UNIVERSITY OF CALIFORNIA, SAN DIEGO

Optimizations of Manufacturability and Manufacturing in Nanometer-Era VLSI

A dissertation submitted in partial satisfaction of the
requirements for the degree Doctor of Philosophy
in
Computer Science

by

Xu Xu

Committee in charge:

Professor Andrew B. Kahng, Chair
Professor Chung-Kuan Cheng
Professor Bill Lin
Professor Tajana Simunic-Rosing
Professor Bang-Sup Song

2006

The dissertation of Xu Xu is approved, and it is acceptable in quality and form for publication on microfilm:

Chair

University of California, San Diego

2006

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

I would like to express my deep and sincere gratitude to my supervisor, Professor Andrew B. Kahng, for his continuous support in the Ph.D. program. His wide knowledge and logical way of thinking have been of great value for me. He is responsible for involving me in the DFM and manufacturing area in the first place. He showed me different ways to approach a research problem and the need to be persistent to accomplish any goal.

I wish to express my warm and sincere thanks to Professor Ion Mandoiu and Professor Alex Zelikovsky for their helpful discussion and suggestion on various projects. Their ideas and concepts have had a remarkable influence on my entire career in the field of VLSI manufacturing research. A special thank goes to Dr. Bao Liu for his insightful comments on timing related projects. His guidance was very important in avoiding pitfalls. I owe my gratitude to Dr. Charles Chiang and Dr. Subarna Sinha, who gave me the opportunity to work with them on the AltPSM project at Synopsys, Inc.

I would also like to thank my thesis committee members, Professor Chung-Kuan Cheng, Professor Tajana Simunic-Rosing, Professor Bang-Sup Song and Professor Bill Lin, for taking time out of their busy schedules to review and evaluate my research work. I am quite grateful for their valuable feedback.

During my Ph.D. study, I have collaborated with many colleagues for whom I have great regard, and I wish to extend my warmest thanks to Puneet Gupta, Swamy Muddu, Chul-Hong Park, Sherief Reda, Kambiz Samadi, Puneet Sharma and Qinke Wang. I wish them the best in the future.

I owe my loving thanks to my wife Yaqing Chen and my coming daughter Jade. Without their encouragement and understanding it would have been impossible for me to finish this work.

I am indebted to my parents for their care and love through my life, their invariant support wherever I go and whatever I do. This thesis is dedicated to them.

The material presented in this thesis is based on the following publications.

- Chapter II is based on the following publication: A. B. Kahng, X. Xu and A. Zelikovsky, "Fast Yield-Driven Fracture for Variable Shaped-Beam Mask Writing", *Photomask and Next-Generation Lithography Mask Technology XI*, April 2006, accepted and to appear.

- Chapter III is based on the following publication: A. B. Kahng, I. I. Mandoiu, X. Xu and A. Zelikovsky, "Enhanced Design Flow and Optimizations for Multi-Project Wafers", *IEEE Transactions on CAD* (2006), accepted and to appear.

- Chapter IV is based on the following publication: C. Chiang, A. B. Kahng, S. Sinha, X. Xu and A. Zelikovsky, "Fast and Efficient Bright-Field AAPSM Conflict Detection and Correction", *IEEE Transactions on CAD* (2006), accepted and to appear.

- Chapter V is based on the following draft: A. B. Kahng, B. Liu and X. Xu, "Perfect Matching Based Optimal Post-Routing Redundant Via Insertion for Manufacturing and Timing Yield Improvement".

The dissertation author was the primary researcher and author. My coauthors (Prof. Andrew B. Kahng, Prof. Ion Mandoiu, Prof. Alex Zelikovsky, Dr. Bao Liu, Dr. Charles Chiang and Dr. Subarna Sinha) have all kindly approved the inclusion of the aforementioned publications in my thesis.

VITA

1975	Born, Maanshan, China
1998	B.S. University of Science & Technology of China
1999	M.S., Carnegie Mellon University
2006	Ph.D., University of California, San Diego

All papers have authors listed in alphabetical order.

PUBLICATIONS

A. B. Kahng and X. Xu, "Accurate Pseudo-Constructive Wirelength and Congestion Estimation", *ACM International Workshop on System-Level Interconnect Prediction*, April 2003, pp. 61-68.

A. B. Kahng and X. Xu, "Local Unidirectional Bias for Smooth Cutsize-Delay Tradeoff in Performance-Driven Bipartitioning", *Proc. ACM/IEEE Intl. Symp. on Physical Design*, April 2003, pp. 81-86.

A. B. Kahng, I. I. Mandoiu, S. Reda, X. Xu and A. Zelikovsky, "Design Flow Enhancements for DNA Arrays", *Proc. IEEE Intl. Conf. on Computer Design*, October 2003, pp. 116-123.

A. B. Kahng, I. I. Mandoiu, S. Reda, X. Xu and A. Zelikovsky, "Evaluation of Placement Techniques for DNA Probe Array Layout", *Proc. IEEE/ACM Intl. Conference on Computer-Aided Design*, November 2003, pp. 262-269.

A. B. Kahng and X. Xu, "Local Unidirectional Bias for Cutsize-Delay Tradeoff in Performance-Driven Bipartition", *IEEE Transactions on CAD* 23(4) (2004), pp. 464-471.

A. B. Kahng, I. I. Mandoiu, Q. Wang, X. Xu and A. Zelikovsky, "Multi-Project Reticle Floorplanning and Wafer Dicing", *Proc. ACM/IEEE Intl. Symp. on Physical Design*, April 2004, pp. 70-77.

A. B. Kahng, X. Xu and A. Zelikovsky, "Yield- and Cost-Driven Fracturing for Variable Shaped-Beam Mask Writing", *Proc. 24th BACUS Symposium on Photomask Technology and Management*, September 2004, pp. 360-371.

C. Chiang, A. B. Kahng, S. Sinha, X. Xu and A. Zelikovsky, "Bright-Field AAPSM Conflict Detection and Correction", *Proc. Design Automation and Testing in Europe*, March 2005, pp. 908-913.

P. Gupta, A. B. Kahng, C.-H. Park, K. Samadi and X. Xu, "Topography-Aware Optical Proximity Correction for Better DOF margin and CD control", *Photomask and Next-Generation Lithography Mask Technology X*, April 2005, pp. 844-854.

A. B. Kahng, I. I. Mandoiu, X. Xu and A. Zelikovsky, "Yield-Driven Multi-Project Reticle Design and Wafer Dicing", *Proc. 25th BACUS Symposium on Photomask Technology and Management*, October 2005, SPIE (5992) pp. 1247-1257. **(1st Place of Best Poster Award and Best Paper Award)**

C. Chiang, A. B. Kahng, S. Sinha and X. Xu, "Fast and Efficient Phase Conflict Detection and Correction in Standard-Cell Layouts", *Proc. ACM/IEEE Intl. Conf. on Computer-Aided Design*, November 2005, pp. 149-156.

A. B. Kahng, I. I. Mandoiu, X. Xu and A. Zelikovsky, "Multi-Project Reticle Design and Wafer Dicing under Uncertain Demand", *Proc. European Mask and Lithography Conference*, January 2006, pp. 45-54.

A. B. Kahng, B. Liu, K. Samadi and X. Xu, "Statistical Crosstalk Aggressor Alignment Aware Interconnect Delay Calculation", *ACM Intl. Workshop on System-Level Interconnect Prediction*, March 2006, pp. 91-97.

A. B. Kahng, B. Liu and X. Xu, "Constructing Current-Based Gate Models Based on Existing Timing Library", *Proc. ACM/IEEE Intl. Symp. on Quality Electronic Design*, March 2006, pp. 37-42.

A. B. Kahng, I. I. Mandoiu, S. Reda, X. Xu and A. Zelikovsky, "Computer-Aided Optimization of DNA Array Design and Manufacturing", *IEEE Transactions on CAD* 25 (2) (2006), pp. 305-320.

A. B. Kahng, X. Xu and A. Zelikovsky, "Fast Yield-Driven Fracture for Variable Shaped-Beam Mask Writing", *Photomask and Next-Generation Lithography Mask Technology XI*, April 2006, accepted and to appear.

A. B. Kahng, B. Liu and X. Xu, "Statistical Gate Delay Calculation with Crosstalk Alignment Consideration", *Proc. Great Lake Symposium on VLSI*, April 2006, pp. 223-228.

A. B. Kahng, I. I. Mandoiu, S. Reda, X. Xu and A. Zelikovsky, "Computer-Aided Optimization of DNA Array Design and Manufacturing", in *Design Automation Methods and Tools for Microfluidics-Based Biochips* (K. Chakrabarty, ed.), Springer, 2006, accepted and to appear.

P. Gupta, A. B. Kahng, C.-H. Park, K. Samadi and X. Xu, "Wafer Topography-Aware Optical Proximity Correction", *IEEE Transactions on CAD* (2006), accepted and to appear.

C. Chiang, A. B. Kahng, S. Sinha, X. Xu and A. Zelikovsky, "Fast and Efficient Bright-Field AAPSM Conflict Detection and Correction", *IEEE Transactions on CAD* (2006), accepted and to appear.

A. B. Kahng, I. I. Mandoiu, X. Xu and A. Zelikovsky, "Enhanced Design Flow and Optimizations for Multi-Project Wafers", *IEEE Transactions on CAD* (2006), accepted and to appear.

A. B. Kahng and X. Xu, "A General Framework for Multi-Flow, Multi-Layer, Multi-Project Reticles Design", *Proc. 27th BACUS Symposium on Photomask Technology and Management*, September 2006, accepted and to appear.

A. B. Kahng, C.-H. Park and X. Xu, "Fast Dual-Graph-Based Hot-Spot Detection", *Proc. 27th BACUS Symposium on Photomask Technology and Management*, September 2006, accepted and to appear.

A. Balasinski, A. B. Kahng, W. Sachs-Baker and X. Xu, "A Procedure and Program to Calculate Shuttle Mask Advantage", *Proc. 27th BACUS Symposium on Photomask Technology and Management*, September 2006, accepted and to appear.

FIELDS OF STUDY

Major Field: Computer Science

Studies in Computer-Aided Design.
Professor Andrew B. Kahng
University of California, San Diego

ABSTRACT OF THE DISSERTATION

Optimizations of Manufacturability and Manufacturing in Nanometer-Era VLSI

by

Xu Xu
Doctor of Philosophy in Computer Science
University of California, San Diego, 2006
Professor Andrew B. Kahng, Chair

As optical lithography advances into the 65nm technology node and beyond, minimum feature size outpaces the lithography wavelength. As a result, mask/wafer manufacturing yield improvement and cost reduction have been widely accepted as key factors for aggressive technology scaling. This thesis is concerned with the following four manufacturability/manufacturing problems.

Fracturing Mask manufacturing for the 90nm and 65nm nodes increasingly deploys variable shaped-beam mask writing machines. The pervasive use of OPC leads to dramatic increase in the number of thin trapezoids, which significantly decrease the mask manufacturing yield. This thesis suggests an optimal integer linear programming based *fracturing* approach and fast heuristics which substantially reduce sliver count in comparison to leading commercial fracturing tools.

MPW *Multi-project wafers (MPW)* provide an attractive mask manufacturing cost reduction solution for low-volume production designs by sharing the cost of mask tooling among up to tens of designs. This thesis proposes a comprehensive MPW flow aimed at minimizing manufacturing cost through (1) multi-project reticle floorplanning, and (2) wafer shot-map and dicing plan definition.

PSM In the context of wafer manufacturing, *Alternating-Aperture Phase Shift Masking (AAPSM)* will be used to image critical features on the polysili-

con layer at advanced technology nodes. This technology imposes additional constraints on IC layouts, beyond traditional design rules. Phase conflicts must be detected and removed to enable the use of AAPSM. This thesis makes two key contributions: (1) a new, computationally efficient approach to detect a minimal set of phase conflicts, which when corrected will produce a phase-assignable layout; and (2) a novel layout modification scheme for correcting phase conflicts in standard-cell blocks.

Redundant Vias Finally, a large part of wafer manufacturing yield loss is due to via voids, which can be relieved by *redundant via* insertion or via doubling. This thesis proposes perfect matching based post-route via doubling which achieves optimum yield improvement. Redundant interconnects or "short loops" are also applied to maximize the number of doubled vias. Experimental results show that near 100% via doubling coverage can be achieved with simultaneously optimal redundant via and short loop insertion in the post-route stage.

Chapter I

Introduction

I.A The VLSI Manufacturing Process

A graphic overview of the VLSI manufacturing process is shown in Figure I.1. The IC manufacturing process starts with cutting a silicon cylinder into silicon wafers. Design patterns are then imaged from the mask to the wafer. After wafer testing, the wafer is sawn with side-to-side slicing lines to get raw dies, which are then packaged and undergo final testing. The dominant manufacturing optimization is to improve yield and thus reduce cost.

The most important and challenging step of the manufacturing process is the wafer patterning or photolithography step. The main operations of a photolithographic cycle are shown in Figure I.2. Two crucial steps which determine the manufacturing yield and cost are mask-making and stepper exposure.

- *Mask-making* is an important process to transfer the design from layout to mask. Photomask creation begins with an ultra-pure glass plate with a surface deposition of chromium. A laser or e-beam writing process is used to selectively remove chromium and create the photomask. With the shrinking of VLSI feature size into the deep-subwavelength regime, the minimum feature size has outpaced the wavelength. Compensating techniques such as optical proximity correction (OPC) and phase-shifting masks (PSM) have been adopted to overcome the gap. As a result, mask patterns have become

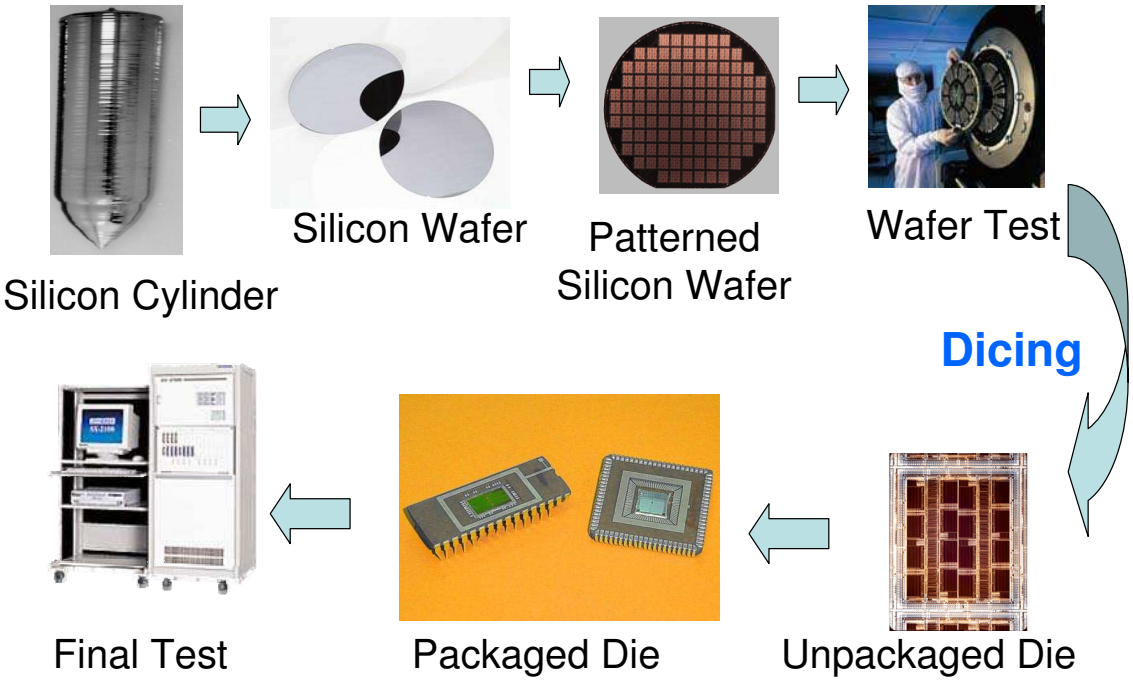


Figure I.1: Overview of VLSI manufacturing process.

much more complicated, significantly reducing the mask manufacturing yield and throughput, and increasing the mask manufacturing cost.

- *Stepper exposure* is a manufacturing step that transfers the design patterns from mask to wafer photoresist. A stepper exposes a photoresist-coated wafer to single-wavelength, deep-UV light passing through a reticle which contains the image of a single device layer. The term "stepper" comes from the "step-and-repeat" action of moving the wafer on its x and y axes to align the reticle with each individual device position as the corresponding mask pattern is transferred.

