ROUTABILITY-DRIVEN FLOORPLANNING OF ANALOG AND MIXED-SIGNAL CIRCUITS

ZHOU HONGXIA

M.Phil

The Hong Kong Polytechnic University

2014

The Hong Kong Polytechnic University
Department of Electronic and Information Engineering

Routability-Driven Floorplanning of Analog and Mixed-Signal Circuits

ZHOU Hongxia

A thesis submitted in partial fulfillment of the requirements for the degree of Master of Philosophy

April 2014

CERTIFICATE OF ORIGINALITY

I hereby declare that this thesis is my own work and that, to the best of my knowledge and belief, it reproduces no material previously published or written, nor material that has been accepted for the award of any other degree or diploma, except where due acknowledgement has been made in the text.

(Signed)

ZHOU Hongxia (Name of Student)

Abstract

The exponential increase in scale and complexity of very large scale integrated circuits (VLSI) poses a great challenge to the present electronic design automation (EDA) techniques. As an essential step in the whole EDA layout synthesis, placement design is attracting more and more attention, especially the one for analog and mixed-signal integrated circuits. Recently, the experts in this field observe a variety of analog-specific layout constraints to obtain high-performance placements. These constraints include symmetry, alignment, boundary, preplace, abutment, range and maximum separation and additionally the routability of the placement. In order to solve this multi-objective placement problem, two different approaches are proposed in this thesis. One employs the sequence pair (SP) representation to solve the placement problem with mixed constraints. The routability of the placement is improved by performing module expansion according to the net congestion probability in the circuits. The other one applies the polish expression (PE) representation and utilizes the characteristics of the slicing structures to achieve better placement results. Experimental results on area and routability demonstrate that the two approaches are effective and feasible in solving the complex placement problem.

Acknowledgments

I would like to express the greatest gratitude to those who have supported me during the period of Mphil study.

The first and deepest thanks I would like to give to my supervisor, Dr. Bruce Chiu-Wing Sham. He helps me to understand the whole structure and details of my research. He further provides me with patient and professional guidance, invaluable suggestions and discussions during two and a half year period.

I am also grateful for the technical support from the staff in the Department of Electronic and Information Engineering for their assistance.

Finally, I am deeply grateful to my beloved family and friends. Without their support and encouragement, I cannot overcome tough situations by myself.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

Electronic design automation (EDA) is defined as a category of software tools for designing electronic systems such as printed circuit boards (PCBs) and integrated circuits (ICs). It is also considered as electronic computer-aided design (ECAD or just CAD). EDA of very large scale integrated circuits (VLSI) and systems has significantly effected the development of information technology in both computer science and engineering [2,3]. EDA has been developing solutions to support circuit design for over 40 years. During this time, with the exponential increase in the scale and complexity of circuit design, it has made great achievements from the first microprocessor (Intel 4004), with 2250 transistors, to the latest multi-core processor, with over a billion transistors. Also, EDA has accelerated the theoretical study in computation and modeling, and successfully employed them into practice. Generally speaking, EDA has completely changed the way that electronic engineers design and manufacture ICs.

In recent years, a single chip composed of a large number of transistors leads to the integration of systems on a chip (SoC), which incorporates both digital parts and analog parts. The increasing levels of integration available in silicon

technology reveals a growing demand for EDA tools. A good EDA tool can improve the quality of ICs and also increase the design productivity [4].

In most analog and mixed-signal integrated circuits, the digital parts account for 90% of an integrated circuit in terms of area. However, the designers are required to put most of the effort into the analog parts where many manual adjustments are required [5]. Thus, automated floorplanning tools for analog ICs are highly desirable from the designers' point of view. In analog and mixed-signal ICs, the designers usually have concerns about the logical correctness and the physical characteristics. The experts in this field observe a variety of analog-specific layout constraints and exploit a range of geometric optimizations to achieve these performance and density goals [6–19]. These layout constraints include alignment, abutment, symmetry pair layouts, common centroid, boundary area, preplace & range and clustering & maximum separation. Obviously, these constraints make the placement problem more complicated. In addition, the separation of placement and routing results in another issue in the place-then-route-style floorplanning. EDA designers concern that how they can estimate accurately the routing space to leave around each module for routing. Over-estimation results in open space, while under-estimation creates too many blockages for routing and placement is required to be done again. Thus, in order to ensure that the routing space can be appropriately estimated, we should consider the routability [1, 11, 20–26] during the floorplanning stage.

1.2 Contribution

In order to address the automated floorplanning issues, it is necessary for us to handle the specified constraints to achieve a satisfying floorplan and simultaneously deal with the routability problem. Two efficient methods are proposed in this thesis that are based on sequence pair (SP) [27] and polish expression (PE) [28], respectively. One applies the methodology presented in [14] and generalizes it to solve the placement problem with mixed constraints and the routability issue. All the required constraints are satisfied by inserting dummy nodes and constraint edges into the constraint graphs. At the same time, the routability of the placement is increased by adjusting the dimensions of corresponding modules according to the net congestion. The other method employs the polish expression and extends the Wong-Liu algorithm [28] to achieve a slicing floorplan. Regularity concept is introduced in this method in order to improve the routability of the placement result. It successfully addresses the constraints for mixed-signal integrated circuits such as symmetry, boundary, clustering and alignment. It also improves the routability of the placement result by placing the modules with a large number of regular structures.

Both methods can successfully solve the mixed-constraint placement problem. In addition, enough white space can be reserved for routing without violating any constraint.

1.3 Organization

This thesis consists of six chapters. Chapter 2 gives a background introduction about EDA technologies for mixed-signal ICs, including its applications and challenges at present. This chapter also presents the analog and mixed-signal

IC design flow and a general introduction about each stage of the layout synthesis. Chapter 3 concerns itself with the placement constraints. It also gives the introduction about two geometric representations and one optimization algorithm employed in this thesis. Chapter 4 focuses on a congestion-oriented approach to dealing with the placement problem for analog and mixed-signal ICs. Chapter 5 covers another method that is proposed to solve the same problem, placement with considering layout constraints and routability simultaneously. Chapter 6 gives conclusions, which mainly includes the overall thesis briefly and the further work.

Chapter 2

EDA of Analog and Mixed-Signal ICs

2.1 Overview

With the rapid development in IC manufacturing technology, IC industry has presented a boom unprecedented in history from a single transistor of the $50's$ to ICs accommodating billions of transistors of the present day. The growing complexity and scale in ICs has brought along design problems as well. These problems can not be totally solved by hiring more engineers, but increasing designing productivity could be the best answer here. This has been possible only by the prevalent use of EDA tools.

In recent years, System-on-Chip (SOC) is becoming more common in modern VLSI industry because the designers would like to put all electronic components for a system into one single chip. Analog circuits generally account for only a small part of the components on these SoC designs or other emerging mixed-signal ICs, since digital parts are more easily handled and the previous trend is to use digital computations in place of analog functions (e.g., replace analog filtering with digital signal processing). But, in many occasions, analog circuits are necessarily needed in electronic systems for reasons of cost and

performance. Firstly, the input side and output side of a system must be analog circuits. For the input side, the signals received through a sensor (e.g., antenna, microphone and wireline) should be amplified and filtered to improve the signal to noise/distortion ratio for digitalizing. The typicallly used analog circuits include variable-gain amplifiers, low-noise amplifiers, oscillators, mixers and filters [4]. For the output side, the digital-to-analog-converted signal must be enhanced in order to drive the external load (e.g., wireline, doudspeaker, antenna, actuator) with less distortion. The typically used analog circuits include filters, buffers, drivers, oscillators and mixers. Secondly, the true mixed-signal circuits of a system which connect the mentioned analog circuits with the digital-signal-processing part remain to be analog circuits. The typically used circuits include the sample-and-hold circuits, analog-to-digital converters, digital-to-analog converters, phase-locked loops and frequency synthesizers. What's more, stable absolute references (generated by voltage and crystal oscillator or current reference circuits) are required for the above circuits. Finally, the largest analog circuits at present are high-performance (low-power, high-speed) digital circuits, such as advanced microprocessors, most of which are custom-sized like analog circuits, to push power or speed limits.

As mentioned above, analog circuits play a crucial role in every electronic application which interfaces with the external systems. When analog and digital parts exist in a system, it is necessary to integrate them together to improve performance and reduce cost. Obviously, the design complexity of today's ICs has increased greatly with circuit complexity growing: 1) an increasing number of transistors are integrated on each IC to perform both analog and digital functions, and they need to be designed together with the embedded software; 2) in order to support performance requirements and new functionalities of emerging applications, it is necessary to develop new signal processing techniques and corresponding system architectures; and 3) the expectation for

changing process technology parameters needs to be accounted for in the design cycle because of the rapid development in process technologies.

The application of EDA and verification tools are necessarily needed in order to deal with the increasing design complexity and meet the time-to-market constraint. In the digital field, EDA is well developed because a digital system can be essentially described by programming language conceptions and Boolean representation. Also, the functionality of a digital system is easily represented in algorithmic form. These advantages provide the digital system design with approaches for logically transition into automation in many aspects. Unfortunately, the progress in EDA tools for analog circuits has been significantly slower. There are several reasons for this different development between digital and analog designs. First, designers usually consider analog design as less systematic than digital design. Second, analog designers still haven't established a higher level of abstraction which can shield all the device-level and process-level details from the higher level design. In addition, analog IC design is a complicated labor, which needs specialized knowledge and rich experience on design techniques. In the analog domain, there exist a variety of circuit schematics and a large quantity of conflicting constraints. Finally, analog circuits are more sensitive to the inherent disturbances such as supply noise, substrate noise and crosstalk. Those differences from digital design are part of the reasons why the digital algorithms cannot be applied to analog EDA tools and special analog-targeted tools are needed to be developed. Therefore, analog designs are mainly achieved manually as a result of the insufficiently mature analog EDA tools. The design of analog or mixed-signal ICs is time-consuming and fallible. Though occupying a small scale on a mixed-signal IC or system, analog circuits usually play as the bottleneck in the whole design in terms of cost and time.

In today's microelectronics industry, analog EDA tools are in great demand to help designers with fast and excellent design of analog ICs due to the decreasing time-to-market restriction and the pressure for electronic products of high quality and low price. Recently, the analog designers are heavily restricted by the more and more integrated systems and the continuous pressure for technology updates and process imgrations. This design difficulty and pressure can be alleviated by analog EDA tools and they can take over a big part of the technology retargeting effort and make analog design easier to port or migrate to new technologies. In addition, analog EDA tools can assist in improving the quality of the design. Therefore, the exploration and application of analog EDA is our indispensable work in the future.

2.2 Analog and Mixed-Signal IC Design Flow

In the following section, Figure 2.1 shows the general flow of analog and mixed-signal IC design. This design flow consists of a top-down synthesis and a bottom-up verification at each level of the hierarchy. Starting from a system concept, it conducts the system design first followed by simulation and verification. Then, the system design is given as an input to the next architectural design, which is input to the circuit design, circuit layout and system layout. Every phase has a feedback loop consisting of simulation and verification. The various phases in the design process are introduced in detail in the following part.

1. Conceptual Design: As the product conceptualization phase, it is responsible for gathering specifications and developing the overall product concepts. Designers often apply mathematical tools like Matlab/Simulink during the phase. Additionally, designers need to set project management goals, which includes project planning, tracking and final product

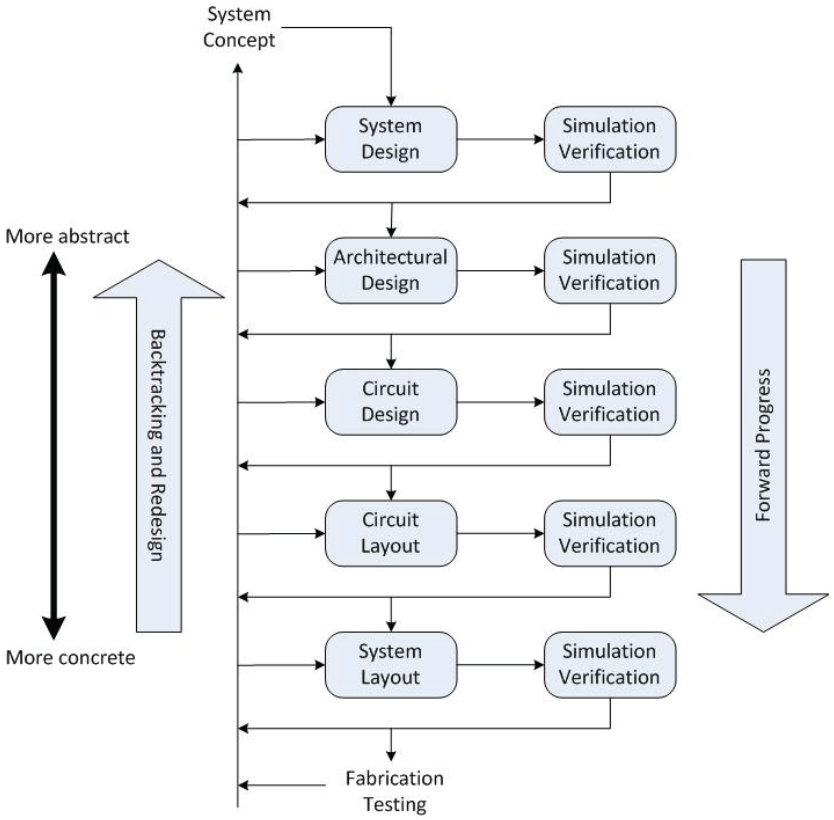


Figure 2.1: General flow of analog or mixed-signal IC design

cost.

2. System Design: As the primary phase of the actual design, it should design and partition the whole architecture of the system. It also needs to define and specify the hardware and software parts in appropriate languages. Specifically, the hardware components need a behavioral-level description and the interfaces should be given a specification. In addition, this phase should determine the implementation issues, including target technology, package selection and general test strategy. Finally, the detailed cosimulation techniques are often used here to verify the system-level partitioning and specifications.

3. Architectural Design: The architectural design phase should decompose the hardware components into an architecture composed of functional modules. Each module is required to achieve the specified behavioral description. In addition, this phase should separate analog modules from digital ones and define the specifications of the functional modules. The behavioral mixed-mode simulations are generally used here to verify the architecture of high level against the specifications.

4. Circuit Design: During this phase, analog modules should be given a concrete implementation for the given specifications and obtain their fully sized device-level circuit schematics in the selected technology process. This phase is also responsible for making decisions about the appropriate circuit topology as well as a dedicated sizing of the circuit parameters. At the same time, the complex analog modules can be further decomposed into a set of sub-modules during this process. To ensure a high yield and robustness, manufacturability considerations like mismatches

and tolerances are often taken into account. Then, the resulting circuit design can be verified by SPICE-type circuit simulations.

5. Circuit Layout: The circuit layout phase plays a role of translating the electrical schematic of different analog modules into a geometric representation in the form of a multi-layer layout. It mainly uses area optimization to generate layouts. To ensure that the performance characteristics have few negative effects from layout parasitics, it will do layout parasitic extraction and provide detailed simulations at the circuit level for the extracted circuit.

6. System Layout: During this phase, the system-level placement-and-routing and power-grid routing are used to generate the system-level layout of an IC. The analysis of crosstalk and substrate coupling should be taken into account in the mixed-signal ICs, and it is also necessary to consider proper measures such as shielding or guarding. In addition, the system layout needs to insert appropriate test structures in order to make the IC testable and it also needs to extract interconnect parasitics and perform detailed verifications like time analysis during this process. The system can be verified by co-simulating the hardware components with the embedded software.

7. Fabrication and Testing: This phase takes a task of generating the masks and fabricating ICs. Specifically, in order to elimimate defective devices, testing is executed during and after fabrication.

In the above IC design flow, any of these simulation and verification phases may detect potential problems, which may make the IC design fail to meet the target requirements. If that happens, backtracking or redesign will be done, as indicated by the upward arrow on the left-hand side of Figure 2.1.

The following section in this chapter will focus on the introduction about the analog and mixed-signal IC layout synthesis, which is also the concentration of my thesis.

2.3 Analog and Mixed-Signal IC Layout Synthesis

One of the most important stages in the design of analog and mixed-signal IC is generating the circuit layout. The layout synthesis has a better development than the circuit synthesis, largely because it can absorb ideas from the mature field of digital layout. However, the layout problem is not a simple geometric one since the layout directly affects the performance of an analog or mixed-signal circuit. It may induce parasitics, such as the parasitic wire capacitance and resistance or the crosstalk capacitance between two crossing or neighboring wires. These induced parasitics have an adverse effect on the circuit performance. Therefore, it is important to generate the circuit layout such that the final circuit can realize all performance specifications and the layout is as compact as possible. The process of layout synthesis is shown in Figure 2.2.

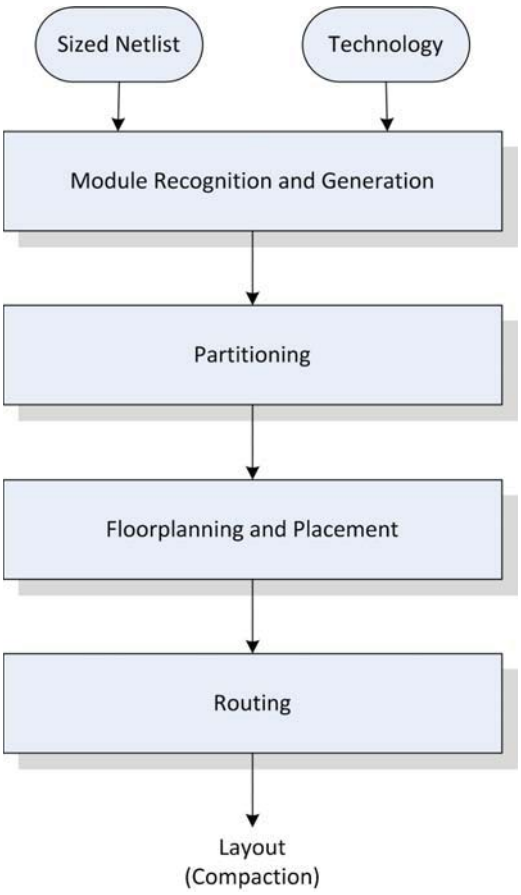


Figure 2.2: The process of analog circuit layout synthesis

2.3.1 Module Generation

Given a circuit netlist, module recognition and generation is the first stage in the layout design. It takes a task of recognizing special analog sections and generating corresponding analog modules. Module generation is simply responsible for generating a single module, or a group of closely related modules, according to the required specifications. Generated modules can be categorized into three types: transistors, resistors, and capacitors. Each group has alternative configurations in order to increase the design flexibility and avoid certain undesired effects, such as mismatch, unwanted parasitic and parameter variation.

