Placement Techniques in Automatic Analog Layout Generation

CUI, Guxin

A Thesis Submitted in Partial Fulfilment

of the Requirements for the Degree of

Master of Philosophy

in

Computer Science and Engineering

The Chinese University of Hong Kong

September 2012

Abstract

Analog layout design is a complicated and time-consuming process. It often takes couples of weeks for the layout designers to generate a qualified layout. The electrical properties of analog circuit are very sensitive to the layout details, and mismatch reduction becomes a very important issue in analog layout design.

In this thesis, we will present some practical placement techniques to reduce mismatch and improve routability. These techniques can be easily integrated into a complete analog placement and routing flow, which can produce in just a few minutes a complete and high quality layout for analog circuits that passes the design rule check, layout-schematic check and with performance verified by simulations. The contents of this thesis will focus on the following two issues:

(1) Symmetry Placement: We consider symmetric placement of transistors, resistors and capacitors, which includes 1-D symmetry and 2-D symmetry (or called common centroid). Different symmetric placement configurations, derived according to the practical needs in analog design, are considered for the matching devices in the simulated annealing engine of the placer in order to generate a placement with high quality.

(2) Congestion-driven Placement: In analog design, wires are preferred not be

routed over active devices, so we need to leave enough spaces properly for routing between the devices during the placement process. To achieve this, we explore congestion estimation and layout expansion during the placement step in order to produce a good and routable solution.

In order to verify the quality of the generated layouts in terms of mismatch, we will run Monte Carlo simulations on them with variations in process and mismatch. Experiments show that our methodology can generate high quality layout automatically in just a few minutes while manual design may take couples of days.

摘要

模擬電路版圖設計是一個非常複雜和耗時的過程。通常情況下，設計一個高質量的模擬電路版圖需要電子工程師花費幾週甚至更長的時間。模擬電路的電子特性對於電路的細節設計非常敏感，因此，減小電路中的失配現象成為模擬電路版圖設計中一個非常重要的課題。

在本論文中，我們提出了一系列實際的佈局技術，來降低電路的失配並提高繞線的成功率。我們可以非常容易的將這些技術整合至一個完整的模擬佈局和佈線的工具中，此工具可以在幾分鐘內生成一個完整的、高質量的模擬電路版圖。同時，該版圖能夠通過設計規則驗證（DRC）和佈局與電路設計一致性檢測（LVS）。模擬結果顯示，它的電路性能能夠與達到甚至超出手工設計的電路版圖。我們的論文主要作出了以下兩方面貢獻。

1. 平衡佈局：對於模擬電路中的電子元器件，如電容、電阻、晶體管等進行一維和二維的平衡佈局。電子工程師可以根據不同的設計需求，通過選擇不同的佈局參數來改變電路的佈局排列方式。同時，在模擬退火算法中，我們著重考慮了器件間的匹配以生成高質量的模擬電路佈局。

2. 消除阻塞的電路佈局：在模擬電路設計中，我們期望盡量避免在電子元器件密度較高的區域進行繞線。因此，我們需要在電路佈局設計過程中在電子元器件間留有足夠的佈線空間。為達到這個目標，我們提出了更精確的阻塞估計方法和版圖拓展方法，使其能夠生成一個高質量、高繞線成功率的電路佈局結果。

為了驗證生成的電路版圖的質量和匹配特性，我們利用蒙地卡羅方法來模擬電路中的製程偏差和失配特性。實驗結果顯示，我們的工具可以在幾分鐘內自動生成高質量的電路版圖，與人工設計通常需要花費數日至數週相比，設計時間大幅縮短，同時電路的匹配特性得以提升。

Acknowledgement

First and foremost, I offer my sincerest gratitude and heartily thankful to my supervisor, Professor Evangeline Fung Yu YOUNG, who gives me most helpful suggestions, comments, guidance and support during my study in CUHK and thesis writing process. From the beginning of my research to the final level of completing thesis enabled me to develop an understanding of my research topic. She has made available her fully support in a number of ways, motivated me for the initial idea generation, methodologies discussion and experiment design. I attribute the level of my MPhil degree to her encouragement and supervision. Without her inspirational guidance, this thesis, would not have been completed successfully. One simply could not wish for a better or more nice supervisor.

Also, I would like to show my gratitude to my committee members Professor Johnny Qiang XU, and Professor Kong Pang PUN, for their encouragement, insightful comments, and contributive questions. Without your knowledge and effort, this study would not have been successful.

My sincere thanks also goes to my fellow lab-mates in CUHK, Dr. Xiao LIU, Dr. Xiao-Qing YANG, Dr. Hai-Le YU, Mr. Feng YUAN, Mr. Rong YE, Mr. Li JIANG, for the sleepless nights we were working and fighting together before deadlines. I offer my regards and blessings to all of my friends who supported me in any respect during the completion of my study, Dr. Fei FEI, Dr. Jian-Zhen LI, Mr. Jian-Peng WANG, Mr. Xian-Shuai CHEN, Mme. Xue-Li LI and Ms. Jing-

Jing LIU, do not forgetting that your best friends who always been there. Also, please accept my apology that I could not mention all of you personally.

The Department of Computer Science and Engineering has provided the equipment and environment I have needed to complete my thesis.

Last but not the least, I would like to thank my family, for supporting me throughout all my life and studies.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

Accompanied with the speed-up of the technology development, *Integrated Circuits* (IC) are used in most of the modern electronic equipments today and have revolutionized the world of electronics. New technology allows us to integrate a large number of devices onto a small chip.

Nowadays, the IC technology virtually created our modern IT industry. Transistors existed in almost every electronic device, from memory sticks to smartphones to computers. In the early period, engineers can only placed a few number of transistors on a chip. Now we are at the stage of "Very Large-Scale Integration (VLSI)" with several billion of transistors per chip, which has been developed for more than 30 years. The design complexity has sharply increased, accompanied with the increase in the number of devices per chip. *Electronic Design Automation (EDA)* becomes more and more important in this modern layout design process because of the high complexity of IC. A lot of new challenges appear in modern VLSI design automation.

Chip design is a complicated and time-consuming process. It contains several

typical steps. The whole design process begins with System Specification, then Architectural Design, Functional Design, Logic Design, Circuit Design, Physical Design, Physical Verification, Fabrication, Packaging and Testing. Physical Design includes steps such as Partitioning, Chip Planning, Placement, Clock Tree Synthesis, Signal Routing and Timing Closure. Our research focus on analog placement is an important step in the physical design process.

An overview of the chip design flow will be shown and discussed briefly in the following sections. Detail discussion on the analog placement problem will be discussed in this thesis.

1.2 Physical Design

Physical design is an important step in the design cycle of integrated circuits, which will map logic to physical implementation. In this stage, the description of a circuit will be converted into a geometric representation or a layout. In a layout, detail geometrical information of components, such as devices locations, wiring and shapes will be specified. Figure 1.1 shows a complete design flow and detail placement design stages:

Placement an important step of physical design. Netlist is generated from the synthesis process. After synthesis, RTL design will be transformed to gate-level netlist, which contains detail information about the cells, such as the used cell, cell area, shape, interconnections between cells and other detail information.

Floorplans are generated from the floorplanning stage of physical design. A floorplan is to plan the positions and shapes of modules at the beginning of design cycle. During this stage, blocks are roughly placed within the chip to achieve a minimize circuit area, according to circuit functional specification and some other constraints. But the details of shapes and I/O pins positions are still not known.

Placement is an essential and important step in EDA. During this process, the

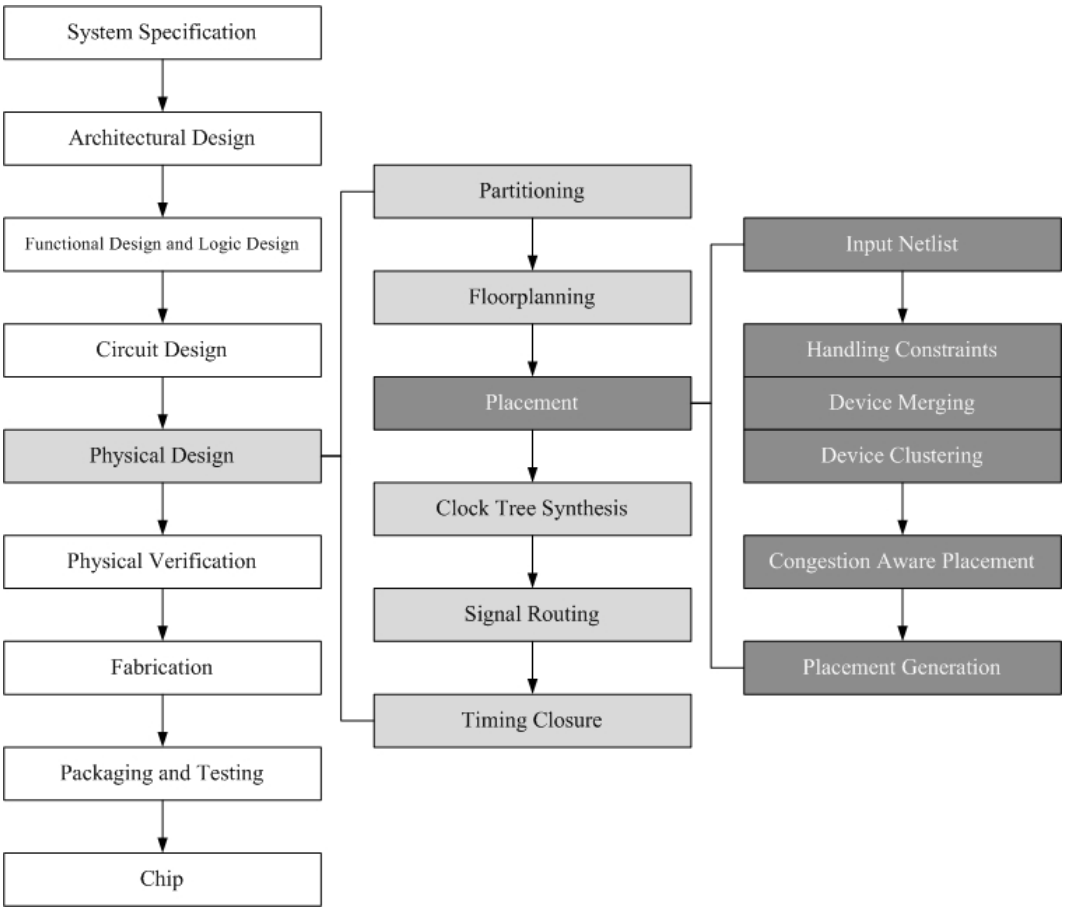


Figure 1.1: Complete Circuit Design Flow.

various circuit components are considered as blocks, and each block is exactly assigned a position. It can achieve the objective that finding a minimum circuit area while ensuring meets the performance demands. A typical placement has two steps, *initial placement* and *iterative improvement*. In addition, most of the placement algorithm will do routing space estimation to further improve the routability. When placement is completed, routing will connect all the pins on placed circuit components. There are two typical phases of routing, named *global routing* and *detailed routing*. Sometimes, *rip-up and reroute* will be used when some connections failed.

Finally, compaction, extraction and verification are performed to get final result.

1.3 Analog Placement

Placement is the step after floorplanning and it plays a key role in the physical design cycle. A good placement solution can lead to smaller circuit area, shorter wirelength, and a better performance of the circuit. A well-placed layout can also lead to high routability and good routing quality. The placement stage is a key step to determine the quality of the final layout.

In recent years, placement becomes an active research topic. There are many publications and researchers focusing on wirelength and area minimization, handling timing, routability, power, delay and synthesis issues. During placement, issues such as interconnect, delay, noise and routability should be considered. The complexity of the placement problem becomes extremely higher than that before with the expanded circuit size.

1.3.1 Methodologies of Analog Placement

As mentioned before, the placement stage is one of the most important stages in VLSI physical design, and the placement result has a key effect on the quality of the final layout solution. As a designer, one of the objectives of placement is to achieve a well placed solution with minimum area, but in analog placement, the analog-specific features should also be taken into account to meet the demands of performance and electrical characteristics.

In the early days, analog placement technique began with *constructive placement*. A constructive technique chooses modules one by one, and places them in the most suitable available positions. A constructive placement technique is the

most straightforward methodology. Its quality is highly dependent on the device placement order, and the placement is not considered globally. Although it is very fast, low quality layouts will be generated especially when the number of modules is larger.

Besides constructive placements, other techniques based on *branch-and-bound* were proposed. The branch-and-bound methodology enumerates all possible solutions and try to find the best solution within the solution space. Similar to the constructive placement technique, this branch-and-bound approach cannot handle circuits with large number of modules. More recently, *stochastic algorithm* is used to solve the analog placement problem. Simulated annealing and genetic algorithm are the most commonly used techniques. They are more efficient on solving analog placement problem with larger number of modules. Local minima can be avoided by employing those stochastic hill-climbing techniques.

1.3.2 Symmetry Constraints of Analog Placement

Symmetry constraints in placement is required in analog design to obtain high-quality and high-performance analog circuit. It can reduce the parasitics effect between two matched groups, and benefit symmetry routing in the next stage. If two devices fail to match with each other, harmful electrical effects might be introduced. For example, mismatch in differential circuit will lead to degraded Common-Mode Rejection Ratio (CMRR), deduction in Power-Supply Rejection Ratio (PSRR) and high offset voltage, which will cause negative effects to circuit performance. Symmetry placement can help to reduce the thermal gradient sensitivity of a circuit. A better thermal-balanced placement can avoid oscillations and reduce mismatch.

Usually, in analog placement, we have symmetry constraints to place devices symmetrically. Two major types of symmetry constraints are: (1) 1-D symmetry:

place devices symmetrically w.r.t. to an x- or y- axis. Devices placed on 1-D symmetry axis and share the same symmetry axis with other symmetry devices often called self-symmetry devices. (2) common centroid placement: devices are placed symmetrically w.r.t. a single point (common centroid). In most cases, a circuit will have symmetric devices and asymmetric devices, simultaneously. Figure 1.2 shows these different symmetry types:

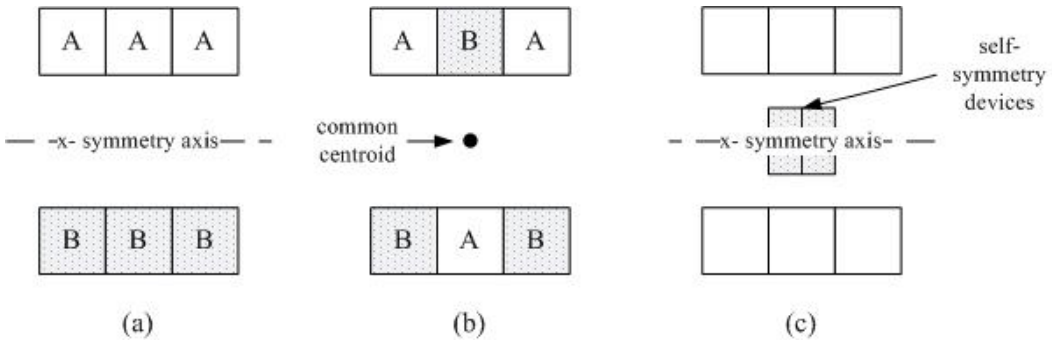


Figure 1.2: Types of Symmetry Devices: (1) 1-D Symmetry; (2) Common Centroid Placement; (3) 1-D Symmetry with Self-symmetry device.

1.4 Process Variation and Layout Mismatch

Matching is one of the most important issues when doing analog IC design. Device mismatches can cause random variations. It can lead to harmful electrical effects, such as parasitic effect, unexpected noise and behavioral variations, etc. These random variations generated from circuit mismatch are time-independent and it's very important to analog, digital and mixed-signal circuit.

1.4.1 Process Variation

Process variation in analog IC design can lead to device mismatch, which can be harmful for the performance of the layout. We denote the process variation pa-

rameter by PV. The parameter PV is the summation of three parameters, *nominal parameter*, *random parameter* and *systematic parameter*.

$$PV = PV_n + PV_r + PV_s; \tag{1.1}$$

The variables PV_n, PV_r and PV_s denote the nominal value, random variation value and the systematic variation value, respectively. Also, we can further classify them into lot-to-lot, wafer-to-wafer, die-to-die and device-to-device, according to the different effective area. The classification is illustrated in Figure 1.3:

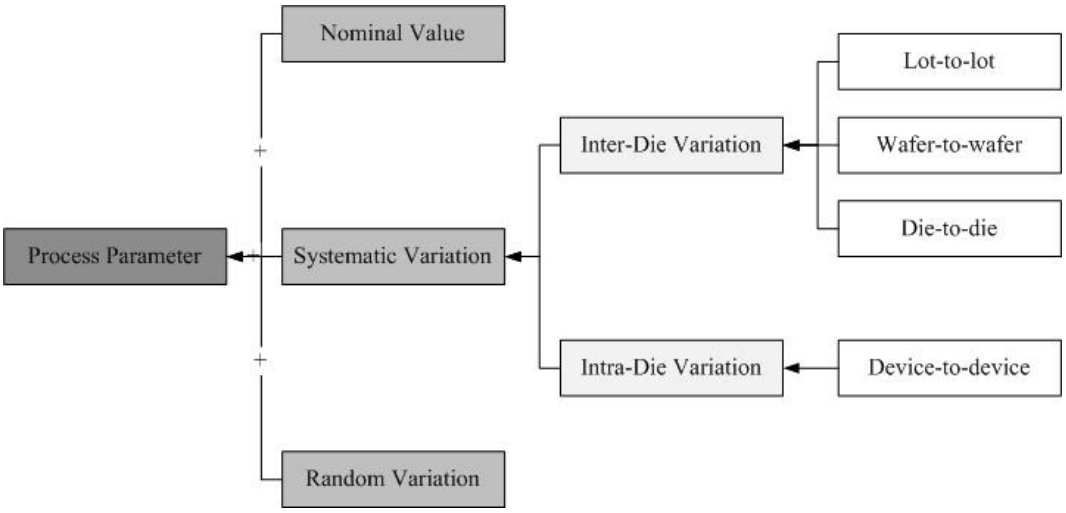


Figure 1.3: Classification of Different Variation Types.

1.4.2 Random Mismatch and Systematic Mismatch

Device mismatches are introduced by process variation. Two major mismatch types are well studied, the *random mismatch* and *systematic mismatch*. In most practical circuit design environment, we can find a combination of both mismatches during the design process. Suppose that two devices (A,B) that need to match with each other have own parameter P_A and P_B, then the mismatch ΔP between device A and B can be calculated by following equation:

$$\Delta P = P_A - P_B; \tag{1.2}$$

We can calculate a series of different values of ΔP and get the mean and standard deviation, denoted by $m(\Delta P)$ and $s(\Delta P)$, respectively. Then $m(\Delta P)$ is called the systematic mismatch, and $s(\Delta P)$ is called the random mismatch.

Random mismatches are mainly due to process variations during the manufacturing process. Some other factors include: edge effects, imperfection of materials and mobility variation. With random mismatch, a circuit might get performance degradation and other electrical errors which may be harmful to the circuit. In general, random mismatch follows some statistical distribution (e.g., gaussian distribution) and leads to random manufacturing inaccuracy. Different device types have different circuit parameters to model the random mismatch. We can find a random mismatch model of regular device as follows:

$$s(\Delta P) = \frac{k_x}{\sqrt{wl}}; \tag{1.3}$$

where $s(\Delta P)$ is the random mismatch, w and l denote the width and length of devices' active area, respectively. The parameter k represents the characteristics of the particular device. Since random mismatch is introduced by imperfect manufacturing process, it cannot be eliminated. According to Equation (1.3), we can proportionally enlarge the area of a device to reduce the random mismatch, it will increase the area usage and conflict with area minimization. It's thus a tradeoff in analog circuit design.

Two major reasons for systematic mismatch are: (1) circuit gradients and (2) asymmetry circuit design. In order to reduce systematic mismatches, common-centroid placement technique is introduced. In real circuit design, parameters such as temperature, pressure and oxide thickness will result in systematic mismatch. Therefore, experienced engineers often divide devices (such as resistors group,

capacitors array) into identical small pieces and place them closely and symmetrically. In Section 3.3, we will propose our method to generate placement with common-centroid constraints.

1.5 Monte Carlo Simulation Procedure

We use Monte Carlo simulations to verify the performance of the generated layout. To do Monte-Carlo simulation, we first get the positions of devices. Next, generate random parameters for width and length to estimate deviation of width w and length l. Then we calculate the mean value of k_x in Equation (1.3) to estimate the systematic mismatch, and obtain the $s(\Delta P)$ to estimate the random mismatch. Finally, we compare these Monte Carlo simulation results with standard simulation results to verify the change in performance and the quality of layout.