I.B Sub-Wavelength Lithography Challenges

With the shrinking of VLSI feature size in the subwavelength regime, process variation has become a critical factor for performance, power, and cost

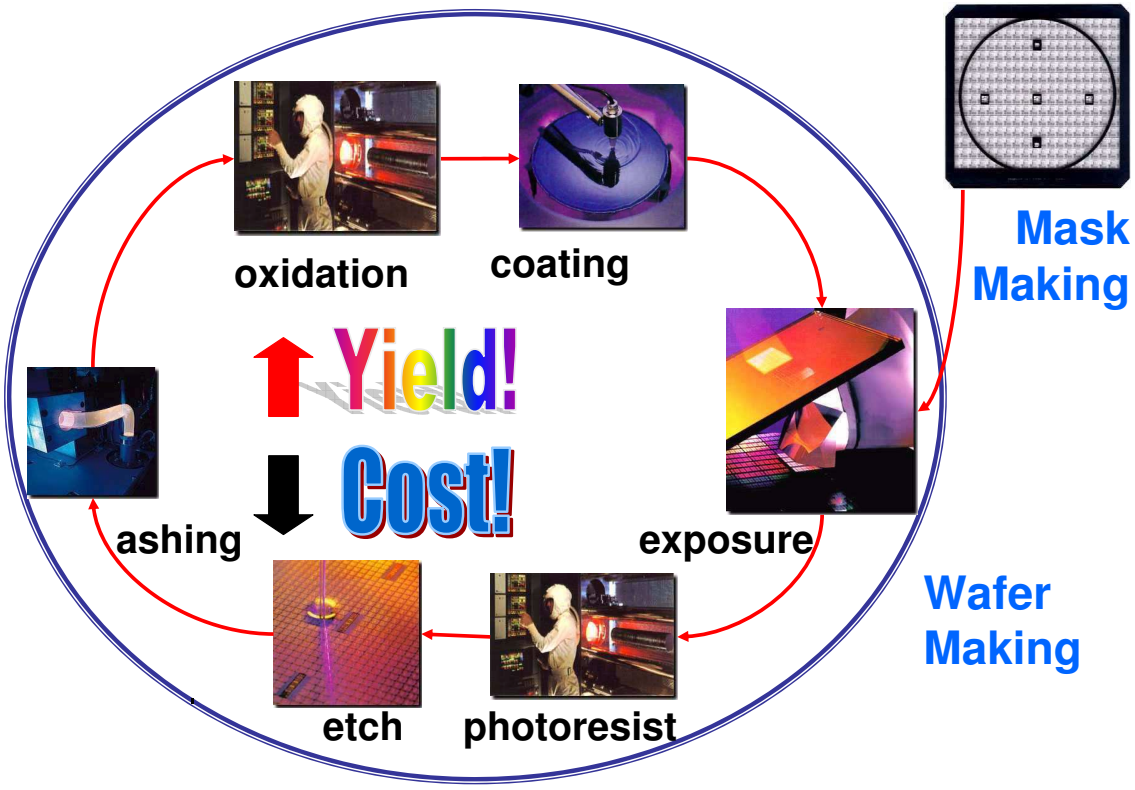


Figure I.2: Overview of a photolithographic cycle.

(i.e., yield). As a result, manufacturability and manufacturing have been widely accepted as key factors for aggressive technology scaling. Optical lithography has been a key technology enabler of the aggressive IC technology scaling implicit in Moore's Law. Although electron-beam and X-ray lithography, as well as maskless direct write techniques have been proposed as promising techniques for the nanometer regime, optical lithography is still the most widely used method in volume VLSI production. However, as shown in Figure I.3, minimum silicon feature sizes have outpaced the introduction of advanced lithography hardware solutions. A popular metaphor is that this is like trying to paint a 1/4-inch wide line using a 1-inch diameter paintbrush. As a result, the critical dimension (CD) variations are extremely difficult to reduce; this in turn significantly reduces the manufacturing yield and increases cost. In order to follow the international technology roadmap for semiconductors (ITRS) requirements for smaller dimensions, adjustment of the

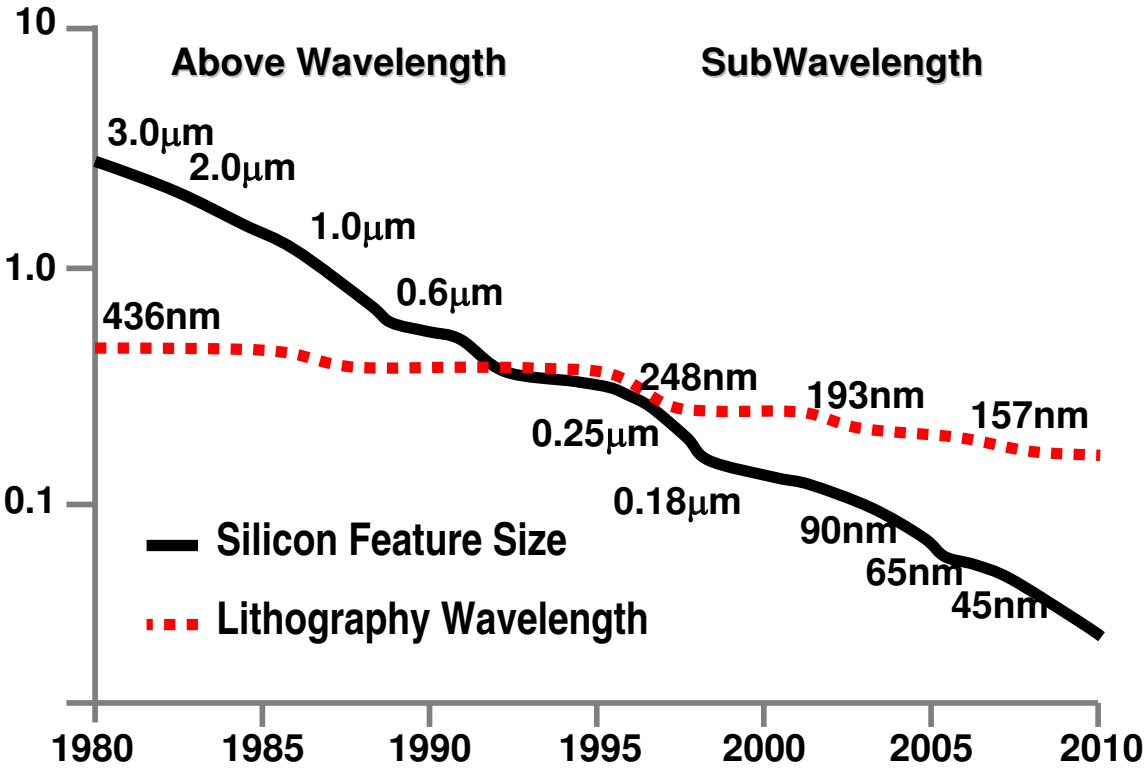


Figure I.3: Predicted manufacturing feature size versus lithography wavelength.

whole lithographic infrastructure, i.e. exposure tool, mask, photoresist, and process, is required. Traditionally, reducing the exposure wavelength has been the trend for the exposure tools. Resolution enhancement techniques (RETs), such as off-axis illumination (OAI), phase-shifting masks (PSMs), and optical proximity correction (OPC), have allowed extended usage of existing tools as cost effective solutions in manufacturing. Due to the ever-reduced development cycles for each wavelength generation, more drastic RETs are required to extend the working lifetime of exposure systems and ensure sufficient technology maturity. Hence, RETs, which are imperative for current photo lithography, bring new challenges to current VLSI industry.

I.C Overview of Manufacturing Problems and Our Solutions

As the semiconductor industry slowly recovers from the economic downturn, it faces an accelerated technology roadmap and shortened product life cycles. The semiconductor manufacturer's challenge is to balance the time to introduce new technologies with reasonable investment costs and minimized risk. Lithography exposure tools are often the most expensive in the wafer fab and therefore dominate miniaturization progress. Patterning of many layers in integrated circuit manufacturing requires a large number of exposure tools, resulting in high total cost for lithography equipment. On the other hand, the increasingly large gap between silicon feature size and lithography wavelength implies a significant yield loss, which becomes the main obstacle of IC manufacturing. Therefore, the key manufacturing optimization goals are yield improvement and cost reduction. This thesis addresses the following four problems related to manufacturing yield improvement and cost reduction.

I.C.1 Fracturing

Mask manufacturing for the approaching 90nm and 65nm nodes increasingly deploys variable shaped beam (VSB) mask writing machines. This has led to high interest in the fracturing methods which are at the heart of layout data preparation for VSB mask writing. The pervasive use of OPC leads to a dramatic increase in the number of thin trapezoids, or *slivers*, which significantly decrease the mask manufacturing yield. In Chapter II, we address the problem of optimizing VSB mask writing by taking into account the large number of constraints imposed by mask writing equipment as well as the manufacturability of optically corrected mask layouts. We suggest a new solution approach based on integer linear programming (ILP). The main advantage of the new method is that the ILP finds optimal solutions while being flexible enough to take into account all specified requirements. We also propose fast heuristics based on ray segment selection to

achieve a good tradeoff between solution quality and runtime. Experimental comparisons with leading industry tools show significant improvement in quality, as well as acceptable scalability, of the proposed methods. In particular, our fracturing solutions on three industry testcases dramatically reduce sliver count (which reflects the risk of mask critical-dimension errors) by 83.7% and 60.5% compared to two commercial fracturing tools while also reducing shot count (which reflects write time and mask cost) by 5.5% and 0.6%. Our results reveal significant headroom that can be exploited by future design-to-mask tools to reduce the manufacturing variability and cost of IC designs.

I.C.2 Multi-Project Wafers

The aggressive scaling of VLSI feature size and the pervasive use of advanced reticle enhancement technologies lead to dramatic increases in mask costs, pushing prototype and low volume production designs to the limit of economic feasibility. Multiple project wafers (MPW) provide an attractive solution for such designs by sharing the cost of mask tooling among up to tens of designs. However, MPW reticle floorplanning and wafer dicing introduce complexities not encountered in typical, single-project wafers. In Chapter III, we propose a comprehensive MPW flow aimed at minimizing the number of wafers needed to fulfill given die production volumes. Our flow includes three main steps: (1) multi-project reticle floorplanning, (2) wafer shot-map definition, and (3) wafer dicing plan definition. For each of these steps we propose improved algorithms as follows. Our reticle floorplanner uses hierarchical quadrisection combined with simulated annealing to generate "diceable" floorplans observing given maximum reticle sizes. The new wafer shot-map definition step fully utilizes the round wafer real estate by extracting the maximum number of functional dies from both fully and partially printed reticle images. Finally, our dicing planner allows multiple side-to-side dicing plans for different wafers, allows different reticle projection rows or columns within a wafer, and improves dicing yield by partitioning each wafer into a small number of parts before individual die extraction. Experiments on industry testcases show

that our methods outperform significantly not only previous methods in the literature, but also reticle floorplans manually designed by experienced engineers. We also initiate the study of MPW use for production under demand uncertainty and propose efficient algorithms for two main optimizations that arise in this context: reticle design under demand uncertainty and on-demand wafer dicing. Preliminary experiments on simulated data show that our methods help reduce the cost overheads incurred by demand uncertainty, yielding solutions with a cost close to what is achievable when a priori knowledge of production volumes is available.

I.C.3 Alternating-Aperture Phase Shift Masking

Alternating-Aperture Phase Shift Masking (AAPSM) is used to image critical features on the polysilicon layer at smaller technology nodes. This technology imposes additional constraints on the layouts beyond traditional design rules. Of particular note is the requirement that all critical features be flanked by opposite-phase shifters, while the shifters obey minimum width and spacing requirements. Phase conflicts must be detected and removed to enable the use of AAPSM. Our work in Chapter IV has two key contributions: (1) a new computationally efficient approach to detect a minimal set of phase conflicts, which when corrected will produce a phase-assignable layout; and (2) a novel layout modification scheme for correcting these phase conflicts in standard-cell blocks. Unlike previous formulations of this problem, the proposed solution for the conflict detection problem does not frame it as a graph bipartization problem. Instead, a simpler and more computationally efficient reduction is proposed. This simplification greatly improves the runtime, while maintaining the same improvements in the quality of results. An average runtime speedup of 5.9x is achieved using the new flow. A new layout modification scheme suited for correcting phase conflicts in large standard-cell blocks is also proposed. Our experiments show that the percentage area increase for making standard-cell blocks phase-assignable ranges from 1.7-9.1%.

I.C.4 Via Doubling

In DFM/nanometer VLSI designs, high defect density is expected in the latest technologies, especially for subsurface defects, or via voids. The causes of the failed vias may be airborne particles, electro-migration, or thermal stress induced voiding. Via doubling, or redundant via insertion, is an effective DFM technique for yield improvement, electro-migration alleviation, and performance enhancement by inserting a second "back-up" via. In Chapter V, we propose a maximum matching based post-route via doubling technique which achieves optimum yield improvement. We introduce redundant interconnect or "short loops" in addition to traditional minimum spacing redundant via insertion, and achieve maximum number of doubled vias. We further propose a timing-driven redundant via insertion method based on a performance sensitivity computation, along with a weighted maximum matching for timing yield improvement. Experimental results show that our perfect matching based redundant via insertion reduces the number of un-doubled (i.e., critical) vias by 99.4% and 98.0% compared with two best previous post-routing via doubling heuristics, and increases the via doubling coverage from 94.5% and 98.2% to 99.97%. One interesting observation is that near 100% via doubling coverage can be achieved with simultaneously optimal redundant via and short loop insertion in the post-route stage. Our timing-driven redundant via insertion achieves up to 3.3% timing yield improvement compared with timing-oblivious redundant via insertion with the same number of doubled vias on the critical paths.

Chapter II

Yield-Driven Fracturing

Mask manufacturing for the approaching 90nm and 65nm nodes increasingly deploys variable shaped beam (VSB) mask writing machines. This has led to high interest in the *fracturing* methods which are at the heart of layout data preparation for VSB mask writing. Some commercial tools are available for handling the sliver minimization problem in fracture, such as CATS from Synopsys and Fracturem from Mentor Graphics. However, the number of slivers in the existing fracturing solutions can be significantly reduced. In this chapter, we set out the main requirements for fracturing and suggest a new solution approach based on integer linear programming (ILP). The main advantage of the new method is that the ILP finds optimal solutions while being flexible enough to take into account all specified requirements. However, the main disadvantage is long runtime. To save the runtime, we also propose a new ray-segment selection heuristic which can find a near-optimal fracturing solution in practical time. Experimental comparisons with leading industry tools show significant improvement in quality, as well as acceptable scalability, of the proposed methods. In particular, our fracturing solutions reduce *shot count* (which reflects write time and mask cost) and dramatically reduce *sliver count* (which reflects the risk of mask critical-dimension errors). Our results reveal significant headroom that can be exploited by future design-to-mask tools to reduce the manufacturing variability and cost of IC designs. The proposed fast heuristics can also solve the reverse-tone fracturing problem in practical time

for large industry testcases.

II.A Introduction

The onset of the 90nm and 65nm technology nodes is accompanied by a sharp increase in mask manufacturing cost, which becomes prohibitive for low-volume designs. The mask cost increase is directly associated with increased write time and data volume, which is caused by highly complex *reticle enhancement techniques* (RETs) that are used to optimize mask apertures for the manufacture of deeply subwavelength features. To decrease the turnaround time of mask manufacturing and to improve mask quality for highly complex layouts, the trend is away from increasingly expensive raster mask writing[1] and toward variable shaped beam (VSB) e-beam mask writing. Today's VSB mask writing machines offer a plethora of rapidly advancing technologies, with beam currents reaching 50kV, support for slanted apertures, and a host of data formats for pattern generation from "fractured" layout information. This chapter addresses the problem of optimizing VSB mask writing to take into account the constraints imposed by mask writing equipment as well as manufacturability of optically corrected layouts.

II.A.1 Definitions and Problem Statement

In the following, we refer to dimensions of various parameters on the *mask*. Corresponding dimensions on the wafer can be obtained by scaling down these values by the *stepper reduction ratio* = the ratio of image size on the reticle to that on the wafer (in each of the x and y dimensions), which is usually between 4 and 5. Exposure data for VSB writing is described as a set of single-exposure units, commonly referred to as *shots*. There are several limitations on the shape and size of a single shot:

[1]The mask writing grid size - roughly 1/50 of the minimum feature size - continues to decrease with technology scaling. Thus, raster writing methods take increasingly large amounts of time to write the mask, since the size of the die and mask remain roughly constant across technology nodes. With their fixed spot size, the raster tools do not achieve the shape resolution of the variable-shaped beam tools.

(a) each shot should be either a rectangle or, more generally, an axis-parallel trapezoid;

(b) the side size of each shot cannot exceed a certain maximum threshold value M; and

(c) the minimum width of each shot should be above a certain minimum threshold value ϵ.

The first two constraints are "hard" since the VSB writing technology cannot reproduce arbitrary shapes, and since exposure quality can be guaranteed only up to a certain extent. Currently, the value of M is between $2\mu m$ and $3\mu m$ (e.g., $2.55\mu m$ for a recent Toshiba VSB writing tool). This corresponds to an image size of $0.5\mu m$ to $0.75\mu m$ on the wafer with a $4\times$ reduction stepper. The third constraint is "soft" – narrow trapezoids having width below the critical value ϵ (which is typically around 100nm on the mask scale), henceforth referred to as *slivers*, can still be reproduced. However, as shown in Figure II.1, small feature size proportional to k_1 [2] will lead to an increase in the mask error enhancement factor (MEEF), which is formally defined as the ratio of changes in the pattern printed on the wafer to the corresponding changes in the pattern written on the reticle [8] [9] [10] . An increase of MEEF will cause larger CD variation and more manufacturing defects, likely reducing mask and / or wafer manufacturing yield. In general, either the number of slivers or the total length of slivers should be minimized.[3]

Besides the above constraints on shot shape and dimension, there are also constraints on how a general layout geometry (polygon) can be represented as a *set* of shots:

(d) shots cannot overlap, i.e., each feature is partitioned into disjoint shots;

[2] $W_{min} = \frac{k_1 \lambda}{NA}$, where W_{min} is the minimum feature size, λ is the exposure wavelength and NA is numerical aperture.