In general, the structures of interdigitization and common centroid [29] can be applied to control the mismatching of the generated modules. At the same time, the two structures reduce relative distances between generated modules thus controlling overall parametric variation [30]. Controlling parameter for interdigitization is folding amount and the increase of folding parameter will also improve the interdigitization density. Common centroid structure is generated by distributing and placing the modules in such a way that they are in rotational symmetry about a common point. Common centroid structure is proposed for transistors and capacitors. In order to alleviate unwanted effects and improve the design flexibility, module generator also supports two more structures: folding and merging [29]. Specifically, merging is used to make two modules abut with each other, so that they can share one of their contacts, thus reducing parasitic capacitance. This method is employed to build stacks of transistors. Stacked transistors are a set of transistors with the drain node of one transistor merged with the source node of the next one. This technique makes it possible to obtain big reductions in area and parasitic capacitance.

Module generation is a straightforward automation stage and it is capable of generating modules when geometric parameters are provided. The next stage will select the best among all possible module configurations.

2.3.2 Partitioning

Partitioning is indispensable in the design automation of VLSI circuits, both in the analog and digital fields. It is a technique of breaking an IC or system into a collection of smaller components [31]. A large system is required to be decomposed into pieces so that each one can be implemented on separate interacting components. Partitioning a circuit is simply dividing the modules or components of the circuit into different groups while minimizing a certain cost, such as the number or the value of the connetions across the partitions. Partitioning has become an indispensable and central stage in today's VLSI physical design, in large part due to the enormous increase of system complexity in the past as well as the expected further advances of microelectronic system design and fabrication technology. As we know, the powerful high-level synthesis tools enable designers to automatically generate huge systems. But, synthesis and simulation tools are not always capable of handling the complexity of the whole system under development, and they need the partitioning of the huge system. Also, the partitioning technique allows designers to focus on crucial components of a system so that the design cycle could be shortened. In addition, fabrication technology makes increasingly smaller feature sizes and augmented die dimensions possible, which allows an IC to accommodate several millions of transistors. But, circuits are restricted by size and the number of outside connections. Thus, fabrication technology requires the partitioning of a large system into small sections. Finally, enormous profit can be obtained by partitioning a system optimally, because the various sections of the system can be implemented in proper methods to achieve optimal system performance,

low-cost fabrication as well as easy adaptation to changing requirements.

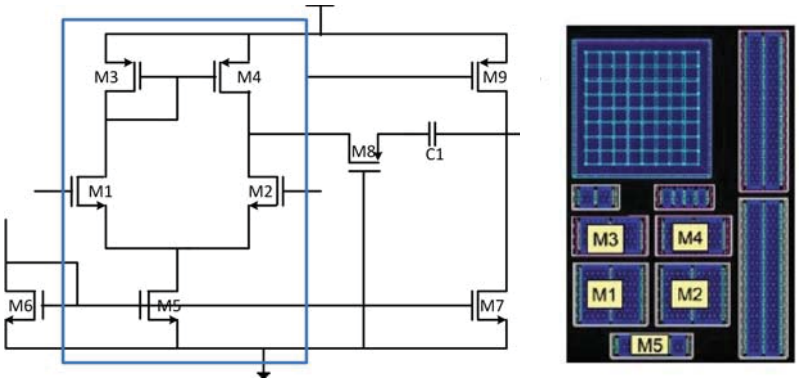
The partitioning problem has been proved to be NP-hard [32], and hence, it is very difficult to obtain the optimal solution. A large quantity of optimization algorithms have been proposed for the partitioning problem. According to the optimization strategy, the techniques are generally categorized into constructive algorithms, which are general clustering-based [33] and iterative improvement algorithms which include Tabu Search heuristic algorithm [34], simulated annealing procedure [35], genetic algorithm [36] and so on. These two types of algorithms can be combined with each other with the constructive algorithm providing a good initial point for further iterative improvement.

2.3.3 Floorplanning and Placement

In layout design, floorplanning is very closely linked with placement. It is sometimes very difficult to determine where one starts and the other ends. I will emphasis on the placement discussion as placement covers floorplanning in many cases.

Placement is a complex process and it takes a task of assigning exact positions to circuit modules within the chip area such that a certain cost is minimized. In general, this cost is simply the total layout area and the estimated interconnect length. For the digital circuits, area minimization and wirelength minimization are the major concerns in this stage. But in the mixed-signal circuit design, placement has become a fairly complicated problem, because of circuit sensitivity to substrate noise, supply noise, parasitic disturbances, crosstalk, mismatch effects, thermal gradients and so on. In order to achieve a

placement with satisfying performance, a set of essential constraints are introduced to placement to reduce these inherent negative effects. These constraints include symmetry, common centroid and other general placement constraints. Figure 2.3 gives an example about how to implement performance-related constraints in the layout synthesis. For symmetry constraint, it requires pairs of



(a) (b)

Figure 2.3: (a)The schematic of a CMOS operational amplifier, in which the differential input sub-circuit forms a symmetry group; (b)A layout design with symmetry and clustering constraints of circuit in (a)

modules to be put symmetrically to a horizontal or vertical axis. For common centroid constraint, it splits modules into a quantity of smaller sub-modules and places them in rotational symmetry about a common central point. Other placement constraints, including alignment, preplace, abutment, boundary, clustering and maximum separation, are also important for placement.

The problem of the constraint-driven placement was extensively studied and great achievements have been made [6–19]. Most previous studies used simulated annealing to floorplan the modules based on a geometric representation. Geometric representations such as sequence pair [27], B*-trees [37], O-tree [38] and TCG-S [39] have been introduced to deal with symmetry constraint [6–14]. Most of these studies addressed only symmetry constraint while [8] addressed

common centroid constraint and [12–14] addressed both. An algorithm called plantage was proposed in [12] which applied a hierarchically bounded enumeration of basic building modules and the B*-tree representation to carry out analog placement with considering symmetry, common centroid, proximity, and minimum distance constraints. Paper [13] made use of the symmetry feasible sequence pair (SP) representation to tackle both symmetry and common centroid constraints efficiently. The other general constraints were handled in [15] by SP with the constraint graphs. In [14], it proposed an approach that could deal with all the constraints simultaneously based on a global SP with center-based corner block list (C-CBL) [8]. In addition, polish expression (PE) [28] is used to respectively solve constraints like boundary [16], clustering [17], preplace [18] and range [19]. As we know, device mismatch may cause significant adverse effects to analog circuits. This kind of mismatch and other effects induced by layout can be reduced by considering constraints during placement.

In addition to the placement constraints, it is desirable to consider routability in this stage due to the technique independence between placement and routing. During the placement stage, white space is often removed from the resulting layout in an attempt to minimize a certain cost such as the chip area or wirelength. But, congestion issue derived from this behavior may deteriorate the circuit performance and lead to unroutable solutions for routing. It may also lead to timing correlation problems because detoured routes can cause mismatch between preroute timing models and post-route timing. Besides, congestion heavily restricts the flexibility of routing to optimize secondarily (e.g. crosstalk, via count, antenna rules). When facing routability problems, designers typically enlarge their floorplans, which may imply schedule delay and extra expense. Worse yet, this measurement can not ensure that the new floorplanner will yield a satisfactory solution. Considering routability earlier in

the design cycle can save substantial time and resources. Thus, it is necessary for us to consider the routability issue during placement in order to guarantee that the final placement can be routable.

There have been a number of methods that were proposed for the routability-driven placement. In general, we can categorize these previous techniques into different groups. For the first group, the routabiity component was formulated and incorporated into the placement optimizing objective. In [20], RUDY congestion estimation technique was proposed for the routaibity-driven placement and the density term was modified to incorporate both the module density and routing density. In [21], the net density was integrated into the analytical placement framework for controlling net congestion. For the second group, it mainly applied the white space allocation (WSA) or the congestion control techniques during or after placement. In [22], it proposed to distribute the white space by adjusting the partition-lines of hierarchically sliced placement according to the congestion and available white space. In [23], a new two-stage placement approach was proposed for congestion optimization. It used the extended bounding box to evaluate the routing of nets and the strategy of cell perturbation to eliminate the net congestion. In [1], it also presented a two-stage routability-driven analog placer based on ASF-B* trees [7] and HB*-trees [40] to control the routing congestion. The third group guided the placement by the global routing. IPR [24] performed a global routing to guide the placer. Some other approaches mixed some of the three features. For instance, an approach was proposed in [25] to optimize RSMT in the global placement and apply WSA in the detailed placement. In addition to these methods, in order to alleviate the congestion and improve the routability, the papers [11, 26] introduced the regular structure to the non-slicing floorplans based on sequence pair and B*-trees [38] respectively. These regular structures enable us to assign channels to the placement for routing.

As discussed above, in order to achieve high-quality layouts, both problems of the layout constraints and routability issues should be taken into account during the placement stage.

2.3.4 Routing

After the accomplishment of the placement design, routing is performed as another crucial step to bring the design for manufacturing. It aims at finding the geometric layout of all nets. VLSI routing is normally implemented through consecutive global routing and detailed routing stages. Global routing is responsible for producing approximate paths for the interconnects of a circuit by the way of finding paths on a coarse routing grid. Detailed routing is the stage where the routing layer and exact position of each net are determined.

Global routing has been proved to be a NP-hard problem and is normally tackled by heuristic algorithms. Global routing approaches are generally classified as either concurrent or sequential techniques. The concurrent algorithms do simultaneous routing for all the nets [41,42], while the sequential ones impose an ordering derived from the perceived importance of the nets on routing [43–45].

Detailed routing plays an key role in the design stage for that it is crucial for design rule satisfaction and routing completion. The quality of its solution affects a variety of design metrics including chip yield, signal integrity, timing and so on. Detailed routing has been studied extensively in past years. In [46], a detailed router was proposed for field-programmable grid arrays based on Boolean satisfiability and achieved good solution quality. In [47], it introduced track assignment as an intermediate stage between global and

detailed routing, which took a task of assigning the segments extracted from global routing to routing tracks. In [48], an efficient technique was proposed to implement escape-routing algorithm for dense pin clusters, which played as the bottleneck of detailed routing. In [49], an algorithm was proposed for the whole-chip routing, in which a combinatorial approximation scheme with min-max resource sharing was introduced for global routing and a shape-based data structure was introduced for detailed routing. Recently, [50] presented an effective detailed routing approach with applying regular routing patterns for potentially better design rule satisfaction. In order to meet all the design rules, [50] proposed an abstract idea of local optimization based on local shift and rip-up-and-reroute, assuming that most design rules were complex functions of local and neighboring geometries.

2.3.5 Compaction

Compaction is carried out after the layout has been created and it is a well-known problem in IC layout generation, whether analog or digital ICs. The compaction procedure tries to find a "rift" in the packing we get from last design phase and eliminate this "rift" from the layout, thus compacting it. However, the major limitation of compaction algorithms is that they can be successfully performed only in one dimension. A two-dimensional compaction is not necessarily the same as applying two one-dimensional compaction successively. Another problem about compaction algorithms is that they may reduce total area at the cost of destroying the symmetry constraint or other constraints that the layout generator created so painstakingly. Thus, simple compaction algorithms should first guarantee the constraints associated with circuits. In [51], it added symmetry constraint into the constraint graph that was used for compaction. It was able to make the circuit smaller and keep the

placement symmetric at the same time. However, the algorithm [51] worked only in the horizontal direction. In [52], an approach that combined the uses of Linear Programming and constraint graph was proposed for the compaction problem. The graph was created not only from topological relations as in the standard algorithm, but also from symmetry constraint as in [51].

Another variation of the compaction problem is whether the modules of the layout have many shape possibilities. If the modules are left flexible until this point or the modules are allowed to change shapes and dimensions during compaction, compaction becomes much more difficult, but much more efficient at the same time. One algorithm for this problem was presented in [53]. However, it is not practical that compaction and shape optimization are performed together, because the compaction phase is too late for changing module shapes.

2.4 Summary

In this chapter, it presented a general introduction about analog and mixed-signal IC design process. It also provided the knowledge about each stage of layout synthesis and the main algorithms applied to these stages. At the same time, my research focused on the floorplanning and placement stage, which was discussed above with a length. As it referred, a large number of excellent approaches were proposed to solve the problems of the constraint-driven placement and the routability-driven placement all the time. However, it still challenges every researcher to propose a novel method to solve the complicated placement problems with less runtime, less algorithm complexity and more excellent circuit performance.

Chapter 3

Constraints and Geometric Representations for Layout Synthesis

3.1 Overview

Floorplanning or placement design is one of the most important steps in VLSI physical design, which takes responsibility for assigning the exact positions to each circuit module on a single chip while optimizing the circuit performance. During this stage, in order to achieve high performance and reduce cost, EDA designers may want to set some restrictions on the positions of some modules in the final layout. For instance, they may try to control the separation between two circuit parts if there exist a lot of interconnections between them, or they may want to place them along the boundary of the final packing for I/O connections. This may also happen in design reuse, designers may want to keep the positions of some modules unchanged in the new placement. In addition, the analog designers are interested in a particular type of placement constraint called symmetry. Some recent literatures on the constraint-driven placement are discussed in section 2.3.3. These constraints are set for placement in order to guarantee the logical correctness and physical characteristics

of the final packing.

This chapter first presents an introduction about the layout-induced constraints. It also introduces two different placement representations and one optimization algorithm "simulated annealing" to well handle the layout constraints and effectively narrow the search space.

3.2 Placement Constraints

In an attempt to improve the circuit performance and guarantee its manufacturability, experts in EDA field treat the industrial requirements as a set of constraints to control the positions of some modules in the resulting placement. Typical placement constraints include symmetry, common centroid, and other general placement constraints, such as alignment, abutment, preplace, range, boundary, range, clustering and maximum separation.

3.2.1 Symmetry and Common Centroid

Symmetry constraint is normally denoted by symmetry groups and gives the constraint that each pair of modules in the group should be placed symmetrically to a common horizontal or vertical axis called symmetric axis as shown in Figure 3.1. The symmetry groups may accommodate self-symmetry modules, whose centers should be put on the symmetric axis. Common centroid constraint means modules belonging to a group should be put in rotational symmetry about a common central point as indicated by Figure 3.2. Before explaining why the two constraints are the most important constraints in analog circuit design, we should have some knowledge about the device matching and parasitic matching issues in analog layout [54].

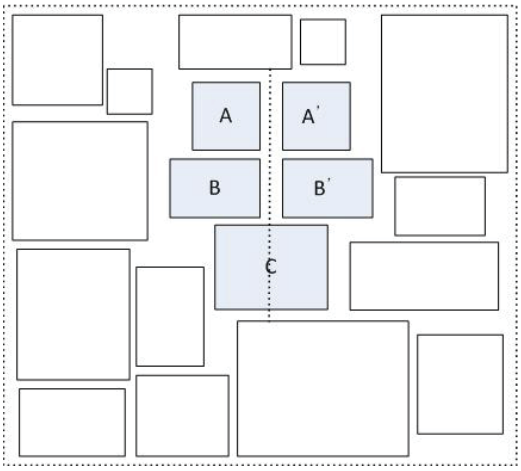


Figure 3.1: Example of symmetry constraint

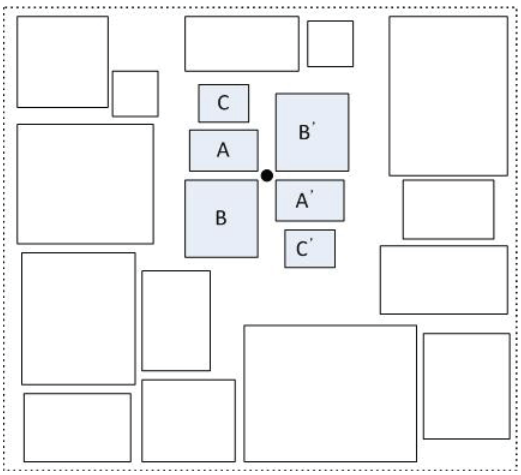


Figure 3.2: Example of common centroid constraint

In analog circuits, unavoidable variations that exist in all processes, can cause small mismatches in electrical characteristics of identical devices. When the mismatches become large enough, they may cause significant adverse effect to circuit performance through inducing electrical problems like offsets. There have been three major layout factors that are shape, orientation and separation. These factors directly impact the matching of identical devices. Device dimension is considered as a factor for that semiconductor processing leads to unavoidable distortions in the geometry that make up devices. Devices composed of identical geometry can improve matching by ensuring that both devices are subject to the same geometric distortions. Also, matching devices should be placed in the same orientation, because many processing effects introduce anisotropic geometric differences. These two factors require the identical devices to be placed symmetrically. Besides, with the separation between devices increasing, the matching characteristics of them seem to be degraded by spatial variations in process parameters. Mainly because process induces gradients in parameters such as oxide thickness or mobility. Placing matching devices in close proximity can reduce the circuit sensitivity to these effects significantly. The well-matching devices may be spatially interdigitated in order to eliminate the effects of global process gradients.

In addition, device matching, particularly of bipolar devices, exhibits a sensitivity to ambient temperature. If two such devices are placed in a random thermal gradient, a mismatch induced by temperature difference may appear. Failing to fully balance thermal couplings in a differential circuit may introduce unwanted circuit oscillation. To suppress this, it is common to place thermally sensitive matching devices symmetrically around thermally generating noise sources. Because the symmetrically placed sensitive components are equidistant from the radiating components, their roughly identical ambient

temperatures will ensure no mismatch induced by temperature disturbance.

Finally, parasitic capacitive and resistive components of interconnect can introduce matching problems in differential circuits, which consist of two matching halves. A mismatch in the parasitic capacitance and resistance between the two matching halves of the circuit may result in offsets and other electrical problems. The most powerful method for improving interconnect parasitic matching is layout symmetry that requires the placement and wiring of matching circuits to be identical, or mirror symmetry in the case of differential circuits.

As mentioned above, symmetry and common centroid arrangement for placement is useful for solving the device matching problem and reducing the layout-induced parasitics and circuit sensitivity to thermal gradients.

3.2.2 Alignment

Alignment constraint indicates that the specified modules are aligned in a row as shown in Figure 3.3. Due to bus structures or pipelines in actual VLSI circuits, alignment is designed to facilitate data transfer in bus structure or a pipeline.