1.6 Problem Formulation of Placement

The input of an analog placement problem includes a netlist, a layout library and a set of constraints. The output will be a layout including detailed information of the block positions satisfying all the constraints. Most of previous works take area and wirelength minimization as major objectives. Our placer emphasizes on the quality of analog placement, aiming at generating a high-quality analog placement. The quality can be measured by the factors shown below:

- Circuit Performance, e.g., matching property, parasitics effect, thermal gradient sensitivity, etc;
- Layout Area;
- Total Wirelength;

- Routability;

We are given a set of n blocks B_i, where $i \in [1 \dots n]$. The height and width of each block B_i are denoted by h_i and w_i, respectively. We are given m nets, a net N_i shows interconnection relationship between blocks, where $i \in [1 \dots m]$. Wirelength L is the total length of all connecting wires, estimated by Half-Perimeter Wire Length (HPWL) model. We use A define the circuit area, which is the total area of layout, including white space and space occupied by blocks and wires.

The placement problem is to find a solution that each block B_i is assigned detailed height and width. In the meantime, we should ensure that there is no overlapping between each pair of blocks.

1. Minimize the total area A and wirelength L;
2. All the nets are routable;
3. Place symmetric devices symmetrically to a 1-D symmetry axis;
4. Use common-centroid placement to achieve better matching characteristics.

1.7 Motivations

The advancement in mixed-signal designs has brought the need of new tools compatible for both the digital parts and the analog parts. However, design automation in the analog domain is not moving as fast as its digital counterpart, and the low acceptance of CAD tools in analog design has presented a serious bottleneck to fast automation of mixed-signal systems. Analog layout design is much more complicated than digital one because of the high sensitivity of the electrical properties in analog circuit. Variations in process and temperature can easily introduce severe mismatch in devices which are designed to have identical performance, and

can cause serious degradation in circuit performance [**?**] [**?**]. Layout design of analog circuit is usually a time-consuming and error prone process. Mismatch in analog circuit can be alleviated by having a symmetric layout, with symmetric and close proximity placement of some devices and symmetric routing of related wires [**?**] [**?**]. Therefore, our research focuses on improving the quality of analog layout. Common centroid placement is studied to reduce device mismatch, and routability is handled by introducing new congestion estimation and layout expansion technique.

1.8 Contributions

The objective of this research is to study and develop a novel and holistic analog automation tool that can automate the whole layout process from a schematic circuit to the final layout effectively. Focuses will be on the quality of the generated output in comparison with manual design on various aspects of performance measures and especially on mismatch reduction, which is an important issue in analog design. The whole project is a collaboration with another PhD student [**?**] in the same department. My major contributions are two important features in our design tool:

Symmetry Placement: Our placement tool considers symmetry of transistors, resistors and capacitors specifically. General 1-D symmetry and 2-D symmetry constraints are already well handled by our tool. Our tool can generate layouts with different placement patterns. We can change the device array (i.e., capacitor array)'s shape by modify the row and column number, to achieve better matching properties of layout. Orientation of device can be changed to fit the symmetric routing and layout design requirements. Two placement options (symmetry placement and interleaving placement) are of-

fered by out tool. We discuss and compare different placement patterns in details, and show how they affect routing and the parasitic mismatch.

Congestion-driven Placement: Routability is also an important issue in analog placement. It is one important criterion to measure the quality of an analog layout. We should leave enough spaces for wires in suitable locations during the placement process. We use a two stages approach to improve the routability of a layout. First, we employ a congestion estimation method to predict congestion. Secondly, we perform layout expansion according to the congestion estimation to improve routability. Experimental results show that the layout will have better routability using this approach.

1.9 Thesis Organization

The remaining part of this thesis is organized as follows. First, we give literal review of analog placement in Chapter 2. We will then present our common centroid analog placement techniques in Chapter 3. Chapter 3 also covers congestion estimation and layout expansion to further improve the routabiliy of the layout. In Chapter 4, we use Monte Carlo simulations to measure the matching properties of the devices, and compare the performance with previous works and manual designs. Finally, Chapter 5 concludes this thesis.

□ End of chapter.

Chapter 2

Literature Review on Analog Placement

Two major considerations for analog layout design are symmetry constraint and relatively regular structure, which guarantee the performance and quality of analog layout. Designer apply symmetry constraints during layout design, place matched components group(s) symmetrically to reduce mismatch and other harmful electrical effect (e. g.: parasitic effect, process variation). Many previous work focus on applying topological representation to handle analog placement problem. Symmetry placement constraints such as 1-D symmetry, 2-D symmetry (as known as common-centroid constraint) imposed on analog layout design, to further enhance the quality of generated analog layout. A relatively regular structure together with symmetry routing can increase the routability and the quality of routing, which, also help reduce the mismatch and other unwanted electrical effects.

Some popular floorplanning representations have been extended to consider 1-D and 2-D symmetry constraints. Among these work, different types of floorplanning representations are embedded in simulated annealing algorithms to generate floorplan while handling symmetry constraints simultaneously. Two major

representations are well studied and applied in both academic and industry:

(1) Absolute representations [**?**] [**?**] [**?**] [**?**] are the most straightforward methodology, for the absolute representations, modules are represented by its *coordinates*, *width* and *height*;

(2) Topological representations, use different methodologies encode the placement into a series of codes, which show an one-on-one matching of placement configuration, simulated annealing algorithm is used to find the topological representation with lowest cost, which is controlled by cost function. There exist a lot of popular topological representations, such as Sequence-Pair [**?**] [**?**], O-Tree [**?**] [**?**], B-Tree [**?**], B*-Tree [**?**], Segment Tree [**?**], and TCG-S [**?**].

2.1 Topological Representations Handling Symmetry Constraints

2.1.1 Symmetry within the Sequence-Pair (SP) Representation

In the paper [**?**], techniques based on sequence-pair representations are used to handle symmetry constraints. The authors discussed detail algorithm that do symmetry placement by using sequence-pair and simulated annealing.

Murata *et al* [**?**] proposed the *Sequence-Pair* representation, which can efficiently encode a non-slicing floorplan. Suppose that we are now given a non-slicing floorplans, with a number of rectangular modules of different sizes. Sequence-pair is a pair of module sequences including the following information: module names, direction information (above, below, to be left of and to be right of). Each sequence-pair represents a particular non-overlap packing of the rectangles. The solution space of sequence-pair has been proven as finite.

Balasa *et al* made some changes to the searching space of SP in order to en-

hance the efficiency. Let m denote the number of modules, p denote the number of symmetry pairs and s denote the number of self-symmetry groups. If there are no symmetry constraints, the searching space is $(m!)^2$. The authors have proved that the size of the solution space with symmetry constraints is $(C_m^{m-2p-s}(m-2p-s)!)^2 \cdot (2p-s)!$. We can find that the searching space with symmetry constraints is smaller than that without symmetry constraints. The approach in the paper will only explore the sequence-pairs which satisfy the symmetry constraints.

Symmetric-Feasible Sequence Pairs

Next, Balasa *et al* showed how to recognize the sequence-pairs satisfying the symmetry constraints. Let (α, β) denote a sequence-pair, and variables x and y denote different cells. Variable $sym(x)$ denotes the symmetric counterpart of cell x, and α_x^{-1} denotes the position of x in sequence α. If a cell x is a self-symmetry cell, $sym(x)$ is equal to x itself. Then, a sequence-pair that satisfies $\alpha_x^{-1} < \alpha_y^{-1} \Leftrightarrow \beta_{sym(y)}^{-1} < \beta_{sym(x)}^{-1}$ is called a symmetric-feasible sequence-pair. The authors proved that there exists at least one symmetric-feasible sequence-pair for any placement containing symmetry group(s).

Derive Optimal Solution from Symmetric-Feasible Sequence Pair

The authors have also proposed techniques to generate an area optimal placement based on a sequence-pair with symmetry constraints. Take vertical symmetry as an example. Let x_i and y_i denote the x-coordinate and y-coordinate of cell i respectively. Let $width_i$ and $height_i$ be the width and height of cell i. First, the authors calculate the x-coordinates of all the cells according to the sequence-pair. Then, they use a *sweep to right* methodology to recompute the positions of the cells on the right side of the vertical symmetry axis. Finally, a similar step called *sweep to left* is used to fix the positions of the cells on the left side of symmetry axis.

Experimental results show that the authors can achieve better performance than their previous works. It shows that sequence pair can handle symmetry constraints very effectively.

2.1.2 Block Placement with Symmetry Constraints Based on the O-Tree Non-Slicing Representation

In the paper [**?**], the authors handle symmetry constraints based on the Ordered Tree (O-tree) representation [**?**]. According to the previous work, O-tree has advantages that: (1) It uses a small number of bits to encode a floorplan. Given a floorplan containing n rectangles, only $n(2+\lceil lgn \rceil)$ bits are used. (2) It uses linear time to generate a placement. By using the O-Tree representation, the 1-D and 2-D symmetry constraints can be handled efficiently and qualified layout can be generated in a shorter time.

In order to take symmetry constraints into account, two constraints are introduced, namely Horizontal Symmetric Constraint (HSC) and Vertical Symmetric Constraint (VSC). An O-tree that satisfies the HSC and VSC is called a *X-Feasible O-tree* and a *Y-Feasible O-tree*, respectively.

Symmetric X-Feasible O-Tree

Given a pair of symmetry device group (a, a'), a tree that satisfies the Horizontal Symmetric Constraint, i. e., $x_a = x_{a'}$, be named as Symmetric X-Feasible O-Trees. A Horizontal Constraint Graph G_h is built to identify the symmetric x-feasibility of a given O-tree. The authors proved that if there are no positive cycles in G_h, O-tree is symmetric x-feasible and a minimum width placement can be generated in $O(n^2)$ time.

Symmetric Y-Feasible O-Tree

Likewise, a Symmetric Y-Feasible O-Tree is a tree that satisfies the Vertical Symmetric Constraint. Suppose that y_s is the y-coordinate of horizontal symmetry axis, and h denotes the height of a device or a device group (a, a'). We can have $h_a = h_{a'}$ and $y_a + y_{a'} + h_a = 2y_s$, that represents the Vertical Symmetric Constraint. The property of Symmetric Y-Feasible O-Tree is similar to that of Symmetric X-Feasible O-Tree. When there are no positive cycles in the Vertical Constraint Graph G_v, the O-tree is symmetric y-feasible. The time for constructing a minimum height placement is $O(n^2)$.

If an O-tree can satisfy both symmetry constraints (HSC and VSC), it is called a *Symmetric Feasible O-Tree*. Given a Symmetric Feasible O-Tree, one can build a packing of symmetry groups with minimum width and height. Experimental result shows that the O-tree representation can handle symmetry constraints in a relatively shorter time with similar quality and performance. Comparing with those methodologies based on sequence-pair [**?**] and absolute coordinates [**?**].

2.1.3 Placement with Symmetry Constraints for Analog Layout Design Using TCG-S

In the paper [**?**], a topological representation called Transitive Closure Graph-Sequence (TCG-S) was introduced to improve the matching property of analog layout design. It can deal with symmetry constraints with better flexibility. The authors provided detailed proof for the necessary and sufficient conditions of the feasibility of a TCG-S satisfying symmetry constraints. An efficient TCG-S based algorithm with polynomial time was also proposed by the authors. The time complexity of the algorithm is $O(m^2)$ given a floorplan with m modules. It can handle the symmetry constraints effectively.

TCG-S Representation Overview

TCG was first presented in the paper [**?**], and extended to TCG-S in the paper [**?**]. For the TCG-S representation, two graphs *Horizontal Transitive Closure Graph* G_h and *Vertical Transitive Closure Graph* G_v are used to represent the geometric relations of the modules. Also, a packing sequence Γ_- represents the topological ordering of the horizontal transitive closure graph and the vertical transitive closure graph. Each module is represented by a node in the graphs and for each pair of nodes, there will be one and only one edge connects them. A TCG-S formed by acyclic transitive closure graphs G_h and G_v, and a packing sequence Γ_-.

Handling Symmetry Constraints in TCG-S

The authors proved that for each TCG-S, one can always find a unique feasible placement. Two more constraints should be satisfied if a TCG-S is symmetric feasible, namely *essence constraint* and *homology constraint*. Essence constraint requires the corresponding edge $(n_b, n_{b'})$ between a symmetric pair (b, b') to be in the vertical transitive closure graph G_v. To make sure that they have the same x-coordinates, module b must below module b' directly and the corresponding edge $(n_b, n_{b'})$ exists in G_v. A homology constraint denotes the internal relationship of two different symmetry pairs (a, a') and (b, b'). That one of the following conditions should be satisfied: (n_a, n_b) and $(n_{a'}, n_{b'})$ and both in G_h, or (n_b, n_a) and $(n_{b'}, n_{a'})$ and both in G_h, or (n_a, n_b) and $(n_{a'}, n_{b'})$ and both in G_v, or or (n_b, n_a) and $(n_{b'}, n_{a'})$ and both in G_v.

In order to pack all regular modules, the authors employ the longest path algorithm on the new TCG. This method can obtain a symmetric-feasible placement. The TCG-S representation is employed in a simulated annealing engine. The authors defined five types of movements to perturb a TCG-S solution:

- Rotation: Randomly rotate a module.

- Swap: Swap two modules and update the corresponding nodes in both G_v and G_h.
- Reverse: Reverse a reduction edge (there exists one and only one edge between n_a and $n_{a'}$) in G_v or G_h.
- Move: Move a reduction edge from G_v to G_h, or in the opposite direction.
- Transpositional Move: When we do the *move* operation, we also transpose the corresponding nodes.

Three representations Sequence-Pair, Segment Tree and TCG-S based on the same simulated annealing engine are compared. Experimental results show that when handling symmetry constraints, TCG-S can generate a solution with the smallest circuit area, while the running time is always shorter than that of the Sequence-Pair and comparable to that of the Segment Tree approach.

2.1.4 Modeling Non-Slicing Floorplans with Binary Trees

In the paper [**?**], Balasa shows a new topological representation based on *Binary Tree* (B-Tree). Before this work, the B-Tree representation is used for slicing floorplan. The authors extended it to symmetric-feasible binary tree dealing with the placement problem with symmetry constraints. Compared to the Sequence-Pair technology and the O-Tree representation, the B-Trees representation has smaller solution space than both Sequence-Pair and O-Tree. Experimental results show that the running time of B-Tree is the shortest among all three representations.

Symmetric-Feasible Binary Trees Handles Symmetry Constraints

Based on the B-Tree representation, Balasa *et al* defined symmetric-feasible binary trees. When we traverse a B-Tree, two sequences α and β can be obtained from

the in-order and pre-order traversal respectively. Suppose that we have a pair of blocks (x, y) within a symmetry group, then a B-Tree is symmetric feasible if and only if $\alpha_x^{-1} < \alpha_y^{-1} \iff \beta_{sym(y)}^{-1} < \beta_{sym(x)}^{-1}$ is satisfied.

Symmetric-feasible binary trees have two major properties that: (1) A symmetric-feasible binary tree can represent any placement containing a symmetry group; (2) One can obtain a minimum area placement solution from a symmetric-feasible binary tree in polynomial time. A quadratic time algorithm was proposed, and the generated placement was proved to have the minimum area and satisfy the symmetry constraints.

Experimental results show that the B-tree representation has equivalent effect when handling general placement problem, compared with sequence-pair and O-Tree. When handling placement with symmetry constraints, symmetric-feasible binary tree performs better than the other two topological representations (sequence-pair and O-Tree). The authors tested the B-tree representation, compared it with sequence-pair and O-Tree, and ran them on five test-cases (with symmetry groups). B-Tree achieved the *minimum running time* in every test-case.

2.1.5 Segment Trees Handle Symmetry Constraints

In the paper [**?**], Balasa proposed a new technique based on the segment tree data structure. Using segment tree, symmetry constraints can be handled before the simulated annealing process, which significantly reduce the size of the searching space. Segment tree is one subset of B-Tree, which is widely used in computational geometry.

Segment Tree Overview

A segment tree is a binary tree that can store intervals. It can be used to find the exact interval that contains a given point. Each leaf node *v* represents an interval

(or a segment). Assume that each node v contains an interval $v.I$, its left descendant v_l and its right descendant v_r. Suppose that we have a segment from point a to point b, denoted by $v.I$. Then we will have $v.I = [a,b]$, $v_l = [a, \frac{a+b}{2}]$ and $v_r = [\frac{a+b}{2}+1, b]$.

We consider symmetric placement along a vertical symmetry axis. The symmetry according to a horizontal symmetry axis is similar. Suppose that we have cell A and B within one symmetry group. Then the symmetry constraint in the segment tree can be described as follows. In the inorder traversal, if A precedes B, then B's symmetry counterpart $sym(B)$ should precede $sym(A)$ in the preorder traversal. It can be described by the following equation:

$$[inorder]\ A \prec B \iff [preorder]\ sym(B) \prec sym(A) \tag{2.1}$$

Calculate y-coordinates of Symmetry Blocks

Assume that there are two blocks B_i and B_j, that form a symmetric pair. Suppose that B_i is on the left and B_j is on the right. Let y_i and y_j denote the y-coordinates of these two blocks. A preorder traversal is performed. Suppose that node B_{i-1} is the closest ancestor of B_i, then the y-coordinate of B_i can be calculated as $y_i = max\{y_i, y_{i-1} + h_{i-1}\}$, where h_{i-1} is the height of B_{i-1}. If B_i doesn't have any ancestor, then $y_i = max\{0, y_i\}$. Next, they will put $y_j = y_i$ to make the y-coordinates of B_i and B_j equal. This calculation can be completed in linear time.

Calculate x-coordinates of Symmetry Blocks

This calculation can be done in two rounds of traversals. In the first traversal, horizontal symmetry will be satisfied by setting $x_i + (x_j + w_j) = 2 \cdot x_{sym}$, where w_j is the width of block B_j and x_{sym} is the x-coordinate of the vertical symmetrical axis. Since there are other constraints such as the non-overlapping constraint, some

symmetric blocks might be placed too far from the vertical symmetry axis on the right side. A second traversal is required to adjust the position of the symmetric blocks. In the second round, the position of left symmetric block of B_i will be re-calculated and updated in the reverse order, by $x_i = 2 \cdot x_{sym} - (x_j + w_j)$.

Experimental results show that the segment tree representation obtains the shortest running time among sequence-pair and O-tree. It can also achieve good performance in terms of circuit area. The authors propose segment tree based technique that can directly handle symmetry constraints. It lead to a significant shrinking of the solution space, which makes it better than O-tree and sequence-pair.

2.1.6 Center-based Corner Block List

In the paper [**?**], the authors proposed a new representation based on Corner Block List (CBL) to handle the common-centroid constrained placement problem, called *Center-based Corner Block List* (C-CBL). It is a complete and non-redundant representation of packings with common centroid devices. Simulated annealing technique is used. Experimental results show that C-CBL can effectively handle the common-centroid placement problem. Furthermore, for large test-cases, it can also obtain high successful rate in shorter running time.

C-CBL Representation

Center-based Corner Block List is derived from the Corner Block List representation, which can handle mosaic packing problem very well. CBL and mosaic packing are one-to-one mapped. Given a mosaic packing, one can obtain the CBL representation by deleting corner blocks iteratively. For horizontal blocks, non-crossing segment of T-shape junction is swept and deleted from left to right. For vertical blocks (rotate T-junction by 90° counterclockwise), the deletion operation

is done from bottom to the top. This process stops until only one block left. Suppose the number of blocks is n, we use a set S to represent a sequence of of blocks b_i ($i \in [1,n]$), and a list L to represent orientation (defined by lower left corner T-junction's orientation). $T = (t_0, t_1 \cdots t_i, \cdots t_{n-1})$ is used to represent a list of information about T-junction. When a corner block b_i is deleted, the number of uncovered T-junction is recorded in t_i. The CBL is represented by a 3-tuple $CBL = (S, L, T)$. Similarly, iteratively inserting corner blocks can transform a CBL representation to a mosaic packing.

C-CBL is an extended version for CBL. It has nice properties to handle the device groups with symmetry and common-centroid constraints. $C-CBL = (C, S, L, T)$, where C is the name of center block, and S, L, T remain the same meaning as CBL.

Methodology

The basic idea is to split each device $d_i (i \in [1,n])$ in a common centroid group $G_i = (d_1, d_2 \cdots d_i, \cdots, d_n)$ into two pieces (a_i, b_i), and pack half of G_i first then pack another half using same common centroid packing methodology. C-CBL is used to represent the common centroid placement.