[3] Nakao et al. [1] have suggested that slivers be counted only if they share a boundary with the fractured polygon.

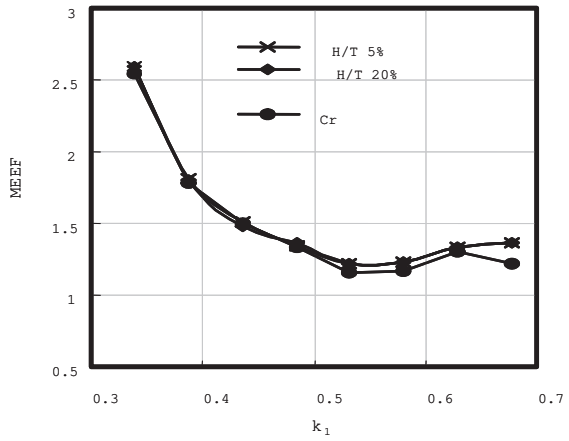


Figure II.1: Relationship between MEEF and k_1 for different mask types, where k_1 is proportional to feature size.

(e) shots should not slice critical features, e.g., minimum-width poly gates, for which special fabrication accuracy is required; and

(f) slant edges should not be partitioned.

These constraints (d-f) are, in general, hard: (1) any overlap between shots would be comparatively overexposed with respect to nonoverlapped shots; (2) critical features are already narrow and their width, even without partitioning, is difficult to control; and (3) slant edges cannot be controlled with the same accuracy as axis-parallel edges. Figure II.2 shows one example of the shots obtained from the fracturing of post-RET layout data.

The design-through-mask data flow is shown in Figure II.3 [15]. In this process, the number of shots is roughly proportional to the mask writing time, which in turn affects the mask cost. Therefore, the main optimization objective for fracturing, i.e., partitioning of polygons into shots, is to minimize the number of shots. Some slivers can be avoided, but a significantly reduced number or total length of slivers may be achievable only at the cost of an increased number of shots. Therefore, we suggest an integrated objective function that seeks to minimize a

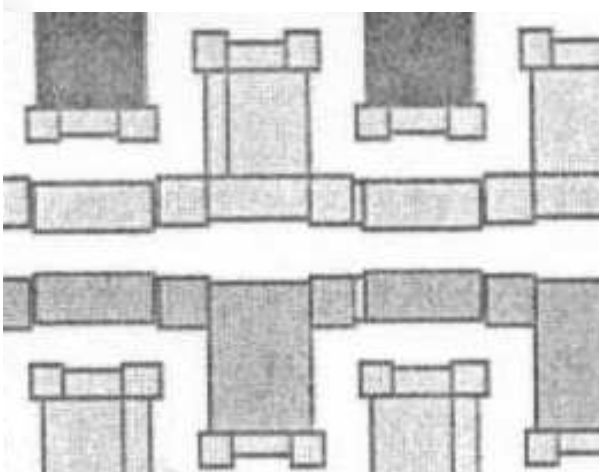


Figure II.2: An example of fractured polygons.

linear combination of the number of shots and the number (or the total length) of slivers, with empirically chosen scaling coefficients.

Fracturing Problem. Given a simple polygon P with axis-parallel and 45-degree slant edges, along with specified critical dimensions, partition P into axis-parallel trapezoidal shots subject to constraints (a),(b),(d),(e),(f) minimizing either

$$\#(shots) + W_C \#(slivers) \tag{II.1}$$

or

$$\#(shots) + W_L L(slivers) \tag{II.2}$$

where $\#(shots)$ and $\#(slivers)$ are respectively the numbers of shots and slivers, $L(slivers)$ is the total sliver length, and W_C and W_L are respectively scaling coefficients for the number of slivers and the total sliver length.

II.A.2 Previous Work

Fracturing of a polygon into basic shapes (rectangles and trapezoids) is a well-studied problem. The standard formulation is to minimize the number of

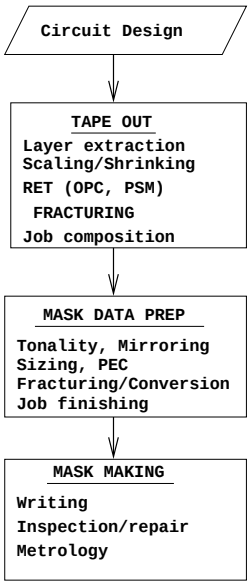


Figure II.3: Mask data process flow.

shots subject to constraints (a) and (d) only, disregarding all other constraints specified above. Ohtzuki [5] has given an exact $O(n^{5/2})$ algorithm for polygon fracturing into rectangles where n is the number of vertices of a polygon. The algorithm is based on finding a maximum independent set in a bipartite graph where vertices correspond to certain lines slicing the given polygon. Imai and Asano [4] have further sped up this algorithm to $O(n^{3/2} \log n)$ and also generalized it to the optimal partition into trapezoids [3]. Unfortunately, these theoretically nice algorithms are not flexible enough to take into account additional important constraints.

Nakao et al. [1] have developed a fairly complicated ad hoc heuristic based on the generalization of the same bipartite graph which takes into account all other constraints except the constraint (b). In fact, they have introduced a different objective – minimize the weighted length of slivers and slices cutting through critical features – while minimizing shot number over all obtained solutions that are (sub)optimal with respect to the new objective. Their heuristic is fast but es-

sentially disregards slant edges during fracturing (rather, slant edges are integrated after rectilinear fracturing). Moreover, the method does not guarantee optimum fracturing and does not appear to allow any way of incorporating the maximum shot size constraint (b). A standard way of fracturing polygons into bounded-size shots is to first partition the polygon into a small number of trapezoids, and then partition these large trapezoids into maximum-size shots. Such an approach is obviously suboptimal; our investigations show that it gives a significant increase in shot count, and we do not pursue it further.

A different technological aspect of the Fracturing Problem has been addressed in a recent series of papers by Shulze et al. [2] and Cobb et al. [6, 7]. In the standard data preparation flow for VSB mask writing, a post-RET layout in GDSII format is transferred into the MEBES mask writing format, which is then further transferred into VSB formats supported by various VSB mask writing machines. The drawback of this flow is that GDSII and VSB formats are hierarchical, while MEBES does not support hierarchy. The cited works suggest ways to avoid layout flattening (e.g., to exclude MEBES format from the flow), which result in drastic reduction of data volumes as well as processing times. The improvements that we develop in the present work are complementary to such methods.

II.A.3 Contributions

In this chapter, we apply an integer linear programming (ILP) approach to the Fracturing Problem. Our contributions include:

- a more adequate formulation of the Fracturing Problem for VSB mask writing machines including maximum shot size constraint (b),
- new ILP formulation for the Fracturing Problem capturing all constraints (a-f),
- fast heuristics based on ray segment selection, and
- validation of the proposed heuristic with available industry tools.

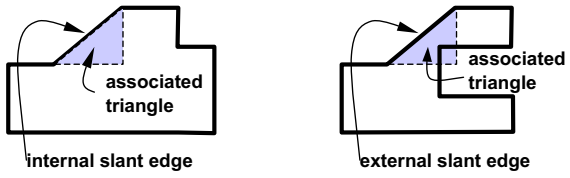


Figure II.4: Internal and external slant edges.

In Section II.B we describe the ILP formulations for the fracturing problem. Section II.C is devoted to the fast ray-segment selection based heuristic. Section II.D expands the solution space with the introduction of auxiliary ray segments. Section II.E compares our results to those of leading commercial fracturing tools, and shows that the proposed approach offers significant improvement in quality (shot count and sliver count), as well as acceptable scalability. Finally, Section II.F gives conclusions and future research directions.

II.B Integer Linear Programming Formulation

In this section, we describe an integer linear programming formulation to optimally solve the Fracturing Problem. We generally consider a simple polygon P with axis-parallel and 45-degree-slant edges. With each slant edge we associate a triangle which internally intersects P, as shown in Figure II.4. If this triangle is completely inside P, the slant edge is called *internal*; otherwise, it is called *external*. In order to simplify exposition of the Fracturing Problem without compromising rigor we will further exclude from consideration external slant edges.[4] Note that partitioning of the internal slants can always be avoided.

In order to partition P into trapezoids, which are convex quadrangles, it is necessary to start at least one *partitioning line*, further referred to as a *ray*, from each *concave point* on the boundary of P, i.e., a point with internal angle greater

[4]In fact, our ILP-based solution can take external slant edges into account and our implementation does not exclude them, but the exposition will be unnecessarily overcomplicated.

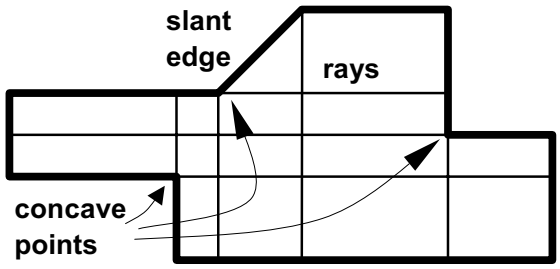


Figure II.5: A polygon P with concave boundary points. The dashed vertical and horizontal rays are originated from concave points. The internal triangle associated with the slant edge is shaded.

than π (see Figure II.5). The rays should be axis-parallel since only axis-parallel trapezoids can be made in a single shot. Besides rays from concave points, there are also rays from the (two) endpoints of slant edges, which form the associated internal triangles. At least one of the two rays from a slant edge should be used since the slant edge can serve only as a side edge of a trapezoid.

Our ILP formulation is based on the following grid graph G (see Figure II.6). For each concave point, there are two rays directed inside the polygon P and drawn to the opposite side of P. For each end point of a slant edge, a single ray is drawn to the opposite side of P. The vertices of the graph G are all intersection points of the rays between themselves and with the boundary of the polygon P. The edges of G are all segments of the rays connecting neighboring vertices on the same ray or boundary segment. We enumerate all vertical rays with x-coordinates $X_i,\ i = 1, \ldots, hr$ and all horizontal rays with y-coordinates $Y_j,\ j = 1, \ldots, vr$. Then, the vertex of G that lies on the i-th vertical ray and j-th horizontal ray is denoted $v_{i,j}$. Each horizontal edge of G between vertices $v_{i,j}$ and $v_{i+1,j}$ is denoted $e^h_{i,j}$, and each vertical edge between $v_{i,j}$ and $v_{i,j+1}$ is denoted $e^v_{i,j}$. The number of variables and constraints in our ILP will be $O(n^2)$, where $n = vr + hr$ is the number of concave points plus the number of slant edges on the boundary of the given polygon P.

We consider a fracturing of polygon P as a subgraph of the grid graph G. Each fracturing should choose certain edges of G as edges of its trapezoids. Let us introduce a Boolean variable $x^d(i,j)$, $d = v, h$, $i = 1, \ldots, hr$ and $j = 1, \ldots, vr$, which is set to 1 if the edge $e^d_{i,j}$ belongs to the chosen fracturing, and 0 otherwise. Note that variables corresponding to the boundary edges are always set to 1 since they belong to any fracturing.

In the rest of this section we show that the ILP objectives (II.1) and (II.2) can be computed based on Formulas (II.9), (II.11) and (II.13), and that the ILP constraints are expressed in (II.3), (II.4), (II.5), (II.6), (II.8), (II.10) and (II.12). We will first describe several types of constraints to ensure:

- convexity of rectilinear fracturing elements;
- convexity of trapezoids with slants;
- keeping intact slant edges and CDs;
- maximum shot size; and
- counting of shots and slivers.

II.B.1 Convexity Constraints

The following constraints force any point $v_{i,j}$ (regardless of whether its position is on the boundary or inside P) to be convex with respect to fracturing edges. In other words, among the four edges incident to an internal point, there could be zero fracturing edges, two fracturing edges along the same ray, or three fracturing edges forming a "T"-shape.

$$\begin{aligned} x^h(i-1,j) + x^v(i,j-1) &\leq 2x^h(i,j) + 2x^v(i,j) \\ x^h(i,j) + x^v(i,j-1) &\leq 2x^h(i-1,j) + 2x^v(i,j) \\ x^h(i,j) + x^v(i,j) &\leq 2x^h(i-1,j) + 2x^v(i,j-1) \\ x^h(i-1,j) + x^v(i,j) &\leq 2x^h(i,j) + 2x^v(i,j-1) \end{aligned} \tag{II.3}$$

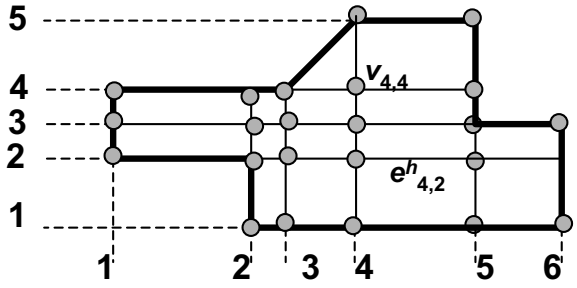


Figure II.6: The grid graph G.

Indeed, let us consider the first constraint. If neither the bottom nor the left edge incident to the point is chosen, then the constraint trivially holds. If only one of the bottom and left edges is chosen, the constraint ensures that at least one more edge should be chosen, since no vertex can have degree 1 in the fracturing. If both bottom and left edge are chosen, then at least one more edge should also be chosen, i.e., the node $v_{i,j}$ cannot be concave since it should have degree at least 3. The next three constraints similarly ensure the same property for the bottom and right edges, top and right edges, and top and left edges.

II.B.2 Constraints for Slant Edges and Critical Features

The endpoints of slant edges are treated in a different manner, as follows. Let a slant boundary edge connect points $v_{i,j}$ and $v_{i+k,j+l}$ (see Figure II.7). If $v_{i,j}$ is concave, then the following constraint ensures that at least one of the two non-boundary incident edges belongs to the fracturing solution.

$$x^v(i, j-1) + x^h(i, j) \geq 1 \tag{II.4}$$

A similar constraint is added if the other endpoint is also concave. To ensure that the slant edge is a part of an axis-parallel trapezoid, the following constraint is introduced:

$$x^v(i, j-1) + x^h(i+k, j+l-1) \geq 1 \tag{II.5}$$

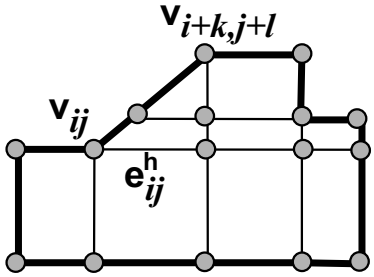


Figure II.7: Treating a slant edge. Since the point v_{ij} is concave, either the edge e^h_{ij} or the edge $e^v_{i,j-1}$ should be used in any fracturing. Since v_{ij} and $v_{i+k,j+l}$ are endpoints of a slant boundary edge, either the edge e^h_{ij} or the edge $e^v_{i+k,j+l-1}$ should be used in any fracturing.

Since the constraints (e-f) on fracturing are hard, we forbid slicing of slant edges and critical features by simply setting to 0 the variables corresponding to slicing edges.

II.B.3 Maximum Shot Size Constraints

No previous fracturing methods directly minimize shot count in the presence of maximum shot size constraints. The ILP approach can smoothly incorporate such constraints, as follows. Let M be the upper bound on horizontal/vertical shot dimension. If the length of any edge in the graph G exceeds M, we add new rays partitioning such edges accordingly and modify the graph G. When combining a chain of edges into the boundary of a single shot we ensure that it never exceeds M. Let X_i, $i = 1, \ldots, vr$, be the x-coordinates of the vertical rays. For each pair (i, i'), $i,\ i' \in \{1, \ldots, vr\}$, satisfying

$$X_{i'+1} - X_{i-1} \geq M$$

and

$$X_{i'} - X_{i-1} < M$$

we introduce the following constraint for each j, $j = 1, \ldots, hr$.

$$x^h(i,j) + x^h(i+1,j) + \ldots + x^h(i',j) \geq 1 \tag{II.6}$$

Similar constraints are also introduced for the vertical dimension.