3.2.3 Boundary

Boundary constraint states that some modules are required to be put along one of the four boundaries: the left, the right, the bottom, or the top boundary in the final layout as shown in Figure 3.4. During the layout design, we have to take the manufacturability into account. Thus, these modules furnished

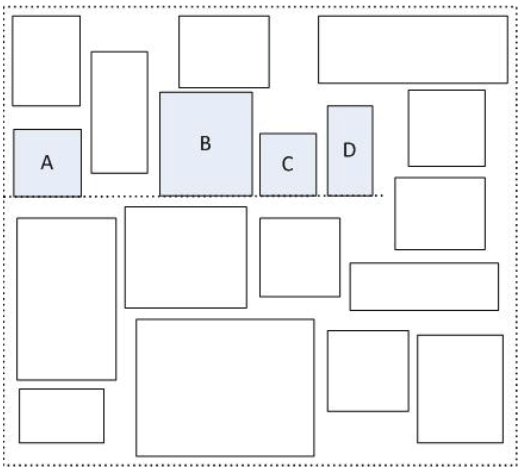


Figure 3.3: Example of alignment constraint

with input and output connections should better be placed along the boundaries such that they are easier to be connected to certain I/O ads. Besides, placement is always implemented hierarchically with grouping modules into different parts and then placement is done independently for each part on the chip. The boundary constraint is of great use when some modules need to be put on the boundary of the part in order to abut with some other modules in the neighboring parts.

3.2.4 Abutment

Abutment constraint means that specified modules should abut with each other in the resulting placement as shown in Figure 3.5, which is helpful to reduce interconnect and junction capacitances, and to obtain substantial gain in chip area. It can also be applied to merge the diffusion regions of MOS transistors or of other components, such as capacitors, BJT's, etc.

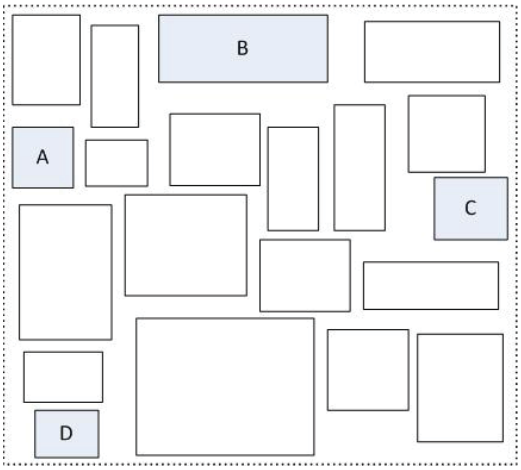


Figure 3.4: Example of boundary constraint

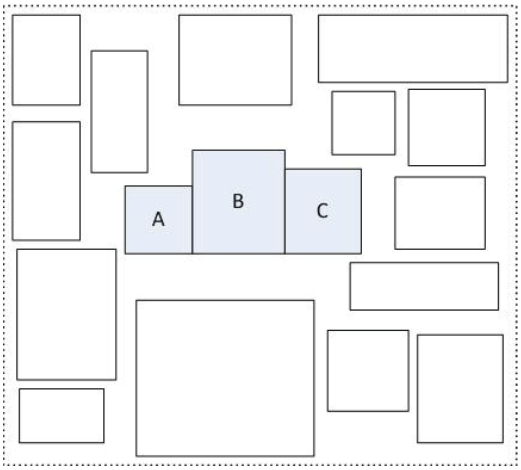


Figure 3.5: Example of abutment constraint

There are capacitance and resistance associated with the geometry of the devices themselves. In CMOS circuits, the dominant layout capacitance is generally associated with MOS gate structures. The gate area, and thus the gate capacitance, is fixed from the beginning and cannot be minimized in layout. However, there exists device capacitance which can be reduced by proper layout. For instance, the *pn* junctions, which form the MOS device source and drain regions, have a non-linear voltage dependent capacitance that is proportional to the junction area and perimeter. This capacitance can be controlled by minimizing the size of all diffusions. An large saving in diffusion capacitance can be made by device merging as shown Figure 3.6(a) and (b), in which devices are placed such that diffusion geometry is shared between electrically connected devices. This pattern of geometry sharing has the additional benefit of improving the packing density of a layout. Besides, like interconnect, each diffused structure has an associated parasitic resistance that is proportional to its aspect ratio. These resistances can be reduced by minimizing the aspect ratio of all diffusions, merging diffusions when possible. Abutment constraint can be designed here to perform this device merging. In addition, if spacing rules permit, additional capacitance and resistance can be saved by making adjacent devices abut with each other instead of explicitly wiring. Routing by abutment is indicated by Figure 3.7(a) and (b).

3.2.5 Clustering and Maximum Separation

Clustering constraint gives the constraint that some modules should be placed close to each other as shown in Figure 3.8(a) and maximum separation constraint gives an upper limit to the separation distance between each pair of modules as shown in Figure 3.8(b). Actually, clustering constraint can be

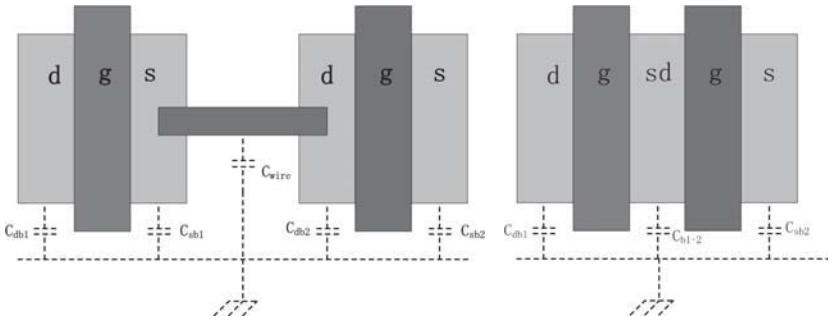


Figure 3.6: (a)Unmerged devices; (b)Merged devices with reduced parasitic capacitance

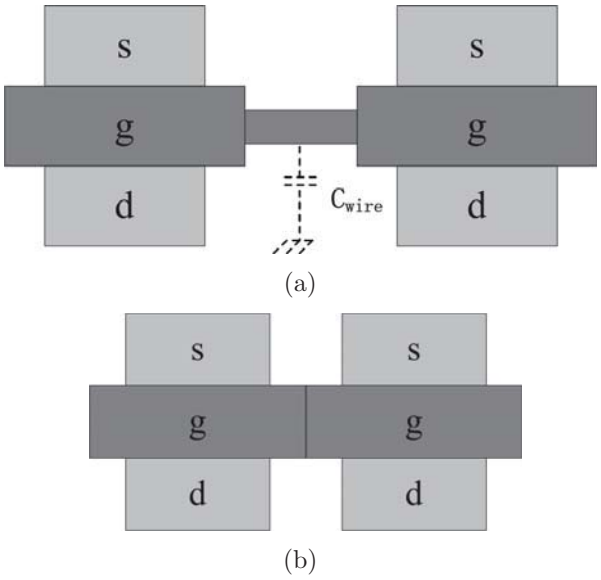
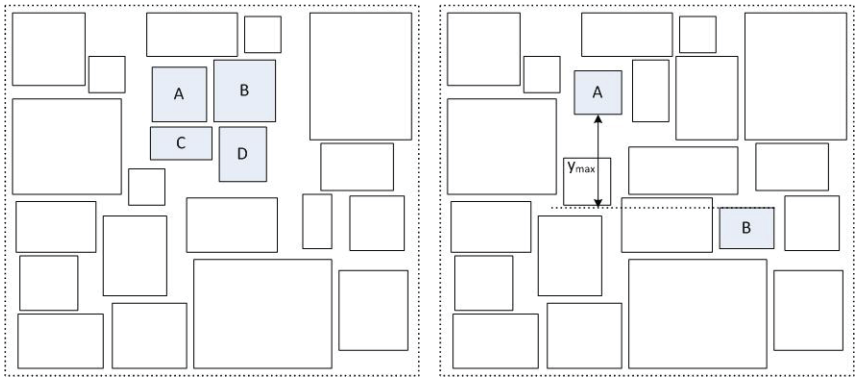


Figure 3.7: (a)Non-abutted devices; (b)Abutted devices with reduced parasitic capacitance

treated as maximum separation constraint by specifying the maximum allowable separation distance between two modules. The two constraints are necessary during placement because they can be designed to reduce the degree of electrical mismatch due to layout effects as mentioned in section 3.2.1. In addition, the nonideality of inter-device wiring introduces capacitive and resistive effects that may degrade circuit performance. Specifically, every conductor has a parasitic capacitance which is proportional to its area. Similarly, every conductor has a finite resistance which is proportional to its aspect ratio like its length to width ratio. Both parasitic capacitance and resistance can be reduced by making critical wires as short as possible. This can be enforced by placing connected modules in close proximity, which means clustering constraint here.



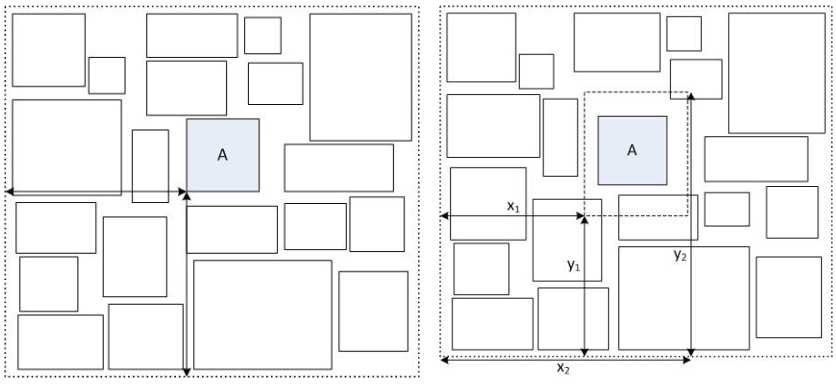
(a) (b)

Figure 3.8: (a)Example of clustering constraint; (b)Example of maximum separation constraint

3.2.6 Preplace and Range

Preplace constraint requires some modules to be preplaced and remain unchanged during the layout design as shown in Figure3.9(a). In VLSI physical design, it is common that the locations of some macro cells, such as ROM,

RAM, and central processing unit core, are fixed in a prior place and the other components of circuits need to be placed in the rest area of the chip. Similarly, in printed circuit board (PCB) design, it often happens that the exact positions of connectors are determined before designing the placement of other components. Expert designers treat these situations as a problem of placement with preplaced modules. Not only the circuit components, but also other obstacles in any type, are candidates to be regarded as preplaced modules. For instance, a substrate or holes of the substrate can be treated as preplaced modules, and the rest components are required to be placed with no overlap with these preplaced modules. In addition, the problem of floorplanning with irregular boundaries can be addressed by treating the protruding parts along the boundaries as preplaced modules. What's more, in practical industrial manufacturing, some areas should be reserved for final package. Hence, during the placement stage, we use some preplaced modules to occupy these areas to handle this problem.



(a) (b)

Figure 3.9: (a)Example of preplace constraint; (b)Example of range constraint

Range constraint requires some modules to be placed within a given rectangular region in the final placement as shown in Figure 3.9(b). Actually, it is a

general situation of the preplace constraint and the preplace constraint can be regarded as a range constraint by specifying the rectangular region such that it has the same dimension as the module itself.

3.3 Slicing and Nonslicing Placement Representations

Floorplan or placement is generally categorized into two types: slicing and non-slicing. A slicing placement, normally represented by slicing tree or polish expression [28], can be obtained by bisecting the chip area recursively with a horizontal or vertical line. The advantage of slicing placement is obvious in that its smaller encoding cost and solution space bring fast runtime and it is flexible to deal with soft, pre-placed, hard and rectilinear modules. However, optimal solutions might not be in the solution space of the slicing placement in real designs. It is only a small subset of all feasible packing and is not general enough. Due to this, a lot of efforts have been devoted to creating representations for non-slicing placement. A nonslicing placement is a placement that is not a slicing one. The non-slicing placement is efficient to represent any kind of packing and more flexible in dealing with existing constraints. Figure 3.10 illustrates the difference between slicing and nonslicing floorplans. In the following section, we will introduce two presentations respectively for slicing and nonslicing floorplans.

3.3.1 Sequence Pair (SP)

In this thesis, sequence pair (SP) [27] is used to represent a placement, which is the most popular non-slicing topological representation. In the majority

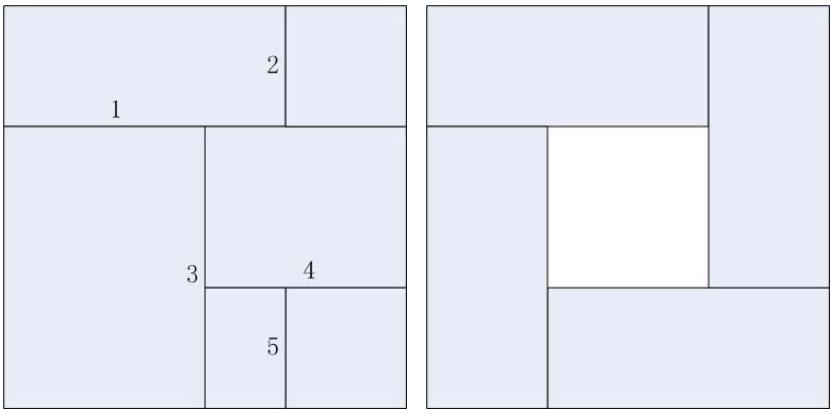


Figure 3.10: (a)Example of a slicing floorplan with a slicing order indicated; (b)Example of a nonslicing floorplan

of cases, sequence pair representation is adequate for high-performance placement, as most constraints can be handled easily. A SP (s_1, s_2) is a pair of sequences of n elements representing a list of n modules and it indicates the relationship between each pair of modules as follows:

$$s_1 = (...A...B...), s_2 = (...A...B...) \qquad (s_1 = (...B...A...)) \tag{3.1}$$

For two modules A and B, $s_1^{-1}(A)$ ($s_2^{-1}(A)$) represents the position of module A in s_1 (s_2), if $s_1^{-1}(A) < s_1^{-1}(B)$ and $s_2^{-1}(A) < s_2^{-1}(B)$ then it means module A is to the left of module B in the placement. If $s_1^{-1}(A) < s_1^{-1}(B)$ and $s_2^{-1}(A) > s_2^{-1}(B)$ then it means module A is above module B. An example of sequence pair is show in Figure 3.11.

Given a sequence pair for a placement, we can use a pair of constraint graphs to represent the horizontal and vertical relationships between the module positions [15, 27]. A horizontal (vertical) constraint graph $G_h(V, E)$ ($G_v(V, E)$) is a directed and vertex-weighted graph, in which the vertices represent the modules and the edges represent the horizontal (vertical) relationships between

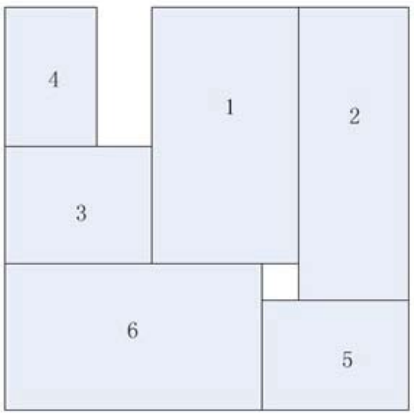


Figure 3.11: A sequence pair to represent a packing $(X, Y) = (< 4, 3, 1, 6, 2, 5 >, < 6, 3, 5, 4, 1, 2 >)$

the modules. A horizontal constraint graph can be constructed by inserting an edge from module A to module B labeled w_A where w_A is the width of module A if $s_1 = (...A...B...)$,$s_2 = (...A...B...)$. Similarly, a vertical constraint graph can be obtained by inserting an edge from module B to module A labeled h_A where h_A is the height of module A if $s_1 = (...A...B...)$, $s_2 = (...B...A...)$. Furthermore, two extra nodes should be inserted to the horizontal constraint graph: a source node representing the left boundary with zero weighted out-going edges to all the leftmost nodes and a sink node representing the right boundary with zero weighted in-coming edges from all the rightmost nodes. Similarly, we can add such nodes to the vertical constraint graph: a source node representing the bottom boundary and a sink node representing the top boundary.

With the constraint graphs, we are capable of obtaining the minimum area packing corresponding to a sequence pair. For the horizontal constraint graph, an edge (A, B) with a weight m indicates that module B must be at least m

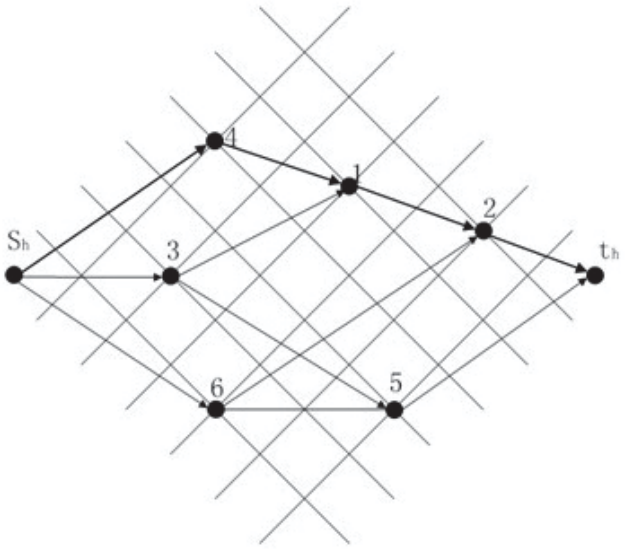
units to the right of module A. Similarly, for the vertical constraint graph, an edge (A, B) with a weight n indicates that module B must be at least n units above module A. Therefore, the x and y coordinates of each module are determined as the minimum by finding the longest path between the source node and the node of the module in G_h and G_v, respectively. The minimum width and height of the packing are computed as the longest path length between the source node and the sink node in G_h and G_v. Figure 3.12 shows the horizontal and vertical constraint graphs constructed according to Figure 3.11. Next, through adding the constraint edges and dummy nodes to the two vertex weighted directed acyclic graphs, all the constraints can be enforced [14].