To realize a placement from a C-CBL representation, the process is similar to that of CBL. Since there are common-centroid constraints, additional adjustment process will be performed. First, a Horizontal Constraints Graph (HCG) and a Vertical Constraints Graph (VCG) will be constructed. Each block's x- and y-coordinates can be obtained by applying the longest path algorithm on HCG and VCG. Assume that w and h are the width and height of block i respectively. Let $w(G_i)$ and $h(G_i)$ be the width and height of a common centroid group G_i. The upper right corner blocks' positions need corresponding adjustment. Assume that we insert block b after a. Then we have new x- and y- coordinate for upper right corner block.

$$x = w(G_i) - w(b) - x(a); \tag{2.2}$$

$$y = h(G_i) - h(b) - y(a); \tag{2.3}$$

Then the center block's position is represented by $(\frac{w(G_i)}{2} - \frac{w(b_c)}{2}, \frac{h(G_i)}{2} - \frac{h(b_c)}{2})$, where b_c denotes the center block. The positions of remaining blocks are adjusted to make them satisfy common-centroid constraint.

For a group containing even number ($2m$, where $m = 1, 2, \ldots$) blocks, we select a block b_{2n} where $n \in [1, m]$. Let $sym_x(b_{2n})$ and $sym_y(b_{2n})$ be the block that placed symmetrically to block b_{2n} in horizontal and vertical direction respectively. We placed them according to following equations:

$$sym_x(b_{2n-1}) = sym_y(b_{2n-1}) = b_{2n},\ sym_x(b_{2n}) = sym_y(b_{2n}) = b_{2n-1} \tag{2.4}$$

For a group containing odd number ($2m+1$, where $m = 1, 2, \ldots$) blocks. The authors select a block b_i as a self-symmetry block and then place remaining blocks w.r.t. block b_i.

These adjustments can make sure that the group satisfies the common-centroid constraint with the self-symmetry block b_c at the center of the group.

Experimental Results and Analysis

The authors use simulated annealing algorithm as the searching method. They define a series of movement operations for C-CBL. The orientation, aspect ratio and information can be changed during the searching process, and two devices can be randomly exchanged for the perturbation. Test cases with common-centroid groups, with a total block number from small to large (e.g., 30 to 300), are tested to verify the effectiveness of the new representation. An average deadspace of 8.29% is achieved, which is comparable to other previously used methodologies.

Furthermore, they can handle mid- to large- size test cases (with more than 200 blocks), which previous methods failed to handle.

2.2 Other Works on Analog Placement Constraints

2.2.1 Deterministic Analog Placement with Hierarchically Bounded Enumeration and Enhanced Shape Functions

A new B*-Tree based algorithm Plantage, is introduced in this paper [**?**]. In this paper, two key techniques, *hierarchically bounded enumeration* and *enhanced shape functions* are presented. The hierarchically bounded enumeration technique can divide the problem into a set of sub-problems. After solving all the sub-problems, the combining procedure combines these solved sub-problem to obtain the final placement result of the completed circuit. This divide-and-conquer algorithm can avoid those time-consuming procedures for evaluating a huge number of B*-Trees. The enhanced shape function is an extended version of shape function, which is used to store pareto optimal results efficiently.

The major contribution of this work are: (1) This work presents the first deterministic algorithm that can handle many analog circuit placement constraints (proximity constraints, symmetry constraints, common-centroid constraints, minimum distance constraints and variant constraints). (2) Hierarchically bounded enumeration and enhanced shape functions are employed to enhance the performance of the algorithm.

Handle Symmetry and Common Centroid Constraints Using B*-Tree

This work based on the B*-tree representation, since B*-tree has little redundancy in the solution space. The authors introduced two new techniques, hierarchically bounded enumeration and enhanced shape functions, that enables customer to

choose the best result from a set of candidate solutions with different aspect ratios, which improve the design flexibility.

Same as B*-Tree, constraint graphs are used to represent the horizontal and vertical relationship between modules. In order to meet the requirement of symmetry constraints and common-centroid constraint, an arbitrary linear constraint is employed to handle these problems. A set of constraints should be satisfied in the vertical and horizontal direction. Two matrix M_v and M_h are defined, together with the corresponding elements d_v and d_h, describing the minimum distance constraints. Where d_v and d_h denote the vertical and horizontal distance between two individual modules' centers. Matrix C_v and C_h represent the symmetry and common-centroid constraints. Vectors $\vec{x}$ and $\vec{y}$ are the x- and y- coordinates of the modules. Symmetry constraints and common-centroid constraints are encoded in two vectors, $\vec{k_v}$ and $\vec{k_h}$. The vertical and horizontal symmetry and common-centroid constraints can be formulated as follows:

$$\vec{C_v} \cdot \vec{y} = \vec{k_v},\ \vec{C_h} \cdot \vec{x} = \vec{k_h}; \tag{2.5}$$

The authors also employ enhanced shape function and hierarchically bounded enumeration to improve their analog placement tool. The tool Plantage can produce high quality layouts, although the running time is a bit longer than that of sequence-pair with dummy nodes [**?**] and symmetry island [**?**].

Experimental Results Analysis and Conclusion

A series of experiments are designed to show the advantages of this work. Plantage with hierarchically bounded enumeration and enhanced shape functions is compared with some popular methodologies, such as sequence-pair [**?**], segment tree [**?**], sequence-pair and linear programming [**?**], sequence-pair with dummy nodes [**?**], symmetry island [**?**], sequence-pair with Johnson's priority queue [**?**].

Results show that Plantage can generate a placement result in acceptable running time, with comparable area usage.

2.2.2 Analog Placement Based on Symmetry-Island Formulation

In the paper [**?**], the authors propose a new concept named symmetry island that place some pairs of matching devices closely and symmetrically, to handle the marching between devices. They modify the B*-tree, and propose a new representation based on it. This new representation is automatically symmetry feasible and is called automatically symmetry feasible B*-Tree (ASF-B*-tree). ASF-B*-tree can be used to model a symmetry island. In order to handle symmetry islands and non-symmetry islands simultaneously, the authors propose hierarchical B*-tree (HB*-tree) framework. An ASF-B*-tree can be a node of HB*-tree. HB*-tree is a novel hierarchical framework which can dynamically update the shapes of symmetry islands and can handle symmetry and non-symmetry devices. It's an analog placement algorithm with linear packing time.

Representing Symmetry Island by ASF-B*-tree

The authors use ASF-B*-trees to represent symmetry groups. They first define the representative B*-tree, constructed by representative nodes. Suppose there is a symmetry pair (b, b'), where b is the left (or bottom) device and b' is the right (or top) device. Then the representative block b^r of this pair is b'. For a self-symmetry device b, the representative block b^r is the right (or top) half of b. The other part of a symmetry pair or self-symmetry device is called non-representative block. A placement is symmetric if the non-representative blocks of pairs or devices can be placed symmetrically with their representative blocks. A symmetry representative B*-tree is an ASF-B*-tree.

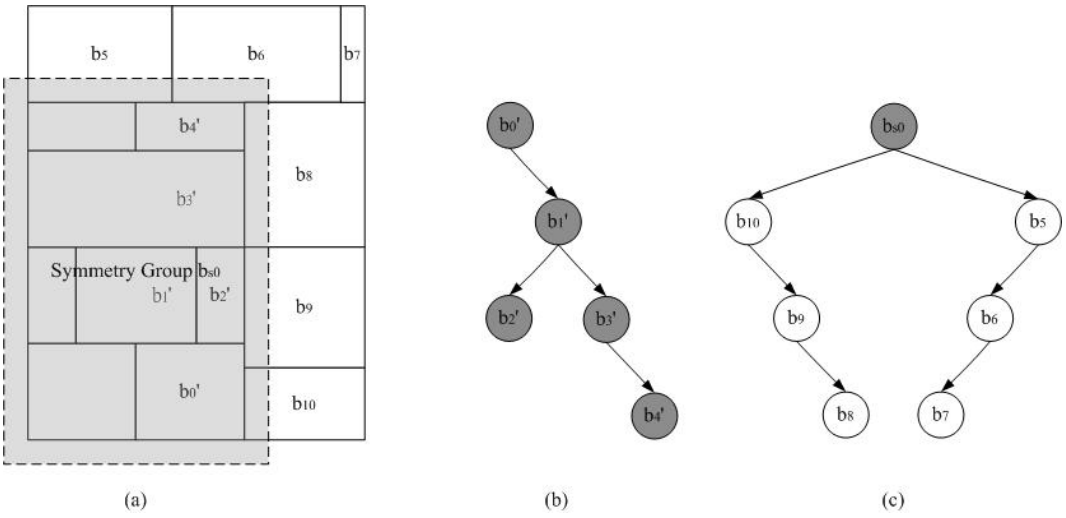


Figure 2.1: HB*-Tree Example: (a) Placement with Symmetry Group b_{s0}. (b) The Corresponding ASF-B*-tree of Symmetry Group. (c) Hierarchical B*-tree.

In order to handle symmetry group(s) and asymmetric devices, the authors propose a hierarchical B*-tree framework. In HB*-tree, symmetry group(s) is treated as a node, together with asymmetry nodes to form a whole B*-tree.

Figure 2.1 shows an example of placement with symmetry group. The corresponding ASF-B*-tree and whole HB*-tree are also shown in this figure.

Experimental results show that this symmetric based approach can obtain the best performance in terms of area and running time. Compared with other works based on sequence-pair, segment tree and TCG-S, this work can effectively handle symmetry island and asymmetric devices and can pack modules in linear time.

2.2.3 Heterogeneous B*-Trees for Analog Placement with Symmetry and Regularity Considerations

The research work done by Chou *et al* [**?**], consider symmetry constraints and regularity during the whole analog placement generating process. Symmetry placement is required to reduce harmful electrical effects. For example, parasitics effect will

be introduced if the devices placement is asymmetric. Regularity is one issue that designer should take into account. Irregular placement will reduce the routability in the following routing step. Sometimes the total wirelength also increase because unnecessary bends are introduced. This paper proposed a heterogeneous B*-trees based technique, handling the two issues simultaneously.

Overview of the Placement Technique

After the input information (modules information, netlist, symmetry and common-centroid constraints) is parsed by the tool, module classification will be performed to identify the sizes and dimensions of the modules. Modules with the same size will be identified and form a regular structure. Empirically, regular group with close to one aspect ratio usually has a good matching property. A *score table* is computed and a score is calculated for every possible regular group. Given m possible regular structures for a regularity hierarchical node, let w_i and h_i be the width and height of the corresponding regular structure r_i $(i \in [1,m])$. The score s will be calculated by following equation:

$$s = \sum_{i=1}^{m} \frac{min(w_i, h_i)}{max(w_i, h_i)} \tag{2.6}$$

A score of regularity node can be used to decide the probability for changing the node when perturbation is performed. Next, new types of moves are defined and corporate with heterogeneous B*-tree moves, e.g., merge/split and reshape. In the final step, new cost function for regularity cost calculation is proposed to guide the placement result.

Handling Symmetry and Regularity by Using Heterogeneous B*-Trees

The heterogeneous B*-trees are derived from the B*-trees representation and inherit the advantages, such as linear time packing and flexible movement. Com-

pared to the original B*-tree. It contains symmetry nodes and regularity nodes. Both nodes are hierarchical nodes of B*-Tree. *Symmetry nodes* are used to represent symmetry group. Symmetry constraints are observed to ensure that the devices in a symmetry group will be placed on both sides and have the same distance from the symmetry axis. *Regularity nodes* are nodes used to model modules that should be placed in a regular structure, e.g., in a row-like, column-like placement pattern or in a rectangular shape with several rows and columns. These regularity constraints are predefined by the designer. By applying these constraints, running time for modules packing is shorten and the wirelength is decreased because detours and bends are eliminated.

The algorithm can handle symmetry constraints and regularity requirement simultaneously. Two cases are studied in this paper: (1) regular structures with symmetry and non-symmetry modules, and (2) regular structures with modules in different symmetry islands. For the first condition, the algorithm allows merging operation between non-symmetry modules and symmetry modules. This operation can form a regular group. It should be pointed out that merging can only be allowed on the boundaries of regular structures to satisfy the symmetry requirement. For the second case, when there are modules belonging to more than one symmetry groups and appearing in same regular structure, the technique allows merging these symmetry groups into a new larger one (by adding *pseudo wires* to connect them). A corresponding symmetrically hierarchical node is created to represent the new symmetry group, which will be included into automatically symmetric feasible B*-tree.

Experiment Design and Result Analysis

Two sets of experiments are performed. In the first phase, three key results (area, half-perimeter wire-length and running time) are measured, the authors compare

their work with other representations, e.g., sequence-pair, HB*-Tree. They also test their heterogeneous B*-Trees with and without the regularity consideration. It turns out that the heterogeneous B*-Trees without regularity constraints obtain $+4\%$ circuit area, $+10\%$ wirelength but the running time is decreased by 4%.

They also compare the routing results for their generated placement. Heterogeneous B*-Trees with regularity constraints also has the best performance in overflow, via cost, wire-length and running time, and the corresponding reductions are 17%, 10%, 16% and 6%, respectively.

2.3 Summary

In this chapter, some previous works are studied and reviewed. We review previous works on topological representations handling symmetry constraints and common-centroid constraints, such as Sequence-Pair [**?**] [**?**], O-Tree [**?**] [**?**], B-Tree [**?**], B*-Tree [**?**], Segment Tree [**?**], and TCG-S [**?**].

Some recent works further extend based on the above representations. For example, some researcher improve and extend on B*-Tree [**?**] [**?**]. Strasser [**?**] proposed a deterministic placement algorithm based on B*-Tree using hierarchically bounded enumeration and enhanced shape functions to handle 1-D and common centroid constraints. For the common centroid constraint. Ma [**?**] proposed a method based on the C-CBL representation.

□ End of chapter.

Chapter 3

Common-Centroid Analog Placement

In analog circuit design, devices need to be placed symmetrically (both 1-D and 2-D symmetry) to satisfy the matching property. Besides, some electrical properties such as parasitic effects and thermal effects should be taken into account among the analog circuit design, which significantly increases the difficulty of the problem. Compared to the general ASIC placement problem, the analog placement problem has more constraints to make sure that the placement result will satisfy those complicated requirements on electrical characteristics. This is usually a time-consuming and error prone process and takes experienced circuit designers couples of days to complete the whole layout design process. Our research focuses on generating qualified layout automatically by using our automatic layout design tool.

In this chapter, we propose our placement method that is based on a simulated annealing engine using sequence-pair as the representation. We are given a netlist of devices and a layout library for each device. The objective is to generate a routable placement of the devices with small area, wirelength and good matching

properties such that all the user specified constraints are satisfied. We assume that the users will input the symmetry constraints and placement constraints. In the input netlist, some devices are specified by the users as belonging to a symmetric group and are thus required to be placed symmetrically with respect to a single point (common centroid constraint) or to an axis (1-D symmetry constraint) to reduce mismatch errors. Our proposed method can handle symmetry constraints and common centroid placement in analog placement very well. We describe the practical technique that places devices (transistors, resistors and capacitors) in a more symmetrical way. A detail interleaving placement method is proposed, with flexible options on device orientation, row (column) number, order (defined by input and output pins' order), etc. It can generate different symmetrical placement by choosing different combination of those placement configuration.

During the placement process, we also need to consider the routablility issue. In analog design, wires will often avoid going above the active area of the devices to reduce crosstalk effect between the devices and the wires. Thus the original compact realization of sequence pair is not practical. A layout expansion of the originally compacted placement is needed to get enough routing resources. We will present an expansion method based on congestion estimation with trade-off between area and routability. We employ a routing tool to test the routability of our placement with this layout expansion technique. Results show that we can obtain higher routing completion rate with this technique.

3.1 Problem Formulation

A formal description of the analog layout automation problem is defined as follows:

- Input:

  - Netlist.
  - Layout library for all the devices supplied by the foundry.
  - Constraints:
    A list of constraints specified by the layout designer, e.g., symmetric pairs, symmetric nets, clustering constraints and merging constraints, etc.

- Output: A layout satisfying all the constraints and can pass the checks and verifications.
- Objective: Minimize the mismatches between the devices.

A set of key parameters are used to measure the quality of placement result, such as the area, HPWL (half-perimeter wire-length), overlap, timing, congestion, etc. Among those factors, *area* and *HPWL* are commonly used in the cost function, helping the search engine to find a placement result with the minimal cost computed by a weighted sum of the area and HPWL. A qualified placement should have the following properties:

- 1. Minimized area: Place all devices without overlapping and to minimize the total circuit area.
- 2. Minimized wirelength after routing.
- 3. Matching properties: Layout those matched devices symmetrically to increase the matching properties of the circuit.
- 4. Routability: Increase the successful rate of routing completion.

After a circuit is designed, we will take the netlist and the layout library from the foundry as inputs of our tool. Simulated annealing will be used to place the

devices satisfying the symmetry and other placement constraints. Congestion estimation and layout expansion will be performed in each iteration of the annealing process in order to produce a placement with good routability. Routing will then be performed on the placement result handling both the complete and partial symmetrical nets. After a final layout is generated, verification including Design Rule Check (DRC), Layout Schematic Check (LVS) and post-simulation will be performed to validate the correctness and quality of the output by extracting the key parameters of the circuits, e.g., DC gain, bandwidth, phase Margin and CMRR values. Monte Carlo simulations will also be performed at the end to evaluate the matching properties of the devices under different kinds of variations.

3.2 Overview of Our Work

Our placement technique is based on simulated annealing using sequence-pair as the representation. The placement generate flow of our tool is : First we export netlist file from Virtuoso. Then device retrieval process is performed to obtain the layout information. Next, a list of constraints (symmetric pairs, symmetric nets, clustering constraints and merging constraints) are passed into our tool. Then our tool handles device merging, device clustering, common-centroid placement for devices (resistors, transistor and capacitors) simultaneously. It can also change positions of devices within symmetric group to obtain optimize circuit area. And finally, a congestion estimation and layout expansion technique is used to increase the routability.

The objective of this whole project is to study and develop a novel and holistic analog automation tool that can automate the whole layout process from a schematic circuit to the final layout effectively. Focuses will be on the quality of the generated output in comparison with manual design on various aspects of performance measures and especially on mismatch reduction, which is an impor-

tant issue in analog design. In order to achieve this, we emphasize the following two features of our analog placement methodology:

Symmetric Placement: We consider symmetric placement of transistors, resistors and capacitors, which includes 1-D symmetry and 2-D symmetry (or called common centroid). Different symmetric placement configurations, derived according to the practical needs in analog design, are considered for the matching devices in the simulated annealing engine of the placer in order to generate a placement with high quality. The proposed approach considers different placement patterns by choosing different row (column) numbers, orientations, I/O pins orders, etc. Our tool can generate a set of candidate placements in different patterns (symmetric placement and interleaving placement), and offer a high flexibility to designer to choose suitable placement according to design requirements and simulation results. In some cases, a large deadspace is introduced by the symmetric group. Two types of symmetric devices are defined: *symmetric devices* and *self-symmetry devices*. Improper placement of self-symmetry devices may result in producing unnecessary deadspace. The cost function of our simulated annealing approach take these symmetric devices into account and give them higher penalty weights to find the minimize area of symmetric group.

Congestion-driven Placement: In analog design, wires are preferred not to be routed over active devices, so we need to leave enough spaces properly for routing between the devices during the placement process. To achieve this, we explore congestion estimation and layout expansion during the placement step in order to produce a good and routable solution.

A brief placement work flow is shown as follow:

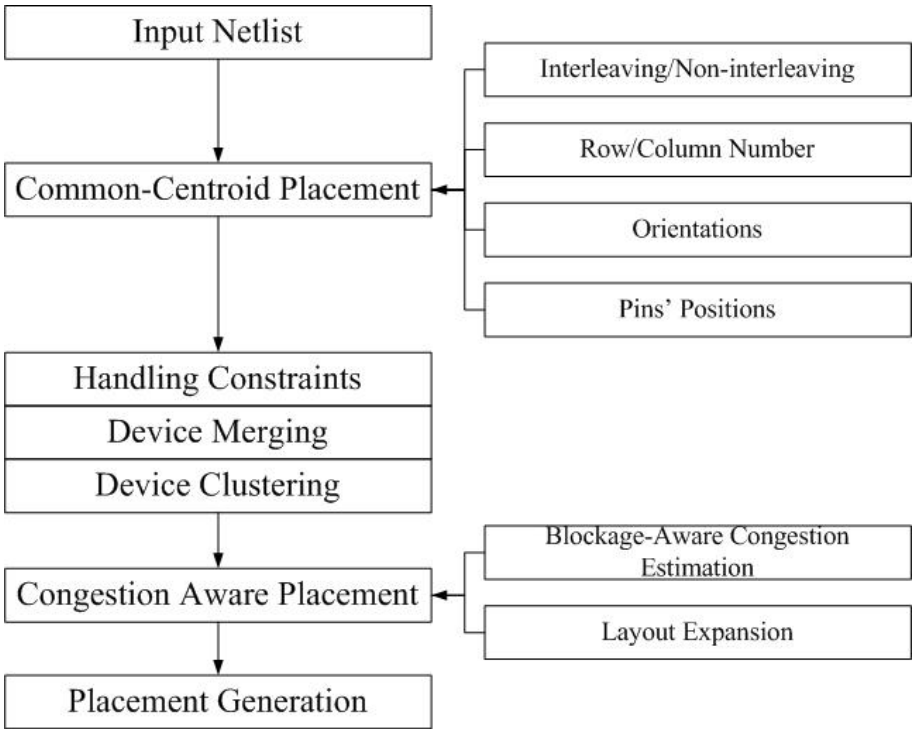


Figure 3.1: Placement Work Flow.