II.B.4 Counting Shots

The objective of the Fracturing Problem is to minimize the number of shots and the number of slivers, so it is necessary to accurately count these numbers in the ILP. The subgraph H of G corresponding to a fracturing is obviously planar. Each shot corresponds to a face of the graph H, and we can apply the Eulerian formula relating the numbers of vertices, edges and faces of a planar graph:

$$\#(shots) = |E(H)| - |V(H)| + 1 \tag{II.7}$$

The number of edges $|E(H)|$ is easily computed, since it is equal to the sum of all variables corresponding to edges of G. To find the number of vertices of H we must exclude from all vertices of G the vertices which become isolated in H. This is done by introducing a variable $y(i,j)$ for each vertex of G which is set to 0 if v_{ij} is isolated, and to 1 otherwise. Since the objective is to minimize the number of shots, (II.7) implies that it is sufficient to introduce an upper bound on $y(i,j)$:

$$y(i,j) \leq x^h(i-1,j) + x^v(i,j-1) + x^v(i,j) + x^v(i,j) \tag{II.8}$$

Finally, the number of shots is counted as:

$$\#(shots) = 1 + \sum_{d,i,j} x^d(i,j) - \sum_{i,j} y(i,j) \tag{II.9}$$

II.B.5 Counting and Finding Length of Slivers

For each possible sliver, i.e., a pair of vertical (or horizontal) rays X_i and $X_{i'}$ such that $|X_i - X_{i'}| < \epsilon$, we introduce a sliver variable $sl(i,i')$ which is set to 1 if there is a sliver in the fracturing, and to 0 otherwise. The corresponding constraints are as follows:

$$sl(i,i') \geq x^v(i,j) + x^h(i',j) - 1, \quad j = 1, \ldots, hr \tag{II.10}$$

The resulting number of slivers is computed as

$$\#(slivers) = \sum_{|X_i - X_{i'}|<\epsilon} sl(i,i') + \sum_{|Y_j - Y_{j'}|<\epsilon} sl(j,j') \tag{II.11}$$

If, instead of counting the number of slivers, we wish to take into account the total length of slivers, we can use variables $sl(i,i',j)$ such that

$$sl(i,i',j) \geq x^v(i,j) + x^h(i',j) - 1 \tag{II.12}$$

The total length of slivers would then be computed as

$$\begin{aligned} L(slivers) \;=\; & \sum_{|X_i - X_{i'}|<\epsilon} \sum_{j} sl(i,i',j)(X_j - X_{j-1}) + \\ & \sum_{|Y_j - Y_{j'}|<\epsilon} \sum_{i} sl(i,j,j')(Y_i - Y_{i-1}) \end{aligned} \tag{II.13}$$

II.C Ray-Segment Selection Based Heuristics

Although the integer linear programming (ILP) method can find the optimal fracturing, it becomes prohibitively slow for polygons with a large number of vertices, and heuristic partitioning of large polygons may severely degrade the solution quality. In this section, we propose a new ray-segment selection heuristic which can find a near-optimal fracturing solution in practical time while being flexible enough to take into account all specified requirements.

II.C.1 Polygon Partitioning With Coincident Rays

Since one ray should be sent from each concave point, the total number of rays is equal to the number of concave points N. Whenever a ray is sent it can stop as soon as it reaches the opposite side of P or another orthogonal ray which has been sent earlier. Two rays may coincide if they reach each other's origin, which is defined as a *coincident ray*. See Figure II.8 for an example. If I denotes the number of pairs of coincident rays, then the total number of different rays

that should be sent in order to partition P into trapezoids is $N - I$. Note that each time we send a ray, we increase the number of faces of the resulting planar graph by 1, and hence the total number of trapezoids in the polygon partitioning is $\#(trapezoids) = N - I + 1$.

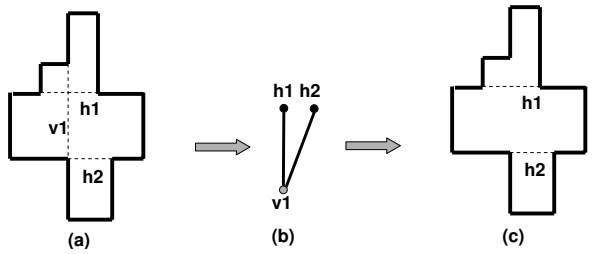


Figure II.8: (a) A polygon with five concave points and three rays between them. (b) The corresponding bipartite graph B, in which the vertices $\{h1, h2\}$ form the maximum independent set. (c) The corresponding partitioning into sub-polygons without coincident rays.

Thus, in order to minimize the number of trapezoids, one should maximize the set of coincident rays. Note that all such rays cannot intersect as well as cannot come from the same concave point. The last constraint can be expressed in a graph B with vertex set $R = RV \cup RH \cup S$, where RV (respectively, RH) is the set of coincident vertical (respectively, horizontal) rays and S is the set of rays sent from concave points which are simultaneously the endpoints of slant edges. Two vertices u and v of the graph B are adjacent if the corresponding rays are in conflict. This may happen in the following two cases (see Figure II.8):

(a) $u \in RV$ and $v \in RH$ and the corresponding rays intersect; and

(b) $u \in RV \cup RH$ and $v \in S$ and the corresponding rays are orthogonal and sent from the same concave point.

The maximum set of coincident rays that we seek corresponds to a maximum independent set in the graph B. In general, finding a maximum independent set

is NP-hard, but in our case the graph B is bipartite since all edges are between vertices corresponding to orthogonal rays. According to König's theorem, finding the maximum independent set in a bipartite graph can be reduced to maximum matching and, therefore, can be done efficiently. Then the polygon will be divided by the selected set of coincident rays into some small polygons without coincident rays.

II.C.2 Ray-Segment Selection Formulation of the Fracturing Problem

The ray-segment selection heuristic is based on the following directed grid graph $G'(V, E)$ (see Figure II.9). For each concave point and the endpoint of slant edges, there are rays directed inside the polygon P and drawn to the opposite side of P. V consists of all intersection points of the rays and the concave vertices of the polygon P. E includes all segments of the directed rays from one vertex to a neighboring vertex on the same ray or boundary segment. We enumerate all vertical rays with x-coordinates $X_i,\ i = 1, \ldots, hr$ and all horizontal rays with y-coordinates $Y_j,\ j = 1, \ldots, vr$. Then, the vertex of G' that lies on the i-th vertical ray and j-th horizontal ray is denoted $v_{i,j}$. From each vertex $v_{i,j} \in V$, there are exactly one horizontal directed edge denoted as $e^h_{i,j}$ and one vertical directed edge denoted as $e^v_{i,j}$. In G', every ray becomes a linked list of ray segments. For a ray segment e, the next ray segment in the list is denoted as $Next(e)$, which is null if e is the end of the list. The number of ray segments will be $O(n^2)$, where n is the number of concave points. The fracturing of a polygon P is denoted as F, which is a subset of $E - P$.

Fracturing can be viewed as a process of "eliminating" all concave points since any rectilinear polygon without concave vertices is a rectangle.

Definition II.1 *Two ray segments form a* **conflict pair** *if they are from the same vertex.*

For a ray segment e, the ray segment form a conflict pair with e is denoted as $C(e)$.

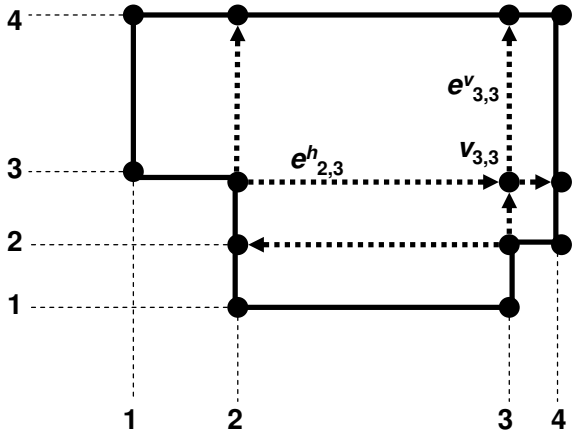


Figure II.9: The directed grid graph G'.

Theorem II.1 *F is a fracturing solution of the rectilinear polygon P with N concave points and the number of rectangles in the fracturing solution is $N+1$, if the following conditions are satisfied: (1) at least one of the ray segments from each concave point of P is in F; (2) if $e \in F$ and $Next(e) \neq$ null, either $Next(e) \in F$ or $C(Next(e)) \in F$; and (3) at most one ray from each conflict pair is in F.*

Proof: We only need to prove that every vertex in G' becomes convex with the edges of $P \bigcup F$. The convexity of any rectilinear concave vertex v of P is guaranteed with the first constraint since the internal angles of v are 90^o and 180^o with one ray segment from v. As shown in Figure II.10, for any internal vertex v:

1. If there is no ray segment point to v (case (a)), it is a convex point;
2. If there is a ray segment $e \in F$ point to v and $Next(e) \in F$ (case (b)), it is a convex point since both internal angles are 180^o;
3. If there is a ray segment $e \in F$ point to v and $C(Next(e)) \in F$ (case (c)), it is a convex point since the internal angles are 90^o, 90^o and 180^o.

Therefore, every vertex in G is convex and hence all fractured polygons are rectangles.

Next, we want to prove that the number of rectangles is $N + 1$. From the Eulerian formula relating the numbers of vertices, edges and faces of a planar graph:

$$\#(rectangles) = |E(H)| - |V(H)| + 1 \tag{II.14}$$

For the polygon P, the edge number is equal to the vertex number. Also, for each internal point v, there is exactly one ray segment from v according to the third constraint if one ray segment from v is in F. Therefore, the edge number is equal to the vertex number for each internal point. For every concave vertex v of P, there is a ray segment from v. There are N such edges and $\#(rectangles) = N + 1$. □

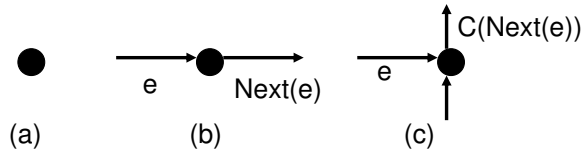


Figure II.10: The cases of internal vertices v: (a) no ray segments point to v; (b) a ray segment $e \in F$ points to v and $Next(e) \in F$; and (c) a ray segment $e \in F$ points to v and $C(Next(e)) \in F$.

II.C.3 Gain-Based Ray Segment Selection Heuristics

Based on Theorem II.1, we propose the "Gain-Based Ray Segment Selection" or **GRSS** heuristics for the fracturing problem.

Definition II.2 *A rectangle is a* **sliver** *if its minimum width* $< \epsilon$*, where* ϵ *is a given threshold value.*

The main objective of our method is to reduce the sliver number of the fracturing solution to increase the mask manufacturing yield.

Definition II.3 *For any edge* $e \in G'$*, its* weight $W(e)$ *is equal to the number of parallel edges in* $F \cup P$ *which are within distance of* ϵ *and their projects in the*

Input: Rectilinear polygon P and its grid graph $G(V, E)$
Output: $F \subset E$ with minimized sliver number
1. $F \leftarrow \emptyset$; $S \leftarrow \{e \mid e$ from a concave point of $P\}$
2. Calculate the gains of edges in S
3. **While** $(S \neq \emptyset)$
4. Select the edge e with the largest gain, $F \leftarrow F \cup \{e\}$
5. $S \leftarrow S - \{e, C(e)\}$
6. **If** $(Next(e) \neq$ null)
7. $S \leftarrow S \cup \{Next(e)\}$
8. Update gains of the edges in S

Figure II.11: Gain-based ray segment selection algorithm.

orthogonal direction overlap with the projection of e. In other words, the weight of e is equal to the increased sliver number with the selection of e.

Definition II.4 *For any edge $e \in G'$, its* gain $g(e) = W(C(e)) - W(e)$. *In other words, the gain of e is equal to the number of "saved" slivers with the selection of e.*[5]

Our proposed gain-based ray segment selection algorithm is shown in Figure II.11. The algorithm starts with a candidate pool S which includes all ray segments from a concave point of P. We calculate the gains of all edges in S. Then in each iteration, one ray segment e is selected and its conflict pair $(\{e, C(e)\})$ is deleted from S (Line 3-4). If e is not the end ray segment in the ray list, the next ray segment of the list $Next(e)$ will be added to S. The gains of the edges in S are updated in Line 6. The iterative selection process continues until S is empty. The intuition behind this gain based algorithm is to greedily select the ray segment which leads to the largest save on the sliver number.

To speed up the gain update and ray segment selection process, we adopt a "linked bucket" structure to store the candidate pool S as shown in Figure II.12. There are five linked lists or buckets, each store the edges in S with a gain between

[5] *Unlike the algorithm presented in [1] which uses weights to choose rays, we adopt gains to advise ray segment selection.*

-2 and 2. The edge deletion, addition and gain update operations take constant runtime with this data structure.

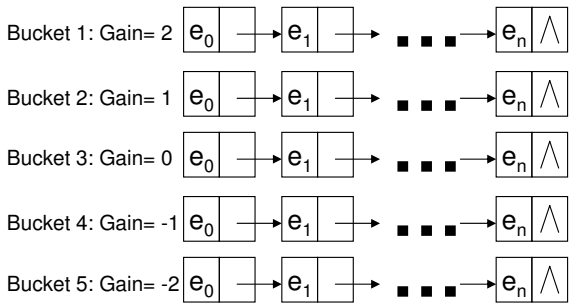


Figure II.12: Bucket structure for candidate pool S.

II.D Auxiliary Ray Segments for Sliver Minimization

Definition II.5 *An* auxiliary ray *is a ray whose starting point is not a concave vertex of* P.

From the Eulerian formula, the number of rectangles will be increased with auxiliary rays. However, the sliver number may be reduced with the additional rays at the expense of rectangle number increase. For the example shown in Figure II.13, the sliver number is one in the fracturing solution without auxiliary ray (case (a)); while the sliver number is zero with one auxiliary ray (case (b)). To the best of our knowledge, no previous methods have been proposed to construct the auxiliary rays to minimize the sliver number. Since adding too many rays may significantly increase the complexity and runtime, it is crucial to limit the number of auxiliary rays being added.

We propose the rule-based auxiliary rays addition method for efficient tradeoff between runtime and sliver number.

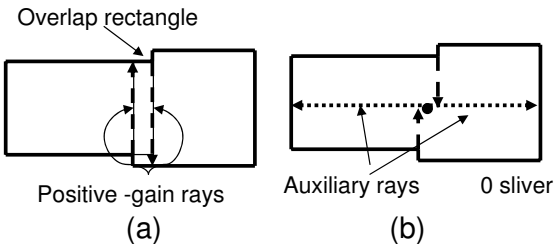


Figure II.13: Fracturing of a polygon: (a) without auxiliary ray and (b) with auxiliary ray.

Definition II.6 *If a ray segment e is from a concave point of P and $g(e) > 0$, the ray containing e is a* positive-gain ray.[6]

Definition II.7 *For two parallel rays of different directions, let one ray from vertex v_0 located at (x_0, y_0) and the other ray from vertex v_1 located at (x_1, y_1), their* overlap rectangle *is a rectangle whose four corners are (x_0, y_0), (x_0, y_1), (x_1, y_1) and (x_1, y_0). Its width is the distance between the two rays and its length is the other dimension of the rectangle.*

Auxiliary Ray Addition Rule: Two auxiliary rays are added if two parallel positive-gain rays of opposite directions satisfy the following constraints: (1) the overlap rectangle located inside P; (2) the length of the overlap rectangle is $> 2\epsilon$ and the width of the overlap rectangle is $< \epsilon$; and (3) no existing rays intersect with the two rays and partition the overlap rectangle into two rectangles whose lengths are $> \epsilon$. Two auxiliary rays from the center of the overlap rectangle are added to G' as shown in Figure II.13 (b) and the two ray segments from the center of the overlap rectangle are added to S in Line 1 of the algorithm shown in Figure II.11.

[6] *There are two positive-gain rays in Figure II.13 (a).*

II.E Experimental Results

We use three industry testcases to evaluate the performance of our fracturing approach. Design A and Design B are from Photronics, Inc. [12]. Design C is a post-RET cell layout from a leading foundry 130nm standard-cell library; RET was inserted by Mentor Calibre. The basic properties of the three designs are listed in Table II.1. For each testcase, the minimum number of fractured trapezoids is calculated using the method given in Section II.C.1. We have implemented our algorithm in ANSI C, and use the CPLEX 8.100 Mixed Integer Optimizer [11] to solve all Integer Linear Programming instances. In all runs, we set the runtime limit for CPLEX to 10 CPU seconds.

We compare our fracturing code against state-of-art commercial fracturing tools. Two specific examples of leading commercial tools are Mentor Calibre v9.3_2.10 Fracturem [13] and Synopsys CATS v2501 [14]. In our experiments, we set the maximum shot size as $2.55\mu m$ and the slivering size as $100nm$, following parameters for a recent Toshiba VSB writing tool. The stepper reduction ratio is four. All tests are run on an Intel Xeon 2.4GHz CPU.[7]

The fracturing results in Table II.2 show that our ILP method can reduce the number of slivers by 82%, 79% and 29% compared to Tool A, Tool B, and Tool C respectively, while also reducing the number of shots by 5.5%, 0.6% and -2.5%.[8] The runtime of our method is much larger than that of the commercial tools due to the large runtime of the Integer Linear Programming solver. However, we note that the runtimes given in Tables II.2 are for flattened layouts. When using hierarchical layout representation, the number of different polygons will be drastically reduced. Hence, the CPU cost of ILP solver can also be amortized in exact correspondence to any amortization of RET insertion costs: as soon as

[7]The results in Table II.2 are anonymized to satisfy no-benchmarking restrictions in the vendor tool licenses. Per the license terms, we do not ascribe any specific results to any specific vendors or tools. However, we believe that our results demonstrate the magnitude of the solution quality gap in current tools.

[8]We have also made a comparison when the slivering size is set to $50nm$. In that experiment, our method ILP reduces the number of slivers by 96%, 97% and 81% compared with Tool A, Tool B, and Tool C respectively, while reducing the number of shots by 6.3%, 4.7% and 1.4%.

optical correction of a feature is fixed, fracturing can also be decided. Our gain based ray segment selection heuristic (GRSS) reduces the number of slivers by 83.7%, 81.1% and 60.5% compared to commercial tools with negligible runtime overhead.

