribution to assign the pin at the boundary of each extra block. With more accurate methodology for pin assignment, the wirelength for critical nets or non-critical nets could be reduced more effectively and, further, the floorplanning/placement can be improved better than the algorithm with normal distribution method.

**The method for calculating bounding box for performance constraints:** In this dissertation, we increase the entire bounding box according to the block with highest voltage. In order to get more accurate estimation, the calculation of bounding box can be divided into several sub-box according to different voltages. Finally, the bounding box could be the combination of rectangles with different size according to the voltages of different island.

**The flow for circuit editing:** Besides observations, there is one feature for FIB techniques, which is *circuit-editing*. The actions about circuit-editing are most often cutting one existing line and reconnecting it to another location, such as another line or pre-placed sparse cell. By circuit-editing with FIB techniques, we can quickly change the function of silicon chips for reducing the time of failure analysis.

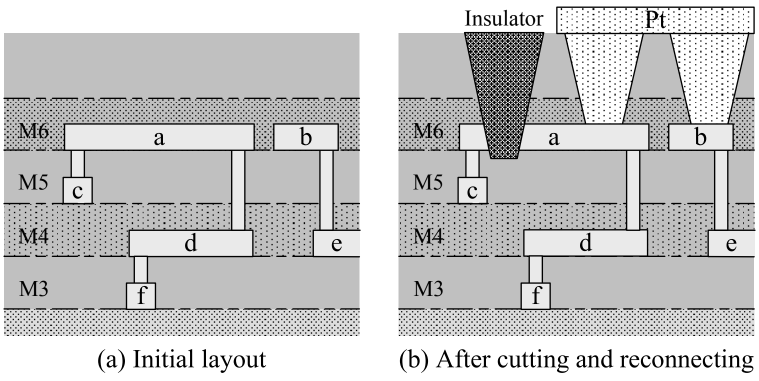


Figure 6.1: (a) The initial layout. (b) Reconnect line $a$ to line $b$ with circuit-editing.

However, there are some different properties compared to FIB observations. First, we only need one location for creating one probing pad, but we have to form two probing pads corresponding to different signals and connect them with extra conductor (usually the same as probing pads) on the surface of chips. Further, we need one more FIB hole for cutting the connection with original signal. Figure 6.1 illustrates an example about circuit-editing. In order to connect line $a$ and $b$, we need two FIB holes with line $a$. One for cutting original connection, and one for reconnecting with line $b$.

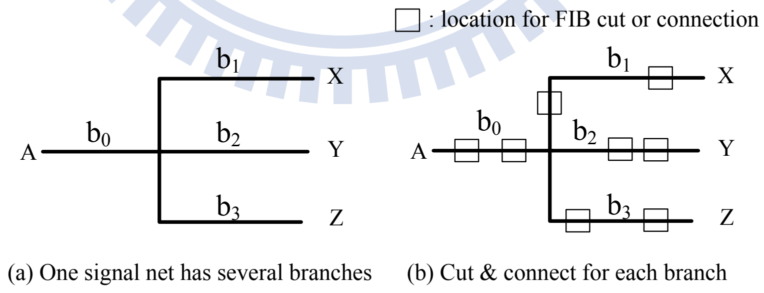

□ : location for FIB cut or connection
A
b0
b1
b2
b3
X
Y
Z
(a) One signal net has several branches
(b) Cut & connect for each branch

Figure 6.2: (a) The observable location could be any branch for signal $A$. (b) The feasible locations for FIB in different branches.

The other property about circuit-editing is shown in Figure 6.2. During FIB observation, the observable location could be at any branch since their source is signal $A$. However, if we want to perform circuit-editing, each branch from $b_0$ to $b_3$

has different meaning and, hence, each branch needs two observable locations for successful circuit-editing. For example, if we need to change the source about all outputs of signal $A$, we could use branch $b_0$ to save the most resource. On the other hand, if we only want to change the input of signal $X$, we just use the branch $b_1$ for cutting and reconnecting.

In order to handle the new properties of circuit-editing, we need to develop new method based on the proposed flow about FIB observation in the future.