3.3 Handling Common Centroid Constraints in Different Devices

Our proposed method further extends the previous work [**?**]. It gives designer more design flexibility to generate layout with different symmetric styles. For example, they can place capacitors symmetrically in an interleaving or non-interleaving fashion. Our tool will choose different placement patterns for a device. Take a resistor group with 24 unit resistors as an example. We will change the placement pattern by modifying the row number. To become a 2×12 array, 3×8 array or 6×4 array. Designer can choose suitable placement configurations according to the simulation result and design requirements.

3.3.1 Common Centroid Placement of Resistors

Some symmetric resistors pairs are needed to be placed in a common centroid way to reduce the mismatch error. In our placement tool, we can change the placement of resistors by modifying the following configurations:

- Row (column) number: This will affect the shape of the resistors group.
- Orientations: Each resistor has it's own orientation. Modifying the orientations will change the wire connection pattern.
- Connection order: Connection order specify the orders of input and output pins.

Figure 3.2 shows an example. This example shows a symmetric pair A and B of 2×6 transistors. Instead of placing them separately on the two sides of a symmetry axis as shown in Figure 3.2(a), we will place them in a common centroid fashion by interleaving the resistors as shown in Figure 3.2(b). Note that each group of resistors are connected in series. Comparing with the placement in Figure 3.2(a), this common centroid way of placement will bring extra cost for interconnection and this occurs only when we need to connect resistors on the two sides of the symmetry axis, e.g., connect the rightmost A on top to the rightmost A below and connect the rightmost B on top to the rightmost B below. This will occur only once for each group with our common centroid placement configurations and the additional cost in interconnection will be very low in practice, comparing with the total wirelength of the whole circuit. The experimental results have also confirmed this.

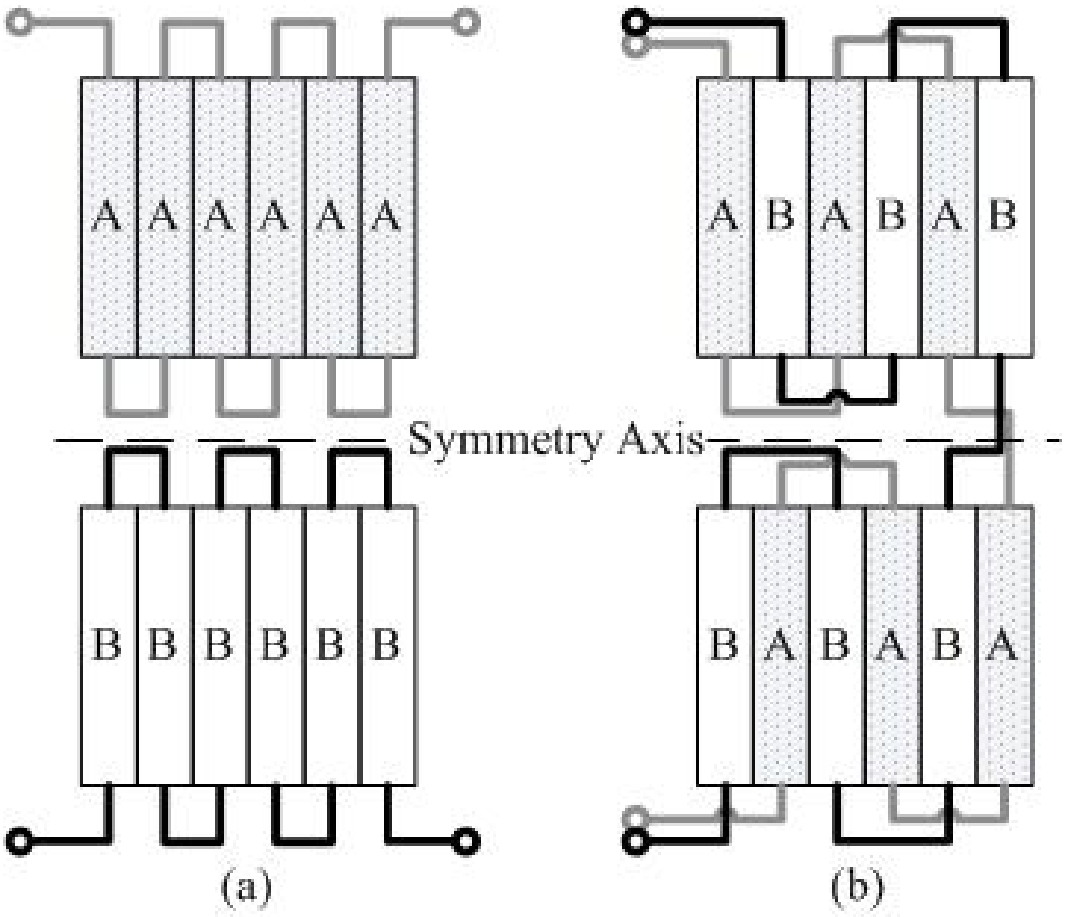


Figure 3.2: Common Centroid Placement for Symmetric Pair with 2×6 Resistors and Their Routing Patterns: (a) 1-D Symmetric Placement. (b) Common Centroid Placement.

Further to the example shown in Figure 3.2(b), we use Figure 3.3 to illustrate how the assignment of input and output pins affect routing. Different input and output positions lead to different routing patterns and affect the total wire length. In this example, we assume that the height and width of each device is $2.0\mu m$ and $0.4\mu m$ respectively, and the spacing between the upper side and the lower side is $1.0\mu m$.

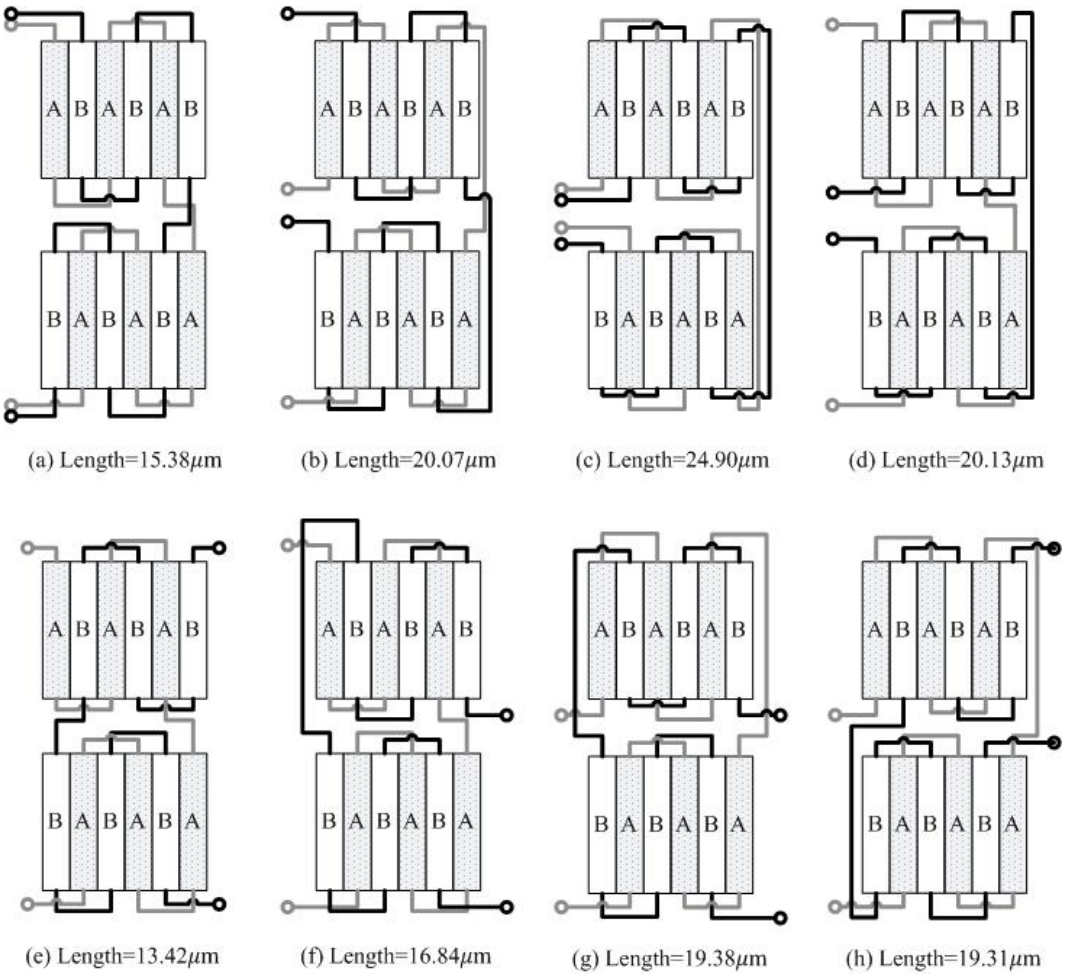


Figure 3.3: Common Centroid Placement for Symmetric Pair with 2×6 Resistors. Different Routing Patterns and Wire Lengths.

Figure 3.3 lists 8 different routing patterns and the corresponding wire lengths. Figure 3.3(a)-(d) shows conditions that pins of device *A* and *B* are all positioned on the left side of the device group. We omit the conditions that pins are assigned on the right side of device group, because they are similar to that of Figure 3.3(a)-(d). Figure 3.3(e)-(h) illustrates the conditions that pins of *A* and *B* placed at different sides of the device group. We get different wire lengths because of different routing methodologies. In Figure 3.3(a)-(h), we can find that different placement

patterns lead to different wire lengths. It ranges between 13.42μm and 24.90μm. Interconnections are the major reason for different wire lengths. In these 2-D symmetric placements, some long connections that across the symmetry axis bring additional wire length in comparison with the 1-D symmetric placements. The distance between the upper and the lower halves will also affect the total wire length.

When we decide which routing pattern to be used, we should consider the wire length globally. We also need to consider the affects of the routing pattern on the symmetric placement and the routings of other nets. In practice, Figure 3.3(a) and (e) are commonly used.

In general, all symmetric resistor pairs of size $2 \times n$ can be divided into four groups depending on the value of $k = n \ mod \ 4$. The common centroid placements for the cases when $n = 4, 5$ and 7 are shown in Figure 3.4 and the other cases are similarly generated. All these configurations can lead to better symmetric routing with shorter wire length.

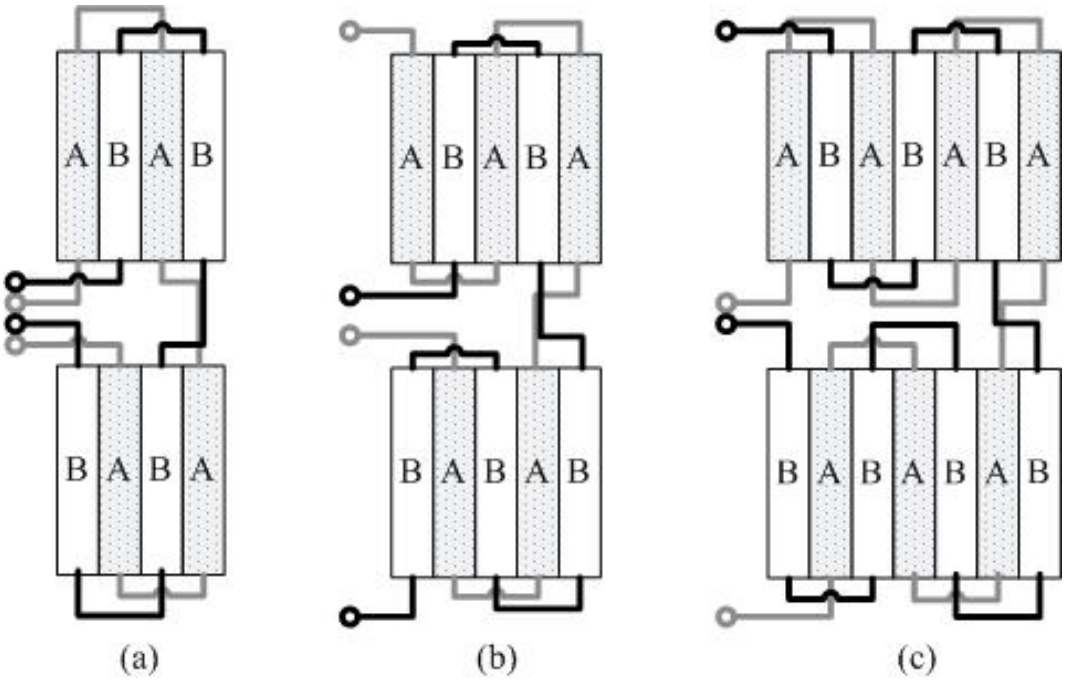


Figure 3.4: Common Centroid Placement of Resistor and Their Routing Patterns: (a) 2×4 Resistors, (b) 2×5 Resistors, (c) 2×7 Resistors.

We discuss common centroid placement of resistors with 2 rows in the above paragraphs. It can be easily extended to resistor pairs with other number of rows (e.g., $row = 3,4,5\ldots$)[1]. For simplicity but without loss of generality, we fix the column number as six and change the row numbers. Figure 3.5 show these placement for resistor group with 3×6, 4×6 and 5×6 resistors, respectively.

[1]It should be noted that in our test cases, only resistor groups with 2 rows are considered because for the available test cases, these 2-row placement patterns are good enough for optimization.

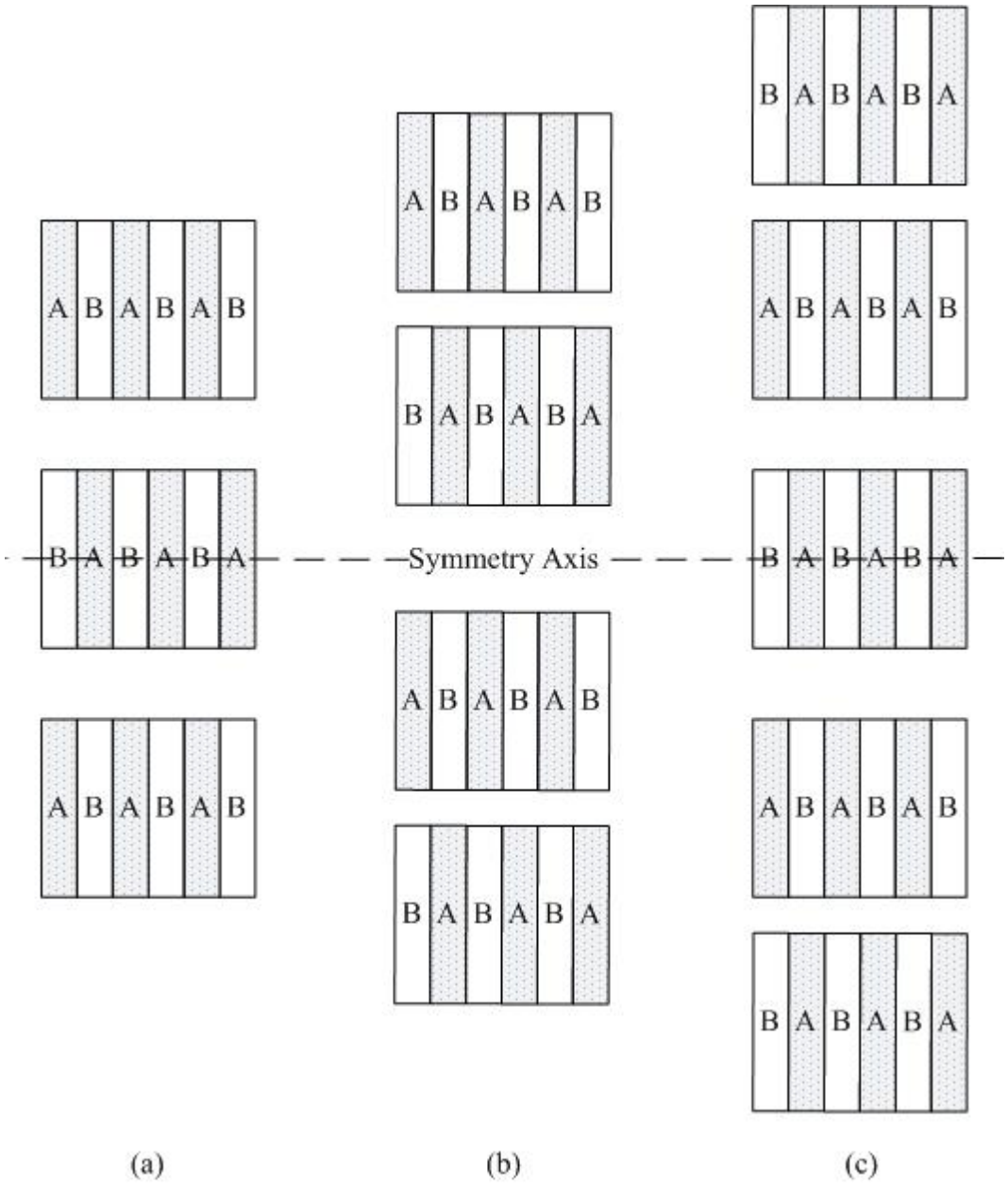


Figure 3.5: Common Centroid Placements of Resistors: (a) 3×6 Resistors, (b) 4×6 Resistors, (c) 5×6 Resistors.

3.3.2 Common Centroid Placement of Transistors

We can also apply similar techniques to symmetric pairs of transistors. This technique is widely used in differential pair transistors. Figure 3.6 shows an example. In this example, two symmetric transistors A and B are divided into six parallel sub-devices respectively. Instead of placing them separately on the two sides of a symmetry axis as shown in Figure 3.6(a), we will place them in a common centroid fashion by interleaving the sub-devices as shown in Figure 3.6(b) and (c). Since each sub-device have their input pin i and output pin o, the locations of these pins will affect the orientation of the symmetric group. The orientations are illustrated by arrows that start from i and point to o. Our tool allows designer to select orientation options (same or opposite) and generate a corresponding layout. Figure 3.6(b) shows the two halves of this group opposite orientations, while Figure 3.6(c) shows a placement with same orientation.

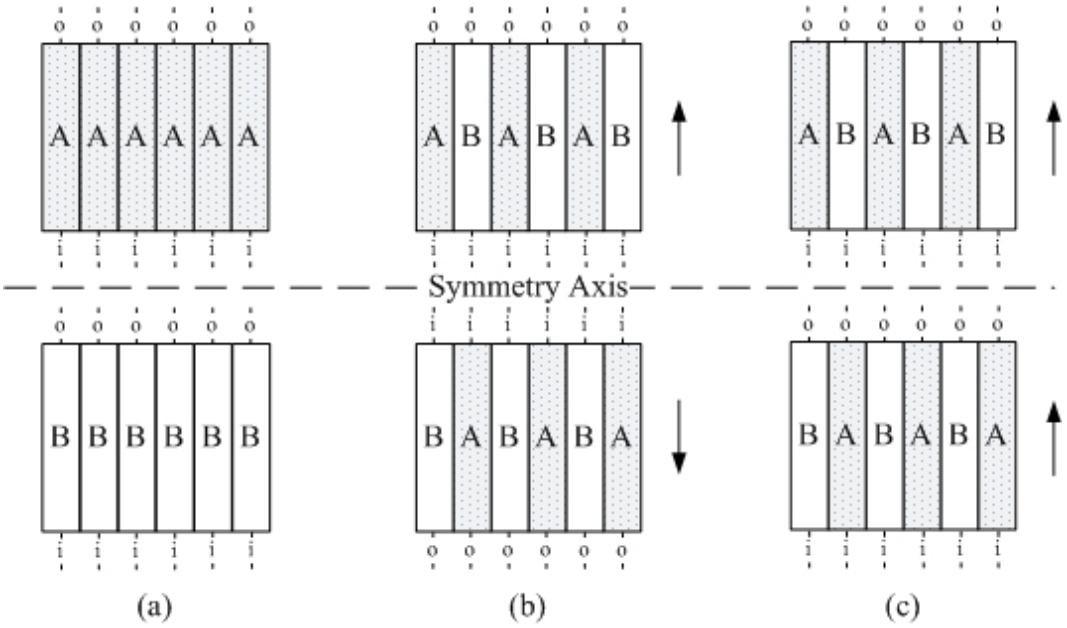


Figure 3.6: Common Centroid Placement of Transistors: (a) 1-D Symmetric Transistor Placement. (b) Common-Centroid Transistor Placement with Opposite Orientations. (c) Common-Centroid Transistor Placement with Same Orientation.