Table II.1: Properties of testcases.

Testcase	# Polygons	Min # trapezoids	slants	# vertices
Design A	602	9613	21	24807
Design B	676	16273	17	38305
Design C	104	476	0	1580

Table II.2: Fracturing results with slivering size $\epsilon = 100nm$ and maximum shot size $M = 2.55\mu m$. CPU time is given in seconds.

	Design A			Design B			Design C		
Method	shots	slivers	CPU	shots	slivers	CPU	shots	slivers	CPU
Tool A	10754	6111	0	17335	11572	0	589	318	0
Tool B	10455	4451	0	17130	10797	0	566	147	0
Tool C	9755	786	2	17195	6502	3	592	66	0
ILP	9750	417	134	17684	2750	222	518	83	8
GRSS	9786	183	1	17656	2691	4	527	61	0

II.F Summary

We suggest a new optimal ILP approach for the fracturing problem in VSB mask writing. Our new approach improves both shot count and, very substantially, sliver count, in comparison to leading commercial fracturing tools. Although the processing of layouts is much slower than in commercial products, we believe that the ILP-based heuristic framework is scalable to full-chip layout processing, particularly when fracturing is done hierarchically (cf., e.g., [2]) and/or in a distributed fashion. We suggest a fast gain-based ray segment selection method

with auxiliary rays for the fracturing problem. Our new approach substantially reduces sliver count with negligible runtime overhead.

From an IC design automation perspective, our work offers the possibility of directly considering yield loss mechanisms such as MEEF into existing layout and RET insertion flows. This would lead, for example, to fracturing-aware "smart OPC". More generally, our results reveal significant headroom in existing tool solution quality; we believe that this can be exploited by future design-to-mask tools to reduce manufacturing variability and cost of IC designs.

The material presented in this chapter is based on the following publication.

- A. B. Kahng, X. Xu and A. Zelikovsky, "Fast Yield-Driven Fracture for Variable Shaped-Beam Mask Writing", *Photomask and Next-Generation Lithography Mask Technology XI*, April 2006, accepted and to appear.

The dissertation author was the primary researcher and author. My coauthors (Prof. Andrew B. Kahng and Prof. Alex Zelikovsky) have all kindly approved the inclusion of the aforementioned publication in my thesis.

Chapter III

Enhanced Design Flow and Optimizations for Multi-Project Wafers

The aggressive scaling of VLSI feature size and the pervasive use of advanced reticle enhancement technologies lead to dramatic increases in mask costs, pushing prototype and low volume production designs to the limit of economic feasibility. Multiple project wafers (MPW), or "shuttle" runs, provide an attractive solution for such designs by providing a mechanism to share the cost of mask tooling among up to tens of designs. However, MPW reticle floorplanning and wafer dicing introduce complexities not encountered in typical, single-project wafers. Recent works on wafer dicing adopt one or more of the following assumptions to reduce the problem complexity: (i) the assumption of equal production volume requirement for all designs, (ii) the assumption that the same dicing plan is used for all wafers or for all rows/columns of reticle images on a wafer, (iii) the assumption of unrealistic wafer models such as a rectangular array of projections, and (iv) the assumption of a fixed wafer shot-map. Although using one or more of the above assumptions makes the problem solvable, the performance of the solutions is degraded.

In this chapter we propose a comprehensive MPW flow aimed at mini-

mizing the number of wafers needed to fulfill given die production volumes. Our flow includes two main steps: (1) multi-project reticle floorplanning, and (2) wafer shot-map and dicing plan definition. For each of these steps we propose improved algorithms as follows. Our reticle floorplanner uses hierarchical quadrisection combined with simulated annealing to generate "diceable" floorplans within given maximum reticle sizes. Our dicing planner allows multiple side-to-side dicing plans for different wafers as well as different reticle projection rows/columns within a wafer, and further improves dicing yield by partitioning each wafer into a small number of parts before individual die extraction. We also employ the dicing plan definition heuristic as a procedure for the wafer shot-map definition in order to fully utilize round wafer real estate by extracting the maximum number of functional dies from both fully and partially printed reticle images. Experiments on industry testcases show that our methods outperform significantly not only previous methods in the literature, but also reticle floorplans manually designed by experienced engineers.

III.A Introduction and Motivation

With the shrinking of VLSI feature size and the pervasive use of advanced reticle enhancement technologies such as Optical Proximity Correction (OPC) and Phase Shifting Masks (PSM), mask costs are predicted to reach $10 million by the end of the decade. These high mask costs push prototyping and low volume production designs to the limit of economic feasibility since the costs cannot be amortized over the volume. Multiple Project Wafers (MPW), or "shuttle" runs, provide an efficient method to reduce the cost [26]. Thus, MPW has now become a commercial service offered by both independent providers such as MOSIS and CMP and semiconductor foundries such as TSMC and IBM. An overview of related multi-layer mask technologies, which rely on sharing the reticle space between multiple layers of the *same* design, typically via blading, is given in [19].

Most previous papers on MPW reticle floorplanning rely on an "ideal-dicing" model which assumes either zero dicing loss [20] or arbitrary margins in the floorplan formulation. Chen and Lynn [21] considered the problem of finding

the minimum area slicing floorplan, with 90 degree chip rotation allowed. Xu et al. [29] studied the MPW mask floorplanning under die-alignment constraints imposed by the use of die-to-die mask inspection. All these approaches assume that all dies can be obtained, which is impractical for current side-to-side wafer dicing technology. A grid-packing formulation for MPW mask floorplanning is proposed in [17] and [18], with the assumption that arbitrary amounts of blank area can be left on a die. However, in practice, arbitrary margins cannot be tolerated due to package requirements.[1]

Side-to-side dicing based floorplanners consider the constraints imposed by the current side-to-side dicing technology. Due to the complexity of the general dicing problem, it is crucial to simplify the dicing problem and use a fast yet accurate wafer cost evaluator in the floorplanners. According to the different dicing simplification methods, the current reticle floorplanners can be divided into two categories.

- *Single wafer dicing plan (SWDP) based floorplanners* assume that all wafers share the same dicing plan. Kahng et al. [23] were the first to consider the side-to-side wafer dicing problem with SWDP assumption. They propose three optimal integer linear programming (ILP) solutions and a fast heuristics for wafer cost evaluation. The fast wafer cost evaluator is used in a sequence pair based simulated annealing floorplanner. Recently, Kahng et al. [25] proposed a grid floorplanner. The wafer cost of grid floorplans can be directly calculated with a closed-form formula. Therefore, it is actually practical to apply the branch and bound algorithm to exhaustively search the whole solution space for small testcases. However, the closed-form wafer cost calculation also depends on the impractical assumption that a wafer is a rectangular array of projections. Also the runtime of the proposed branch and bound algorithm may explode for large testcases.

- *Single row and column dicing plan (SRCDP) based floorplanners* assume that all rows and columns of reticle images within a wafer are diced using the

[1]No margins are allowed for our industry testcases from CMP.

same set of cuts. Xu et al. [30, 31] formulate the dicing problem as a minimum coloring problem. Wu et al. [27] extend the min-coloring based dicing approach by proposing three ILP formulations for optimal minimum coloring. In [28], they propose to perform chip replication and give integrated ILPs for simultaneous floorplanning and dicing which are impractical even for small testcases due to large runtime.

In this thesis we propose a comprehensive MPW flow aimed at minimizing the number of wafers needed to fulfill given die production volumes. Our flow includes two main steps: (1) multi-project reticle floorplanning, in which the reticle floorplan is designed for the given list of dies with fixed shot-map and simplified dicing cost evaluation; and (2) wafer shot-map and dicing plan definition, in which the exact dicing plan and wafer center location is determined for the floorplan generated in Step 1. In Step 2, the dicing plan generation algorithm is included in the wafer shot-map definition algorithm for accurate wafer cost calculation. Our contributions are as follows. For the first flow step, we propose an algorithm based on fixed hierarchical quadrisection structure which is suitable for fast wafer cost evaluation with simulated annealing to generate "diceable" floorplans observing given maximum reticle sizes. Our algorithm leads to an average reduction of 10-20% in the required number of wafers compared to reticle floorplans manually designed by experienced industry engineers. For the second step, we give an integer program which can be used to find in practical time the *optimal* dicing plan under the SRCDP assumption. We also give a two-level optimization algorithm that simultaneously allows multiple dicing plans for different wafers and for different reticle projection rows/columns within a wafer. We also show the advantages of partitioning each wafer into a small number of parts before individual die extraction. For a fixed reticle floorplan, the two-level optimization algorithm gives an average reduction in the required number of wafers of 42% without wafer partition, and of 47% and 63%, when partitioning into 2 and 4 parts, respectively. Finally, we propose to include wafer shot-map definition, which has not been previously considered in the context of MPW, in order to fully utilize the real estate on round

wafers by extracting the maximum number of functional dies from both fully and partially printed reticle images. This optimization is shown to yield an average reduction of 13.6% in the required number of wafers for a reticle floorplan.

The rest of the chapter is organized as follows. In Section III.B we describe the basics of MPW with side-to-side wafer dicing. Section III.C presents a novel hierarchical quadrisection method for reticle floorplanning. Section III.D describes the advantages of multiple dicing plan (MDP) and gives a novel two-level optimization algorithm. Section III.E combines the wafer shot-map definition with dicing plan definition for further wafer cost reduction. Section III.F presents an algorithm for robust reticle floorplanning with die cloning. Section III.G addresses the on-demand wafer dicing issues. Finally, in Section III.H we give experimental results comparing proposed methods on industrial testcases.

III.B Preliminaries

A wafer consists of a number of reticle projections arranged in a number of reticle image *projection rows* and *projection columns*. Each projection is a copy of the same reticle image. In the prevalent "side-to-side" wafer dicing technology, the diamond blades cannot stop at arbitrary points during cutting; consequently, all projections in the same projection row (or column) will share the same horizontal (or vertical) cutlines. In this chapter, we extend side-to-side dicing to allow preliminary partitioning of each wafer into a small number of parts (e.g., halves or quarters) as shown in Figure III.1 so that the side-to-side dicing plans for the parts can be independent from each other.

Following [23], two dies D and D' on a reticle are said to be in *vertical (resp. horizontal) dicing conflict* if no set of vertical (resp. horizontal) cuts can legally dice both D and D'. Let $\mathcal{D}$ denote the set of dies on a given reticle. The *vertical reticle conflict graph* $R_v = (\mathcal{D}, E_v)$ is the graph with vertices corresponding to the dies and edges connecting pairs of dies in vertical dicing conflict. The *horizontal reticle conflict graph* $R_h = (\mathcal{D}, E_h)$ is defined similarly. As usual, a set of vertices in a graph is called independent if they are pairwise nonadjacent. A

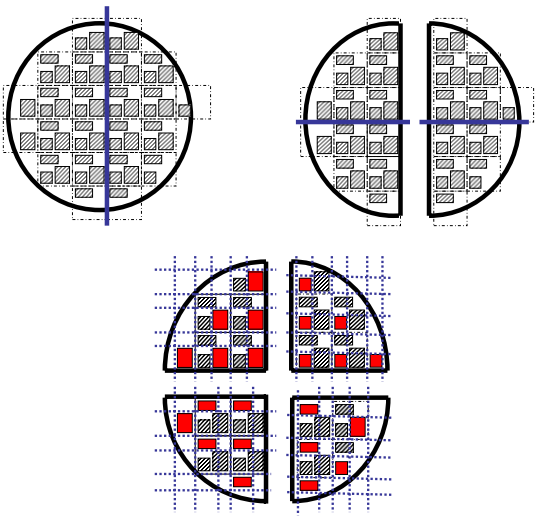


Figure III.1: Four quadrant dicing: the wafer is first divided into four quadrants, then each quadrant is diced independently using side-to-side cuts.

maximum horizontal (or vertical) independent set is a subset of $\mathcal{D}$ which can be sliced out by a set of horizontal (or vertical) cutlines; the set of cutlines used for a wafer are called a *wafer dicing plan*. The *dicing yield* of a die D is defined as the number of legally diced copies of D divided by its volume requirement. The *wafer dicing yield* is defined as the minimum dicing yield over all dies $D \in \mathcal{D}$, which needs to be at least 1.

III.C Reticle Floorplanning

In this section, we focus on the following MPW reticle floorplanning problem. Given a maximum reticle size, and the size and required volume for each die, find a reticle floorplan (allowing die rotations) and a wafer dicing plan minimizing the number of used wafers.

Compared with other floorplanning problems, the main difficulty of the MPW reticle floorplanning problem lies in the wafer cost calculation. To simplify and speed up the estimation of wafer cost and dicing plan yield, we use hierarchical quadrisection-based floorplanning. The reticle floorplan is based on a hierarchical

quadrisection mesh which is constructed in the following recursive way.

- At Level 1, the reticle area is divided into 4 regions with one horizontal line and one vertical line: $R(1,1), R(1,2), R(1,3)$ and $R(1,4)$, where $R(i,j)$ is the j^{th} Region for Level i.

- At Level $i+1$, each region at Level i, $R(i,j)$, is divided into 4 regions with one horizontal line and one vertical line: $R(i+1, 4^{j-1}+1), R(i+1, 4^{j-1}+2), R(i+1, 4^{j-1}+3)$ and $R(i+1, 4^{j-1}+4)$.

Finally, there are 4^l regions at Level l. Figure III.2(a) shows a mesh of Level 2. The constructed mesh is "soft" since the dimensions of regions are determined by the dies within the regions. The number of level l is chosen such that 4^l is greater than the number of dies.[2] Then we place the dies in the regions of Level l mesh such that each region $R(l,j)$ $(j = 1...4^l)$ contains at most one die. Different die placements lead to different reticle floorplans. Figure III.2 (b) and (c) show two different reticle floorplans for a set of 10 dies based on the same mesh in Figure III.2(a). A simulated-annealing based algorithm is used to find the best die placement.

We denote the width of the region $R(i,j)$ as $W(R(i,j))$ and the height as $H(R(i,j))$. The hierarchical quadrisection allows computing height and width in a bottom-up manner.

- At Level l, if there is a die in the region $R(l,j)$, $W(R(l,j))$ is equal to the width of the die and $H(R(l,j))$ is equal to the height of die; otherwise, $W(R(l,j)) = H(R(l,j)) = 0$.

- At Level i,$W(R(i,j)) = Max(W(R(i+1, 4^{j-1}+1)), W(R(i+1, 4^{j-1}+4))) + Max(W(R(i+1, 4^{j-1}+2)), W(R(i+1, 4^{j-1}+3)))$.

[2]It is sufficient to choose $l = 3$ in practice since the case of putting more than 64 dies in one reticle is very rare. However, we may choose $l = \lceil log_4 Number\ of\ dies \rceil$ if the number of dies is larger than 64.

$H(R(i,j)) = Max(H(R(i+1, 4^{j-1}+1)), H(R(i+1, 4^{j-1}+2))) + Max(H(R(i+1, 4^{j-1}+3)), H(R(i+1, 4^{j-1}+4)))$.

There are two main advantages of the proposed floorplan structure. First, the structure is suitable for conflict elimination since there are no conflicts between dies located in diagonal regions. Second, the wafer cost can be easily evaluated with the following lemma.

Lemma III.1 *All dies can be divided into at most* 2^l *conflict-independent sets of dies for the floorplan in a Level* l *mesh such that any two dies in the same set are not in conflict.*

Proof: The lemma is true for $l = 1$ since the dies in $R(1,1)$ and $R(1,3)$ are not in conflict and the dies in $R(1,2)$ and $R(1,4)$ are not in conflict.

Suppose the lemma is true for $l = i$, for $l = i + 1$: the reticle is first divided into four regions $R(1,1), R(1,2), R(1,3)$ and $R(1,4)$ and each region is further divided into a Level i mesh. Since the lemma is true for $l = i$, there are at most 2^i conflict-independent sets for each of the four regions. We denote the k^{th} conflict-independent set of the region $R(1,j)$ $(j = 1..4)$ as $S(1,j,k)$. Since any die in $R(1,1)$ is not in conflict with any die in $R(1,3)$, we can have the 2^i combined conflict-independent sets $S(1,1,k) \bigcup S(1,3,k)$ $(k = 1..2^i)$. Similarly, we can have another 2^i combined conflict-independent sets $S(1,2,k) \bigcup S(1,4,k)$ $(k = 1..2^i)$. Therefore, there are at most 2^{i+1} conflict-independent sets. □

It is obvious that all copies of the dies in the same conflict-independent set can be simultaneously sliced out since they are not in conflict. If we assume that only the dies of one conflict-independent set are obtained for each wafer, the *wafer requirement* for a conflict-independent set S is $MAX_{D \in S}(\lceil \frac{N(D)}{Q(D)} \rceil)$, where $N(D)$ is the volume requirement of the die D and $Q(D)$ is the number of copies of die D per wafer.[3] The total wafer requirement is the sum of the wafer requirements of all the conflict-independent sets.

[3]In order to speedup the wafer cost evaluation in the floorplanning step, we fix the wafer center at the point $(0,0)$ and set $Q(D)$ to the number of dies D on the wafer.