# Bibliography

[1] M. T. Abramo and L. L. Hahn, "The application of advanced techniques for complex focused-ion-beam device modification", *Microelectronics Reliability*, pp. 1775-1778, vol. 36, 1996.

[2] M. Abramovici, P. Bradley, K. Dwarakanath, P. Levin, G. Memmi, and D. Miller, "A reconfigurable design-for-debug infrastructure for SoCs", *Design Automation Conference*, pp. 7-12, 2006.

[3] E. Anis and N. Nicolico, "On using lossless compression of debug data in embedded logic analyis", *International Test Conference*, pp. 1-10, 2007.

[4] E. Anis and N. Nicolico, "Low cost debug architecture using lossy compression for silicon debug", *Design, Automation and Test in Europe*, pp. 1-6, 2007.

[5] M. L. Bushnell and V. D. Agrawal, "Essentials of electronic testing," *Kluwer*, Boston, 2000.

[6] J.-A. Carballo, J.L. Burns, S.-M. Yoo, I. Vo, and V.R. Norman, "A semi-custom voltage-island technique and its application to high-speed serial links," *IEEE International Symposium on Low Power Electronics and Design*, pp.60-65, 2003.

[7] Y.-C. Chang, Y.-W. Chang, G.-M. Wu, and S.-W. Wu, "B*-trees: A new representation for non-slicing floorplans," *Design Automation Conference*, pp. 458-463, 2000.

[8] H. Chen, C.-K. Cheng, A.B. Kahng, Ion Măndoiu, and Q. Wang, "Estimation of wirelength reduction for $\lambda$-Geometry vs. Manhattan placement and routing," *International Workshop on System-Level Interconnect Prediction*, pp. 71-76, 2003.

[9] H. Chen, C.-K. Cheng, A.B. Kahng, Ion Măndoiu, and Q. Wang, "The Y-architecture for on-chip interconnect: Analysis and methodology," *International Conference on Computer-Aided Design*, pp. 13-19, 2003.

[10] T.-C. Chen, Y.-L. Chuang, and Y.-W. Chang, "Effective wire models for X-architecture placement," *IEEE Transactions on Computer-Aided-Design of Integrated Circuits and Systems*, pp. 654-658, vol. 27, 2008.

[11] R.L.S. Ching, E.F.Y. Young, K.C.K. Leung, and C. Chu, "Post-placement voltage island generation," *International Conference on Computer-Aided Design*, pp. 641-646, 2006.

[12] P. Coussy, A. Baganne, and E. Martin, "A design methodology for integrating IP into SoC systems," *IEEE International Custom Integrated Circuits Conference*, pp. 307-310, 2002.

[13] Y.S. Dhillon, A.U. Diril, A. Chatterjee, and H.-H. Sean Lee, "Algorithm for achieveing minimum energy consumption in COMS circuits using multiple supply and threshold voltages at the module level," *International Conference on Computer-Aided Design*, pp.693-700, 2003.

[14] P. Goel, "An implicit enumeration algorithm to generate tests for combinational logic circuits," *IEEE Transactions on Computers*, pp. 215-222, vol. C-30, 1981.

[15] R. Goering, "Post-silicon debugging worth a second look", *EETimes*, Feb. 05, 2007.

[16] P.-N. Guo, C.-K. Cheng, and T. Yoshimura, "An O-tree representation of non-slicing floorplans and its applications," *Design Automation Conference*, pp. 268-273, 1999.

91-101, vol. 23, 2004.

[17] J. Hu and Y. Shin and N. Dhanwada and R. Marculescu, "Architecting voltage islands in core-based system-on-a-chip designs," *IEEE International Symposium on Low Power Electronics and Design*, pp. 180-185, 2004.

[18] W.-L. Hung, G.M. Link, Y. Xie, N. Vijaykrishnan, N. Dhanwada, and J. Conner, "Temperature-aware voltage island architecting in system-on-chip design," *International Conference on Computer Design*, pp. 689-694, 2005.