3.3.2 Polish Expression (PE)

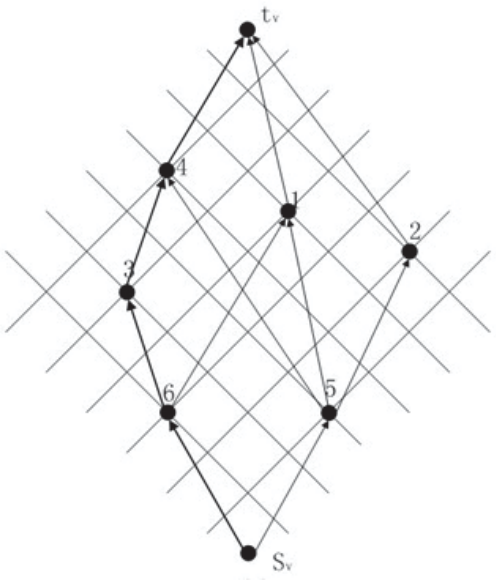
A slicing floorplan can be represented by an oriented rooted binary tree, called a slicing tree as shown in Figure 3.13. Each internal node of the tree is denoted by a $+$ or a $*$, corresponding to a horizontal or a vertical cut respectively. Each leaf represents a basic module and is denoted by a number from 1 to n. A polish expression (PE) [28] can be obtained by traversing a slicing tree in postorder. A normalized polish expression is a PE with no consecutive $*$ or $+$ in it. Previous work in [28] proved that a normalized PE with length $2n - 1$ can generate only one corresponding slicing floorplan with n modules.

3.4 Simulated Annealing

In this thesis, I have chosen to base the layout optimization on simulated annealing [54], which is perhaps the best known and most mature in terms of wide-spread application on industrial layout problems. In the following section, we will review the basic idea of simulated annealing in more detail.



(a)



(b)

Figure 3.12: (a) Horizontal constraint graph $G_h(V, E)$ of Figure 3.11; (b) Vertical constraint graph $G_v(V, E)$ of Figure 3.11

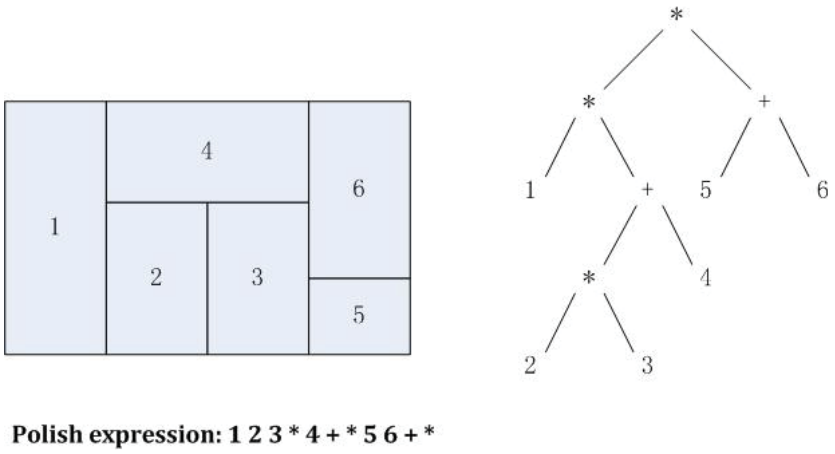


Figure 3.13: Polish expression representation and slicing tree representation for a slicing floorplan

Simulated annealing is based on an analogy with thermodynamics of a crystalline solid solidifying from a melt. When temperature is sufficiently high, the atoms in the melt move randomly in some degree. Individual atoms are free to move such that the total energy of the system either increases or decreases. With the temperature of the melt decreasing, atomic perturbations that may result in an increment in the system energy are less likely to occur and the total energy of the system decreases. When the temperature of the melt is lowered slowly enough, all atoms will eventually reach their lowest energy stage and the system will reach its maximum ordered crystalline state.

The mapping of the placement problem into simulated annealing is accomplished as follows. Given a group of placeable modules in a random state with specifying an initial temperature, T_0, the placer is able to relocate one or more modules by performing many small placement moves. After each move, a cost-function is evaluated to determine the effect of this move on such quality measures as the total chip-area or the estimated net-length. If the change

in the cost-function, $\triangle C$, is less than or equal to zero, the overall quality of the placement has improved or remained unchanged and the new placement is retained. If $\triangle C$ is positive, the overall quality of the placement has decreased. These uphill moves are accepted with probability based on the Metropolis relation [55], that is

$$Pr[uphill] \propto exp(- \triangle C/T) \tag{3.2}$$

If the move is rejected, the placement is returned to its previous state. The system is considered to be in thermal equilibrium after a sufficiently large number of successful moves. At this point, a new lower temperature is calculated and the process begins anew. Eventually, as the placement no longer improves, the layout is considered to be frozen and the optimization is complete.

3.5 Summary

This chapter gave an overview of specified placement constraints for analog layout synthesis. It also introduced two representations and the main optimization algorithm that were applied to solve the complicated placement problem. In addition, this chapter explored the difference between slicing and nonslicing floorplans.

Chapter 4

Congestion-Oriented Approach for Non-Slicing Floorplans

4.1 Overview

In section 2.3.3, it gave a detailed discussion about the specified constraints for analog placement and the necessity of incorporating routing issue during placement. In this chapter, I have applied the method proposed in [14] and generalized it to handle the constraint-driven placement problem. At the same time, I propose a new approach to adjusting the resulting placement to allow routing. Experimental results show that this approach is effective to minimize routing congestion with less placement area and time-consuming.

4.2 Problem Formulation

In this work, I have an input of a set of n modules of areas A_i and aspect ratio bounds $[l_i, u_i]$, where $n = 1, 2, ..., n$, together with the following:

1. A set of m symmetry groups $H_1, H_2, ..., H_m$, where each symmetry group H_i has self-symmetry modules and symmetry pairs.

2. A set of n general placement constraints $G_1, G_2, ..., G_n$, where each placement constraint G_i represents a constraint in placement between two arbitrary modules, including abutment, boundary, alignment, maximum separation, preplace and range constraints.

3. A set of k multi-pin nets $N_1, N_2, ..., N_k$, where each net N_i denotes its connected modules.

The objective of this work is to output a high routability placement result such that the estimated wirelength and total layout area are minimized. At the same time, the placement result satisfies all the specified constraints.

4.3 Methodology

Simulated annealing is employed as the basic search-engine to optimize the placement solution with minimum chip area and wirelength. Figure 4.1 illustrates the flow of the congestion-oriented placement algorithm. From the placement flow, we can see that a preliminary candidate placement that meets all the specified constraints can be obtained from the first stage and it is given as an input to the second stage to estimate the net congestion of each module. The method of getting the candidate placement is proposed in [14, 15] and it will be discussed in the following part. With the net congestion, the size of corresponding modules will be expanded to reserve enough space for the subsequent channel routing.

4.3.1 Approach to Obtaining A Candidate Placement

Firstly, a random SP is created and an initial scan should be carried out to see whether its corresponding placement satisfies certain constraints. Then, constraint graphs are built according to the generated SP. In the next stage, new constraint edges and dummy nodes are added into G_h and G_v to enforce all

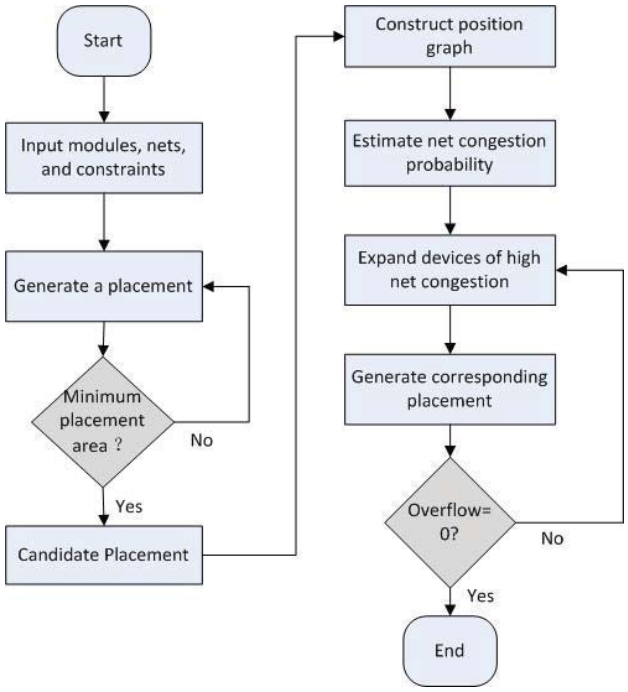


Figure 4.1: The flow of the congestion-oriented placement algorithm

the specified constraints. Some of the new inserted edges have variable weights and it is necessary to calculate their weights with getting rid of positive cycles and minimizing the packing area. A set of random moves are performed to generate different placements during the annealing process. Finally, one placement with minimum cost and satisfying all the constraints can be obtained from the annealing process.

Fast Initial Scan

Each sequence pair represents a placement packing, but not all of them are subject to these constraints. It needs to identify these feasible candidates from all possible sequence pairs in order to narrow the search space. The fast initial scan is used to screen out some infeasible ones. This step can check the

constraints as follows:

1. Alignment constraint $align(h, A, B)$ requires modules A and B to keep the same order in s_1 and s_2 , while $align(v, A, B)$ requires modules A and B to keep the reverse order in s_1 and s_2 .

2. Abutment constraint $abut(l, A, B)$ and $abut(r, A, B)$ mean modules A and B must at least keep the same form in s_1 and s_2 , while $abut(t, A, B)$ and $abut(b, A, B)$ mean modules A and B must at least keep the reverse form in s_1 and s_2 .

3. Boundary constraint $boundary(x, A)$ means if module A abuts with the left (right) boundary of the packing, there is no module B such that B is before (after) A in $s1$ and $s2$. Similarly, there is no module B such that B is before (after) A in $s1$ and after (before) A in $s2$ if module A abuts with the bottom (top) boundary.

4. Symmetry constraint requires the modules which belong to the symmetry group to meet the two conditions:
   For horizontal symmetry groups:

$$s_1^{-1}(A) < s_1^{-1}(B) \Longleftrightarrow s_2^{-1}(sym(B)) < s_2^{-1}(sym(A)) \tag{4.1}$$

   For vertical symmetry groups:

$$s_1^{-1}(A) < s_1^{-1}(B) \Longleftrightarrow s_2^{-1}(sym(A)) < s_2^{-1}(sym(B)) \tag{4.2}$$

   Note that A and B are any two distinct modules in the symmetry group, $s_1^{-1}(X)$ $(s_2^{-1}(X))$ represents the position of module X in s_1 (s_2), and $sym(X)$ represents the symmetry counterpart of X ($sym(X)$ of a self-symmetry module X is X itself).

Handling of General Placement Constraints

According to the introduction about SP and the corresponding constraint graphs in last chapter, constraint edges can be added to constraint graphs to enforce the general placement constraints. For example, $align(v, A, B)$ requires an edge between A and B with weight 0 in the vertical constraint graph. The other general constraints can be handled in a similar way. The method is proposed in paper [15].

Handling of Symmetry Constraint

For a symmetry group H_i including $s_i = pair(H_i)$ (symmetry pairs (X_1, Y_1), $(X_2, Y_2) \cdots (X_{s_i}, Y_{s_i})$) and $r_i = self(H_i)$ (self-symmetry modules $Z_1, Z_2 \cdots Z_{r_i}$), we should first see if H_i is symmetric horizontally or vertically with an initial scan as described above. Then constraint edges are inserted into the horizontal constraint graph to align the symmetry pairs in H_i vertically (if the symmetry axis is in the vertical direction) as shown in Figure 4.2. Also, a dummy node d_i is inserted into the horizontal constraint graph to denote the symmetric axis of H_i. In order to guarantee equidistance between symmetry pairs with respect to the axis, four constraint edges should be added, $e(d_i, X_j), e(X_j, d_i), e(d_i, Y_j)$ and $e(Y_j, d_i)$ with weights $(-x_{ij}, x_{ij}, x_{ij} - w(Y_j)$, and $-(x_{ij} - w(Y_j))$, respectively, for each $j = 1 \cdots s_i$ where $w(Y_j)$ denotes the width of module Y_j (note that $w(X_j) = w(Y_j)$) and $x_{ij} - w(Y_j)$ is a positive real number. For each self-symmetry module Z_j where $j = 1 \cdots r_i$, a pair of edges should be added, $e(Z_j, d_i)$ and $e(d_i, Z_j)$ with weights $w(Z_j)/2$ and $-w(Z_j)/2$ to guarantee that Z_j is placed symmetrically to the axis. Next, we needs to calculate the value of x_{ij} for $j = 1 \cdots s_i$ such that there exist no positive cycle in the graph and $max_{1 \leqslant j \leqslant s_i} x_{ij}$ is minimized. The algorithm to determine the value of x_{ij} is discussed in [14].

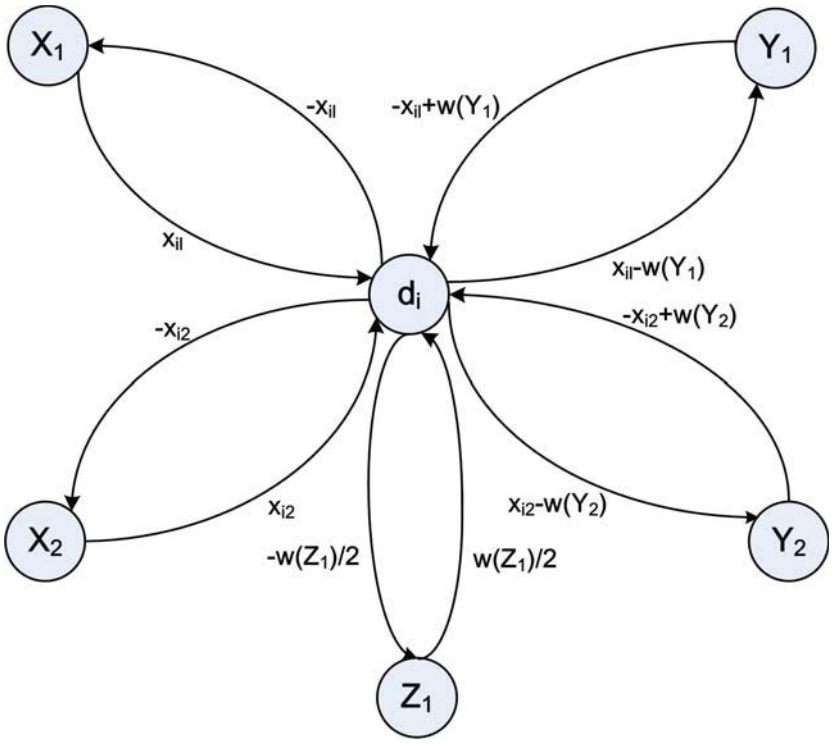


Figure 4.2: Dummy nodes and constraint edges for a symmetry group

Simulated Annealing Process

Simulated annealing is employed as the basic searching engine to identify the optimum result among all the feasible placement candidates. During the annealing process, a set of moves are performed to perturb a current candidate solution. These moves includes the five types as follows:

1. Swapping two symmetry groups;
2. Swapping two modules of the same symmetry group;
3. Moving an asymmetric module;
4. Rotating a symmetry group;
5. Changing the aspect ratio of a soft module.

During the simulated annealing process, the cost function $cost(F) = area(F) + \lambda * wire(F)$ is used to evaluate a placement F, where $area(F)$ denotes the area of F, and $wire(F)$ denotes the total wirelength obtained by the half perimeter method. The parameter λ is a factor that specifies the relative importance between area and wirelength.

4.3.2 Net Congestion in the Approach

A placement with high compaction (Figure 4.3(a)) can be achieve by the implementation of the mentioned placement. Obviously, compaction in placement may lead to unroutable solutions and result in timing correlation problems because of detoured nets. To make the resulting placement favorable for routing, we need separate the modules with high net congestion to leave enough space so that all nets can be routed successfully in the next stage(Figure 4.3(b)).

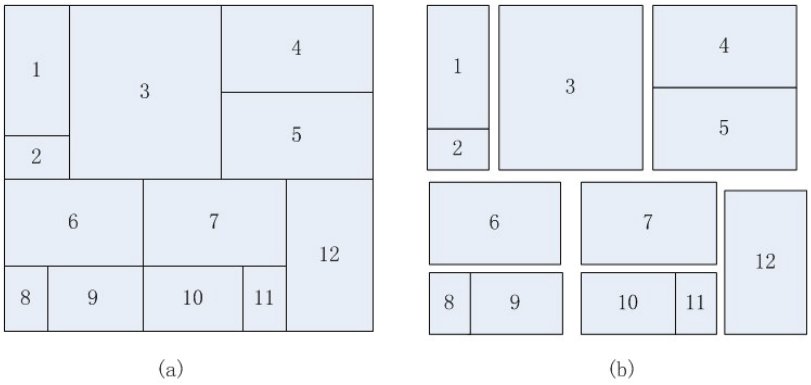


Figure 4.3: (a) Placement with high compaction; (b) Placement with considering the routability issue

Assuming that route nets in Figure 4.4, my approach mainly focuses on leaving channels for the subsequent routing. The modules are allowed to occupy their respective dash area first as shown in Figure 4.4 and then release them to construct the routing channels in the final placement. This approach emphasizes on expanding the module dimension appropriately to control the final packing area. In order to achieve this goal, the probabilistic analysis [56] model is generalized here to determine which modules need to expand and the expanding level. In the new proposed model, a module with high net congestion probability means that more nets are likely to route through its surrounding area. Accordingly, enough white space should be left for routing these nets. Therefore, this module needs a dimension expansion during placement to occupy more area in an attempt to leave enough routing space in the end.

To estimate the net congestion probability, the probabilistic congestion model should be built first.

The following assumptions are proposed for this model.

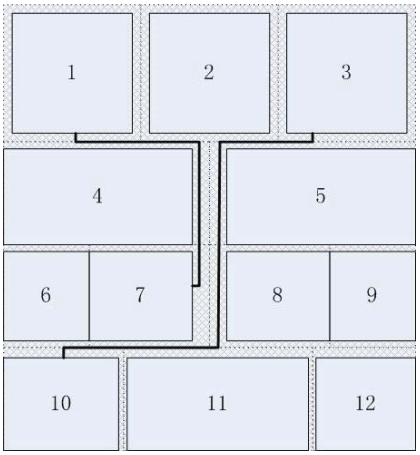


Figure 4.4: Placement with leaving routing channels

Assumption 1: All nets are routed through the locations of the modules.

Assumption 2: All nets are optimally routed with shortest length. Therefore, detoured nets are not allowed and all nets should be routed within a rectangular region determined by the pins of the nets.

Assumption 3: The net congestion probability analysis is done exclusively for two-pin nets. Multi-pin nets are handled by being divided into a set of two-pin nets by Minimum Spanning Tree.