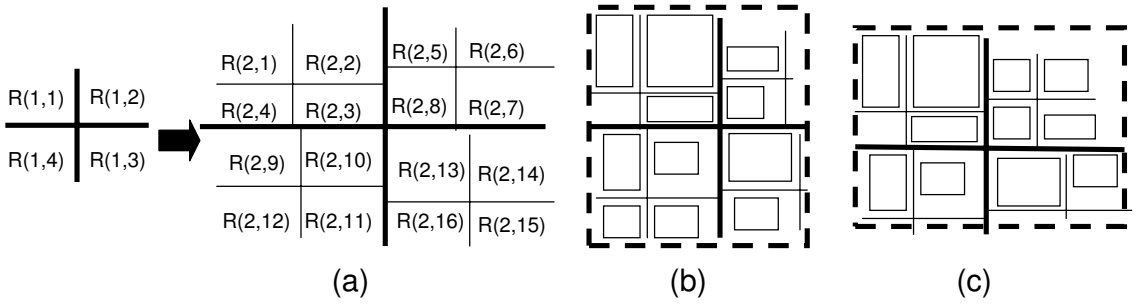


Figure III.2: Two-level hierarchical quadrisection floorplan.

We give a generic simulated annealing placement algorithm for finding reticle floorplan in Figure III.3. The algorithm starts with the floorplan with each die randomly placed in the 4^l regions as its initial placement. First, the algorithm tries to minimize the floorplan area in order to find a feasible solution. After a feasible solution is found, the objective switches to minimize the *total wafer requirement* whose calculation is specified in Lemma 1 and the paragraph following the proof of Lemma 1. Note that for speeding up the algorithm, quadrisection floorplan evaluation does not include the dicing plan and the shot-map definition. At each step we find a neighbor solution based on the following moves:

- Region exchange move, which exchanges the dies in two regions if at least one of the regions contains a die;
- Orientation move, which rotates one die by 90 degrees if the width and height of the die are different.

Each generated solution is evaluated and kept with a probability dependent on the current temperature (see Figure III.3). Finally, we inset additional copies of dies if the reticle dimension is not increased (Lines 12-15).[4][5]

[4]Whether a die D can be inserted is decided by finding a free room for D on the reticle: we place the left-bottom corner of D and its 90^o rotation at the corners of each die in the reticle and check whether D overlaps with other dies in the reticle.

[5]In practice, there is not too much empty space left in the reticle even when the number of dies is substantially smaller than the grid-number of the mesh. There are two reasons: (1) if there is no die in a region of Level l, the region area is zero; (2) if a floorplan has too much empty area, its dimension will exceed the maximum reticle dimension and this floorplan will be discarded.

Input: Dimensions of n dies, β: $0 \leq \beta < 1$

Output: Reticle floorplan and wafer dicing plan

1. Construct the hierarchical quadrisection floorplan mesh
2. Assign the n dies to regions at random
3. If (floorplan width and height < maximum reticle dimensions)
   then $FoundFeasible \leftarrow True$, else $FoundFeasible \leftarrow False$
4. While (not converged and # of moves < $Move\_Limit$)
5. Pick a move at random
6. If (width and height < maximum reticle dimensions) then
7. $FoundFeasible \leftarrow True$;
   $\delta \leftarrow$New Wafer requirement - Old Wafer Requirement
8. Else if ($FoundFeasible = False$)
   then $\delta \leftarrow$ New Area - Old Area, else $\delta \leftarrow \infty$
9. If ($\delta < 0$) then accept the move, else accept with probability $e^{-\frac{\delta}{T}}$
10. $T \leftarrow \beta T$
11. While ($\exists$ a die that can be inserted)
12. Sort all dies that can be inserted in descending order of $N(D)/A(D)$
13. For each die D_i do
14. If (D_i can be inserted) then insert it

Figure III.3: Hierarchical quadrisection floorplanning algorithm.

III.D Multiple-Dicing-Plan Dicing

The following problem has been introduced in [23].

Side-to-Side Wafer Dicing Problem (SSWDP). Given a reticle floorplan with dies $\mathcal{D} = \{D_1, \ldots, D_n\}$, required production volume for each die $N(D_i)$, $i = 1, \ldots, n$, and the positions of the reticle projections of the wafer, find the minimum number of wafers N_w and the corresponding dicing plan for each wafer such that the wafer dicing yield is at least 1.

In [23] and [25], the authors adopt the SWDP assumption, which limits the solution space. The IASA method proposed for SDP in [23] can be extended to solve the MDP by placing N_w wafers into one "super-wafer" whose row (column) number is N_w times the initial row (column) number as shown in Figure III.4. However, the runtime will increase rapidly when N_w is large since we need to check all rows and columns of the "super-wafer" in each iteration. The large runtime makes it unsuitable to be used in our proposed flow since the wafer shot-map

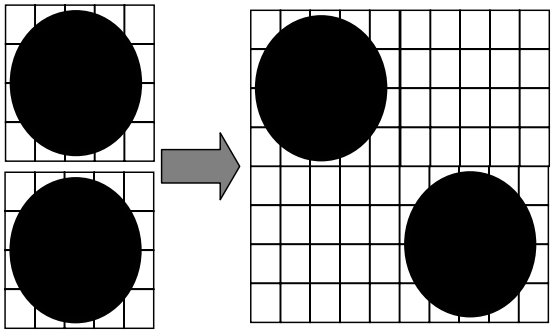


Figure III.4: Placing two wafers on one "super-wafer".

definition step requires accurate wafer cost calculation for each candidate wafer center location.

III.D.1 Integer Linear Program for Restricted MDPs

In [30], the authors assume that each wafer uses exactly one horizontal dicing plan and one vertical dicing plan for all projection rows/columns within a wafer. This assumption allows them to use a coloring based heuristic which gives good results for testcases with a large volume requirement. In this section we give an integer linear programming formulation which allows finding optimal MDPs restricted in this way.

As in [30], two dies D and D' on a reticle are said to be in *dicing conflict* if they are either in horizontal dicing conflict or vertical dicing conflict. The *conflict graph* $R_c = (\mathcal{D}, E_c)$ is the graph with vertices corresponding to the dies and edges connecting pairs of dies in dicing conflict. A *maximum conflict independent set* is a subset of $\mathcal{D}$ which can be sliced out by a set of horizontal and vertical cutlines. We use $MCIS$ to denote the set of all maximal independent sets in the conflict

graph.[6] For each independent set $C \in MCIS$, let f_C denote the number of wafers which use the dicing plan defined by C, MDP can be formulated as the following integer linear program:

Minimize N_w (ILP1)

subject to

$$\sum_{D \in C} Q(C, D) f_C \geq N(D), \qquad \forall D \in \mathcal{D}$$
$$\sum_C f_C = N_w$$
$$f_C \in \mathbb{Z}_+, \qquad \forall C \in MCIS$$

where $Q(C, D)$ is a constant which represents the number of copies of die D obtained from a wafer diced according to C. The ILP can be optimally solved in a short time since there are only $|MCIS|$ variables and $|\mathcal{D}|+1$ constraints. As shown in Section III.H, the runtimes of ILP are within 0.03 second in all the experiments on industry testcases with up to 30 dies.

III.D.2 Two-Level Optimization Algorithm for MDPs

Although the ILP method can solve the MDP problem quickly, its performance will be degraded for the small volume requirement cases. Extended IASA for MDP can produce a good solution but suffers from large runtime with large N_w. In order to rapidly find a near optimal solution for MDP, we propose the Two-level Optimization (TLO) heuristic shown in Figure III.5. We first solve ILP1 to obtain an upper bound on N_w. Then we gradually reduce the number until the yield is smaller than 1. In Lines 04-08, we assume that all rows (columns) of each wafer use the same horizontal (vertical) dicing plan. The dicing plan for each wafer is obtained by solving the following ILP:

[6] $MCIS$ can be found as follows. We denote the $MCIS$ for i dies as $MCIS(i)$.

1. Sort all dies according to max_x.
2. $MCIS(1) \leftarrow \{D_1\}$.
3. For $(i = 2; i \leq n; i++)$
4. Find the last die D_j which satisfies $max\_x(D_j) < min\_x(D_i)$
5. Add D_i to every set in $MCIS(j)$ and $MCIS(i) \leftarrow MCIS(j) \cup MCIS(i-1)$.

Minimize Y (ILP2)

subject to

$$N(D) - \sum_{D \in C} Q(C, D) f_C \leq y_D, \qquad \forall D \in \mathcal{D}$$
$$\textstyle\sum_C f_C = N_w$$
$$\textstyle\sum_D y_D = Y$$
$$f_C \in \mathbb{Z}_+, \qquad \forall C \in MCIS$$
$$y_D \in \mathbb{Z}_+, \qquad \forall D \in \mathcal{D}$$

where Y is the total number of unsatisfied volume requirement and y_D is the number of unsatisfied volume requirement for the die D. Since one maximal conflict independent set may belong to several maximal horizontal (vertical) independent sets, we use y_D as the weight of D and choose the maximal horizontal (vertical) independent set with the maximum total weight for each wafer. Then we perform the "row-and-column level" dicing plan replacement in Lines 10-13 to improve the yield.[7] A *candidate pool* is employed to speed up the process. Since the wafer yield depends on the dies with the minimum dicing yield, the dicing plans which can slice out at least one of these dies are put into the candidate pool. Only the dicing plans in the candidate pool will be tried in each iteration. The candidate pool will be updated with the change of min-yield dies. This process is greedy which requires the yield to increase with each dicing plan replacement. If a die D does not belong to any chosen horizontal or vertical dicing plan, we need to simultaneously change a horizontal and a vertical dicing plan to obtain one copy of D and increase the yield. Therefore, a *cross selection* process in Lines 14-17 is used to choose the dicing plans for one row and one column simultaneously. Since the "cross selection" process is extremely time-consuming, we do it only for the center row and column of each wafer.

[7] In the process of yield and wafer cost evaluation, we may take the dicing operation setup cost and lithography cost into consideration. Here, yield improving is equal to total manufacturing cost reduction.

Input: $MHIS$, $MVIS$, $MCIS$
Output: N_w and dicing plan for N_w wafers

```
01. Solve ILP1 to obtain the N_w upper bound
02. while (yield≥ 1 )
03.     N_w − −
04.     Solve ILP2 and choose one set C for each wafer
05.     Set the weight of each die D as y_D
06.     For (each wafer)
07.         Choose max horizontal (vertical) independent set
08.     While (improve==true)
09.         While (improve==true)
10.             For (each row and column)
11.                 try other horizontal (vertical) dicing plans
12.                 If (wafer-dicing yield increases)
13.                     Replace the current dicing plan
14.         For (the center row and column of each wafer)
15.             Simultaneously try other pairs of dicing plans
16.             If (wafer-dicing yield increases)
17.                 Replace the current dicing plan
```

Figure III.5: Two-level optimization heuristic.

III.E Wafer Shot-Map Definition

In the previous section we have fixed reticle images in order to reduce problem complexity. However, if we allow the reticle images position to freely move on the wafer, then the wafer cost can be reduced even more. The wafer shotmap definition step, which determines the position of reticle images printed on a wafer, was previously investigated for general wafers to maximize the wafer yield [22]. However, it was ignored in the previous papers in the MPW context. In both [25] and [23], the wafer is modeled as a rectangular array of projects, which is not true for actual round wafers. This simplification may lead to wrong dicing yield estimation since (i) the projection rows (columns) do not have equal contributions to the wafer dicing yield – the rows/columns near the center contain more reticle images, and (ii) fully printed dies within partial reticle projection are ignored. For a round wafer with the radius r and the center (x_0, y_0), a die image D is *on wafer* if and only if $(x - x_0)^2 + (y - y_0)^2 \leq r^2$ for all $(x, y) \in D$ (see Figure

III.6). Given a rectangular reticle image, a *shotmap* is a regular tiling of the plane with identical copies of the reticle. The corresponding problem of wafer position with respect to shot-map is formulated as follows.

Wafer Shot-Map Definition Problem (WSMDP). Given a projection plane and the wafer radius r, find the position of the wafer minimizing the number of wafers required to meet the given production volumes.

The periodic property of the projection plane implies the following lemma.

Lemma III.2 *The optimal solution of WSMDP can be achieved when the location of the wafer center is restricted to be within one reticle projection L.*

Proof: Let the reticle width and height be R_W and R_H and the optimal solution of WSMDP can be achieved when the wafer center located in $(i \times R_W + x, j \times R_H + y)$, where i, j are integers and $0 \leq x < R_W, 0 \leq y < R_H$. For any copy of a die located in the wafer centered in $(i \times R_W + x, j \times R_H + y)$, there is a corresponding copy of the same die located in the wafer centered in (x, y) and vice versa. Therefore, the optimal solution can also be achieved when the wafer center is located in (x, y). □

Therefore, the wafer center is constrained in one projection. The wafer center location is further constrained by the following lemma.

Lemma III.3 *The optimal solution of WSMDP can be achieved when at least two die corners are located on the circular boundary of the wafer and the dies having these corners are located within the wafer.*

Proof: Suppose the optimal solution of WSMDP can be achieved when the wafer center is located in (x, y). Let S be the set of the four corner coordinates of all dies on the wafer. Consequently, one die is on the wafer if and only if its four corners are in the wafer, so the solution remains optimal if all the points in S are in the wafer. Suppose no points $\in S$ are on the wafer boundary, then for any point $(x_i, y_i) \in S$, $(x_i - x)^2 + (y_i - y)^2 < r^2$. Let $t_i = x_i - x + \sqrt{r^2 - (y_i - y)^2}$. It is easy to prove that $t_i > 0$ and $(x_i - x - t_{i,1})^2 + (y_i - y)^2 = r^2$ (intuitively, this means that the wafer center is shifted to the right by t_i to make the point (x_i, y_i) on the boundary). Let t be the smallest value of all the t_i values and move the wafer center

to $(x+t, y)$. Then at least one point in S will be located on the wafer boundary. Also if any point $(x_i, y_i) \in S$ is out of the wafer, then $(x_i - x - t)^2 + (y_i - y)^2 > r^2$ and $(x_i - x - t)^2 + (y_i - y)^2 > (x_i - x)^2 + (y_i - y)^2 \Rightarrow r^2 - (y_i - y)^2 < (x_i - x - t)^2$ and $t > 2(x_i - x) \Rightarrow t_i < x_i - x + \sqrt{x_i - x - t)^2} = t$. However, this is impossible since t is the smallest value. Therefore, all points in S are still in the wafer when the wafer center is located at $(x+t, y)$.

Next, if there is only one die corner (x_1, y_1) on the boundary, we can perform coordinate transformation such that (x, y) in the original coordinates becomes $((x+t-x_1)\cos\phi + (y-y_1)\sin\phi, (y-y_1)\cos\phi - (x+t-x_1)\sin\phi)$ in the new coordinates, where $\phi = \arctan(\frac{y-y_1}{x+t-x_1})$. The points (x_1, y_1) and $(x+t, y)$ become $(0,0)$ and $(r, 0)$ in the new coordinates. Let the (x'_i, y'_i) be the new coordinates of the points $(x_i, y_i) \in S$, $(x'_i - r)^2 + y'^2_i < r^2 \Rightarrow x'^2_i + y'^2_i < 2rx_i$. Let $\theta_i = \arcsin(\frac{\sqrt{x'^2_i + y'^2_i}}{2r}) - \arcsin(\frac{x'_i}{\sqrt{x'^2_i + y'^2_i}})$. $\theta_i < 0$ and $(x'_i - r\cos\theta_i)^2 + (y'_i - r\sin\theta_i)^2 = r^2$ (intuitively, this step equals the rotation of the wafer center around the point $(0,0)$ by θ_i to make the point (x'_i, y'_i) on the boundary). Let θ be the largest value of all the θ_i values and rotate the wafer center around the point $(0,0)$ by θ to $(r\cos\theta, r\sin\theta)$. Then at least two points in S will be located on the wafer boundary: $(0,0)$ and the point (x'_i, y'_i) whose θ_i is θ. If any point $(x'_i, y'_i) \in S$ is out of the wafer, that is, $(x'_i - r\cos\theta)^2 + (y'_i - r\sin\theta)^2 > r^2 \Rightarrow \frac{\sqrt{x'^2_i + y'^2_i}}{2r} > \sin(\theta + \arcsin(\frac{x'_i}{\sqrt{x'^2_i + y'^2_i}})) \Rightarrow \theta_i > \theta$. However, it is impossible since θ is the largest value. Therefore, all points in S are still in the wafer when the wafer center is located at $(r\cos\theta, r\sin\theta)$. □

If two points (x_1, y_1) and (x_2, y_2) on the wafer boundary are known, the wafer center is located at either $(\frac{x_1+x_2-(y_1-y_2)t}{2}, \frac{y_1+y_2+(x_1-x_2)t}{2})$ or $(\frac{x_1+x_2+(y_1-y_2)t}{2}, \frac{y_1+y_2-(x_1-x_2)t}{2})$, where $t = \sqrt{\frac{4r^2}{(y_1-y_2)^2+(x_1-x_2)^2} - 1}$.