[19] Y.-C. Hsu, F. Tsai, W. Jong, and Y.-T. Chang, "Visibliltiy enchancement for silicon debug," *Design Automation Conference*, pp. 13-18, 2006.

[20] W. Hwang, "New trends in low power SoC design technologies," *IEEE SOC Conference*, pp. 422, 2003.

[21] J. Kao, A. Chandrakasan, and D. Antoniadis, "Transistor sizing issues and tool for multi-threshold CMOS technology," *Design Automation Conference*, pp. 409-414, 1997.

[22] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecci, "Optimization by simulated annealing," *Science*, pp. 671-680, 1983.

[23] C.-K. Koh, P.H. Madden, "Manhattan or non-Manhattan?: A study of alternative VLSI routing architectures," *Great Lakes Symposium on VLSI*, pp. 47-52, 2000.

[24] A. Krstic, L.-C. Wang, K.-T. Cheng and T.M. Mak, “Diagnosis-based post-silicon timing validation using statistical tools and methodologies,” *International Test Conference*, pp. 339-348, 2003.

[25] D.E. Lackey, P.S. Zuchowski, T.R. Bednar, D.W. Stout, S.W. Gould, and J.M. Cohn, “Managing power and performance for system-on-chip designs using voltage islands,” *International Conference on Computer-Aided Design*, pp. 195-202, 2002.

[26] W.-P. Lee, H.-Y. Liu, and Y.-W. Chang, “Voltage island aware floorplanning for power and timing optimization,” *International Conference on Computer-Aided Design*, pp. 389-394, 2006.

[27] J. Li, T. Yan, B. Yang, J. Yu, and C. Li, “A packing algorithm for non-Manhattan hexagon/triangle placement design by using an adaptive O-tree representation,” *Design Automation Conference*, pp. 646-651, 2004.

[28] J.-M. Lin and Y.-W. Chang, “TCG: A transisitive closure graph-based representation for non-slicing floorplans,” *Design Automation Conference*”, pp. 764-769, 2001.

[29] J.D. Meindl, “Low power microelectronics: Retrospect and prospect,” *Proceedings of the IEEE*, pp. 619-635, 1995.

[30] H. Murata, K. Fujiyoshi, S. Nakatake, and Y. Kajitani, “VLSI modules placement based on rectangle-packing by the sequence-pair,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, pp. 1519-1524, vol. 15, 1996.

[31] J. Nonaka, “Design for failure analysis by using LVP measurement elements”, *Semi Technology Symposium*, pp. 45-48, 2003.

[32] J. Nonaka, T. Ishiyama, and K. Shigeta, “Design for failure analysis inserting replacement-type observation points for LVP”, *International Testing Conference*, pp. 1-10, 2009.

[33] K. Nose andT Sakurai, “Optimization of Vdd and Vth for low-power and high-speed applications,” *Design Automation Conference*, pp. 25-28, 2000.

[34] R.H.J.M. Otten, “Automatic floorplan design,” *Design Automation Conference*, pp. 261-267, 1982.

[35] M. Paluszewski, P. Winter, and M. Zachariasen, “A new paradigm for general architecture routing,” *Great Lakes Symposium on VLSI*, pp. 202-207, 2004.

[36] S. Pateras, “Embedded diagnosis IP,” *Design, Automation and Test in Europe*, pp. 242-243, 2002.

[37] S. Pateras, “IP for embedded diagnosis,” *IEEE Design & Test of Computers*, pp. 44-53, vol. 19, 2002.

[38] M. Pedram and J. Rabaey, "Power aware design methodologies," *Kluwer Academic Publishers*, 2002.

[39] R. Schlangen, R. Leihkauf, U. Kerst, C. Boit, and B. Kruger, "Functional IC analysis through chip backside with nano scale resolution - E-beam probing in FIB trenches to STI level", *International Symposium on the Physical and Failure Analysis of Integrated Circuits*, pp. 35-38, 2007.

[40] D. C. Shaver and B. W. Ward, "Integrated circuit diagnosis using foucused ion beams", *Journal of Vacuum Science & Technology B; Microelectronics and Nanometer Structures*, pp. 185-188, vol. 4, 1986.