Assumption 1 gives a constraint that the nets should be routed through the corresponding modules. This assumption makes it possible to build a position graph for net congestion calculation according to the candidate placement result as indicated by Figure 4.5. It also ensures that the module expansion approach can be used to gain enough space for channel routing. This congestion model is a graph-based model. Each node in Figure 4.5 represents a module in placement and the position graph is obtained from Figure 4.4 by

inserting an edge between any two adjacent modules. Take module 4 in Figure 4.4 as an example, it is adjacent to modules 1, 2, 5, 6, 7 and 8 in different directions. Accordingly, edges between node 4 and nodes 1, 2, 5, 6, 7 and 8 are added to the position graph when converting Figure 4.4 to Figure 4.5. Assumption 2 ensures that all nets can be routed within a length range, since the router is more likely to choose the shortest path for each net. Therefore, modules with high net congestion should be assigned to enough capacity to reduce the number of detoured nets and avoid unroutable solutions in routing. Assumption 3 is proposed to simplify the model.

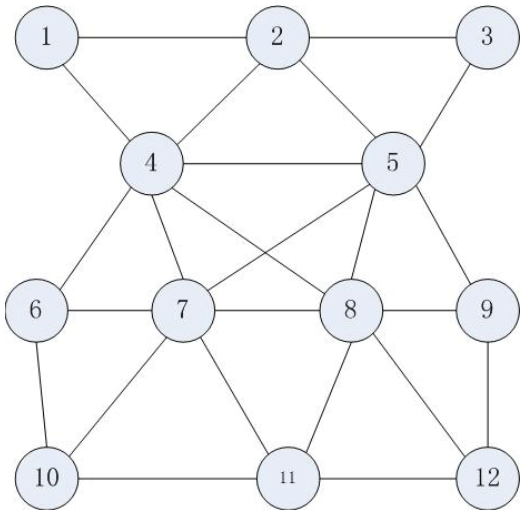


Figure 4.5: The position graph of the placement in Figure 4.4

According to assumption 1 and assumption 2, a net connecting module 1 and module 5 has to be routed via module 2 or module 4 in Figure 4.4. Assumption 2 can be used to determine a monotonic direction when routing a net and the work is carried out according to the Manhattan distance between the net source pin and all the possible via modules. The placement of Figure 4.4 can be taken as a detailed example to illustrate my approach. Assuming that

we need to find a route between module 1 and module 12, the routing region will be rectangular with the source pin as the upper left corner and the drain pin as the lower right corner. Thus, all possible modules for routing the net will be those within this area including modules 1, 2, 3, 4, 5, 6, 7, 8 and 12.

In my graph mesh, the probabilistic congestion of each node is defined to be its total probabilistic usage contributed by routing all nets. The congestion probability of each node represents the number of possible routing tracks via the corresponding module since the routing area is incorporated in the module. In my work, the congestion probability can be explained in two steps. First, for a single net, the net congestion probability can be defined as the ratio between the number of routes via this node and the number of total routes for routing this net. Then, based on this definition, the congestion probability of each module can be defined as the sum of the net congestion probability of routing all nets. The larger the probability is, the worse the congestion will be. Given a placed netlist, we can analyse the congestion probability for every node in the mesh.

4.3.3 Number of Possible Routes of Each Node

In order to illustrate this congestion model, a simple example is given to show the calculation of possible routes passing through each node. In Figure 4.6, for the single node 4, there are 3 and 2 edges from two different directions. It is obvious that there are 6 possible routes travelling through node 4. The number of possible routes is determined by the number of edges from two directions and the following discussion is based on this rule.

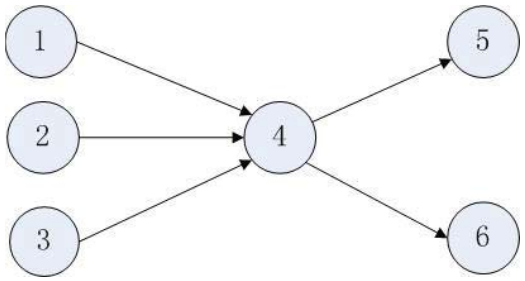


Figure 4.6: Number of possible routes of a single node

4.3.4 Total Number of Possible Routes

Definition 1: Define F_i, B_i respectively as the number of possible routes entering each node from the forward direction and reverse direction to route a two-pin net, where i represents the module sequence number. For each net, we only need to consider these nodes within the rectangular routing region. Given a two-pin net, assuming that the source pin is located in module m, and the drain pin in module l. The two pins are equivalent and they can exchange their roles during the calculation. Then, F_l and B_m can represent the total number of possible ways to route a two-pin net.

Lemma 1:

$$F_m = B_l = 1 \tag{4.3}$$

Proof: If there are only two nodes to route this net, then there is only one possible route. For the nodes that come after the source node in the position graph, there is only one route entering them from the source node as shown in Fig.4.7. Therefore, F_m and B_l are always equal to one.

Definition 2: Define N_i^F and N_i^B as the aggregations of nodes that have edges entering node i from two directions. Take Figure 4.6 as an example, then

$$N_4^F = \{1, 2, 3\}, N_4^B = \{5, 6\} \tag{4.4}$$

Corollary 1: Assuming that the number of nodes to route a net is more than 2, the total number of possible routes F_l or B_m can be computed using following equations.

$$F_l = \sum_{k \in N_l^F} F_k \tag{4.5}$$

$$B_m = \sum_{k \in N_m^B} B_k \tag{4.6}$$

Proof: In order to optimally route the two-pin net, all routes must entering the drain pin or leaving the source pin as shown in Figure 4.7. The sets of routes from the previous nodes of the drain pin are mutually exclusive. According to this corollary, the total number of routes can be calculated iteratively.

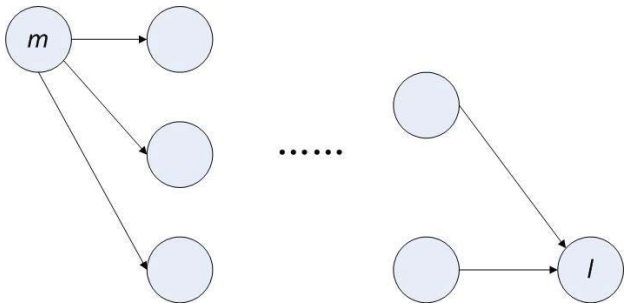


Figure 4.7: Explanation of the source pin and drain pin

4.3.5 Probabilistic Usage Array

Definition 3: Define array P as the probabilistic usage of all nodes. P_i represents the probabilistic usage of node i. For each net, P_i has a real value when the node i is located within the routing area of this net. For any net, the P array has the following properties:

$$P_m = P_l = 1 \tag{4.7}$$

Equation 4.7 indicates that routes consume exactly one track of the source pin and drain pin, because of the fact that any optimal route must travel through

the two pins.

For the other nodes within the routing area, routes enter the node and leave the node at the same time. Let us denote I_i and O_i to represent the number of routes entering node i from the two directions. According to *Corollary 1*, P_i can be computed by the following equations.

$$I_i = \sum_{k \in N_i^F} F_k \tag{4.8}$$

$$O_i = \sum_{k \in N_i^B} B_k \tag{4.9}$$

$$P_i = \frac{I_i * O_i}{F_l} \tag{4.10}$$

4.3.6 Detailed Example

I will further introduce the method of computing probabilistic usage array with a simple example. Assuming that we need to find a route between module 1 and module 12 in Figure 4.4, the iterative calculation of possible routes is carried out from both directions. The whole process is shown in Figure 4.8, Figure 4.9 and Figure 4.10. Take node 2 in Figure 4.5 as an example, its net congestion probability will be the ratio of the number of possible routes via node 2 and the total number of possible routes between node 1 and node 12. Given a set of two-pin net in Figure 4.4, Table 4.1 shows the value of P array of every single net and its total value of all nets. Table 4.1 also demonstrates that the three biggest net congestion areas are distributed in modules 2, 5 and 8. Accordingly, these three modules can be expanded to reserve enough space for routing as shown in Figure 4.4.

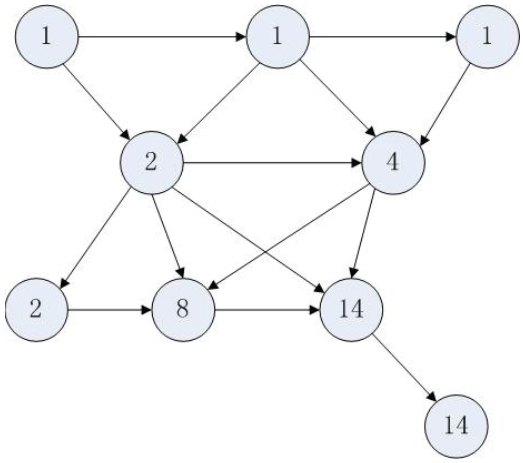


Figure 4.8: Process of calculating possible routes from the forward direction (F_i)

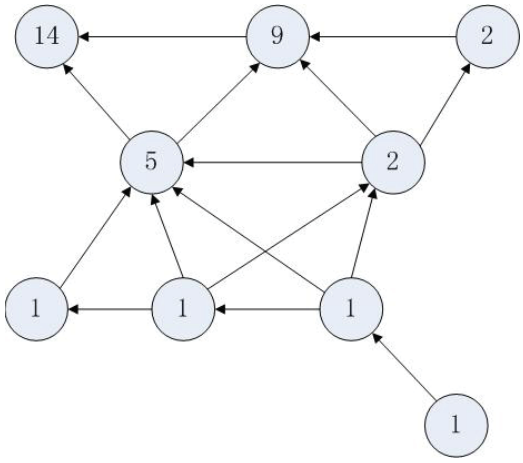


Figure 4.9: Process of calculating possible routes from the reverse direction (B_i)

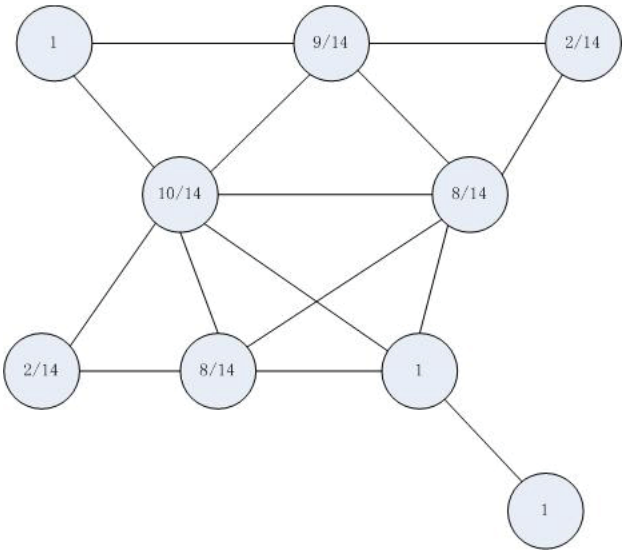


Figure 4.10: Single probabilistic usage of all nodes (P_i)

P_i	P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8	P_9	P_{10}	P_{11}	P_{12}
net 1,12	1	0.64	0.14	0.71	0.57	0.14	0.57	1	0	0	0	1
net 1,7	1	0	0	1	0	0	1	0	0	0	0	0
net 3,10	0.19	0.69	1	0.75	0.63	1	0.75	0.38	0	1	0	0
net 3,8	0	0	1	0	1	0	0	1	0	0	0	0
net 2,12	0	1	0.5	0	1	0	0	1	0	0	0	1
net 2,8	0	1	0	0	1	0	0	1	0	0	0	0
P_i'	2.19	3.33	2.64	2.46	4.20	1.14	2.32	4.38	0	1	0	2

Table 4.1: Detailed example of calculating P_i

4.3.7 Adjustment of Module Size

When accomplishing the work of prerouting all nets, the net congestion probability can be used to expand the dimensions of high-congested modules for the following placement. The criterion to implement this work is as follows:

$$A'_i = A_i * (1 + \lambda * P'_i) \tag{4.11}$$

Where P'_i is the sum probability of routing all nets via module i, the constant λ is determined by the width of wires and the compaction of the candidate placement, and A_i, A'_i represent the original area and adjusted area of module i.

4.4 Experimental Results

In this section, the experimental results are presented. My placer is carried out in C and run on a Linux operating system with an Intel Xeon 1.6GHz CPU and 16GB memory. Two groups of experiments are conducted to test the effectiveness of our approach.

4.4.1 Comparisons with Previous Approach

I have compared my approach mainly with previous approach [1] on handling the same problem. In this group of experiment, I use the same parameters and evaluation metrics with that in [1], based on the same data sets, ami33 and ami49, from MCNC benchmarks circuits. The two columns in Table 4.2 respectively show the results with placement expansion by performing my approach and previous approach [1]. In order to compare the two approaches, I list the placement area, the wirelength estimated by half perimeter method, the estimated number of routing overflow from FastRoute 4.1 [43], and the

runtime. We can see that my approach performs much better in terms of both area and runtime.

Data	Our Approach				Previous Approach [1]			
	area(%)	WL(mm)	OF	Time(s)	area(%)	WL(mm)	OF	Time(s)
ami33	105.01	45.08	0	18.69	113.25	47.73	0	556
ami49	103.67	775.63	0	35.24	115.32	677.24	0	1337

*WL represents wirelength and OF represents overflow.

Table 4.2: Comparisons with previous approach [1]

4.4.2 Detailed Experiments

In order to study the effectiveness of this congestion-oriented approach, I have set another data for experiments with more modules, more complicated constraints and more nets. The information of benchmark circuits is shown in the Table 4.3. Table 4.4 contains the results of placement before and after module expansion. The comparison between them can illustrate the feasibility of my approach in improving the routability of the placement. Figure 4.11 and Figure 4.12 respectively represent the placement of the same circuit before and after module expansion, the differences between them testify that my placer can assign corresponding channels for routing. Considering the regularity methodology in papers [26, 57], if the modules are of different sizes, it will increase the deadspace significantly to leave some channels in the placement result. However, my method performs module expansion according to the possible net congestion and avoids the problem of introducing much extra deadspace. In this section, FastRoute 4.1 [43] is employed to confirm the effectiveness of this approach. There are two routing layers in the experiment and each module is implemented as routing blockage. Hence, the placement result should be expanded large enough to enable all the nets to be routed

successfully. Table 4.5 is a detailed result about the corresponding changes of overflow and wirelength when adjusting the degree of expansion. Table 4.5 shows that the wirelength fluctuates all the time, because in the given range, although the augmenting area determines the increase of total wirelength, the more corresponding routing space increases the possibility for these nets to find shorter paths simultaneously.

Data Set	Module No.	Sym. Group No.	General Const. No.	Net No.
Data40a	40	1	4	60
Data80a	80	5	6	98
Data80b	80	5	10	98

Table 4.3: Information of benchmark circuits with mixed constraints

Data	$\lambda = 0.0$			$\lambda = 0.085$			$\lambda = 0.15$		
	DS	OF	WL	DS	OF	WL	DS	OF	WL
Data40a	0.08	257	N	0.20	0	5833	0.29	0	5812
Data80a	0.14	895	N	0.43	0	11933	0.58	0	12217
Data80b	0.13	775	N	0.38	0	12031	0.52	0	12129

*DS represents deadspace and N means when the value of overflow is too large, wirelength is meaningless. And actually the FastRoute 4.1 can not route all the nets in the experiments at this time.

Table 4.4: Experimental results of different expansions

λ	0.065	0.070	0.075	0.080	0.085	0.090	0.095	0.1	0.11	0.12	0.13	0.14	0.15
DS	0.33	0.35	0.36	0.37	0.38	0.40	0.41	0.42	0.45	0.46	0.48	0.50	0.52
OF	54	42	16	0	0	0	0	0	0	0	0	0	0
WL	11603	11443	11553	12031	12013	11423	11379	11637	11479	11676	11774	12151	12129

Table 4.5: Detailed changes of overflow and wirelength of data80b

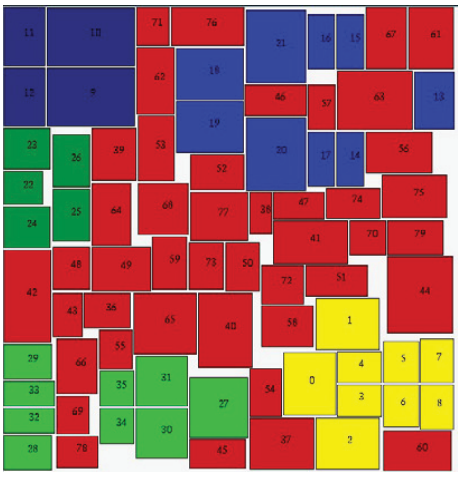


Figure 4.11: Resultant packing with high compaction of data80b

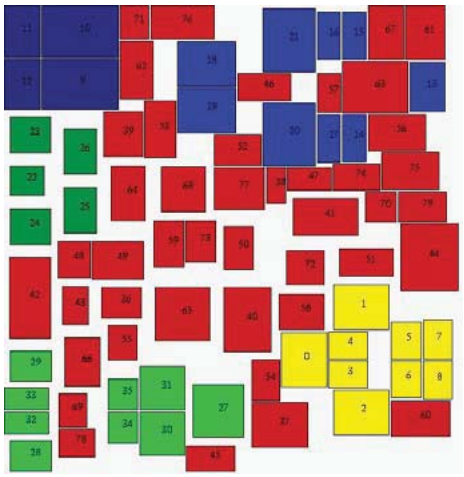


Figure 4.12: Resultant packing of data80b after adjustment

4.5 Summary

In this chapter, an approach was proposed in order to solve the routability-driven placement problem with mixed constraints in analog and mixed-signal integrated circuits. The specified constraints were enforced by augmenting additional constraint edges and dummy nodes to the constraint graphs. The net congestion probability analysis was used to successfully tackle the routability issue during the placement stage. Experimental results demonstrated the feasibility of my method. It not only improved the routability of the final packing without consuming much placement area, but also guaranteed that the final placement satisfied all the constraints after placement expansion.

Chapter 5

Regularity-Oriented Approach for Slicing Floorplans

5.1 Overview

In last chapter, an approach based on non-slicing structures was proposed for the multi-objective placement problem and the algorithm of solving the whole problem was apparently complicated. In this chapter, another approach is proposed in order to solve the same problem. This approach can take into account some specified constraints, especially the symmetry constraint and it tackles the routability issue based on the characteristics of polish expression. Experimental results demonstrate that my method is effective and feasible in solving the constraint-driven slicing floorplanning problems.

5.2 Problem Formulation

In this section, I have an input of a set of n modules of areas A_i and aspect ratio bounds $[l_i, u_i]$, where $n = 1, 2, ..., n$, together with the following:

1. A set of m symmetry groups $H_1, H_2, ..., H_m$, where each symmetry group H_i has self-symmetry modules and symmetry pairs.