Figure 3.7 shows the corresponding wiring topologies of Figure 3.6. Orientations will affect routing pattern and wire length. In common centroid placements, there are both *A* and *B* on each side of the symmetry axis. Wire length is increased because of the additional wires needed to connect sub-devices of *A* (or *B*) on the two sides of the symmetry axis. The length of the additional wires is related to the distance between the lower and upper sub-groups and the height[2] of the transistors. We can see from Figure 3.7(a) that 1-D symmetry has the shortest wire length of 29.00μm, and Figure 3.7(b)(c) show 47.74μm and 45.00μm, respectively. Wires 1, 2, 3 and 4 marked in Figure 3.7(c) are the additional wires in comparison of the 1-D symmetric placement shown in Figure 3.7(a). In real design, we usually use the placement pattern shown in Figure 3.6(c) because it's easier for symmetric

[2]It's related to the device width when the symmetry axis is vertical.

routing. This placement pattern also reduces the wire density in the area between the two symmetric halves. Note that transistors belonging to the same group will be connected in parallel. The wiring topology is the same no matter what the value of m is in a symmetric pair of size $2 \times m$.

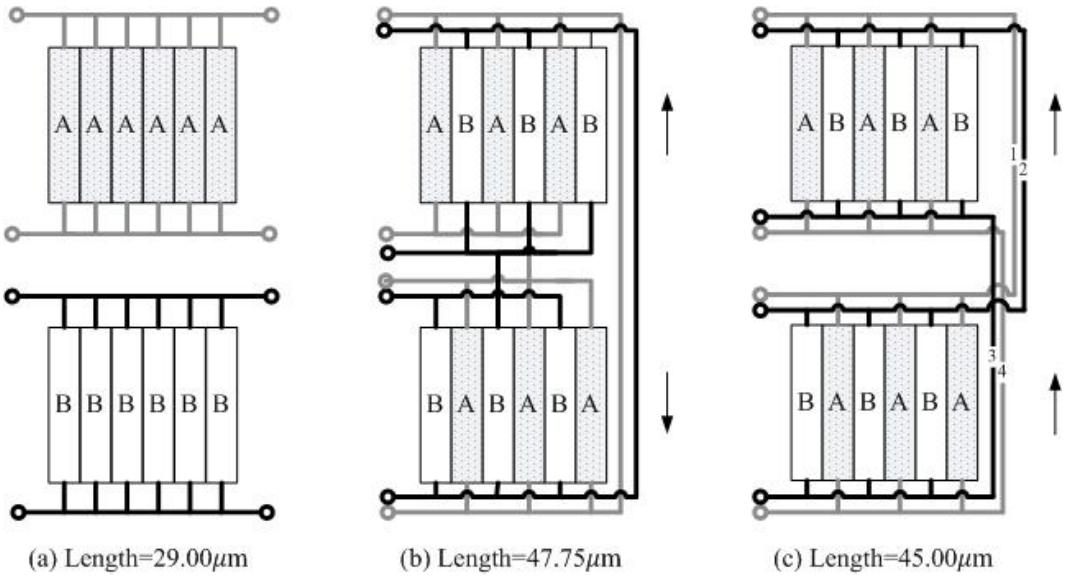


Figure 3.7: Common Centroid Placement of Transistors and Their Routing Patterns: (a) 1-D Symmetric Placement. (b) Common Centroid Placement with Opposite Orientations. (c) Common Centroid Placement with Same Orientation.

In Figure 3.7, we assume that each device's height and width is $2.5\mu m$ and $0.5\mu m$, respectively. The spacing between the upper side and lower side is $1.2\mu m$. The percentage of increase on wire length becomes smaller when m becomes larger. That's because it's only need to add four additional wires (see wires 1, 2, 3 and 4 in Figure 3.7(c)) no matter which number m is. Comparing Figure 3.7(a) and Figure 3.7(c), the wire length increases from $29.00\mu m$ to $45.00\mu m$. If we have more sub-devices, the percentage of increase will further drop.

3.3.3 Common Centroid Placement of Capacitors

Matching of capacitors is also a common and important concern in analog design. There are some rules of thumb to follow in general to get better matching for capacitors:

Rule 1: Use square-like geometry for the unit capacitors.

Rule 2: Use identical geometry for all the unit capacitors.

Rule 3: Place the matching capacitors adjacent to one another.

Rule 4: Place dummy capacitors on the boundaries of capacitor group to shield the matching capacitors from lateral electrostatic fields.

Rule 5: Cross couple arrayed capacitors to minimize the effects of oxide gradients.

In our proposed common-centroid placement of capacitor pairs, according to the matching rules above, we use square shape unit capacitors to form large capacitor groups. Consider a symmetric pair of capacitors with size $2 \times m$, we will generate a *set* of common centroid placement configurations to describe the exact placement and routing methodology according to the value of m. The simulated annealing engine will then help to choose a configuration to optimize the objective function.

Consider an example of a symmetry pair with size 2×12. There are thus totally 24 unit capacitors. First of all, we will generate a set of placement configurations. Assuming that the symmetry axis is horizontal and we will consider even row number in this case.[3] Therefore, the set of possible sizes for this symmetry pair will be $(row, column) = \{(2,12),(4,6),(6,4),(8,3),(12,2)\}$. For each of these

[3] If the symmetry axis is vertical, we will consider even column number.

candidate sizes, there are two choices for the relative placement between A and B and there are four choices for the placement orientation. To illustrate this better, we use the size of $(row, column) = (4, 6)$ as an example in the following figures. Figure 3.8 shows the two different configurations with different relative placement between A and B.

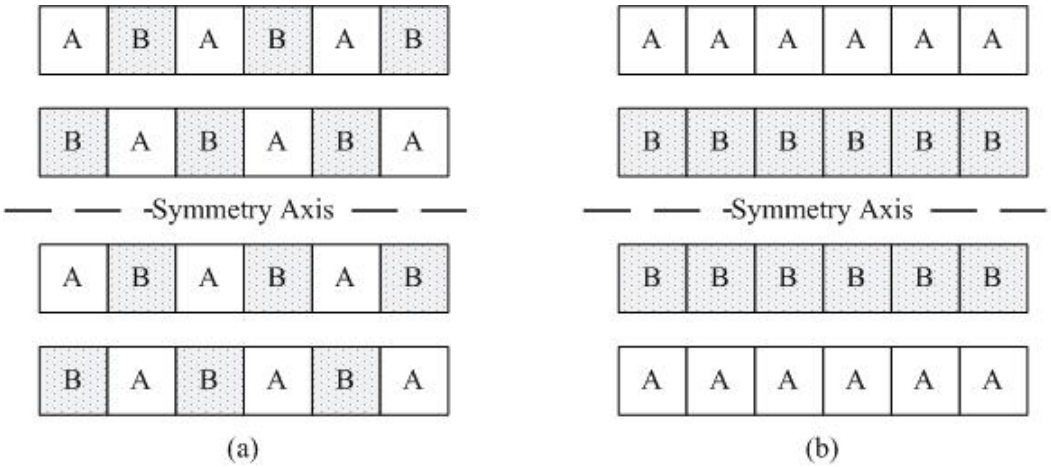


Figure 3.8: Two Different Common Centroid Placement of Capacitor Pairs: (a) Interleaving Placement; (b) Symmetric Placement *ABBA*.

For each of these possible configurations, there are four possible choices for the placement orientation as shown in Figure 3.9. Note that different orientations will lead to different pin positions and the routing will be affected. In our methodology, we choose to have the same orientation for capacitors in the same row and we allow the rows on one side of the symmetry axis to either have the same orientation or have alternating orientations. Therefore, for this example of symmetric capacitor pair, there will be a set of $5 \times 2 \times 4$ possible candidate placement configurations to choose from.

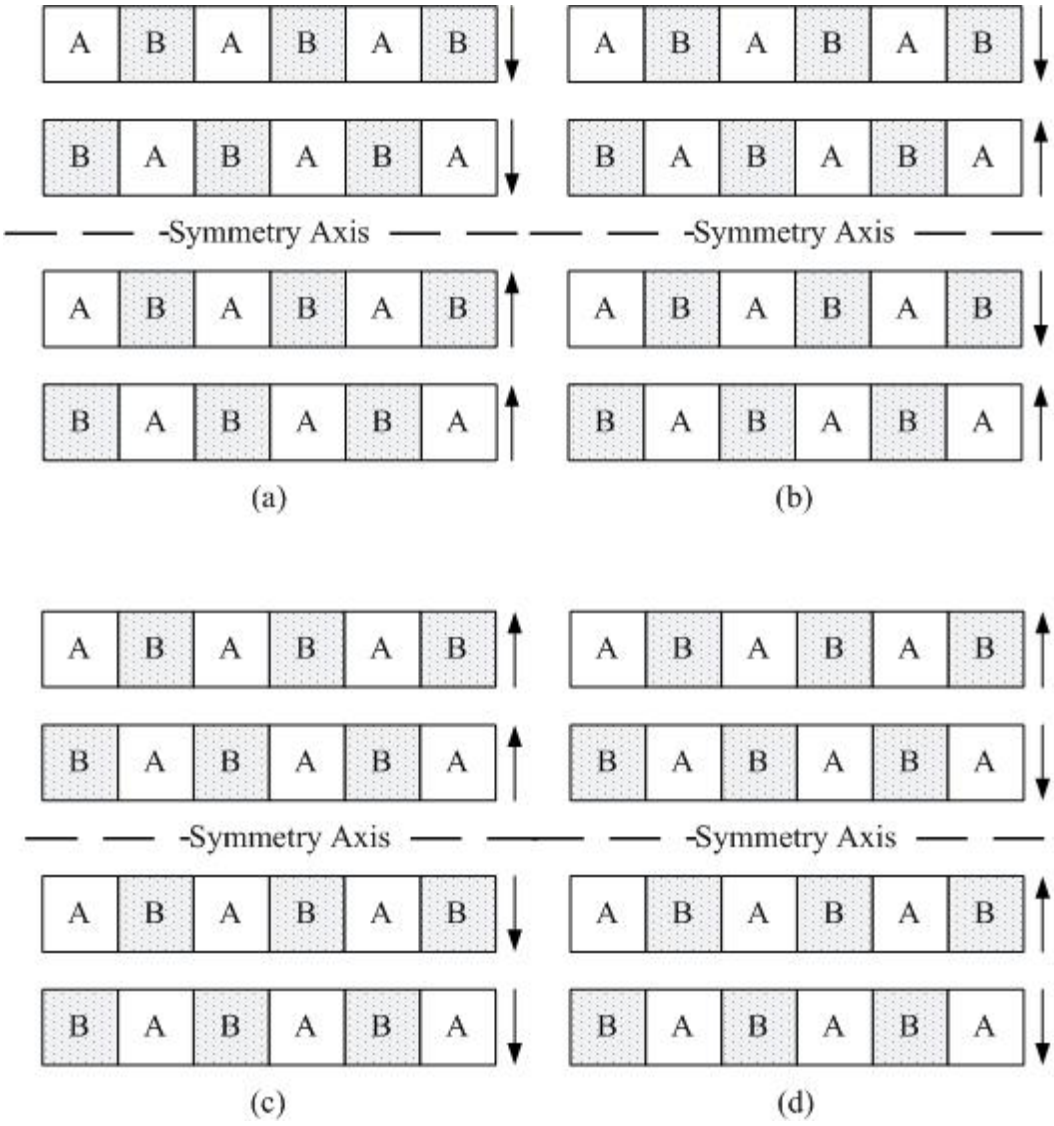


Figure 3.9: Four Orientation Options.

Figure 3.10 shows two examples of different routing patterns. Figure 3.10(a) corresponds to the placement with orientation choice shown in Figure 3.9(b). Figure 3.10(b) shows the routing pattern of a symmetric placement. Although Figure 3.10(b) has a shorter wire length, the matching property might not be as good as that of Figure 3.10(a). We give designer plenty of flexibilities to generate different layouts by selecting different combinations of placement configurations. They

can choose suitable placement according to design requirements and circuit performance.

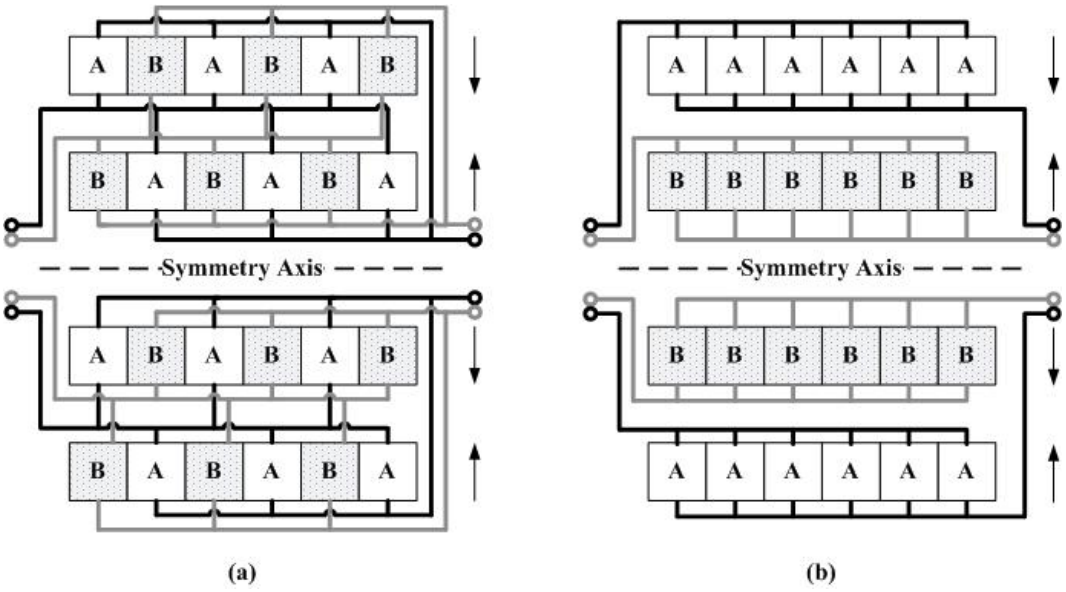


Figure 3.10: Two Different Routing Pattern: (a) Interleaving Placement; (b) Symmetric Placement *ABBA*.

3.4 Congestion Estimation and Layout Expansion

In analog designs, wires are preferred not to be routed over active devices to reduce the crosstalk effect between devices and wires. Therefore, we need to leave enough spaces for routing between the devices during the placement process. To achieve this, we need a good congestion estimation and layout expansion method to produce a good and routable placement solution. Using the traditional method of computing longest paths in constraint graphs for sequence pair representation will give a lower-left compact packing.

We will first overlay a $n \times n$ grid on it for routing estimation ($n = 40$ in our implementation). We will then compute the routing congestion according to the

approach in [**?**]. Additional steps will be taken to distribute the congestion measures over the blockages (which are active devices in our case) to the surrounding in order to obtain an accurate estimation. After that, each room in the $n \times n$ grid will be *expanded* proportionally according to the congestion estimations while the devices in the room will be moved correspondingly. Details of these two steps will be described below.

- Divide the layout into a 40×40 mesh.
- Calculate the vertical (c_v) and horizontal (c_h) wire capacities: $c_v = h/z$ and $c_h = w/z$, where h and v are the width and height of a room, and z denotes the sum of the minimum wire width and the minimum wire spacing.
- For each room (i, j), use the method proposed in [**?**] to calculate the congestion value. Then we can get vertical congestion and horizontal congestion denoted by $cong_v$ and $cong_h$ respectively.
- Expand the room to different size according to the routable space pencentage of each room.
- Calculate and update the new positions of the devices by computing longest paths in the horizontal and the vertical constraints graphs.

3.4.1 Blockage-Aware Congestion Estimation

There are many methods to calculate the congestion values. Jiang *et al* [**?**] proposed a method to estimate congestion by calculating the percentage of congested bins in the bounding box of a net. Sham [**?**] also proposed a method to do estimation of congestion. A 3-steps congestion estimation methodology is used in [**?**] starting with a preliminary estimation, detailed estimation and congestion redistribution are performed repetitively.

We use the detailed congestion estimation model proposed by Sham [**?**]. It is one of the best efficient approaches to estimate congestion. The authors use a three steps estimation methodology to obtain a relatively accurate congestion estimation. We calculate the probability of it passed by other vertical (horizontal) nets, and compare it with vertical (horizontal) wire capacities. If it's smaller than the wire capacity, then there are still enough space to let other nets pass. If it's larger than the wire capacity, then this tile is over congested. This preliminary estimation can find out which regions have a high possibility to be over congested.

We proposed a novel blockage-aware congestion distribution method in which the horizontal and vertical congestion measures over a blockage will be distributed to the surroundings appropriately, since wires cannot be routed over the blockage. Assume that we have a grid $G_{(i,j)}$, and have already obtain the horizontal and vertical congestion measures $cong_h[i][j]$ and $cong_v[i][j]$, respectively. If the grid at (i, j) is occupied by a blockage B, it means that no more wires can go through the grid $G_{(i,j)}$. Then it's horizontal and vertical congestion measures should be distributed to the surrounding grids that still have spaces for wires. We classify the grids into two categories:

- Boundary grid: the grid that lies on the left or right (top or bottom) boundary of a blockage.
- Inside grid: the grid that lies inside the boundary of a blockage.

For each grid covered by blockages, the vertical congestion distribution and the horizontal congestion distribution will be performed respectively. Assume that the x-coordinates of the left and right boundaries of a blockage B are L and R. Denote by B and T the y-coordinates of the bottom and the top boundaries of blockage B. We will distribute the horizontal congestions to the bottom and top neighboring grids according to the distances between the grid and $G_{(i,j)}$. The percentage of

distribution is inversely proportional to the distance. That is, the vertical congestion of $G_{(i,j)}$ will be distributed to left and right side. Suppose that $G_{(i,j)}$'s vertical congestion value is $cong_v[i][j]$, we will distribute $(\frac{R-i}{R-L})cong_v[i][j]$ to the left neighboring grid. The remaining $(\frac{i-L}{R-L})cong_v[i][j]$ will be distributed to right neighboring grid of right boundary. Similarly, we will distribute the horizontal congestion $cong_h[i][j]$ to the bottom and top neighboring grids. The top neighbor grid will get $(\frac{j-B}{T-B})cong_h[i][j]$, and the bottom neighbor grid will get $(\frac{T-j}{T-B})cong_h[i][j]$. After all these distributions, the horizontal and vertical congestions of $G_{(i,j)}$ is 0. Figure 3.11 illustrates how the congestion measures in the middle part of a blockage can be distributed to the sides.

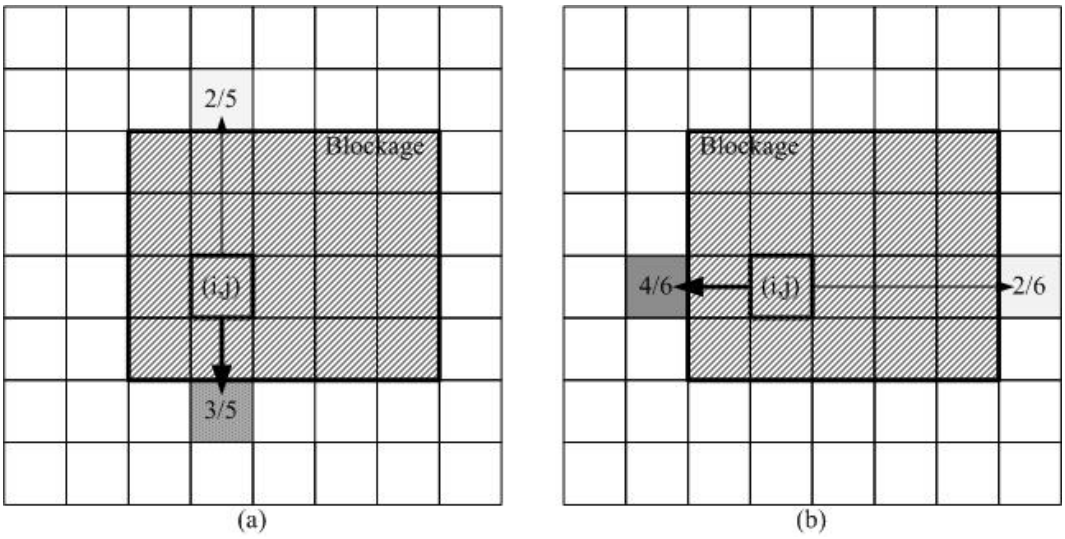


Figure 3.11: Congestion Redistribution: (a) Vertically Congestion Redistribution; (b) Horizontally Congestion Redistribution.

Figure 3.11 shows a blockage that covers 4×5 grids. Take one grid $G_{(i,j)}$ as example, Figure 3.11(a) and Figure 3.11(b) show how to redistribute congestion vertically and horizontally. In Figure 3.11(a), we distribute horizontal congestion $cong_h[i][j]$ to the bottom and top. Let the distance between $G_{(i,j)}$ and the top neighboring grid be 3, and the distance to the bottom neighboring grid be 2. Then

$\frac{2}{5}cong_h[i][j]$ will be distributed to the top neighbor grid, and $\frac{3}{5}cong_h[i][j]$ goes to the bottom. Similarly, vertical congestion $cong_v[i][j]$ is distributed to the left and right neighbors using the same method.