As in Figure III.7, when the wafer center is constrained in one projection L, all dies within Region 1 can be on the wafer. All dies within Region 2, which is the intersection of four cycles with radius of r whose centers located at the four corners of L, will be within the wafer no matter where within L the wafer center is. From Lemma 3, it is sufficient to consider the points in L which are determined by at least two corners of dies in Region 1 − Region 2, when the corners are on the wafer boundary. An optimal solution can be achieved by checking all these points.

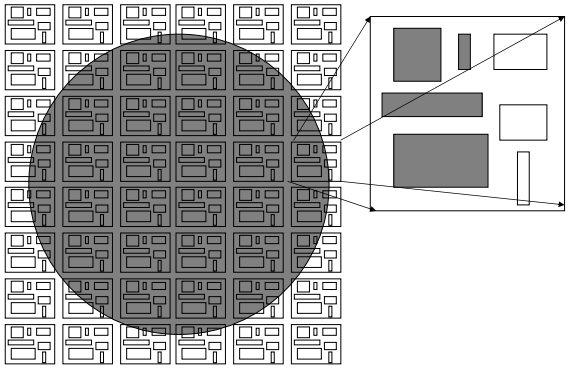


Figure III.6: A periodic shot-map with dark circular wafer. A partially printed reticle contains dark completely printed projects.

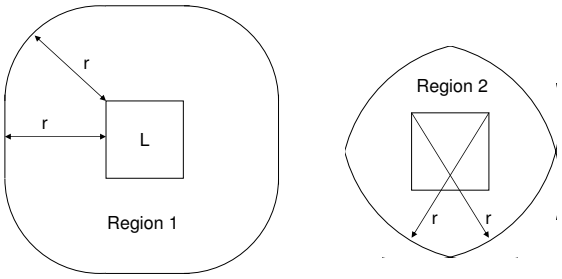


Figure III.7: Region 1 and Region 2 for the projection L.

However, obtaining the optimal solution is impractical due to the large number of points to be checked and the relatively long runtime of wafer cost calculation procedure TLO. Therefore, we propose a hierarchical wafer shot-map definition algorithm as summarized in Figure III.8, which only calls TLO dicing procedures at a few hierarchically selected locations. We first divide the projection L into several grids, then run TLO when wafer centering at each grid center. The "best" grid will be chosen for the next iteration. The following trick is employed in the algorithm to speed up the process. For each candidate wafer center location p, there is a *feasible set* of dies, $F(p)$, on the wafer when the wafer center is at p. Obviously, the wafer cost will not be reduced if $F(p)$ is a subset of one feasible set whose wafer cost is already calculated. We store all feasible sets whose wafer costs are calculated for comparison. In Line 6, if $F(p)$ is included in any stored set, p

Input: wafer radius r, reticle dimensions, one projection L
Output: wafer center location minimizing manufacturing cost

```
1.  L_0 ← L
2.  For (level =0; level < l; level++ )
3.      Divide L_0 into k × k uniformly-spaced grid
4.      For (all the grids)
5.          choose the grid center p as the wafer center
6.          If (F(p) not included in any stored feasible set)
7.                  calculate N_w with TLO, store F(p)
8.      Find the grid g with the minimum N_w
9.      Min_N_w ← N_w; L_0 ← g;
```

Figure III.8: Hierarchical wafer shot-map definition algorithm.

will be skipped to avoid redundant wafer cost calculation.

III.F Robust Floorplanning With Die Cloning

Reticle floorplanning is perhaps the most important optimization step of any MPW flow. Compared with traditional chip floorplanning, the difficulty of MPW reticle floorplanning lies in the complex relationship between the reticle floorplan and overall manufacturing cost, via conflicting factors such as reticle area and dicing compatibility [28]. Manufacturing cost estimation becomes even more difficult when production volumes are uncertain. Two reticle floorplans that require the same number of wafers for satisfying a certain set of production volumes may require significantly different numbers of wafers even for slight changes in the production volumes. Ideally, we would like a reticle floorplan that leads to low production costs under most possible production requirements. This *robustness* objective for reticle design under production demand uncertainty can be formulated as follows.

Robust Reticle Design Problem (RRDP).
Given: Maximum reticle size, dies $\mathcal{D} = \{D_1, \ldots, D_n\}$, and probability distribution of customer orders for each die.
Find: A reticle floorplan for $\mathcal{D}$ that maximizes the *expected* number of wafers

required to satisfy customer orders over the time horizon for which production is being planned.

Our RRDP algorithm uses the simulated annealing (SA) framework introduced in Section III.C. A key issue in the algorithm is how to evaluate the objective value, i.e., the average number of required wafers, after each move. In our algorithm we employ a Monte-Carlo simulation: 100 random production requirement vectors are generated according to the given probability distributions, and then the average number of required wafers is estimated over this fixed set of vectors.

Most previous works on MPW reticle design assume that the reticle contains a single copy of each die. This is appropriate for prototype manufacturing, when only a small number of wafers is produced, since it minimizes reticle area (and hence total cost). For low- and medium-production volume the number of manufactured wafers is larger, and die cloning (i.e., using multiple copies of a die in the reticle) may be justified even when it leads to some increase in reticle area, since cloning can improve dicing yield and thus decrease the number of required wafers.

A simple die cloning method was proposed in [28], based on the insertion of additional die copies within the white space available in a floorplan that was constructed starting with a single copy of each die. However, this post-processing approach has limited potential for improvement since typically there is not much empty space left on the reticle. Here, we propose a comprehensive approach to die cloning, which involves making cloning decisions *before*, *during*, and *after* running the simulated annealing algorithm in Figure III.3.

First, we set the initial number of copies c_D for die D with average volume requirement V_D to

$$c_D = \max\{1, \beta f(V_D)\}.$$

Here, $f(V_D)$ is a monotonically increasing function of V_D. We use $f(V_D) = \sqrt{V_D}$

Input: reticle floorplan with dies $\mathcal{D} = \{D_1, \ldots, D_n\}$, wafer shot-map, customer orders Q_i, $1 \leq i \leq m$

Output: number of wafers N_i to be diced after receiving each order Q_i

```
01. For each k = 1, ..., n, in_stock(k) ← 0
02. For i = 1, ..., m do
03.     For each k = 1, ..., n
04.       If in_stock(k) ≥ Q_i(k) then
05.         in_stock(k) ← in_stock(k) − Q_i(k)
06.         Q_i(k) ← 0
07.       Else
08.         Q_i(k) ← Q_i(k) − in_stock(k)
09.         in_stock(k) ← 0
10.     If Q_i ≠ 0
11.       Run the SSWDP algorithm with production volumes given by Q_i
12.       Let N_i be the number of wafers required by the algorithm, and
13.         yield(k) be the resulting number of copies of die D_k
14.       For each k = 1, ..., n
15.         in_stock(k) ← in_stock(k) + yield(k) − Q_i(k)
16.     Else N_i ← 0
17. Return N_i, i = 1, ..., m
```

Figure III.9: Greedy ODSSWDP algorithm.

in our experiments. Parameter β is a scaling factor chosen such that

$$\sum_{D \in \mathcal{D}} c_D a_D \leq \alpha A$$

where, a_D denotes the area of die D, A denotes the maximum reticle area and $\alpha \leq 1$ is a maximum reticle utilization factor, which is set to 0.6 in our experiments. To facilitate dicing, all copies of a die are arranged in a $k \times l$ "clone array" which is always assigned to a single floorplan mesh region.

We also modify the simulated annealing algorithm in Figure III.3 by adding four new moves: addition/deletion of a row/column of copies from a clone array. Finally, after the completion of the algorithm, we try to insert additional rows or columns into the clone arrays without increasing reticle size.

III.G On-Demand Wafer Dicing

The SSWDP formulation in Section III.D is appropriate in the context of current shuttle services, which focus on serving the prototyping needs of independent design companies. In this context, there is no uncertainty on the number of prototype copies required for each project since these are specified by the customers before reticle design, and all dicing can be done in a single batch. However, for low- and medium-volume production the exact customer demand may not be known *a priori*. When a single company owns all designs on the MPW, it is advantageous to manufacture a large wafer lot in anticipation of future customer demand, and then dice the wafers only in response to incoming customer orders. This motivates the study of the following *on-demand* version of the dicing plan optimization problem.

On-Demand Side-to-Side Wafer Dicing Problem (ODSSWDP).
Given:

- A multi-project reticle floorplan with dies $\mathcal{D} = \{D_1, \ldots, D_n\}$,
- A wafer shot-map, i.e., the position of reticle images on the wafer, and
- A sequence of customer orders Q_i, $1 \leq i \leq m$, where each Q_i is an n-dimensional vector of non-negative integers

Find: number of wafers N_i to be diced after receiving each order Q_i and corresponding dicing plans
Such that:

- each customer order is satisfied before receiving the next order, i.e., for every $k \in \{1, \ldots, n\}$ and $j \in \{1, \ldots, m\}$, the number of copies of die D_k that result from dicing the first $\sum_{i=1}^{j} N_i$ wafers is at least $\sum_{i=1}^{j} Q_i(k)$ (we assume that excess die copies obtained in a dicing step are stored at no cost and can be used to satisfy customer orders in subsequent steps);

Input: reticle floorplan with dies $\mathcal{D} = \{D_1, \ldots, D_n\}$, wafer shot-map, customer orders Q_i, $1 \leq i \leq m$

Output: number of wafers N_i to be diced after receiving each order Q_i

```
01. For each k = 1,...,n, in_stock(k) ← 0, past_demand(k) ← 0
02. For i = 1,...,m do
03.     For each k = 1,...,n
04.       past_demand(k) ← past_demand(k) + Q_i(k)
05.       If in_stock(k) ≥ Q_i(k) then
06.         in_stock(k) ← in_stock(k) − Q_i(k)
07.         Q_i(k) ← 0
08.       Else
09.         Q_i(k) ← Q_i(k) − in_stock(k)
10.         in_stock(k) ← 0
11.     If Q_i ≠ 0
12.       α ← max{Q_i(k)/past_demand(k)| past_demand(k) ≠ 0}
13.       For each k = 1,...,n
14.         Q'(k) ← max{0, ⌈α · past_demand(k)⌉ − in_stock(k)}
15.       Run the SSWDP algorithm with production volumes given by Q'
16.       Let N_i be the number of wafers required by the algorithm, and
17.        yield(k) be the resulting number of copies of die D_k
18.       For each k = 1,...,n
19.         in_stock(k) ← in_stock(k) + yield(k) − Q_i(k)
20.     Else N_i ← 0
21. Return N_i, i = 1,...,m
```

Figure III.10: History-based ODSSWDP algorithm.

- dicing decisions are made on-demand, i.e., for every i, the number of wafers N_i and the associated dicing plans are chosen *without* any knowledge of Q_j for $j > i$; and
- the total wafer and dicing cost is minimized.

We remark that, although we refer to the demand vectors Q_i as "customer orders", they may represent customer orders aggregated over certain periods of time (e.g., daily, weekly, etc.). In Section III.H we will use this flexibility to gauge the benefits of batching customer orders for dicing purposes.

A simple greedy ODSSWDP algorithm is given in Figure III.9. The

algorithm keeps track of the existing die stock, which changes after each dicing step. For every incoming customer order, the algorithm tries first to use existing dies to satisfy as much as possible of the order. If the order is fulfilled using existing stock, no additional wafers are diced. Otherwise, the SSWDP algorithm in Section III.D is invoked with the remaining order balance as required production volume to determine the wafer cost. Finally, the die copies thus obtained are used to complete the order, and any leftover copies are stored for future use.

The algorithm in Figure III.9 is attractive for its simplicity, but has a number of weaknesses. The SSWDP instances solved in Line 11 of the algorithm in Figure III.9 will typically require only a few wafers. This means that a large fraction of the resulting die copies will end up being stocked for future use. These die are chosen by the SSWDP algorithm without considering the already existing stock or the demand trends that can be inferred from past customer orders. An improved ODSSWDP algorithm correcting these weaknesses, which we call "history-based", is given in Figure III.10. In addition to tracking the existing stock, the improved algorithm also tracks the past order history. When a customer order cannot be fulfilled using the existing stock, instead of calling the SSWDP algorithm with production volumes given by the remaining balance, we call it with a vector of production volumes given by the past demand scaled down as much as possible while still ensuring that we can satisfy the remaining order balance, and further adjust this by subtracting existing stock quantities (Lines 12-14).

III.H Experimental Results

We use six industry testcases from CMP [32] to evaluate the performance and scalability of the proposed algorithms, each having between 12 and 31 dies with varying sizes and production volume requirements. For the wafer shot-map and wafer dicing problem, we use the reticle floorplan of the actual industry MPW runs which were manually designed by an experienced engineer. The basic parameters of the six testcases are listed in Table III.1.

Table III.1: CMP testcase parameters.

Cases	# dies	Total volume	Max Vol.	Min Vol.	Die area(cm^2)	$\|MCIS\|$
Ind 1	12	330	40	25	1.13	19
Ind 2	14	275	25	6	1.36	19
Ind 3	24	775	67	25	1.82	56
Ind 4	31	755	30	8	1.62	242
Ind 5	14	250	25	12	0.86	18
Ind 6	24	625	35	25	2.26	127

Reticle Floorplanning. We have implemented our hierarchical quadrisection floorplan algorithm in C++. The maximum reticle dimension is set as 2cm. After the placement, we use a fixed wafer shot-map and TLO dicing method to generate the dicing plans for all the wafers. The reticle floorplan results are summarized in Table III.2. Here "CMP" denotes the original industry floorplan used by CMP, "IASA+SA" is the SDP driven floorplanner used in [23] and HQ is our proposed hierarchical quadrisection floorplan algorithm. The results show that our proposed hierarchical quadrisection floorplan can save the wafer cost by 9.1%, 23.5% and 16.1% for one part, two parts and four parts compared with the original industry floorplan. On the other hand, "IASA+SA" increases the wafer cost by 18.2%, 14.7% and 17.8%, which indicates that "IASA+SA" is not a good choice for MDP on round wafers.

Wafer Dicing. We implement the wafer dicing algorithms in the C++ language. We set the wafer diameter as six inches and use a fixed wafer shot-map for all testcases. The number of wafers used (N_w) and runtime of four methods are shown in Table III.3, where IASA is the method used in [23], E-IASA is our extended IASA heuristic, ILP is the integer linear programming restricted MDP method and TLO is the proposed two-level MDP optimization method. Each method was run without any wafer partition and with wafer partition into 2 or 4 parts prior to dicing. The results show that compared with the original IASA with one part, the wafer cost can be reduced by 34.2% by using four parts. E-IASA can reduce the wafer cost by 39.5% for one part at the expense of long runtime.

ILP can reduce the cost by 5.3% for one part and can reduce the cost by 57.9% for four parts. Therefore, ILP is more efficient for multiple part dicing. TLO achieves the best solution quality in a short time. TLO reduces the wafer cost by 63.2% for four parts.

Wafer Shot-Map Definition. Our algorithm for the wafer shot-map definition problem is implemented in C++. The wafer cost and runtime results are summarized in Table III.4. l is the number of levels and k is the grid size used in each level. Compared with the fixed shotmap, the wafer cost can be reduced by 9.1% by using 10×10 grid at the expense of increased runtime. Runtime will significantly increase when N_w is reduced since there will be more dicing plan replacement iterations in the TLO procedure. Although using 100×100 grid can further reduce the wafer cost (13.6%), the runtime becomes impractical (over $100X$). However, a good tradeoff between solution quality and runtime can be achieved by using our proposed hierarchical wafer shotmap definition algorithm with $l = 3$ and $k = 10$. The wafer cost is reduced by 13.6% while the runtime is within $2.5X$ compared with using 10×10 grid.

Robust Reticle Design. We include in our comparison several floorplans:

- The industry floorplan designed by CMP engineers (CMP);
- The floorplan obtained by running the hierarchical quadrisection algorithm in [24] with production volumes set to the median point of the expected distribution, i.e., to the average between the minimum and the maximum expected customer orders (HQ);
- The floorplan obtained by running the simulated annealing algorithm in Figure III.3 driven by the uniform, respectively normal distributions for customer orders (SA-unif and SA-norm); and
- The floorplans obtained by running the simulated annealing algorithm with cloning, using an initial number of clones which is proportional to the square root of the average production volume (SA-clone). In these versions of the

algorithm we use a simpler SA objective instead of the Monte Carlo simulation used in SA-unif and SA-norm, namely the number of wafers needed to satisfy average production volumes.

Table III.5 gives the observed average and standard deviation of the number of wafers required to fulfill 100 random production requirement vectors generated for each die according to uniform, respectively normal probability distributions. The observed average is an unbiased statistical estimator for the expected number of wafers. We assume that the dicing is done using the integer linear programming SSWDP algorithm in Section III.D.1.

For comparison, we include in the table the observed average and standard deviation for the number of wafers obtained by independently running the hierarchical quadrisection algorithm of Section III.C for each of the 100 random production volume requirements (HQ*). HQ* can be used to estimate the reticle floorplanning suboptimality incurred due to demand uncertainty, since in its computation we allow selecting an individual floorplan for every production volume vector. (Note that HQ* is not a true lower-bound since HQ does not guarantee optimality.) The results show that the two cloning based SA algorithms give the best results, often better even than HQ*, despite the unfair advantage of the latter. The cloning algorithm which starts with a number of clones proportional to the square root of average production volumes gives best results, improving over CMP floorplans by an average of 33% for production volumes generated from uniform distributions, and by an average of 28% for production volumes generated from normal distributions. Most of the improvement is due to the use of cloning, as can be seen from the fact that the SA algorithms without cloning give significantly worse results (although still better than CMP when driven by the correct distribution).