2. A set of n general placement constraints $G_1, G_2, ..., G_n$, where each placement constraint G_i represents a constraint in placement between two arbitrary modules, including boundary, clustering, alignment constraints.

3. A set of k multi-pin nets $N_1, N_2, ..., N_k$, where each net N_i denotes its connected modules.

The goal of this approach is to obtain a high routability placement result while minimizing the total area and wirelength. At the same time, the placement result satisfies all the specified constraints.

5.3 Overview of the Approach

In this chapter, I propose another efficient approach that can handle different constraints simultaneously, including symmetry constraint, boundary constraint, alignment constraint and clustering constraint. Also, the new proposed floorplanner is capable of generating a placement that has a better performance in the routing stage because of high regularity [26].

I have chosen to base the layout optimization on simulated annealing and selected the polish expression [28] as the representation in this approach. In order to handle the symmetry constraint in the slicing floorplan without violating other constraints and adding extra chip area, each symmetry group is regarded as a super-module in PE that represents a floorplan. The internal structures of every symmetry group are handled specifically to satisfy the symmetry constraint. During the annealing process, different PEs are generated by a set of random moves and each PE consists of the super-modules representing the symmetry groups and other modules in the circuits. In every step of the annealing process, the symmetry groups are handled first and then packed,

and a PE x is then automatically generated. After that, the other general constraints can be checked by just scanning the PE. If all the constraints are satisfied at the same time and we can pack accordingly to obtain one feasible floorplanning result and get the coordinates by building its corresponding slicing tree. Details on the approach to handling the symmetry constraint and other placement constraints are presented in the following parts.

5.4 Handling of Symmetry Constraint

5.4.1 Placement of Symmetry Group

In this part, a symmetric-feasible polish expression (SF-PE) is proposed to handle the symmetry constraint. The symmetry constraint states that each pair of modules in the symmetry group should be placed symmetrically to a common horizontal or vertical axis called symmetric axis. The symmetry group may contain some self-symmetry modules, the centers of which should be put on the symmetric axis. To suppress the circuit sensitivity due to process variations and thermal gradients, modules of a symmetry group are normally placed quite close to each other. In order to achieve such a placement, the symmetry modules of a group are treat as a symmetry island [7] in my approach. This is why the symmetry groups are treated as super-modules in a PE.

Before the introduction of SF-PE, representatives for symmetry pairs or self-symmetry modules should be defined first.

Definition 1: The representative m_i^l of a symmetry pair (m_i,$m_i^{'}$) is m_i (the left or the lower module of the symmetry pair), while the representative m_i^l of a self-symmetry module is itself m_i^s.

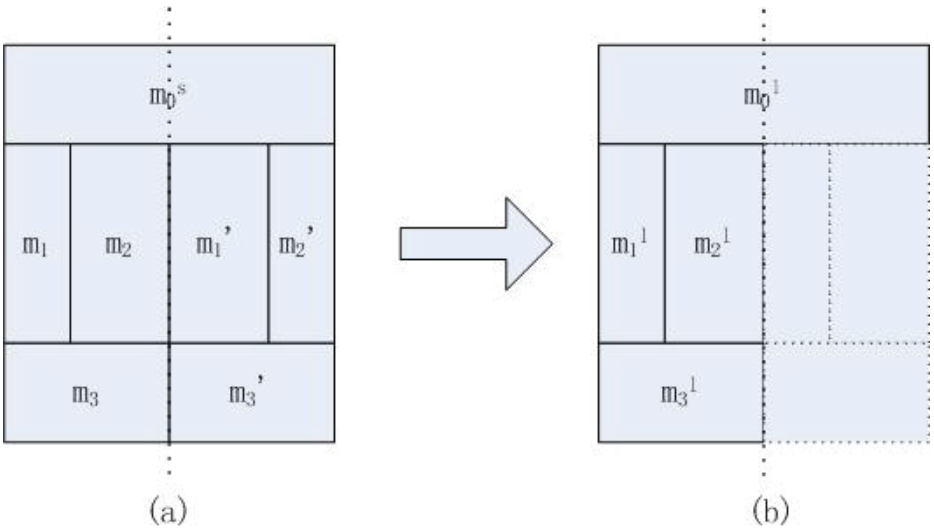


Figure 5.1: (a)A placement of a symmetry group; (b)Representative selection for symmetry pairs and self-symmetry modules

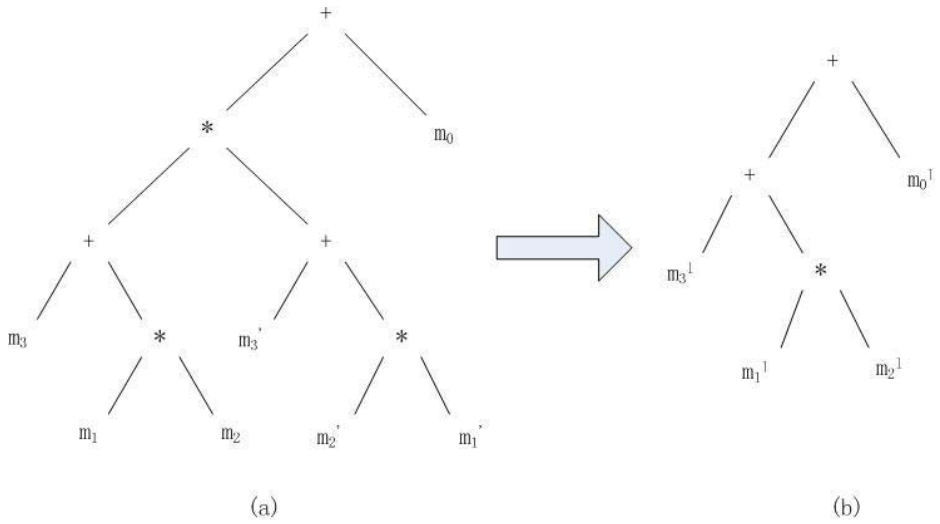


Figure 5.2: (a) The slicing tree representation of PE in Figure 5.1(a); (b) The slicing tree representation of SF-PE in Figure 5.1(b)

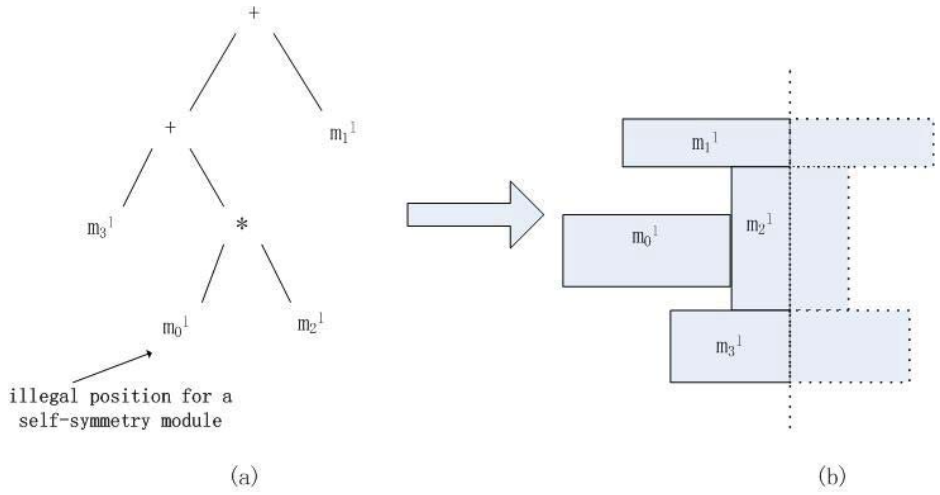


Figure 5.3: A invalid slicing tree and its slicing floorplan

Take the symmetry group in Figure 5.1(a) as an example, the representative m_1^l of the symmetry pair $(m_1, m_1^{'})$ is m_1, and m_0^l is the representative of self-symmetry module m_0^s. The representative of each module in a symmetry group is shown in Figure 5.1(b).

From Figure 5.1(b), we can see that each symmetry pair or self-symmetry module must have its representative module when dealing with the symmetry constraint. So, the number of the representative modules in a symmetry group should be equal to the number of symmetry pairs and self-symmetry modules. So, the SF-PE has a definition as follows.

Definition 2: A SF-PE representation is a PE containing only the representative modules that correspond to the symmetry groups.

Based on the two definitions above, it is obvious that the corresponding PE of Figure 5.1(a) is $m_3 m_1 m_2 * + m_3^{'} m_2^{'} m_1^{'} * + * m_o +$, and a SF-PE for this symmetry

group should be $m_3^l m_1^l m_2^l * + m_o^l +$. Their respective slicing tree representations are shown in Figure 5.2.

From the new representation of a symmetry group, we can see that the representative of a symmetry pair is flexible in a SF-PE, which means that it can be placed in any position in its corresponding slicing tree. But in order to make the SF-PE really feasible for a symmetry placement, the self-symmetry modules should have some restrictions in a SF-PE representation.

Lemma 1: Given a SF-PE with a vertical axis, the self-symmetry modules should be placed if and only if there is no module to the right of them. This feature shown in the slicing tree is that the self-symmetry module is not in the left subtree of any internal node labeled $*$.

Proof: Because in a symmetry group, the self-symmetry modules are placed symmetrically only if their centers are placed on the symmetric axis. Hence, in a SF-PE, there is no module placed to the right of the self-symmetry modules, which also indicates they should be placed on the right boundary in their corresponding sub-floorplan. For example, the illegal location for a self-symmetry module m_0^l is show in Figure 5.3(a). Based on the SF-PE of this slicing tree, it is impossible to obtain a symmetry placement by adjusting the coordinate of m_0^l. In a regular PE, there are three legal moves [28] to perform to get different placements. Similarly, these moves are allowed in a SF-PE. But after each move, it is necessary to take an initial scan to check the boundary information for self-symmetry modules, which aims to see whether the new SF-PE is feasible to achieve a symmetry sub-floorplan.

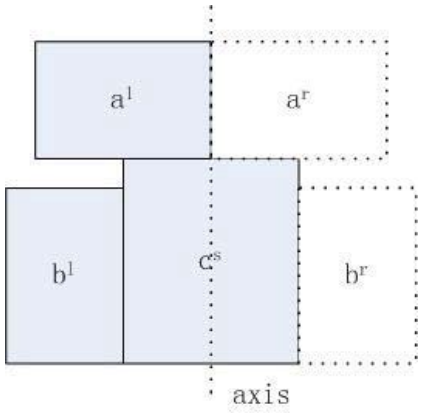


Figure 5.4: Placement with satisfying symmetry constraint (symmetry pair modules a^l, a^r and b^l, b^r and self-symmetry module c^s)

5.4.2 Linear Constraint Expressions from Symmetry Constraint and the Corresponding SF-PE

In order to get a valid packing for a symmetry group from a SF-PE and its slicing tree, the coordinates of these modules need to be adjusted according to a group of linear constraint expressions.

First, I will introduce the approach to obtaining the linear constraint expressions from the SF-PE representation of a horizontal-symmetry group. If the modules are represented as $ab*(ab+)$, which indicates that module a is to the left of (under) module b, the following inequality in $x(y)$ direction can be derived[8]:

$$x(a) + w(a) \leq x(b) \qquad (y(a) + h(a) \leq y(b)) \tag{5.1}$$

Note that $x(a)$ and $x(b)$ are the *x_coordinate* of the left edge of modules a and b, and $w(a)$ is the width of module a. ($y(a)$ and $y(b)$ are the *y_coordinate* of the lower edge of modules a and b, and $h(a)$ is the height of module a.)

Basically, the linear constraint expressions are set to guarantee that the self-symmetry modules satisfy the constraint, because each symmetry pair is represented by the left half of it and its representative is flexible in the SF-PE. Therefore, the complexity of the expressions is determined by the number of self-symmetry modules. In Figure 5.4, two module pairs (a^l, a^r) and (b^l, b^r) and a self-symmetry module c^s are symmetric to a vertical symmetric axis. Let $Axis_x$ represent the *x_coordinate* of the symmetric axis. Symmetry constraint can be converted into linear constraint expressions using SF-PE. For self-symmetry module c^s with its representative c^l, the linear constraint expression in x direction is as follows:

$$x(c^l) + w(c^l)/2 = Axis_x \tag{5.2}$$

In y direction, the linear constraint expression is

$$y(b^l) = y(c^l) \tag{5.3}$$

For the symmetry pair modules, the linear constraint expressions in x direction are

$$x(a^l) + w(a^l) \leq Axis_x \tag{5.4}$$

$$x(b^l) + w(b^l) \leq x(c^l) \tag{5.5}$$

$$x(b^l) \leq x(a^l) \tag{5.6}$$

The linear constraint expression in y direction we have here is

$$y(a^l) \geq max(h(b^l), h(c^l)) + y(b^l) \tag{5.7}$$

Because module b^l is supposed to be the left-bottom module of the symmetry group, $x(b^l)$ and $y(b^l)$ are already known. Therefore, it is possible to obtain the set of constraint expressions.

The objective function is that the area of the symmetry group should be as compact as possible since it is part of the total area cost. In y direction, the expression is

$$H = max(h(b^l), h(c^l)) + h(a^l) \tag{5.8}$$

Where H represents the minimum total height of this symmetry group. Since H is constant in this equation, the total area is determined by the longest path in x direction. The objective function in the linear programming is $Axis_x - x(b^l)$, whose value should be as small as possible.

$$W = 2 * (Axis_x - x(b^l)) \tag{5.9}$$

Where W is the total width of this symmetry group.

After obtaining the value of $Axis_x$, $x\_coordinate$ and $y\_coordinate$ of all the modules in this group can be determined.

$$Axis_x - (x(a^l) + w(a^l)) = x(a^r) - Axis_x \tag{5.10}$$

$$y(a^l) = y(a^r) \tag{5.11}$$

$$Axis_x - (x(b^l) + w(b^l)) = x(b^r) - Axis_x \tag{5.12}$$

$$y(b^l) = y(b^r) \tag{5.13}$$

Calculation in the two directions is considered independently in the work. For a horizontal symmetry group, it only needs to list the linear expressions in x direction due to the compaction requirement. Take Figure 5.4 as an example, when SF-PE ($b^l c^l * a^l +$) and the width of modules $w(a^l) = 3, w(b^l) = 2$ and $w(c^l) = 3$ are given. Then the following expressions in x direction can be obtained:

$$x(a^l) + 3 \leq Axis_x \tag{5.14}$$

$$x(b^l) + 2 \leq x(c^l) \tag{5.15}$$

$$x(c^l) + 1.5 = Axis_x \tag{5.16}$$

$$x(a^l) \leq x(b^l) \tag{5.17}$$

Then, remove redundant expressions and obtain

$$x(a^l) \leq x(b^l) \tag{5.18}$$

$$x(a^l) + 3 - Axis_x \leq 0 \tag{5.19}$$

$$Axis_x - 3.5 \geq x(b^l) \tag{5.20}$$

Finally, a group of constraint expressions are determined with two variables and three expressions.

Linear programming is used just for the symmetry group. The algorithm [28] is executed to calculate the rest modules' coordinates. During the simulated annealing process, the coordinate $(0, 0)$ is assigned to the left-bottom module of the symmetry group in each perturbation in order to obtain the size information of this super-module. When getting the final packing, the new coordinate information will substitute $(0, 0)$ to determine the absolute position of symmetry modules.

5.5 Handling of Other Constraints

In this thesis, the methods in paper [16] and paper [17] are respectively applied to satisfy the boundary and clustering constraints. These methods are implemented by only looking at the valid PE and no real packing is needed. The following part provides a detailed introduction about how to satisfy alignment constraint in placement.

Alignment constraint requires the modules to align vertically along the left or the right side, or align horizontally along the top or the bottom side. Due to the features of a slicing floorplan that it can be obtained by recursively partitioning a rectangle in two by either a vertical or a horizontal line, it is not a difficult job to satisfy the alignment constraint during floorplanning. All the modules in Figure 5.5 that align along the same partitioning line must align with each other. For instance, modules B, C and D, modules E and F are able to form two alignment groups. Therefore, the problem of checking the specified alignment modules can be treated as the problem of checking that if these modules share one partitioning line.

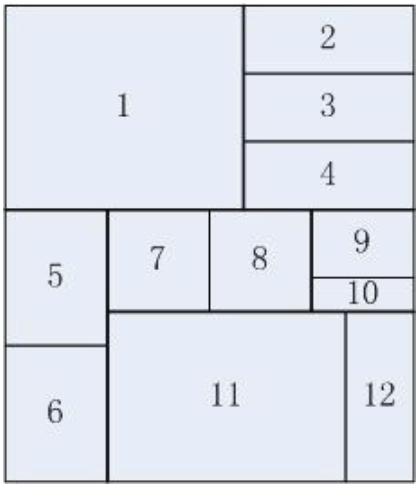


Figure 5.5: Example of alignment in slicing floorplans

In paper [28], it introduced the definition that *a slicing structure* is a rectangle dissection that can be obtained by recursively cutting rectangles into smaller rectangles. It is obvious that modules sharing one cutting line must belong to such a slicing structure. For a polish expression $\alpha = \alpha_1\alpha_2, ..., \alpha_n$, we define a subexpression $\beta = \alpha_i\alpha_{i+1}, ..., \alpha_{i+m}$ where $k \geq 1$ and $i + m \leq n$ to represent

such a slicing structure. To make this subexpression valid to represent a subtree in the whole slicing tree, the first element α_k in β should be an operand and the number of operands should be equal to the number of operators plus one.

My method emphasizes on checking the alignment constraint by figuring out the partitioning line of these modules. It is implemented by scanning the PE with no need to build the slicing tree. In order to achieve this goal, the shortest subexpression including these modules should be picked out first. The algorithm about how to find the shortest valid subexpression for alignment group (a, b) (a is before b in PE) is shown in Algorithm 1.