[41] C. Shawn, C. C. Tsao, and T. R. Lundquist, "Measuring back-side voltage of an integrated circuit", *U.S. Patent 6,872,581 B2*, 2005.

[42] N. Sridhar and M.S. Hsiao, "On efficient error diagnosis of digital circuits," *International Test Conference*, pp. 678-687, 2001.

[43] C. G. Talbot, M. Park, N. Richardson, P. Alto, and D. Masnaghetti, "IC modification with foucused ion beam system", *U.S. patent 5,140,164*, 1992.

[44] X. Tang and D.F. Wong, "Floorplanning with alignment and performance constraints," *Design Automation Conference*, pp. 848-853, 2002.

[45] S.L. Teig, "The X architecture: Not your father's diagonal wiring," *International Workshop on System-Level Interconnect Prediction*, pp. 33-37, 2002.

[46] S. Venkataraman and S.B. Drummonds, "POIROT: Applications of a logic fault diagnosis tool," *IEEE Design & Test of Computers*, pp. 19-30, vol. 18, 2001.

[47] B. Vermeulen, C. Hora, B. Kruseman, E.J. Marinissen, and R. van Rijsinge, "Trends in testing integrated circuits," *International Test Conference*, pp. 688-697, 2004.

[48] B. Vermeulen, "Design-for-debug to address next-generation SoC debug concerns," *International Test Conference*, pp. 1, 2007.

[49] B. Vermeulen, S. Oostdijk, and F. Bouwman, "Test and debug strategy of the PNX8525 Nexperia$^{TM}$ digital video platform system chip," *International Test Conference*, pp. 121-130, 2001.

[50] A. Vorg and M. Radetzki and W. Rosenstiel, "Measurement of IP qualification costs and benefits," *Design, Automation and Test in Europe*, pp. 996-1001, 2004.

[51] L. Wei, Z. Chen, K. Roy, M.C. Johnson, Y. Ye, and V.K. De, "Design and optimization of dual-threshold circuits for low-voltage low-power applications," *IEEE Transactions on Very Large Scale Integration Systems*, pp. 16-24, March, 1999.

[52] D.F. Wong and C.L. Liu, "A new algorithm for floorplan design," *Design Automation Conference*, pp. 101-107, 1986.

[53] G.-M. Wu and Y.-C. Chang and Y.-W. Chang, "Rectilinear block placement using B*-trees," *ACM Transactions on Design Automation of Electronic Systems*, pp.188-202, vol. 8, 2003.

[54] H. Wu and I-M. Liu and M.D.F. Wong and Y. Wang, "Post-placement voltage island generation under performance requirement," *International Conference on Computer-Aided Design*, pp.309-316, 2005.

[55] M.-C. Wu and Y.-W. Chang, "Placement with alignment and performance constraints using the B*-tree representation," *International Conference on Computer Design*, pp. 568-571, 2004.

[56] Y.-R. Wu, S.-Y. Kao, and S.-A. Hwang, "Minimizing ECO routing for FIB", *VLSI Design Automation and Test*, pp. 351-354, 2010.

[57] J.-S. Yang and Nur A. Touba, "Expanding trace buffer bbservation window for in-system silicon debug through Selective Capture", *VLSI Tset Symposium*, pp. 345-351, 2008.

[58] W.-M. Yee, M. Paniicia, T. Eiles, and V. Rao, "Laser Voltage Probe (LVP): A novel optical probing technology for flip-chip package microprocessors", *Internation Symposium on the Physical and Failure Analysis of Integrated Circuits*, pp. 15-20, 1999.

[59] C.-C. Yen, T. Lin, H. Lin, K. Yang, T. Liu, and Y.-C. Hsu, "Diagnosing silicon failures based on functional test patterns," *International Workshop on Microprocessor Test and Verification*, pp. 94-98, 2006.

[60] FEI Company, "Focused ion beam technology, capabilites and applications", http://www.fei.com.

[61] Cadence, "Encounder®User Guide," version 8.1.2, 2009.

[62] Synopsys, PrimeTime, version B-2008.12-SP3-2, 2009.