The methodology can be described by the following pseudo code of Algorithm.

Algorithm 1 Distribution of Congestion over Blockages

1: **for** each blockage B **do**
2: **for** each grid $G = (i, j)$ occupied by B **do**
3: **if** G on the left boundary of B **then**
4: $cong_v[i-1][j] += cong_v[i][j]$ and $cong_v[i][j] = 0$
5: **end if**
6: **if** G on the right boundary of B **then**
7: $cong_v[i+1][j] += cong_v[i][j]$ and $cong_v[i][j] = 0$
8: **end if**
9: **if** G on the bottom boundary of B **then**
10: $cong_h[i][j-1] += cong_h[i][j]$ and $cong_h[i][j] = 0$
11: **end if**
12: **if** G on the top boundary of B **then**
13: $cong_h[i][j+1] += cong_h[i][j]$ and $cong_h[i][j] = 0$
14: **end if**
15: **if** G lies inside B **then**
16: Let L and R be the x-coordinates of the left and right boundaries of B respectively.
17: $cong_v[L-1][j] += cong_v[i][j] \times \frac{R-i}{R-L}$
18: $cong_v[R+1][j] += cong_v[i][j] \times \frac{i-L}{R-L}$
19: $cong_v[i][j] = 0$
20: Let B and T be the y-coordinates of the bottom and top boundaries of B respectively.
21: $cong_h[i][T+1] += cong_h[i][j] \times \frac{j-B}{T-B}$
22: $cong_h[i][B-1] += cong_h[i][j] \times \frac{T-j}{T-B}$
23: $cong_h[i][j] = 0$
24: **end if**
25: **end for**
26: **end for**

After congestion redistribution over blockages to the boundary neighboring grids, these neighbor grids might have too much congestion. A smoothing step will be performed at the end of the procedure to get more accurate estimation of congestion. We distribute the congestion horizontally and vertically. Assume that the congestion value of grid $G_{(i,j)}$ is $cong[i][j]$. The vertical congestion of the d^{th} grid from $G_{(i,j)}$ will get an additional congestion measures of $2^{-d}cong_v[i][j]$. We use the same smoothing strategy for the horizontal congestion of $G_{(i,j)}$. Pseudo code is shown in Algorithm 2.

Algorithm 2 Congestion Smoothing

1: //Smoothing the congestions:
2: **for** each column **do**
3: Find the contiguous grids whose horizontal congestion values are greater than one, then average out their horizontal congestion values within the same group of contiguous grids.
4: **end for**
5: **for** each row **do**
6: Find the contiguous grids whose vertical congestion values are greater than one, then average out their vertical congestion values within the same group of contiguous grids.
7: **end for**
8: Output the horizontal and vertical congestion values.

3.4.2 Layout Expansion

After congestion estimation and re-distribution, we will perform an expansion step on the originally compacted placement according to the estimated congestion measures. We need an appropriate congestion estimation and layout expansion step so that the expanded layout will be routable on one hand and will not lead to an

over-sized layout on the other hand. Besides, since this congestion estimation and layout expansion step will be performed in every iteration of the annealing process, a fast approach is needed.

Similar to the congestion estimation step, the originally compacted packing will be divided into a $n \times n$ grid and the routable space percentage $X[i][j]$ will be calculated for each room (i, j). The routable space percentage is the white space percentage in a room that is not occupied by any blockages. We then compute the overflow value for each room. For each room (i, j), we have already obtain congestion information from the congestion estimation step. A variable δ is defined as the difference between the horizontal (vertical) congestion and horizontal (vertical) wire capacity. We use following equations to calculate the horizontal overflow δ_h and vertical overflow δ_v, respectively:

$$\delta_h = cong_h[i][j] - c_h \times X[i][j] \tag{3.1}$$

$$\delta_v = cong_v[i][j] - c_v \times X[i][j] \tag{3.2}$$

If the horizontal (vertical) overflow δ_h (δ_v) is smaller than 0, it means that the grid has enough space for routing wires. Then the room does not need to be further expanded in the horizontal (vertical) direction. If δ_h (δ_v) is positive, it indicates that there still need at least additional δ_h of height (δ_v of width) for successful routing. Furthermore, we should update the horizontal (vertical) constraint graph, according to the expansion in the horizontal (vertical) direction. Details of this layout expansion step are given by the following pseudo code.

Algorithm 3 Placement Expansion

1: Divide the layout into a $n \times n$ mesh. Assume that the room width and height are w and h respectively.

2: Assume that the minimum wire width is z,

$c_v = h/z$; $c_h = w/z$

3: Calculate the routable space percentage X for each room.

4: Perform congestion estimation.

5: // Expand grid:

6: **for** each room (i, j) **do**

7: $\delta = cong_v[i][j] - c_v \times X[i][j]$.

$$expand_h[i][j] = \begin{cases} 0 & : \delta < 0 \\ \delta & : otherwise \end{cases} \tag{3.3}$$

8: $\delta = cong_h[i][j] - c_h \times X[i][j]$.

$$expand_v[i][j] = \begin{cases} 0 & : \delta < 0 \\ \delta & : otherwise \end{cases} \tag{3.4}$$

9: **end for**

10: **for** each device D **do**

11: Let the x-coordinates of the left and right boundaries of D be L and R. Let the y-coordinates of the bottom and top boundaries of D be B and T.

12: Find the horizontal displacement $expand_x$ by accumulating the $expand_h$ values: $expand_x = \max_{j=B}^{T}(\sum_{i=L}^{R} expand_h[i][j])$

13: Add an edge from source to D in the horizontal constraint graph with weight $expand_x$.

14: Find the vertical displacement $expand_y$ by accumulating the $expand_v$ values: $expand_y = \max_{i=L}^{R}(\sum_{j=B}^{T} expand_v[i][j])$

15: Add an edge from source to D in the vertical constraint graph with weight $expand_y$.

16: **end for**

17: Update the coordinates of all the devices by computing the longest paths in the constraint graphs.

3.5 Simulated Annealing

Simulated annealing is used in our analog placer. New types of moves and new cost function are defined in our simulated anneal approach.

3.5.1 Types of Moves

The moves in simulated annealing change the placement configuration. Besides commonly used moves such as interchanging two modules in the sequence pair, rotation, merging, etc, some new moves related to our common centroid placement for different devices are described as follows:

- Move 1: Interchange two modules in the first sequence.
- Move 2: Interchange two modules in both sequences.
- Move 3: Rotate a device or a cluster.
- Move 4: Merge or separate two devices.
- Move 5: Change the row and column numbers. This move affects the shape of the device group and the detail routing pattern for those devices.
- Move 6: Change the relative placement. A placement pattern interleaving or symmetric placement will be selected.
- Move 7: Change the orientation. This move will affect the I/O pin positions and thus the routing pattern.

3.5.2 Handling Devices in Symmetry Group

In order to reduce the placement area and parasitic effect, we introduce a technique that can change devices' position in symmetry group to shrink the size of symmetry

group. In analog design, some pairs of devices need to be placed symmetrically to get a better performance. We put them into the same symmetry group. We have two different types of symmetry devices, the *symmetry devices* and the *self-symmetry devices*. For the symmetry devices, we place them separately on both sides of 1-D symmetry axis. Self-symmetry devices have connections with devices on both sides of symmetry axis. We put them on the 1-D symmetry axis to achieve better device matching property and easier symmetry routing.

Symmetry group is good for symmetry routing and device matching. However, it might cause a large deadspace if we fail to place the symmetry devices within it properly. Figure 3.12(a) shows an example that an unnecessary deadspace is generated because of improperly placement of self-symmetry device. In order to avoid it, the devices may have chance to change there positions within the symmetry group if the deadspace exceeds a threshold, i.e., 20% larger deadspace than that of manual layout. The cost function takes the symmetry groups' area into account and gives it a higher penalty. It can help us select solution with the minimize value.

The following example illustrates the process referred above:

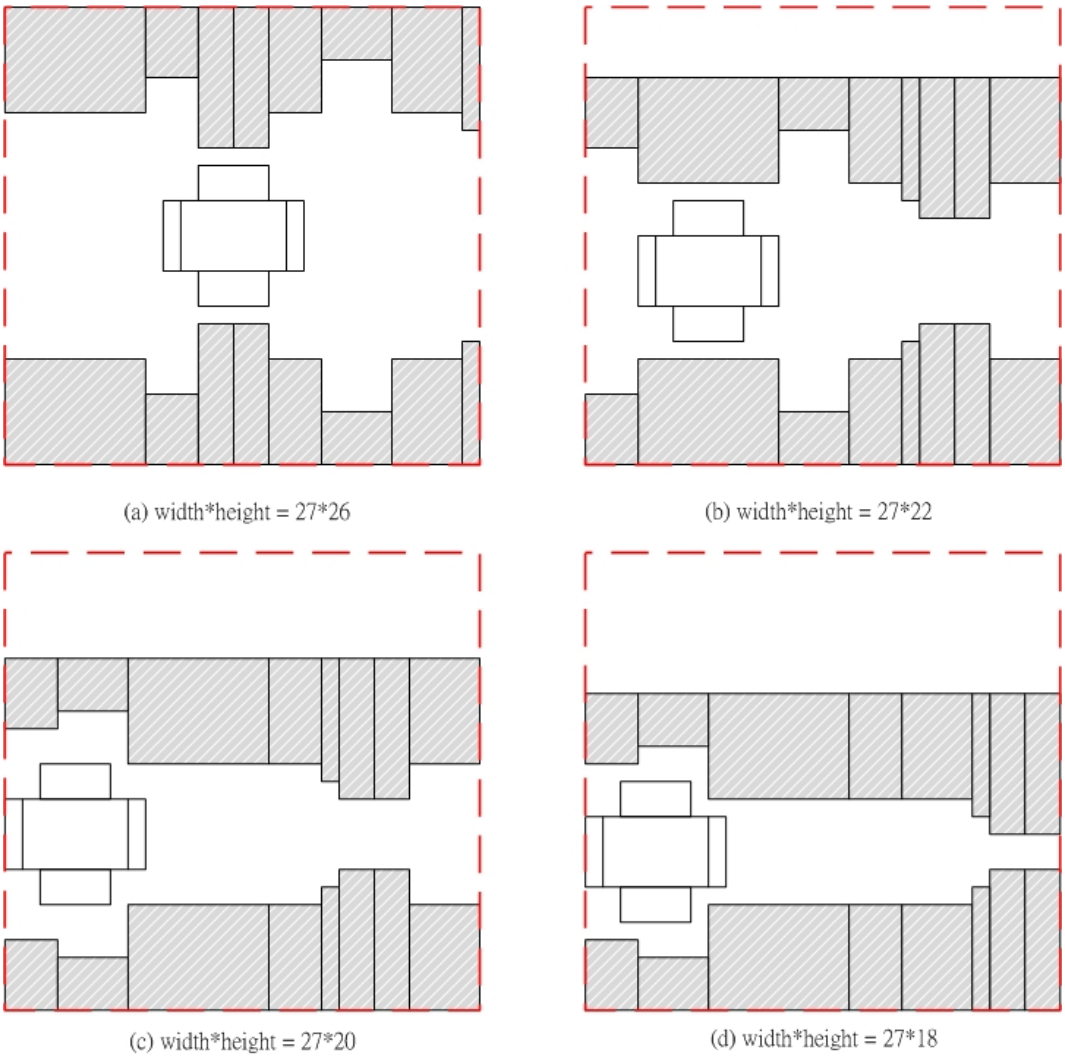


Figure 3.12: Symmetry Group Shrinking Process.

3.5.3 Cost Function of Simulated Annealing

The quality of a placement result is highly related to the cost function used. In order to further reduce the deadspace introduced by symmetry group(s), we employ a new cost function in the simulated annealing approach.

$$Cost(F) = Area(F) + \alpha Wire(F) + \beta \sum_{i \in [1...n]} Area_i(sym\ group); \qquad (3.5)$$

where $Area(F)$ is the area of the placement, $Wire(F)$ is the HPWL wire length, and $\sum_{i \in [1...n]} Area_i(sym\ group)$ is the total area of symmetry groups. The parameter α is set by running a random walk for 1000 iterations before the simulate annealing process to find a value that balances the average area and wire-length costs. The parameter β is the weight of symmetry group. Empirically, β ranges from 0.5 to 5 and it can be chosen by the users according to different requirements of different testcases. If the users are very sensitive to deadspace then they can choose a high β, and vice versa. We use the third term $\beta \sum_{i \in [1...n]} Area_i(sym\ group)$ to control the quality of the placement result. In most cases, setting $\beta = 1$ can decrease the deadspace by 5% to 20%. In our experiments, we set the value of β is equal to one.

According to the previous experimental results, a lot of deadspace was related to those self-symmetry devices in a symmetry group. Our new cost function takes all of these factors into account. In our approach, the areas of the symmetry group are considered and our simulated annealing search engine can find a placement with smaller deadspace.

3.6 Summary

In this chapter, a complete and high quality layout methodology for analog circuits is presented. Our proposed common centroid placement is integrated into the tool to reduce the mismatch errors. It offers higher flexibility and to change the placement of devices (transistors, resistors and capacitors), especially for those within symmetry group. An accurate congestion estimation model and a layout expansion technique are applied during the simulate annealing process to further improve the routability.

□ **End of chapter.**

Chapter 4

Experimental Results and Monte-Carlo Simulations

This chapter includes the experimental results of the following study:

1. Study of the effectiveness of deadspace reduction and better routability performance.

2. Study of correlate coefficients selection and different types of layouts (layouts with and without symmetry groups, layouts with and without self-symmetry groups, layouts with different numbers of symmetry groups, large and small size capacitors array).

3. Study and comparison of automatically generated layouts and manual designed layouts using Monte Carlo simulation.

4.1 Study of Congestion-driven Layout Expansion

Our proposed approach in Chapter 3 and our analog layout design tool is implemented in C++ and complied with g++ 4.3. All results and experiments are performed on an Intel ®Core ^{TM}2 Duo CPU, running at 2.20GHz with 4GB memory

Table 4.1: Testcase Information for OTA1, OTA2, OTA3 and AMP1

Data Set	Device Number					Device Area (μm^2)	Net Number	Sym Group Number
	Capacitor	Resistor	PMOS	NMOS	Total			
OTA1	20	17	12	16	65	17057.8	32	42
OTA2	16	19	12	16	63	15094.7	34	44
OTA3	50	23	12	16	101	39777.1	40	52
AMP1	14	104	11	21	150	61419.6	130	62

on Linux workstation. To demonstrate the quality of our automatically generated layout and the effectiveness of our proposed approach, three real test cases are used in our experiments. Due to a limited availability of public test cases, we test our tool with circuits designed by experienced analog designers in the department of Electronic Engineering. We compare the manual layouts and the layouts generated by us. We also compare our results with those of our previous work [**?**].

In the simulations, the UMC 0.13μm process technology is used for the analog design environment. In annealing schedule, we will start with initial temperature 10^6°C and gradually decrease it at a constant rate of 0.95. At each temperature step, the annealing iterations will be repeated k times, where k is proportional to the number of blocks. The test circuits are Operational Trans-conductance Amplifiers (OTA) and Amplifier (AMP). Table 4.1 lists the circuit information, such as the number of devices, number of symmetry groups and number of nets.

We design a set of experiments to show how our methodology can reduce deadspace and improve routability using three test cases (OTA1, OTA2, OTA3). For each test case, key variables such as *via*, *wire length* and *deadspace* are measured. We compare these values with the corresponding results obtained without using this technique [**?**]. We run each test case 20 times to get the deadspace and wire-length information and compare them with the results of [**?**]. In order to test the improvement in routability, each test case is run 100 times. The numbers of successful and unsuccessful routing are recorded:

Table 4.2 shows the results of our new congestion estimation and layout expansion technique make improvement for the testcase OTA 1. We can reduce the deadspace percentage, minimize the wirelength and decrease the number of vias using our technique. We obtain a result of 0.91%, 0.97% and 0.92% for deadspace, wire length and via respectively, compare with our previous work. Similar to OTA 1, the experiments on OTA 2 also get all factors decrease. It leads to the biggest improvement of deadspace, which is 17% better than before. Improvement also occurs on the wirelength (2%) and vias (5%), respectively. OTA 3 has the largest number of devices and circuit size. Experimental results show that our approach can make all required factors decreased, see Table 4.4.

Besides considering the circuit area and wire length, routability is also a very important issue. The quality of placement will affect the routing process significantly. We test on 3 test cases (OTA 1, OTA 2 and OTA 3) to verify the routability of our placement. Compare the routabiligy of the placements generated from our tool (with congestion estimation and layout expansion technique[1]) and those from the previous work [**?**]. In the experiments, we use a variable *seed* to generate different initial placement. Then we use our router to perform routing on those placements to obtain the number of successful routing out of 100. We also show the running time, area, wire length and via information for reference.

Table 4.5 shows that our new approach can improve the routability property for all 3 test cases. The routability of OTA1, OTA2 and OTA3 are improved by 7%, 8% and 4%, respectively.

[1]Detail about these techniques is described in Section 3.4

Table 4.2: Testcases OTA1: Routing Result Comparison.

	Previous Approach [?]			Our Approach					
	Dead Space	Wire Length (μm)	Via	Dead Space$'$	$\frac{DeadSpace'}{DeadSpace}$	Wire Length$'$	$\frac{WireLength'}{WireLength}$	Via$'$	$\frac{Via'}{Via}$
1	25.15%	5283	505	21.79%	0.87	5400	1.02	593	1.17
2	26.52%	5344	479	25.65%	0.97	5583	1.04	479	1.00
3	26.23%	5044	452	17.55%	0.67	5082	1.01	343	0.76
4	10.52%	4279	442	14.55%	1.38	4726	1.10	294	0.67
5	17.33%	5148	425	23.31%	1.35	5986	1.16	419	0.99
6	17.22%	4927	517	14.51%	0.84	4334	0.88	432	0.84
7	17.40%	5128	515	26.55%	1.53	4065	0.79	396	0.77
8	15.77%	4512	365	11.44%	0.73	3342	0.74	395	1.08
9	27.49%	5774	386	16.15%	0.59	5174	0.90	401	1.04
10	18.82%	4126	416	15.78%	0.84	4551	1.10	370	0.89
11	21.98%	4521	522	18.01%	0.82	4298	0.95	388	0.74
12	29.31%	4955	487	25.49%	0.87	4879	0.98	429	0.88
13	24.59%	5269	501	18.22%	0.74	5078	0.96	472	0.94
14	18.27%	4530	484	17.39%	0.95	4331	0.96	473	0.98
15	15.94%	5214	497	15.01%	0.94	4809	0.92	452	0.91
16	26.39%	4893	510	21.49%	0.81	4559	0.93	503	0.99
17	25.14%	4925	498	18.79%	0.75	4887	0.99	469	0.94
18	18.29%	4539	457	15.63%	0.85	4519	1.00	420	0.92
19	19.35%	5017	480	17.25%	0.89	4877	0.97	460	0.96
20	24.58%	4871	429	18.34%	0.75	4621	0.95	435	1.01
Average	21.31%	4915	468	18.65%	**0.91**	4755	**0.97**	431	**0.92**

Table 4.3: Testcases OTA2: Routing Result Comparison.

	Previous Approach [?]			Our Approach					
	Dead Space	Wire Length (μm)	Via	Dead Space$'$	$\frac{DeadSpace'}{DeadSpace}$	Wire Length$'$	$\frac{WireLength'}{WireLength}$	Via$'$	$\frac{Via'}{Via}$
1	27.32%	4148	443	14.49%	0.53	3277	0.79	357	0.81
2	43.82%	3023	380	42.19%	0.96	3464	1.15	386	1.02
3	22.79%	2940	403	17.48%	0.77	2863	0.97	309	0.77
4	30.55%	3894	382	29.09%	0.95	3870	0.99	445	1.16
5	27.58%	2925	349	29.20%	1.06	4853	1.66	403	1.15
6	31.50%	3803	461	15.52%	0.49	2995	0.79	372	0.81
7	23.15%	3894	442	18.89%	0.82	3879	1.00	346	0.78
8	23.35%	2843	405	27.36%	1.17	3516	1.24	344	0.85
9	34.63%	3776	406	30.72%	0.89	3411	0.90	357	0.88
10	44.33%	3823	383	42.34%	0.96	3212	0.84	425	1.11
11	32.64%	4019	421	28.14%	0.86	3987	0.99	410	0.97
12	28.97%	4227	394	22.42%	0.77	4109	0.97	388	0.98
13	37.09%	4396	405	32.79%	0.88	4022	0.91	419	1.03
14	25.01%	3679	487	17.88%	0.71	3297	0.90	462	0.95
15	23.72%	4015	450	15.49%	0.65	3698	0.92	439	0.98
16	36.73%	3843	397	31.09%	0.85	3581	0.93	374	0.94
17	32.03%	3695	329	24.57%	0.77	3127	0.85	301	0.91
18	41.67%	3701	409	37.24%	0.89	3019	0.82	354	0.87
19	25.39%	4269	457	20.39%	0.80	4055	0.95	439	0.96
20	29.58%	3897	468	23.36%	0.79	3704	0.95	470	1.00
Average	31.09%	3741	414	26.03%	**0.83**	3597	**0.98**	390	**0.95**

Table 4.4: Testcases OTA3: Routing Result Comparison.