Algorithm 1: FindSubexpression(a,b)

Input:
$\alpha = \alpha_1\alpha_2, ..., \alpha_{2n-1}$ is a given PE;
Output:
Shortest valid subexpressions containing modules a and b;

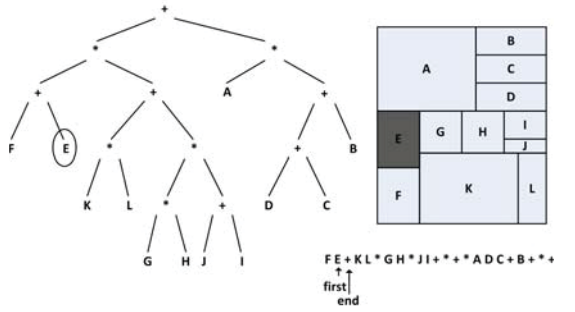
1: $\alpha_t = a$;
2: $first = end = t$;
3: **while** (b is not in e yet) **do**
4: $end = end + 1$;
5: **if** (α_{end} is an operator) **then**
6: Find k such that $e = \alpha_{first-k}\alpha_{first-k+1}, ..., \alpha_{end}$ is the shortest valid subexpression containing α_t and α_{end};
7: $first = first - k$;
8: **else**
9: Find k such that $e = \alpha_{first}, ..., \alpha_{end+k-1}\alpha_{end+k}$ is the shortest valid subexpression containing α_t and α_{end};
10: $end = end + k$;
11: **end if**
12: **end while**
13: **return** e

The complexity of this algorithm is $O(n)$. Figure 5.6 illustrates the steps of the algorithm in picking out the shortest subexpression containing modules E and

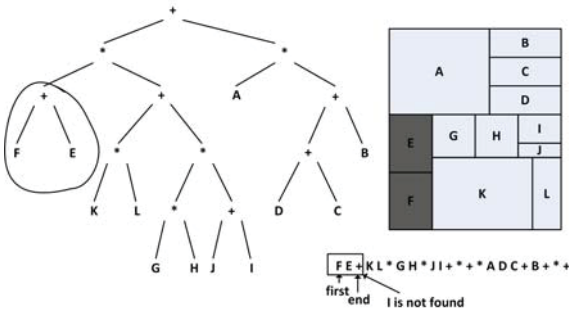
I. If the alignment group has more than two members, we can check them from the left member to the right member one by one through this algorithm. After the subexpression is picked out, the alignment constraint can be converted into a boundary constraint in this subexpression. For example, if the modules are required to align horizontally to the top side, it indicates that all the modules in the sub-floorplan have to be placed on the upper boundary, which also means that all the modules are not in the left subtree of any internal node labeled + in the slicing tree. The algorithm proposed in [16] can be applied here to solve the boundary constraint. This can be done efficiently in linear time by scanning the subexpression once. Figure 5.7(a) shows that modules E and I are on the upper boundary of their corresponding sub-floorplan. It is apparent that they satisfy the alignment constraint. In comparison, modules E and L in Figure 5.7(b) do not share the same boundary and are not able to align with each other. If modules in the subexpression do not satisfy the alignment constraint, random moves will be performed in this subexpression until we get a valid one.

5.6 Regularity of Placement

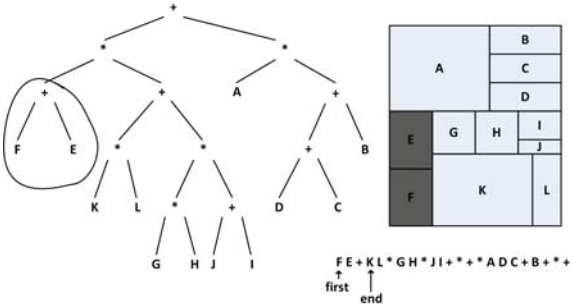
In the analog layout, it is perceived that regular structures such as symmetry and array serve suppression of device mismatch and high routability. As is discussed in section 3.2.1, common centroid and symmetry are common analog styles to reduce such mismatching. But such constraints are normally restricted to only critical modules whose mismatching affects the circuit performance due to the complex of constraint control. Then, regular structures in placement begin to draw attention from expert designers, for that it is not restricted to certain modules. It is obvious that regular structures can contribute to improving the routability by minimizing the number of vias and wire



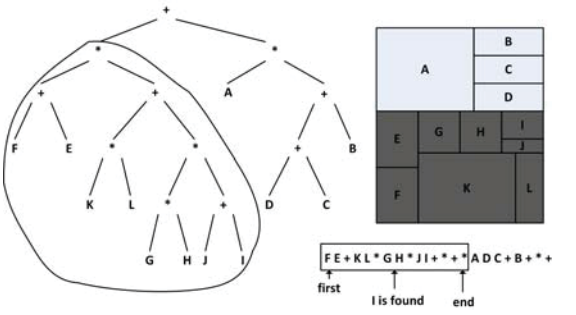
(a)



(b)

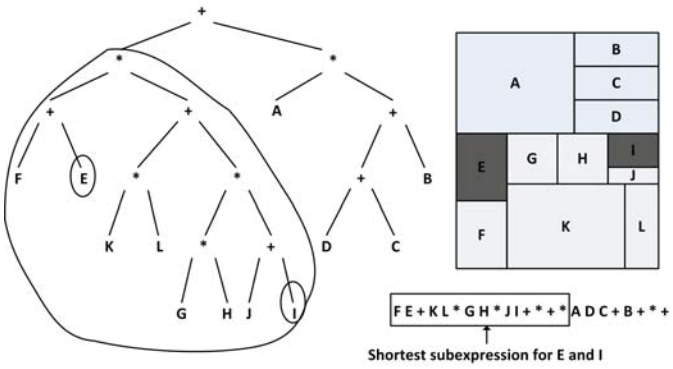


(c)

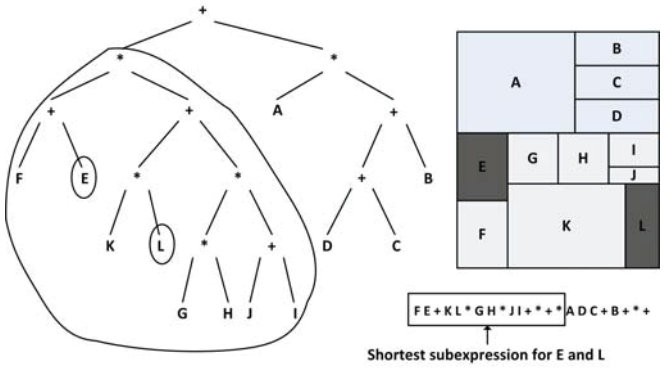


(d)

Figure 5.6: Example to illustrate the algorithm–FindSubexpression(E,I)



(a)



(b)

Figure 5.7: Illustration of alignment satisfaction

bends which are also crucial to control the layout parasitic. At the same time, surrounding the regular structures, it is possible for us to leave white space for routing during the placement stage, thus resulting in improving the packing performance significantly. Consequently, regularity is of important value in the research of placement design. My new method is proved to be effective in improving the routability of placement by calculating the regularity, which is proposed in paper [26]. The method is discussed as follows.

5.6.1 Regular Structure in Placement

It is widely accepted that there are four types of placement structures, array, row, symmetry and random. Among these structures, the intensity of regularity is perceived as $array > row > symmetry > random$. On the contrary, the flexibility of the area efficiency of them is regarded as $array < row < symmetry < random$. It is perceived that the regularity of a typical analog placement often assists to suppress device mismatch and wire parasitic as well as improve placement routability, while the flexibility contributes to yielding smaller chip area. Therefore, it is important to balance the regularity and flexibility during placement.

5.6.2 Representation of Regular Structure

In the research of regularity, a new representation [26] was proposed to represent a topology of a placement, which is derived from sequence pair.

A single-sequence can be represented by $S = (a_1, a_2, \cdots, a_n) | a_n \in 1, 2, \cdots, n$. It is defined as $a_n = s_1^{-1}(s_2(n))$ from a SP (s_1, s_2), which means S will be the same as s_2 if each module is renamed as $s_1 = (1, 2, \cdots, n)$. For instance, given

a sequence pair, $(s_1, s_2) = (d, b, c, a; a, b, c, d)$, we can get its corresponding single-sequence $S = (4, 2, 3, 1)$.

5.6.3 Type of Regular Structure

In [26], it mainly discussed three regular structures based on a single-sequence. Their definitions are given in the following.

Given a subsequence X ($X \subset S$) with two or more numbers, the maximum and minimum numbers of X are M_X and m_X. If $M_X - m_X + 1 = |X|$, X is thought to represent a rectangular extractable subsequence. Take $S = (3, 4, 1, 6, 7, 5, 8, 2)$ as an example, subsequences $(3, 4)$, $(6, 7)$, $(6, 7, 5)$, $(6, 7, 5, 8)$ and $(3, 4, 1, 6, 7, 5, 8, 2)$ are rectangular extractable.

A rectangular extractable X with $a_{k+1} - a_k = 1$ ($a_{k+1}, a_k \in X$) is able to form a horizontal single row. Similarly, X with $a_{k+1} - a_k = -1$ ($a_{k+1}, a_k \in X$) is a vertical single row. The row structure is shown in Figure 5.8.

Given an adjacent subsequence pair (X_i, X_{i+1}) of S, X_i and X_{i+1} are vertical stackable if $m_{X_i} - M_{X_{i+1}} = 1$. If a rectangular-extractable subsequence X consists of more than one horizontal single rows that are vertical stackable, X is regarded as a topology of horizontal multi-row. Similarly, vertical multi-row can be defined. Moreover, if each row of the multi-row subsequence X has the same length, X constitutes a topology of an array.

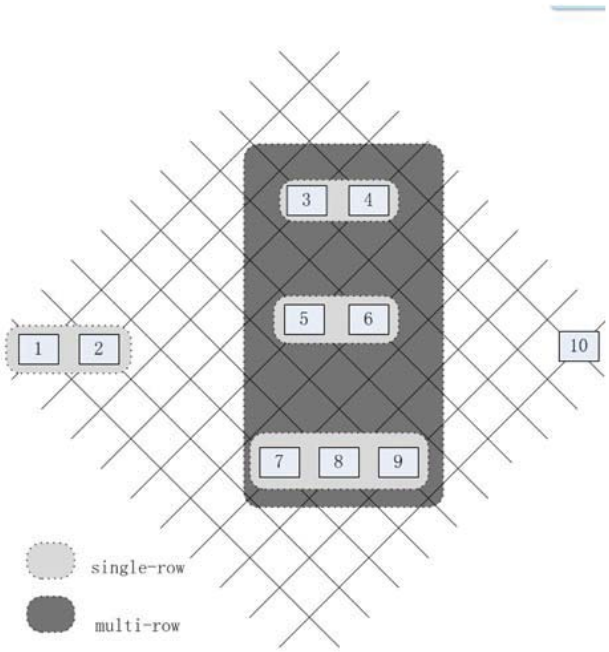


Figure 5.8: Extraction of row structures

5.6.4 Regular Structure Evaluation

A new evaluation method was proposed in [26] to continuously calculate the amount of regular structures in placement.

Given S, we can generate a different sequence $S' = (a'_1, a'_2, \cdots a'_{n-1})$, where $a'_k = a_{k+1} - a_k$. For example, given $S = (2, 1, 3, 5, 4, 6)$ as shown in Figure 5.9, we can get the corresponding $S' = (-1, 2, 2, -1, 2)$. S' is regarded as the difference of S.

The regularity value can be calculated from the following 5 steps:

1. $I_{k,l}$ is subsequence of S: $I_{k,l} = a_i | k \leq i \leq k + l - 1$, where $I_{k,l}$ represents the $k - th$ subsequence with length l.

2. Maximum and Minmum of $I_{k,l}$: $M_{k,l} = \max_{a_i \in I_{k,l}}(a_i), m_{k,l} = \min_{a_i \in I_{k,l}}(a_i)$.

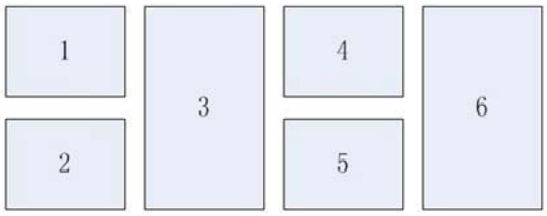


Figure 5.9: A placement for $S = (2, 1, 3, 5, 4, 6)$

3. Accumulation of Row structures: $\triangle_{k,l}$

$$
\begin{aligned}
& l = 1 : \triangle_{k,l} = 0 \\
& l = 2 : \triangle_{k,l} = \begin{cases} 1 & if |a'_k| = 1 \\ 0 & otherwise \end{cases} \\
& l = 3 : \triangle_{k,l} = \triangle_{k,l-1} + \triangle_{k+1,l-1}
\end{aligned} \tag{5.21}
$$

As mentioned above, $|a'_k| = 1$ is the necessary condition to form a horizontal or vertical single-row with two modules. Therefore, $\triangle_{k,l}$ is used to accumulate the amount of row structures.

4. Accumulation of repetitive structures: $R_{k,l}$

$$
\begin{aligned}
& l \leq 2 : R_{k,l} = 0 \\
& l \geq 3 : R_{k,l} = R_{k,l-1} + R_{k+1,l-1} - R_{k+1,l-2} + r, \qquad r = \begin{cases} 1 & if a'_k = a'_{k+l-2} \\ 0 & otherwise \end{cases}
\end{aligned} \tag{5.22}
$$

Note that a'_k denotes the relation between a_k and a_{k+1}, $a'_k = a'_{k+l-2}$ indicates that the relation of a_k and a_{k+1} equals that of a_{k+l-2} and a_{k+l-1}. Given $S = (\cdots, v+3, v+2, v+1, v, \cdots, u+3, u+2, u+1, u, \cdots)$, its

corresponding placement has a repetitive structure, because $(u+3)-(v+3)=(u+2)-(v+2)=(u+1)-(v+1)=u-v$.

5. Maximum and minimum of width and height:

$$W_{(k,l)}=\max_{a_i\in I_{k,l}} wid(s_1(a_i)), w_{(k,l)}=\min_{a_i\in I_{k,l}} wid(s_1(a_i))$$
$$H_{(k,l)}=\max_{a_i\in I_{k,l}} hei(s_1(a_i)), h_{(k,l)}=\min_{a_i\in I_{k,l}} hei(s_1(a_i)) \quad (5.23)$$

Note that $wid(x)$ and $hei(x)$ represent the width and the height of module x, respectively.

Finally, we can get the regularity from the following equation:

$$C_{reg}=\sum_{1\le l\le n}\sum_{1\le k\le n-l+1}\frac{C_{uni}(k,l)}{\triangle_{k,l}\cdot R_{k,l}+\epsilon} \quad (5.24)$$

Where ϵ is a sufficient small constant. $C_{uni}(k,l)=W_{(k,l)}-w_{(k,l)}+H_{(k,l)}-h_{(k,l)}$, and it is designed to uniform the physical dimensions inside a regular structure.

5.7 Simulated Annealing Process

5.7.1 Set of Moves

Simulated annealing is carried out in my approach by employing the following four kinds of moves $M1, M2, M3$ and $M4$. Each move can help to generate a new PE, which results in a different floorplan.

$M1$: Swapping two adjacent operands in the global expression.

$M2$: Interchanging the operators in a chain, which is a substring of operators in PE.

$M3$: Swapping two adjacent operand and operator.

$M4$: Employing M1, M2 or M3 in a SF-PE to obtain a different super-module consisting of symmetry modules.

5.7.2 Feasible Scan

In Wong-Liu algorithm [28], there is a $1-1$ correspondence between the set of normalized polish expression of length $2n-1$ and the set of slicing structures with n basic rectangles. In order to achieve the feasible floorplan from a PE, a new PE from each perturbation during the annealing process should be checked to see whether it has the properties of normalized polish expression.

5.7.3 Simulated Annealing

In the simulated annealing, the temperature is set to be 10^6 in the beginning and drops at a rate of 0.99999. n random moves are performed at each temperature until we get a different valid PE.

5.8 Experimental Results

In this section, the experimental results are presented. The floorplanner is carried out in C language and run on a Linux operating system with an Intel Xeon 1.6GHz CPU and 16GB memory. Two groups of experiments are conducted to show the effectiveness of the new proposed approach.

I compare this approach mainly with the previous approach [58] (introduced in chapter 4) on handling the same problem. I use the same data sets on the two floorplanners. In order to study their efficiency, I list the area, running time and regularity value. The method of computing regularity is introduced in section 5.6 and we can see that it is defined by SP. Hence, a PE should be first converted to a SP to compute the regularity value. The information

of circuits is shown in Table 5.1, including circuit module number, symmetry constraint number, other constraint number and net number.

Data Set	Module No.	Sym. Group No.	Other Const. No.	Net No.
ami33	33	1	2	49
ami49	49	1	2	151
Data80a	80	5	3	98
Data80b	80	5	3	98
Data80c	80	5	3	98
Data100a	100	6	3	141
Data100b	100	6	3	141
Data100c	100	6	3	141
Data80_4	80	4	3	98
Data80_5	80	5	3	98
Data80_6	80	6	3	98
Data100_4	100	4	3	141
Data100_5	100	5	3	141
Data100_6	100	6	3	141

Table 5.1: Information of circuits with constraints

Table 5.2 respectively shows the results from the approach in this chapter and the one in chapter 4. It is obvious that the newly proposed approach performs better in terms of placement area. In the runtime column, the newly proposed approach is becoming to show its advantage as the circuit module number increases. In addition, it can produce a better placement result with high routabiliy before extra adjustment, which is supported by the result of regularity. In the regularity column, the lower of the regularity indicates the more regular structures in the placement. By assigning channels around these regular structures, the placement results are more suitable for routing. Apparently, there will be less difficulty if the problems of layout constraints and regularity are solved independently. Paper [14] successfully solved the problem

of mixed constraints by SP and paper [26] improved the regularity of the placement by incorporating it as part of the primary objective. If the two problems are treated by SP at the same time, the conflicts between various objectives will lead to the consumption of extra chip area. In this section, I propose to use PE as the representation for the constraint-driven placement problem. The features of PE ensure the regularity of the final packing. Therefore, both goals can be achieved simultaneously with less chip area and less runtime.

Data	PE				SP [58]			
	DS (%)	time (s)	reg [26]	str(%) [26]	DS (%)	time (s)	reg [26]	str(%) [26]
ami33	11.5	65.83	281.9	81.8(%)	12.0	18.72	1092.0	24.2(%)
ami49	3.0	107.81	2090.6	79.6(%)	5.1	34.72	183600.9	0.0(%)
Data80a	9.9	134.24	113.9	82.5(%)	13.0	124.55	9713.9	10.0(%)
Data80b	8.5	130.05	351.3	85.0(%)	11.6	166.46	28303.3	7.5(%)
Data80c	8.6	123.25	175.7	91.25(%)	11.0	151.58	44058.2	7.5(%)
Data100a	11.3	166.11	924.1	90.0(%)	14.8	212.87	20083.9	6.0(%)
Data100b	9.4	175.96	562.4	83.0(%)	12.2	280.63	18364.1	4.0(%)
Data100c	11.6	162.99	751.5	89.0(%)	12.7	280.81	11313.0	6.0(%)

*DS represents deadspace, reg [26] represents regularity value and str [26] represents the ratio to total modules of modules in topological arrays and rows[%].

Table 5.2: Experimental results of SP and PE

Table 5.3 gives the placement results from circuits with different symmetry groups. For the approach in chapter 4, as the number of symmetry groups increases, the floorplanner takes more time to find the optimum result during the annealing process. On the contrary, the approach proposed in this chapter considers the symmetry group as a super-module, which indicates that more symmetry groups lead to less elements in PE. Therefore, this floorplanner performs faster when the number of symmetry groups increases.

Finally, two placement results are presented in Figure 5.10 and Figure 5.11,

Data Set	PE		SP [58]	
	deadspace(%)	time(s)	deadspace(%)	time(s)
Data80_4	11.1	143.38	12.7	121.25
Data80_5	9.9	134.26	12.9	125.11
Data80_6	9.2	123.59	12.1	134.96
Data100_4	9.6	188.44	12.4	232.67
Data100_5	10.9	171.86	12.8	245.94
Data100_6	11.7	161.72	12.7	280.50

Table 5.3: Experimental results of different symmetry constraint settings

which are respectively obtained from the regularity-oriented approach and the previous approach [58]. The differences between them testify that the approach proposed in this chapter makes it possible to assign proportional channels for routing.

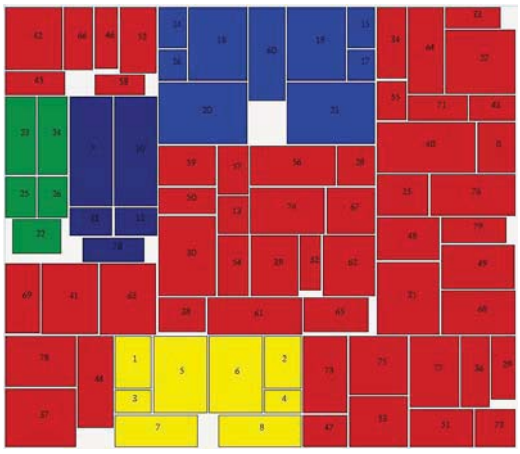


Figure 5.10: Resultant packing of Data80_4 by using PE

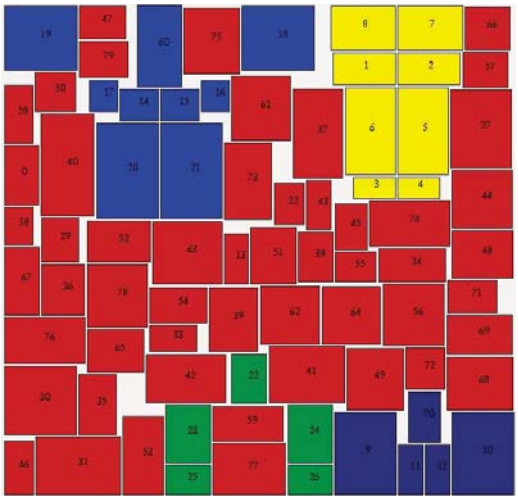


Figure 5.11: Resultant packing of Data80_4 by using SP

5.9 Summary

In this chapter, an efficient approach was proposed for the slicing floorplanning problem with mixed layout constraints. The complicated floorplanning problem was successfully tackled with less area cost and runtime. Experimental results demonstrated the feasibility of my approach. It not only dealt with the symmetry constraint in slicing floorplans which had never been accomplished before, but also improved routability significantly by containing a large number of regular structures in the final packing.

Chapter 6

Conclusion

6.1 Conclusion

In this thesis, I focused on the placement design for analog and mixed-signal ICs. In recent years, a series of layout constraints and the issues of floorplan routability have been presented in EDA field to obtain satisfactory placements. Hence, two novel approaches were proposed in order to solve this multi-objective placement problem.

In chapter 4, a two-stage approach was proposed for the placement problem which could satisfy mixed constraints and improve the routability of the placement simultaneously. All the specified constraints are enforced by inserting constraint edges and dummy nodes into the constraint graphs during the placement stage. The preliminary placement is used to estimate the net congestion probability of each module. Then based on that, the dimensions of modules with high net congestion are adjusted to reserve enough space for the subsequent channel routing. Compared with previous approach [59], a graph-based model is used in this thesis. As compared to the grid-based model, the graph-based model can leave appropriate routing space between the modules in an attempt to allow routing at the end. At the same time, the expansion process does not violate any constraint. Compared with another approach [1],

our congestion probability model contains less over-congested regions. This can avoid the problem of over-expanding.

In chapter 5, I proposed another approach by extending a previous algorithm [28] for the same placement issue. The extended algorithm was successfully applied, and therefore the regularity of the final packing could be guaranteed by the slicing structures. The symmetry constraint was satisfied by treating the symmetry groups as super-modules in the global PEs. The proposed approach is a novel method with handling the symmetry constraint in slicing floorplans which has never been accomplished before. In addition, the floorplanner generates a large number of regular structures and the corresponding placement result is more suitable for routing.

Both approaches successfully solved the complex placement problem and the experimental results showed the efficiency of the approaches on area cost, runtime and routability.

6.2 Publication

My research works have been published in the 5th Asia Symposium on Quality Electronic Design (ASQED 2013) and the IEEE Computer Society Annual Symposium on VLSI (ISVLSI 2014), respectively. The first paper [58] proposed a congestion-oriented approach for routablity-driven nonslicing floorplans. The other paper [60] proposed a regularity-oriented approach for slicing floorplans.

6.3 Future Work

The drawback of my placement design is that the approaches lack manufacture technology for further testing. I propose the following tasks that can be done

in the future. Some industry benchmarks can be used to test the efficiency and commercial values. In addition, I only used a global router "FastRoute 4.1" [43] for evaluations in this thesis. Detailed router may be used and the results will be more convincing.

Bibliography

[1] C. W. Lin, C. C. Lu, J. M. Lin, and S. J. Chang. Routability-driven Placement Algorithm for Analog Integrated Circuits. *Proceedings of ACM International Symposium on Physical Design*, pages 71–78, Mar 2012.

[2] R. Brayton and J. Cong. NSF Workshop on EDA: Past, Present, and Future (Part 1). *IEEE Design and Test of Computers*, pages 68–74, 2010.

[3] R. Brayton and J. Cong. NSF Workshop on EDA: Past, Present, and Future (Part 2). *IEEE Design and Test of Computers*, pages 62–74, 2010.

[4] G. G. E and R. A. Rutenbar. Computer-Aided Design of Analog and Mixed-Signal Integrated Circuits. *Proceedings of IEEE Journals and Magazines*, pages 1825–1854, Dec 2000.

[5] C. Toumazou, G. S. Moschytz, and B. Gilbert. Trade-Offs in Analog Circuit Design. *Kluwer Academic Publishers*, 2002.

[6] F. Balasa, S. C. Maruvada, and K. Krishnamoorthy. On the Exploration of the Solution Space in Analog Placement with Symmetry Constraints. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, pages 177–191, Feb 2004.

[7] P. H. Lin and S. C. Lin. Analog Placement Based on Novel Symmetry-Island Formulation. *Proceedings of IEEE/ACM Design Automation Conference*, pages 465–470, Jun 2007.

[8] Q. Ma, E. F. Y. Young, and K. P. Pun. Analog Placement with Common Centroid Constraints. *Proceedings of IEEE/ACM International Conference on Computer-Aided Design*, pages 579–585, Nov 2007.

[9] Y. C. Tam, E. F. Y. Young, and C. Chu. Analog Placement with Symmetry and Other Placement Constraints. *Proceedings of IEEE/ACM International Conference on Computer-Aided Design*, pages 349–354, Nov 2006.

[10] J. M. Lin, G. M. Wu, Y. W. Chang, and J. H. Chuang. Placement with Symmetry Constraints for Analog Layout Design Using TCG-S. *Proceedings of Asia and South Pacific Design Automation Conference*, pages 1135–1138, Jan 2005.

[11] P. Y. Chou, H. C. Ou, and Y. W. Chang. Heterogeneous B*-trees for Analog Placement with Symmetry and Regularity Considerations. *Proceedings of IEEE/ACM International Conference on Computer-Aided Design*, pages 512–516, Nov 2011.

[12] M. Strasser, E. Eick, H. Grab, U. Schlichtmann, and F. M. Johannes. Deterministic Analog Circuit Placement Using Hierarchically Bounded Enumeration and Enhanced Shape Functions. *Proceedings of IEEE/ACM International Conference on Computer-Aided Design*, pages 306–313, Nov 2008.

[13] L. F. Xiao and E. F. Y. Young. Analog Placement with Common Centroid and I-D Symmetry Constraints. *Proceedings of Asia and South Pacific Design Automation Conference*, pages 353–360, Jan 2009.

[14] Q. Ma, L. F. Xiao, Y. C. Tam, and E. F. Y. Young. Simultaneous Handling of Symmetry, Common Centroid , and General Placement Constraints. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, pages 85–95, Jan 2011.

[15] E. F. Y. Young, C. C. N. Chu, and M. L. Ho. Placement Constraints in Floorplan Design. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, pages 735–745, Jul 2004.

[16] E. F. Y. Young, D. F. Wong, and H. H. Yang. Slicing Floorplans with Boundary Constraints. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, pages 1385–1389, Sep 1999.

[17] W. S. Yuen and E. F. Y. Young. Slicing Floorplan with Clustering Constraint. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, pages 652–658, May 2003.

[18] E. F. Y. Young and D. F. Wong. Slicing Floorplans with Pre-Placed modules. *Proceedings of IEEE/ACM International Conference on Computer-Aided Design*, pages 252–258, Nov 1998.

[19] E. F. Y. Young, D. F. Wong, and H. H. Yang. Slicing Floorplans with Range Constraint. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, pages 272–278, Feb 2000.

[20] P. Spindler and F. M. Johannese. Fast and Accurate Routing Demand Estimation for Efficient Routability-Driven Placement. *Proceedings of Design, Automation and Test in Europe Conference and Exhibition*, pages 1–6, Apr 2007.

[21] K. Tsota, C. K. Koh, and V. Balakrishnan. Guiding Global Placement with Wire Density. *Proceedings of IEEE/ACM International Conference on Computer-Aided Design*, pages 212–217, Nov 2008.

[22] C. Li, M. Xie, C. K. Koh, J. Cong, and P. H. Madden. Routability-Driven Placement and White Space Allocation. *Proceedings of IEEE/ACM International Conference on Computer-Aided Design*, pages 394–401, Nov 2004.

[23] F. B. Mao, Y. C. Ma, N. Xu, S. H. Liu, Y. Wang, and X. L. Hong. Congestion-Driven Floorplanning Based on Two-Stage Optimizaiton. *Proceedings of IEEE International Conference on ASIC*, pages 1298–1301, Oct 2009.

[24] M. Pan and C. Chu. IPR: An Integrated Placement and Routing Algorithm. *Proceedings of IEEE/ACM Design Automation Conference*, pages 59–62, Jun 2007.

[25] J. A. Roy and I. L. Markov. Seeing the Forest and the Trees: Steiner Wirelength Optimization in Placement. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, pages 632–644, Apr 2007.

[26] S. Nakatake, M. Kawakita, T. Ito, M. Kojima, M. Kojima, K. Izumi, and T. Habasaki. Regularity-Oriented Analog Placement with Diffusion Sharing and Well Island Generation. *Proceedings of Asia and South Pacific Design Automation Conference*, pages 305–311, Jan 2010.

[27] H .Murata, K. Fujiyoshi, S. Nakatake, and Y. Kajitani. VLSI Module Placement Based on Rectangle-Packing by the Sequence-Pair. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, pages 1518–1524, Dec 1996.

[28] D. F. Wong and C. L. Liu. A New Algorithm for Floorplan Design. *Proceedings of IEEE/ACM Design Automation Conference*, pages 101–107, Jun 1986.

[29] E. Yilmaz and G. Dundar. New Layout Generator for Analog CMOS Circuits. *Proceedings of European Conference on Circuit Theory and Design*, pages 36–39, Aug 2007.

[30] M. J. M. Pelgrom, A. C. J. Duinmaijer, and A. P. G. Welbers. Matching Properties of MOS Transistors. *IEEE Journal of Solid-State Circuits*, pages 1433–1439, Oct 1989.

[31] F. M. Johannes. Partitioning of VLSI Circuits and Systems. *Proceedings of IEEE/ACM Design Automation Conference*, pages 83–87, Jun 1996.

[32] Y. C. Wei and C. K. Cheng. Ratio Cut Partitioning for Hierarchical Designs. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, pages 911–921, Jul 1991.

[33] J. H. Li and L. Behjat. A Connectivity Based Clustering Algorithm with Application to VLSI Circuit Partitioning. *IEEE Transactions on Circuits and Systems II:Express Briefs*, pages 384–388, May 2006.

[34] S. K. Lodha and D. Bhatia. Bipartitioning Circuits Using TABU Search. *Proceedings of Annual IEEE International ASIC Conference*, pages 223–227, Sep 1998.

[35] G. F. Nan, M. Q. Li, and J. S. Kou. Two Novel Encoding Strategies Based Genetic Algorithm for Circuit Partitioning. *Proceedings of International Conference on Machine Learning and Cybernetic*, pages 2182–2188, Aug 2004.

[36] F. Lin and G. M. He. An Improved Genetic Algorithm for Multi-Objective Optimization. *Proceedings of International Conference on Parallel and Distributed Computing, Applications and Technologies*, pages 938–940, Dec 2005.

[37] P. N. Guo, C. K. Cheng, and T. Yoshimura. An O-Tree Representation of Non-Slicing Floorplan and its Applications. *Proceedings of IEEE/ACM Design Automation Conference*, pages 268–273, Jun 1999.

[38] Y. C. Chang, Y. W. Chang, G. M. Wu, and S. W. Wu. B*-Trees: A New Representation for Non-Slicing Floorplans. *Proceedings of IEEE/ACM Design Automation Conference*, pages 458–463, 2000.

[39] J. M. Lin and Y. W. Chang. TCG-S: Orghogonal Coupling of P*-Admissible Representations for General Floorplans. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, pages 968–980, Jun 2004.

[40] M. P. H. Lin, H. Zhang, M. D. F. Wong, and Y. W. Chang. Thermal-driven Analog Placement Considering Device Matching. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, pages 325–336, Mar 2011.

[41] M. Cho, K. Lu, K. Yuan, and D. Z. Pan. BoxRouter 2.0: A Hybrid and Robust Global Router with Layer Assignment for Routability. *ACM Transactions on Design Automation of Electronic Systems*, Mar 2009.

[42] T. H. Wu, A. Davoodi, and J. T. Linderoth. GRIP: Global Routing via Integer Programming. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, pages 72–84, Jan 2011.

[43] Y. Xu, Y. H. Zhang, and C. Chu. FastRoute 4.0: Global Router with Efficient Via Minimization. *Proceedings of Asia and South Pacific Design Automation Conference*, pages 576–581, Jan 2009.

[44] Y. J. Chang, T. H. Lee, and T. C. Wang. GLADE: A Modern Global Router Considering Layer Directives. *Proceedings of IEEE/ACM International Conference on Computer-Aided Design*, pages 319–323, Nov 2010.

[45] T. H. Lee, Y. J. Chang, and T. C. Wang. An Enhanced Global Router with Consideration of General Layer Directives. *Proceedings of ACM International Symposium on Physical Design*, pages 53–60, 2011.

[46] G. J. Nam, K. A. Sakallah, and R. A. Rutenbar. A New FPGA Detailed Routing Approach via Search-Based Boolean Satisfiability. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, pages 674–684, Jun 2002.

[47] S. Batterywala, N. Shenoy, W. Nicholls, and H. Zhou. Track Assignment: A Desirable Intermediate Step between Global Routing and Detailed Routing. *Proceedings of IEEE/ACM International Conference on Computer-Aided Design*, pages 59–66, Nov 2002.

[48] M. M. Ozdal. Detailed-Routing Algorithms for Dense Pin Clusters in Integrated Circuits. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, pages 340–349, Mar 2009.

[49] M. Gester, D. Muller, T. Nieberg, C. Panten, C. Schulte, and J. Vygen. Algorithms and Data Structures for Fast and Good VLSI Routing. *Proceedings of IEEE/ACM Design Automation Conference*, pages 459–464, Jun 2012.

[50] Y. H. Zhang and C. Chu. RegularRoute: An Efficient Detailed Router Applying Regular Routing Patterns. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, pages 1655–1668, Sep 2013.

[51] R. Okuda, T. Sato, H. Onodera, and K. Tamariu. An Efficient Algorithm for Layout Compaction Problem with Symmetry Constraints. *Proceedings of IEEE/ACM International Conference on Computer-Aided Design*, pages 148–151, Nov 1989.

[52] E. Felt, E. Charbon, E. Malavasi, and A. Sangiovanni-Vincenteli. An Efficient Methodology for Symbolic Compaction of Analog ICs with Multiple Symmetry Constraints. *Proceedings of European Design Automation Conference*, pages 148–153, Sep 1992.

[53] K. Okada, H. Onodera, and K. Tamaru. Compaction with Shape Optimization. *Proceedings of IEEE Custom Integrated Circuits Conference*, pages 545–548, May 1994.

[54] J. M. Cohn, D. J. Garrod, R. A. Rutenbar, and L. R. Carley. Analog Device-Level Layout Automation. *Kluwer Academic Publishers*, 1994.

[55] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller. Equation of State Calculations by Fast Computing Machines. *Journal of Chemical Physics*, Jun 1953.

[56] J. Lou, S. Thakur, S. Krishnamoorthy, and H. S. Sheng. Estimating Routing Congestion Using Probabilistic Analysis. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, pages 32–41, Jan 2002.

[57] S. Nakatake. Structured Placement with Topological Regularity Evaluation. *Proceedings of Asia and South Pacific Design Automation Conference*, pages 215–220, Jan 2007.

[58] H. X. Zhou, C. W. Sham, and H. L. Yao. Congestion-Oriented Approach in Placement for Analog and Mixed-Signal Circuits. *Proceedings of the Fifth Asia Symposium on Quality Electronic Design*, pages 97–102, Aug 2013.

[59] L. F. Xiao, E. F. Y. Young, X. Y. He, and K. P. Pun. Practical Placement and Routing Techniques for Analog Circuit Designs. *Proceedings of IEEE/ACM International Conference on Computer-Aided Design*, pages 675–679, Nov 2010.

[60] H. X. Zhou, C. W. Sham, and H. L. Yao. Slicing Floorplans with Handling Symmetry and General Placement Constraints. *Proceedings of IEEE Computer Society Annual Symposium on VLSI*, pages 112–117, Jul 2014.