	Previous Approach [?]			Our Approach					
	Dead Space	Wire Length (μm)	Via	Dead Space$'$	$\frac{DeadSpace'}{DeadSpace}$	Wire Length$'$	$\frac{WireLength'}{WireLength}$	Via$'$	$\frac{Via'}{Via}$
1	16.21%	3024	463	14.48%	0.89	2897	0.96	387	0.84
2	21.06%	4038	486	5.86%	0.28	2722	0.67	382	0.79
3	23.62%	3667	447	16.50%	0.70	4094	1.12	408	0.91
4	13.41%	3681	443	19.21%	1.43	3734	1.01	452	1.02
5	15.45%	3043	409	10.01%	0.65	2959	0.97	376	0.92
6	17.50%	3630	458	16.51%	0.94	3697	1.02	441	0.96
7	29.80%	4319	457	30.55%	1.03	4300	1.00	470	1.03
8	25.17%	3320	396	17.87%	0.71	3106	0.94	418	1.06
9	16.56%	2727	398	10.81%	0.65	2623	0.96	367	0.92
10	19.81%	2629	348	14.20%	0.72	2842	1.08	360	1.03
11	17.35%	5534	457	25.63%	1.48	4220	0.76	475	1.04
12	36.05%	4886	342	37.20%	1.03	4461	0.91	376	1.10
13	25.46%	3856	491	19.83%	0.78	4099	1.06	365	0.74
14	25.66%	5773	262	21.65%	0.84	5096	0.88	337	1.29
15	18.97%	3448	472	25.27%	1.33	4063	1.18	493	1.04
16	22.93%	4697	464	15.37%	0.67	4834	1.03	383	0.83
17	17.72%	4356	363	20.01%	1.13	3346	0.77	346	0.95
18	55.25%	3558	458	53.59%	0.97	4106	1.15	479	1.05
19	14.02%	4186	396	9.02%	0.64	4715	1.13	353	0.89
20	10.84%	4825	454	14.60%	1.35	4986	1.03	448	0.99
Average	22.14%	3960	423	19.91%	**0.91**	3845	**0.98**	406	**0.97**

Table 4.5: Routability Test: Routing Successful Rate Comparison.

	Previous Approach [?]					Our Approach				
	Dead Space	Wire (μm)	Via	Time (s)	Routability	Dead Space	Wire	Via	Time	Routability
OTA1	21.82%	5344	460	131	**74%**	18.63%	4648	425	117	**81%**
OTA2	32.91%	3812	420	143	**71%**	28.17%	3629	407	185	**79%**
OTA3	23.75%	4530	437	476	**67%**	20.97%	3907	421	507	**71%**

4.2 Monte Carlo Simulations

A tool from Cadence Design Systems named Spectre is used for running Monte Carlo simulation [**?**]. The result of Monte Carlo simulation shows that our automatically generated layout have a better or comparable performance than manual designed layout. In this section, Monte Carlo simulation is used to carry out the following four studies:

- Study of layouts with and without symmetry group(s).
- Study of layout with and without self-symmetry devices.
- Study of layout with symmetry group as (1) one whole group. (2) being divided into several sub-groups.
- Study of layout with (1) large size capacitor arrays and (2) small size capacitor arrays.

4.2.1 Devices Modeling

Spectre Monte Carlo Simulation is employed to model different devices that need matching with each other in an analog circuit. We use the modeling method described in the guide book [**?**]. In order to run Monte Carlo simulation, we should also specify the process parameter during the manufacturing process and the statistical variation parameters. We also need to specify the correlation coefficients for the matching pairs. Matching can be classified into two major categories, *common matching* and *well matching*. In our designs, the devices that interleavely placed in a close distance with their matching pair can be modeled as well matching devices. The correlation coefficient of those devices is equal to 1. Devices in a symmetry group that are placed in 1-D symmetry can be treated as common matching de-

Table 4.6: Notations of Monte Carlo Simulation.

Notation	Description
XRSP	Mismatch parameter, $'1'$ for perfect match and $'0'$ for not match
dist	Statistic distribution
std	Standard deviation
cc	Correlate Coefficient, $'1'$ for perfect match and $'0'$ for not match

vices. The correlation coefficient for common matching devices is related to the center-to-center distance d between two matching device.

Common-Mode Rejection Ratio (CMRR) is the key parameter to measure the mismatch property of a layout. Generally speaking, a value of 70dB is adequate for amplifier output. Some high-end devices may requires higher CMRR of 120dB or even more. In our test cases, OTA1, 2 and 3 require a CMRR of 70dB, and AMP1 requires a CMRR value larger than 90dB.

4.2.2 Study of Layouts with and without Symmetry Groups

In analog layout design, designers may place devices closely to reduce circuit area and wirelength. However, asymmetric and may result in downgrading of the matching property. Placing devices in symmetry is a common method to achieve better matching property. A symmetry group may contain several symmetry pairs. We can specify symmetry group information in the specification file of our placement tool.

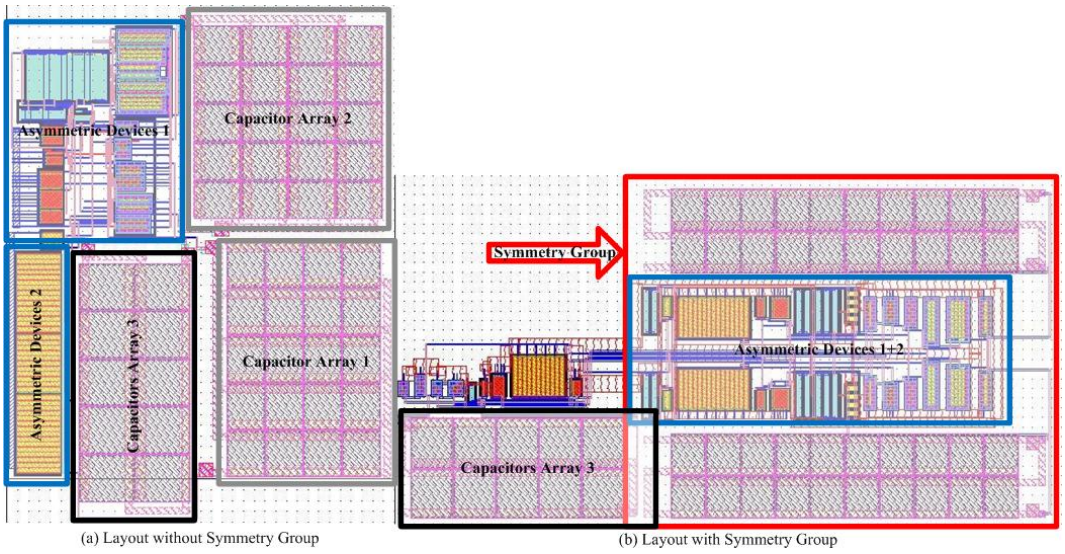


Figure 4.1: Layouts with and without Symmetry Group(s), (a) Layout without Symmetry Group, and (b) Layout with Symmetry Group

An example is shown in Figure 4.1. Figure 4.1(a) is a layout in which asymmetric devices group 1 and asymmetric devices group 2 (blue color in Figure 4.1(a)) are placed closely but without symmetry. Since devices in asymmetric group 1 and asymmetric group 2 both have connection with capacitor array 1 and 2, they can form a symmetry group. In Figure 4.1(b), devices in asymmetric group 1 and 2 are placed with respect to a horizontal symmetry axis, together with capacitor array 1 and 2 to form a new symmetry group. In our four test cases, we apply similar strategy to choose symmetric devices. According to the schematic and the connection relationship between devices.

Table 4.7 shows the simulation results of layouts with/without symmetry group. Symmetric placement and adjacent placement are trade-off when they contribute to the mismatch property. An average their is a 3.5% improvement on the CMRR.

Table 4.7: Monte Carlo Simulation Result for Layouts with/without Symmetry Group.

Layout without Symmetry Group										
Test Case	DC Gain(dB)		UGB(MHz)		Phase Margin		CM Gain		CMRR	Comp.
	Mean	SD	Mean	SD	Mean	SD	Mean	SD		
OTA1	55.50	1.20	69.40	0.30	64.30	0.80	-20.80	1.20	76.30	1.00
OTA2	66.10	0.66	65.25	0.09	57.24	1.48	-23.60	10.00	89.70	1.00
OTA3	62.49	1.07	65.71	0.57	65.59	1.80	-20.34	5.16	82.83	1.00
AMP1	82.00	0.60	26.50	0.48	58.20	1.10	-22.50	2.40	104.50	1.00
Layout with Symmetry Group										
	DC Gain(dB)		UGB(MHz)		Phase Margin		CM Gain		CMRR	Comp.
	Mean	SD	Mean	SD	Mean	SD	Mean	SD		
OTA1	57.39	1.51	68.26	0.34	65.07	1.12	-22.01	1.20	79.40	1.04
OTA2	69.99	0.92	65.70	0.18	56.93	1.97	-23.47	8.98	93.46	1.04
OTA3	61.73	1.98	67.04	0.74	60.90	1.07	-21.40	5.16	83.13	1.00
AMP1	85.18	0.92	27.13	0.70	56.84	1.21	-25.73	2.59	110.91	1.06

4.2.3 Study of Layouts with and without Self-Symmetry Devices

Self-symmetry devices are part of a symmetry group. We can specify some particular devices as self-symmetry devices in a symmetry group. Figure 4.2 shows layouts that with and without self-symmetry device. Device group 1 and device group 2 (in red color) are placed independently in Figure 4.2(a). They have connections with the upper and lower half of a symmetry group. In order to study the effect of having self-symmetry devices, we form them as a self-symmetry devices in the symmetry group (red rectangle in Figure 4.2(b)). For our four test cases, we select transistors and form them in a self-symmetry group because they have connections with resistor and capacitor symmetry pairs.

Table 4.8 compare the Monte Carlo simulation results of layouts with and with-

out self-symmetry group.

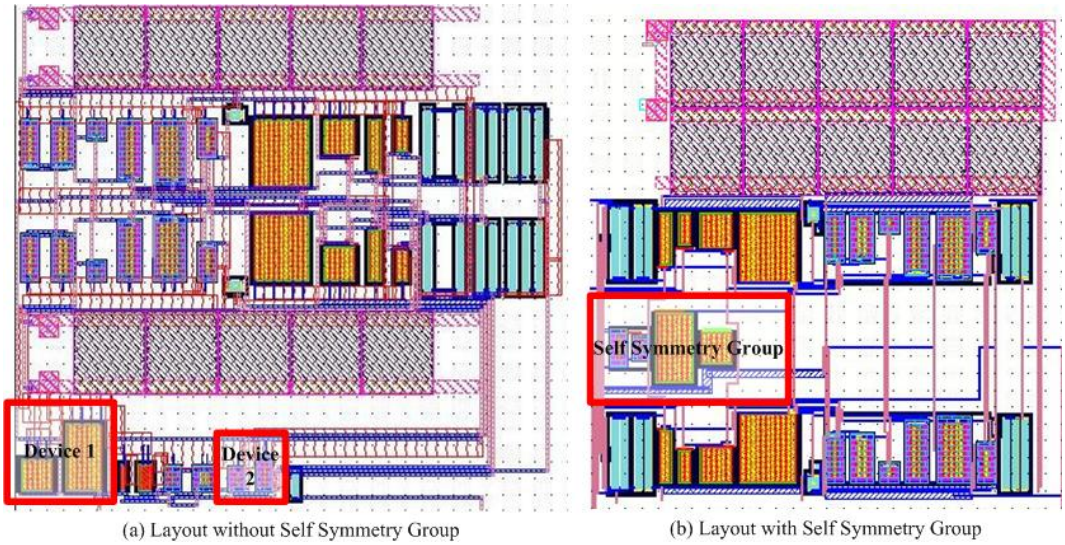


Figure 4.2: Layouts with and without Self-Symmetry Group(s), (a) Layout without Self-Symmetry Group, and (b) Layout with Self-Symmetry Group

4.2.4 Study of Layouts with Different Number of Symmetry Groups

In this section, we want to study the effect of having different number of symmetry groups. Having small symmetry groups will allow higher flexibility in placement and deadspace might be reduced.

We test our test cases with 2, 3 and 4 small symmetry groups. For the layout with 2 symmetry groups, resistors are formed in a symmetry group, transistors and capacitors are formed in another symmetry group since the connection type of these two devices are both parallel. For the layout with 3 symmetry groups, resistors, transistors and capacitors are grouped separately. For the layout with 4 symmetry groups, we apply similar strategy with 3 symmetry groups, choose the group with the largest number of devices and divide it into two smaller ones. Fig-

Table 4.8: Monte Carlo Simulation Result for Layouts with/without Self-Symmetry Group.

Layout without Self-Symmetry Group										
Test Case	DC Gain(dB)		UGB(MHz)		Phase Margin		CM Gain		CMRR	Comp.
	Mean	SD	Mean	SD	Mean	SD	Mean	SD		
OTA1	56.30	1.45	68.92	0.54	67.31	0.75	-23.79	2.16	80.09	1.00
OTA2	66.41	0.76	66.49	1.21	59.31	1.69	-23.82	9.78	90.23	1.00
OTA3	61.39	1.92	64.95	0.83	66.94	1.39	-19.59	7.98	80.98	1.00
AMP1	79.40	0.78	28.69	0.59	59.78	2.09	-22.31	3.01	101.71	1.00
Layout with Self-Symmetry Group										
	DC Gain(dB)		UGB(MHz)		Phase Margin		CM Gain		CMRR	Comp.
	Mean	SD	Mean	SD	Mean	SD	Mean	SD		
OTA1	57.90	2.98	68.47	1.09	66.30	1.32	-24.09	1.97	81.99	1.02
OTA2	66.21	0.83	67.32	1.22	68.04	1.09	-20.91	9.97	87.12	0.97
OTA3	61.73	1.03	65.42	1.04	68.35	1.24	-20.02	6.17	81.75	1.01
AMP1	87.22	1.27	28.49	1.37	57.42	1.97	-25.30	3.90	112.52	1.11

ure 4.3 shows layouts with different numbers of symmetry groups. Figure 4.3(b) shows a layout in which the big symmetry group is divided and placed in three smaller symmetry groups (in three different colors).

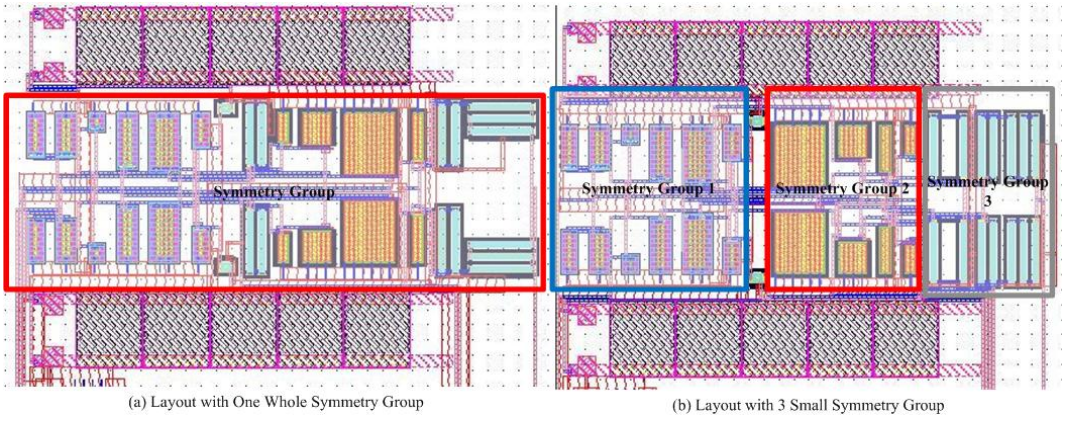


Figure 4.3: Layouts with Different Number of Symmetry Group(s), (a) Layout with 1 Whole Symmetry Group, and (b) Layout with 3 Symmetry Groups

Table 4.9 shows the simulation results of layouts with one symmetry group, and layouts with smaller symmetry groups. The number in the brackets after the name of test case is the number of symmetry groups, e.g.: OTA2(3) means OTA2's with 3 sub-groups. Simulation results show that when we divide a symmetry group into smaller ones, the CMRR will be reduced.

4.2.5 Study of Large and Small Size Capacitors Array

As mentioned in Section 3.3.3, we divide capacitors into smaller pieces and place them in different orientations and relative positions. We split each capacitor into m (m is an even number) small ones. Here, we want to study whether a big m or a small m is better. An example is shown in Figure 4.4. Figure 4.4(b) and (c) show two other choices of implementing the capacitors.

Table 4.9: Monte Carlo Simulation Result for Layouts with Different Number of Symmetry Group(s).

Layout with One Whole Symmetry Group										
	DC Gain(dB)		UGB(MHz)		Phase Margin		CM Gain		CMRR	Comp.
	Mean	SD	Mean	SD	Mean	SD	Mean	SD		
OTA1	57.39	1.51	68.26	0.34	65.07	1.12	-22.01	1.20	79.40	1.00
OTA2	69.99	0.92	65.70	0.18	56.93	1.97	-23.47	8.98	93.46	1.00
OTA3	61.73	1.98	67.04	0.74	60.90	1.07	-21.40	5.16	83.13	1.00
AMP1	85.18	0.92	27.13	0.70	56.84	1.21	-25.73	2.59	110.9	1.00
Layout with Several Symmetry Groups										
	DC Gain(dB)		UGB(MHz)		Phase Margin		CM Gain		CMRR	Comp.
	Mean	SD	Mean	SD	Mean	SD	Mean	SD		
OTA1(2)	56.48	1.09	69.77	0.50	66.02	1.36	-21.87	2.09	78.35	0.99
OTA1(3)	53.82	1.24	69.21	0.42	65.38	1.44	-21.20	2.69	75.02	0.94
OTA1(4)	52.75	2.00	70.31	0.10	66.50	2.65	-20.66	2.43	73.41	0.92
OTA2(2)	69.41	2.36	66.86	1.30	58.41	2.74	-21.49	6.34	90.90	0.97
OTA2(3)	68.24	1.23	64.80	0.94	57.92	2.00	-20.38	8.98	88.62	0.95
OTA2(4)	66.79	1.48	67.25	1.37	57.28	3.59	-20.67	7.77	87.64	0.94
OTA3(2)	61.09	1.37	65.63	2.03	64.55	0.67	-20.82	7.29	81.91	0.99
OTA3(3)	62.27	1.09	65.84	1.23	61.27	0.78	-19.96	5.16	82.23	0.99
OTA3(4)	59.05	1.20	63.58	1.97	66.09	1.24	-17.42	10.3	76.47	0.92
AMP1(2)	84.47	1.61	26.32	2.81	57.00	1.93	-19.48	1.39	103.9	0.94
AMP1(3)	79.28	1.92	30.17	1.60	54.39	0.92	-21.30	2.77	100.6	0.91
AMP1(4)	76.09	1.47	28.98	1.05	56.08	1.44	-20.07	2.59	96.16	0.87

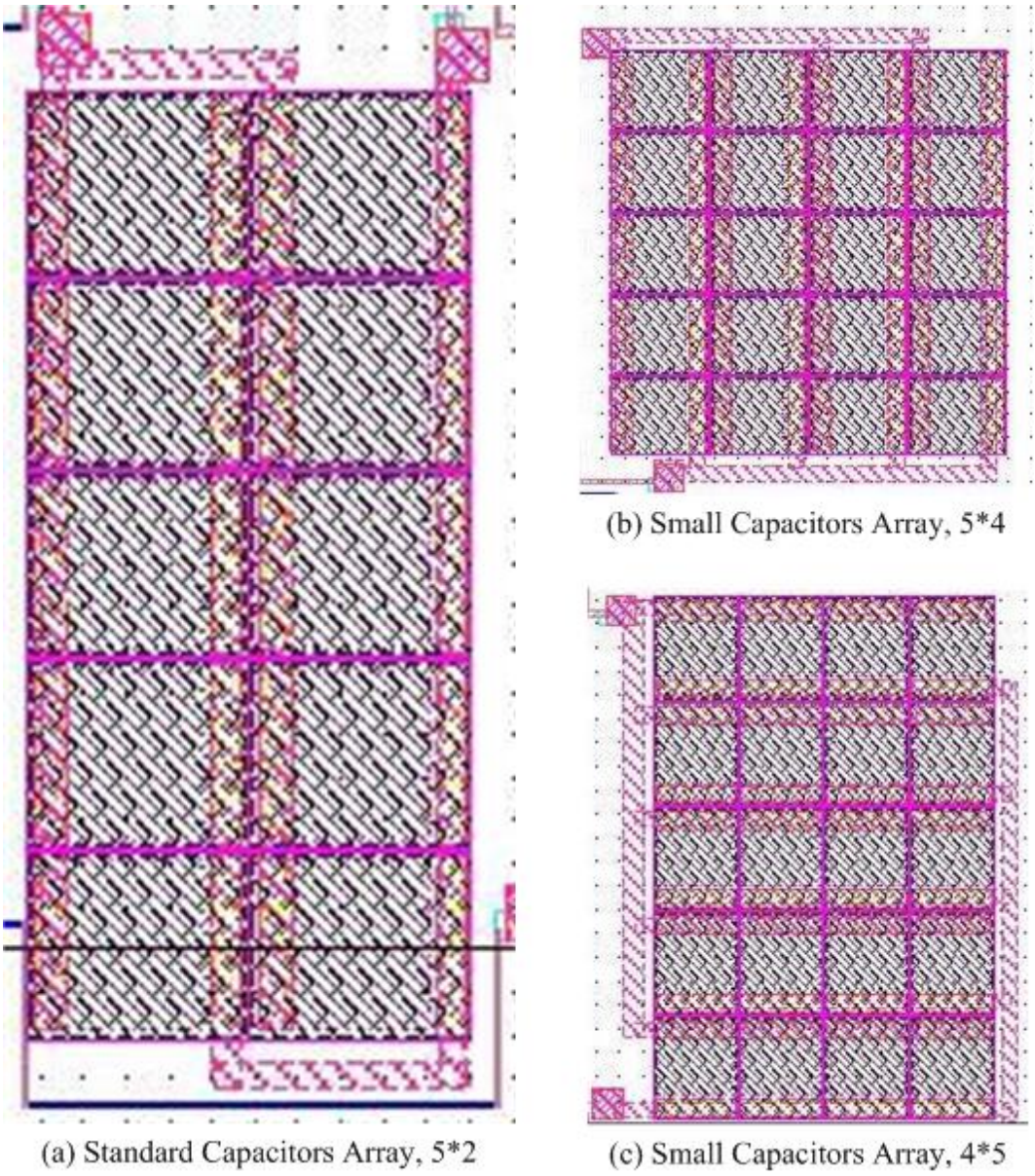


Figure 4.4: Different Placement of Capacitors Array, (a) Standard Capacitors Array, 5×2, and (b) Small Capacitors Array, 5×4, and (c) Small Capacitors Array, 4×5

Table 4.10 shows the simulation results comparing with different capacitors arrays. Experimental results show that a bigger value of *m* results in higher CMRR

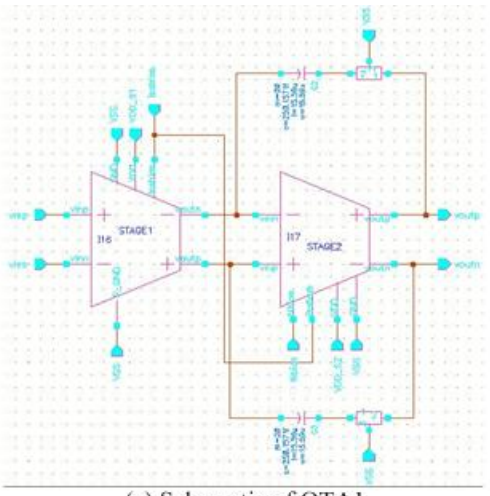
Table 4.10: Monte Carlo Simulation Result for Layouts with Different Capacitors Arrays.

Standard Capacitors Array										
	DC Gain(dB)		UGB(MHz)		Phase Margin		CM Gain		CMRR	Comp.
	Mean	SD	Mean	SD	Mean	SD	Mean	SD		
OTA1	57.39	1.51	68.26	0.34	65.07	1.12	-22.01	1.20	79.40	1.00
OTA2	69.99	0.92	65.70	0.18	56.93	1.97	-23.47	8.98	93.46	1.00
OTA3	61.73	1.98	67.04	0.74	60.90	1.07	-21.40	5.16	83.13	1.00
AMP1	85.18	0.92	27.13	0.70	56.84	1.21	-25.73	2.59	110.91	1.00
Small Capacitors Array ($\frac{1}{4}$ original size)										
	DC Gain(dB)		UGB(MHz)		Phase Margin		CM Gain		CMRR	Comp.
	Mean	SD	Mean	SD	Mean	SD	Mean	SD		
OTA1	60.21	1.39	66.94	0.58	64.79	1.49	-25.48	0.89	85.69	1.08
OTA2	70.13	1.59	66.34	0.43	59.86	2.41	-24.48	10.40	94.61	1.01
OTA3	65.06	1.24	68.37	0.77	58.43	1.30	-23.94	4.78	89.00	1.07
AMP1	84.20	0.71	28.50	0.83	51.02	1.58	-26.82	3.95	111.02	1.00

on average 4.0%.

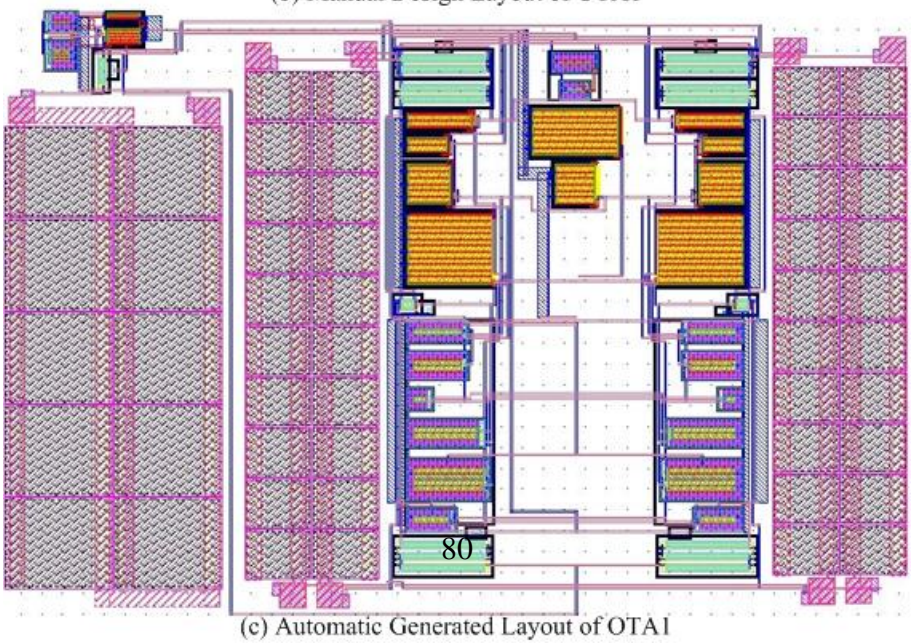
4.3 Comparison of Automatic and Manual Layouts using Monte Carlo Simulations

Figure 4.5, 4.6, 4.7 and Figure 4.8 shows the result of OTA1, OTA2, OTA3 and AMP1, respectively. In each figure, we draw schematic, manual design layout and layout generated by our tools for comparison.

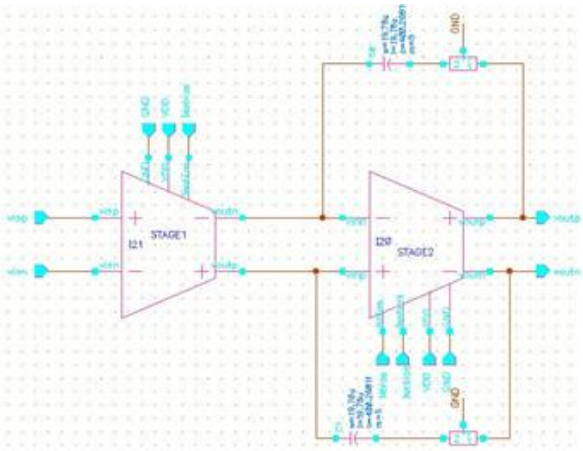


(a) Schematic of OTA1

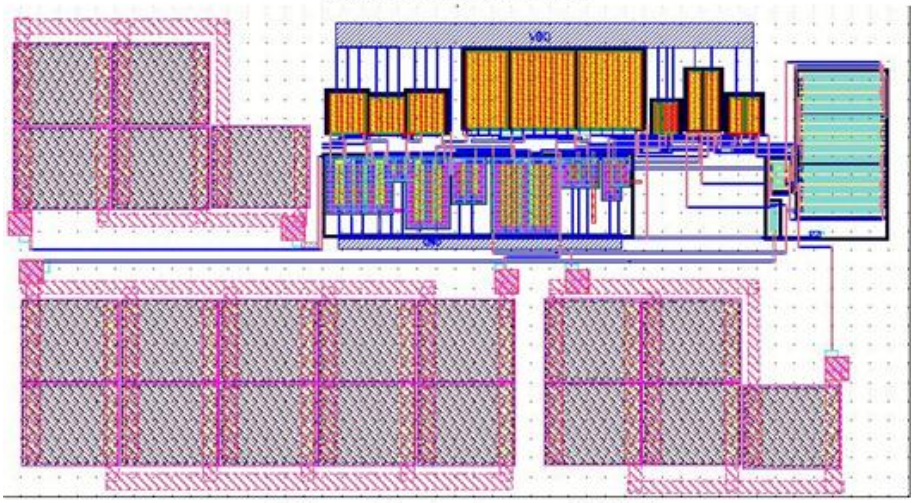
(b) Manual Design Layout of OTA1



(c) Automatic Generated Layout of OTA1

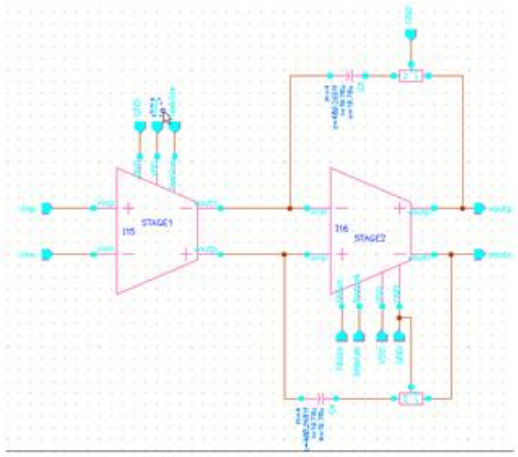


(a) Schematic of OTA2



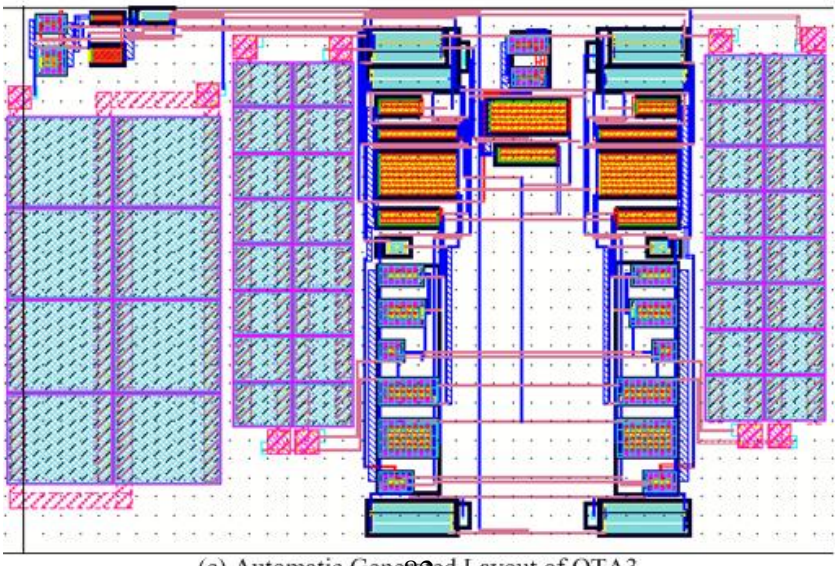
(b) Manual Design Layout of OTA2

(c) Automatic Generated Layout of OTA2



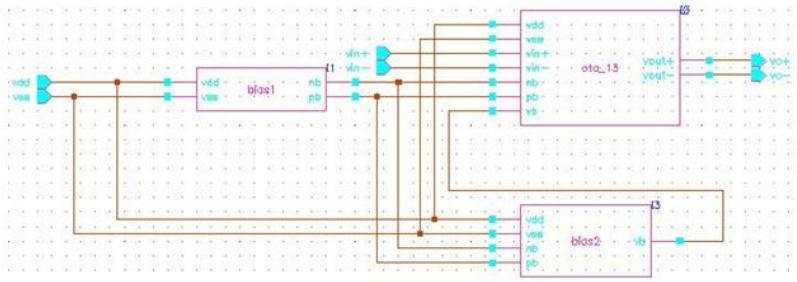
(a) Schematic of OTA3

(b) Manual Design Layout of OTA3

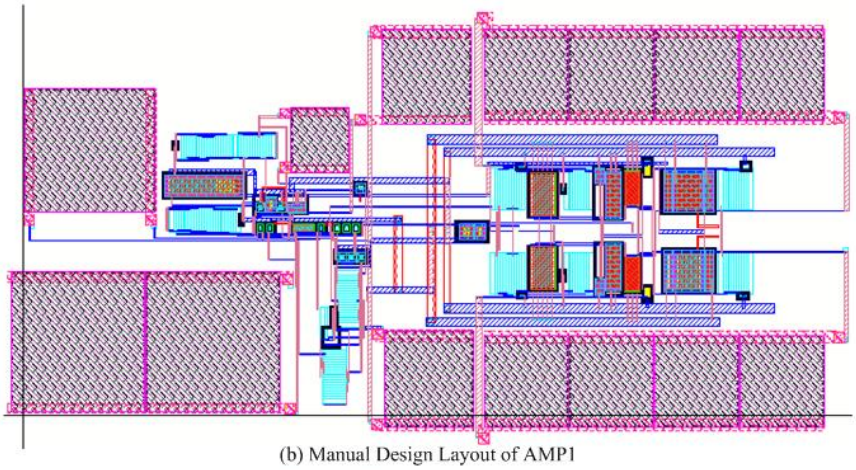


(c) Automatic Generated Layout of OTA3

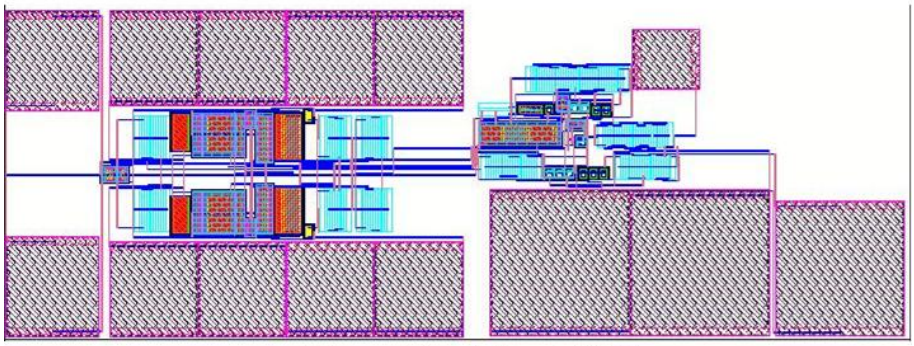
Figure 4.7: Schematic and Layouts of OTA3: (a) Schematic, (b) Manual Design



(a) Schematic of AMP1



(b) Manual Design Layout of AMP1



(c) Automatic Generated Layout of AMP1

Figure 4.8: Schematic and Layouts of AMP1: (a) Schematic, (b) Manual Design Layout, (c) Automatic Generated Layout.

Our tool uses different placement strategies on four test cases. For OTA1 (Figure 4.5), we divide the capacitors into smaller pieces and have interleaving placement for the capacitors.

For OTA2, the CMOS/PMOS and resistors form a symmetry group in our gen-

erated layout, while the manual layout does not have them symmetrically placed to save circuit space. In order to achieve a comparable circuit area, self-symmetry devices are taken out of the symmetry group to achieve a smaller deadspace. A capacitors group containing 2×5 capacitors in irregular shape is interleavely placed at both sides of symmetry axis to get a better matching property. In the manual designs, the 5 capacitors are placed in '$2+3$' format, that may cause mismatch and lead to performance degradation.

OTA3 (Figure 4.7) also has divided capacitors, but the capacitors are placed in symmetric way. Both OTA1 and OTA3, have self-symmetry devices, in the middle of the symmetry group.

For AMP1, the manual layout has symmetric placement. In our generated layout, the area of the symmetry group is further shrink with the technique in Section 3.5.2 while keeping good symmetry.

We can see that the quality of our automated layouts is comparable to the manual ones in terms of gain, unity gain bandwidth (UGB), phase margin and CMRR. Our automatic generating tool can handle symmetry constraints, common-centroid constraints and devices matching requirement while minimize the area and wirelength. However, our tool takes just a few minutes, while manual design will takes a couple of weeks to get a qualified layout.

□ End of chapter.

Table 4.11: Comparison of Simulation Results of Schematic, Manual Layout and Our Automated Layout

Data	Method	Standard Simulation					Monte Carlo Simulation (100 times)								
		DC Gain	UGB	Phase	Dead	CMRR	DC Gain (dB)		UGB (MHz)		Phase Margin		CM Gain		CMRR
		(dB)	(MHz)	Margin	Space		Mean	SD	Mean	SD	Mean	SD	Mean	SD	
OTA1	Schematic	55.90	66.80	62.40°	-	**119.0**	60.50	1.20	72.30	0.70	66.70	1.30	-39.70	2.10	**100.2**
	Manual	52.10	64.60	65.90°	38.4	**83.80**	50.90	1.20	70.50	0.90	68.30	1.00	-31.10	7.50	**82.00**
	Previous	55.60	65.90	63.10°	38.30	**90.90**	50.90	0.90	77.00	0.80	66.10	1.00	-37.20	5.50	**88.10**
	Ours	55.70	75.00	61.20°	39.90	**93.40**	59.20	1.50	69.90	1.00	64.40	0.80	-39.20	8.30	**98.40**
OTA2	Schematic	65.10	66.00	61.90°	-	**114.7**	65.90	1.20	68.50	0.90	55.30	1.20	-30.30	10.0	**96.20**
	Manual	59.30	61.90	65.20°	35.80	**101.2**	64.40	1.00	67.20	0.90	58.41	1.20	-24.90	9.40	**89.30**
	Previous	54.60	62.60	59.90°	40.90	**69.00**	59.90	0.70	60.40	0.80	60.70	0.90	-18.00	8.10	**77.90**
	Ours	61.72	65.00	68.90°	43.50	**68.20**	70.70	0.50	60.70	0.50	61.90	0.60	-20.90	2.00	**91.60**
OTA3	Schematic	60.20	67.90	67.50°	-	**91.20**	70.40	0.80	69.88	0.11	66.76	1.79	-20.90	5.92	**91.30**
	Manual	58.90	65.80	65.70°	35.80	**84.70**	67.90	0.89	68.26	0.14	67.74	1.22	-20.61	8.70	**88.51**
	Ours	57.20	65.70	67.20°	48.00	**83.50**	70.30	1.29	68.31	0.21	68.02	0.86	-20.32	7.09	**90.62**
AMP1	Schematic	59.75	26.40	79.50°	-	**123.8**	79.90	0.60	26.52	0.58	59.91	0.97	-41.93	10.1	**121.83**
	Manual	58.82	26.10	75.50°	58.70	**108.3**	82.09	0.60	26.59	0.48	58.27	1.12	-22.56	2.47	**104.65**
	Ours	59.11	25.70	81.50°	46.30	**80.81**	58.41	1.00	26.90	0.51	75.83	0.75	-44.01	8.09	**102.42**

Chapter 5

Conclusion

Technology development is scaling down in an unimaginable speed. Integrated Circuits (IC) are used in most of the modern electronic equipments today and have revolutionized the world of electronics. Functionalities become more complicated, while the sizes become much more smaller. Analog placement is an important problem in the circuit design processes. Our proposed approach can handle some practical analog placement problems. Monte Carlo simulation is employed to study the effectiveness of our methodology.

Our placement method uses simulated annealing engine in the search with sequence-pair representation. Starting with the netlist exported from Virtuoso and the device information retrieval process. Finally, a layout is generated fully automatically. A congestion estimation and layout expansion technique is used to increase the routability. Experiment results demonstrate the effectiveness of our approach and our methodology can generate high quality layout automatically with improvement of routability compare with the previous works. Our automatically generated layout also demonstrates high performance in comparison with manual design layout. Design time is also shorten significantly. A layout can now be generated in just a few minutes while manual design may take a few weeks.

□ **End of chapter.**