In Figure III.11 we explore another measure of floorplan robustness. Here, we plot the tradeoff curve between the number of wafers and the probability of satisfying customer orders generated from the underlying distributions. To efficiently determine these tradeoffs, we use the following Monte-Carlo simulation:

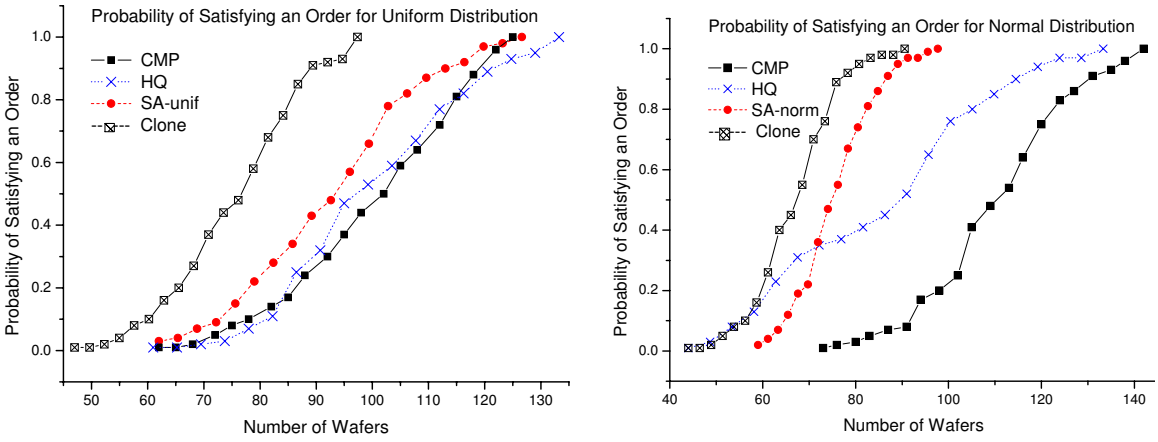


Figure III.11: Tradeoff curves between the probability of satisfying an order and the number of wafers for CMP testcase "Ind5" with production volumes generated from the (a) uniform and (b) normal distributions.

- First, we generate a large number of random production volume vectors according to the given distributions.
- Then, for each production volume vector q we compute the minimum number of wafers $N(q)$ required to satisfy it using the integer linear programming SSWDP algorithm in Section III.D.1.
- Finally, for every number of wafers N, we estimate the probability of satisfying customer order by $\{q|N(q) \leq N\}/N$.

The curves show that cloning based floorplans yield the highest success probability *over the entire range of number of wafers*. Besides showing the intrinsic qualities of a selected multi-project floorplan, estimates of success probability in Figure III.11 can be useful, e.g., in determining how many wafers to manufacture in order to maximize expected profit.

On-demand Wafer Dicing. We have implemented in the C++ language the greedy and history-based algorithms for on-demand side-to-side wafer dicing as described in Section III.G. In the basic OSSWDP formulation we assume that side-to-side cutting is done on whole wafers. When production volumes are

known, significant dicing yield improvement can be achieved by first partitioning each wafer into four equal quadrants, then dicing each quadrant independently using side-to-side cuts. To explore the advantages of quadrant-based methods in an on-line dicing environment we implement quadrant-based versions of the two algorithms.

To compare the quality of the two on-demand dicing algorithms, we generate for each die 100 individual customer orders with quantities coming from either a uniform or a normal distribution, and then randomly permute their arrival order. Tables III.6 and III.7 give (in the columns for batch size of 1) the number of wafers required by the two algorithms for each of the 6 industry testcases when run on individual customer orders. To gauge the benefits of batching customer orders we also include in the two tables results obtained by running the algorithms on "batched orders" that combine groups of 10 or 100 consecutive customer orders. Finally, to estimate the overhead in the number of wafers due to demand uncertainty during dicing, we run the two algorithms on a single batch combining all orders (both algorithms reduce to running the integer programming SSWDP algorithm on the production volume totals in this case, and therefore both give the same result).

The results show that, compared to the simple greedy algorithm, the history-based algorithm reduces wafer overhead by an average of 19.2% (respectively 8.9%) for the uniform (normal) distributions. This consistent improvement suggests that the proposed history tracking scheme is effective in "learning" the demand distribution. As expected, regardless of the algorithm used, batching leads to significant reduction in wafer overhead, i.e., the required number of wafers required by on-demand dicing gets closer to the number of wafers required when knowing all orders in advance. Somehow surprisingly, the results show that the more complex quadrant-based dicing does not help on-line dicing unless using large batch sizes.

III.I Summary

In this chapter we propose improved algorithms for multi-project reticle floorplanning, wafer shot-map definition, and wafer dicing. Experiments on industry testcases show that our methods significantly outperform previous methods in the literature as well as floorplans manually designed by experienced engineers. Our methods can also be extended to handle additional constraints such as die-alignment constraints imposed by the use of die-to-die mask inspection [29] by merging two copies of a die into a single "super-die". We also explore the use of multiple project wafers for production under demand uncertainty. We propose novel algorithms and methodologies for robust multi-project reticle floorplanning and on-demand wafer dicing, and have shown that our algorithms come close in solution quality to algorithms relying on a priori knowledge of production volumes.

The material presented in this chapter is based on the following publication.

- A. B. Kahng, I. I. Mandoiu, X. Xu and A. Zelikovsky, "Enhanced Design Flow and Optimizations for Multi-Project Wafers", *IEEE Transactions on CAD*, (2006) accepted and to appear.

The dissertation author was the primary researcher and author. My coauthors (Prof. Andrew B. Kahng, Prof. Ion Mandoiu and Prof. Alex Zelikovsky) have all kindly approved the inclusion of the aforementioned publication in my thesis.

Table III.2: Reticle floorplan results for six industry testcases. CMP is the original industry floorplan used in CMP, "IASA+SA" is the floorplanner used in [23] and HQ is our proposed hierarchical quadrisection floorplan algorithm.

Cases	# part	CMP		IASA+SA			HQ		
		N_w	area	N_w	area	CPU(s)	N_w	area	CPU(s)
Ind 1	1	3	1.13	3	1.58	24.2	3	1.42	0.00
Ind 2	1	3	1.36	3	1.83	39.2	2	1.65	0.00
Ind 3	1	4	1.82	7	1.96	1031	4	2.26	0.01
Ind 4	1	4	1.62	5	2.72	2351	4	1.82	0.01
Ind 5	1	2	0.86	2	1.77	51.7	2	1.19	0.00
Ind 6	1	6	2.26	6	3.60	795	5	2.66	0.01
Total		22		26			20		
Red.(%)				-18.2			9.1		
Ind 1	2	2	1.13	2.5	1.58	24.2	1.5	1.42	0.00
Ind 2	2	2	1.36	2	1.83	39.2	1.5	1.65	0.00
Ind 3	2	3	1.82	4	1.96	1031	3	2.26	0.01
Ind 4	2	3.5	1.62	3.5	2.72	2351	2.5	1.82	0.01
Ind 5	2	1.5	0.86	1.5	1.77	51.7	1.5	1.19	0.00
Ind 6	2	5	2.26	6	3.60	795	3	2.66	0.01
Total		17		19.5			13		
Red.(%)				-14.7			23.5		
Ind 1	4	1.5	1.13	1.75	1.58	24.2	1.25	1.42	0.00
Ind 2	4	1.5	1.36	1.75	1.83	39.2	1.5	1.65	0.00
Ind 3	4	2.75	1.82	4	1.96	1031	2.75	2.26	0.01
Ind 4	4	2.75	1.62	3.25	2.72	2351	2.25	1.82	0.01
Ind 5	4	1	0.86	1.25	1.77	51.7	1	1.19	0.00
Ind 6	4	4.5	2.26	4.5	3.60	795	3	2.66	0.01
Total		14		16.5			11.75		
Red.(%)				-17.8			16.1		

Table III.3: Wafer dicing results for six testcases. IASA is the algorithm proposed in [23]; E-IASA is our extended IASA heuristic; ILP is the proposed integer linear programming approach; TLO refers to our two level optimization algorithm.

Cases	# part	IASA		E-IASA		ILP		TLO	
		N_w	CPU	N_w	CPU	N_w	CPU	N_w	CPU
Ind 1	1	4	0.9	3	21.4	6	0.0	3	0.14
Ind 2	1	3	0.9	3	20.9	5	0.01	3	0.18
Ind 3	1	9	4.8	5	617	5	0.03	4	4.59
Ind 4	1	7	26.1	4	1631	8	0.03	4	73.6
Ind 5	1	2	1.9	2	15.5	4	0.0	2	0.21
Ind 6	1	13	13.2	6	2634	8	0.00	6	3.57
Total		38		23		36		22	
Red.(%)				39.5		5.3		42.1	
Ind 1	2	3	2.6	2.5	37.0	3	0.0	2	0.05
Ind 2	2	3	2.3	2	18.8	2.5	0.0	2	0.06
Ind 3	2	7	16.8	4.5	1485	3.5	0.01	3	3.98
Ind 4	2	5	76.9	3.5	3041	4	0.02	3.5	0.76
Ind 5	2	2	5.7	1.5	17.7	2	0.0	1.5	0.21
Ind 6	2	9	37.4	5	4457	5	0.02	5	0.04
Total		29		18.5		20		17	
Red.(%)		23.7		51.3		47.4		55.3	
Ind 1	4	2	6.5	1.75	31.4	1.75	0.01	1.5	0.02
Ind 2	4	2	6.3	1.75	29.9	2.25	0.0	1.5	0.02
Ind 3	4	7	44.8	3.75	2246	3	0.01	2.75	0.17
Ind 4	4	4	225	3	6176	3.25	0.03	2.75	0.72
Ind 5	4	1	13.6	1	17.9	1	0.0	1	0.01
Ind 6	4	9	91.6	4.75	10606	4.75	0.02	4.5	0.82
Total		25		16		16		14	
Red.(%)		34.2		57.9		57.9		63.2	

Table III.4: Cost efficiency of wafer shot-map definition step for six industry test-cases. l is the number of levels and k is the grid size used in each level.

Cases	# part	Fixed		$l = 1$,$k = 10$		$l = 1$,$k = 100$		$l = 3$,$k = 10$	
		N_w	CPU	N_w	CPU	N_w	CPU	N_w	CPU
Ind 1	1	3	0.14	3	13.7	2	2496	2	31.7
Ind 2	1	3	0.18	2	32.9	2	2790	2	68.5
Ind 3	1	4	4.59	4	442.7	4	39763	4	1069
Ind 4	1	4	73.6	4	7455	4	635342	4	18674
Ind 5	1	2	0.21	2	20.9	2	1762	2	48.3
Ind 6	1	6	3.57	5	1024	5	96521	5	2581
Total		22		20		19		19	
Red.(%)				9.1		13.6		13.6	
Ind 1	2	2	0.05	2	4.87	2	372	2	9.73
Ind 2	2	2	0.06	2	5.73	2	461	1.5	46.8
Ind 3	2	3	3.98	3	376	3	28909	3	937
Ind 4	2	3.5	0.76	3	1937	3	93402	3	4852
Ind 5	2	1.5	0.21	1.5	17.6	1	3594	1	79.8
Ind 6	2	5	3.57	4	479	4	38721	4	971
Total		17		15.5		15		14.5	
Red.(%)				8.8		11.8		14.7	
Ind 1	4	1.5	0.02	1.5	1.76	1.25	634	1.25	16.8
Ind 2	4	1.5	0.02	1.25	3.51	1.25	337	1	39.1
Ind 3	4	2.75	0.17	2.75	19.4	2.5	45827	2.5	673
Ind 4	4	2.75	0.72	2.5	173	2.5	14523	2.5	483
Ind 5	4	1	0.01	1	0.97	0.75	1877	0.75	13.5
Ind 6	4	4.5	0.82	4	567	4	30469	4	1235
Total		14		13		12.25		12	
Red.(%)				7.1		12.5		14.3	

Table III.5: Average and standard deviation of the number of wafers assuming fixed whole wafer dicing.

Case	CMP	HQ	SA-unif	SA-norm	HQ*	SA-clone
Uniform distribution						
ind1	166.9/48	138.7/38	138.7/38.7	154.2/42.2	128.6/31.8	114.0/23.6
ind2	116.7/20	107.3/18	103.1/13.1	132.9/27.6	101.7/16.2	85.9/13.6
ind3	352.2/52	348.6/51	347.2/48.2	417.7/67.4	257.6/35.0	210.0/38.4
ind4	101.9/22	103.7/21	100.2/20.7	108.3/19.9	89.7/19.6	60.8/6.6
ind5	107.3/16	104.8/16	99.6/15.7	107.3/21.3	84.3/12.4	75.3/13.1
ind6	80.9/9.6	85.6/12	77.3/10.2	78.9/9.9	72.8/8.4	74.0/8.2
Imp	0	4.02%	6.46%	-7.93%	20.65%	33.04%
Normal distribution						
ind1	137.3/23	144.2/24	144.2/24.0	131.5/20.9	127.0/22.3	108.4/15.2
ind2	149.7/20	135.3/16	161.1/24.8	127.3/16.1	114.9/14.9	113.7/14.7
ind3	375.0/37	355.6/27	361.8/26.7	352.8/31.3	270.3/19.2	272.4/21.5
ind4	120.2/29	97.0/10	104.6/11.5	95.3/10.4	88.2/9.9	81.0/26.2
ind5	119.5/16	96.0/10	93.2/10.0	81.0/8.4	75.7/8.3	73.6/8.3
ind6	105.3/8	78.4/6	89.2/7.4	76.1/5.8	74.1/4.7	73.2/5.6
Imp	0	9.98%	5.25%	14.20%	25.50%	28.27%

Table III.6: On-demand wafer dicing results for six industry testcases with customer orders generated from a uniform distribution.

#Parts	Whole Wafer				4 Quadrants			
Batch size	1	10	100	all	1	10	100	all
Test	Greedy							
ind1	64	62	54	54	64	59	48	47
ind2	356	289	268	253	356	272	250	239
ind3	228	186	153	144	211	178	146	135
ind4	72	68	57	51	69	62	50	45
ind5	185	160	156	148	185	152	148	141
ind6	86	84	76	76	86	83	70	67
Overhead	36.5%	16.9%	5.2%	0	44.1%	19.6%	5.6%	0
Test	History-based							
ind1	59	57	55	54	58	53	50	47
ind2	301	272	282	253	294	264	267	239
ind3	182	161	157	144	177	161	150	135
ind4	63	58	53	51	61	57	52	45
ind5	167	156	163	148	159	156	162	141
ind6	80	77	79	76	81	75	73	67
Overhead	17.3%	7.6%	8.6%	0	23.1%	13.6%	11.8%	0

Table III.7: On-demand wafer dicing results for six industry testcases with customer orders generated from a normal distribution.

#Parts	Whole Wafer				4 Quadrants			
Batch size	1	10	100	all	1	10	100	all
Test	Greedy							
ind1	215	183	177	177	215	176	167	167
ind2	123	108	108	104	123	107	104	97
ind3	375	354	338	322	372	361	325	306
ind4	96	95	93	87	92	90	88	83
ind5	118	107	90	83	118	101	88	80
ind6	99	89	81	81	101	98	77	74
Overhead	20.1%	9.6%	3.9%	0	26.5%	15.6%	5.2%	0
Test	History-based							
ind1	193	177	185	177	187	174	175	167
ind2	110	105	108	104	106	102	102	97
ind3	354	344	339	322	346	343	326	306
ind4	94	95	87	87	90	91	87	83
ind5	107	103	85	83	99	96	86	80
ind6	92	86	82	81	93	86	81	74
Overhead	11.2%	6.5%	3.7%	0	14.1%	10.5%	6.2%	0

Chapter IV

Bright-Field AAPSM Conflict Detection and Correction

Alternating-Aperture Phase Shift Masking (AAPSM), a form of strong Resolution Enhancement Technology (RET), will be used to image critical features on the polysilicon layer at smaller technology nodes. This technology imposes additional constraints on the layouts beyond traditional design rules. Of particular note is the requirement that all critical features be flanked by opposite-phase shifters, while the shifters obey minimum width and spacing requirements. A layout is called phase-assignable if it satisfies this requirement. Phase conflicts have to be removed to enable the use of AAPSM for layouts that are not phase-assignable. Previous work has sought to detect a suitable set of phase conflicts to be removed, as well as correct them.

This chapter has two key contributions: (1) a new computationally efficient approach to detect a minimal set of phase conflicts, which when corrected will produce a phase-assignable layout; (2) a novel layout modification scheme for correcting these phase conflicts with small layout area increase. Unlike previous formulations of this problem, the proposed solution for the conflict detection problem does not frame it as a graph bipartization problem. Instead, a simpler and more computationally efficient reduction is proposed. This simplification greatly improves the runtime, while maintaining the same improvements in the quality of

results obtained in [34]. An average runtime speedup of 5.9x is achieved using the new flow. A new layout modification scheme suited for correcting phase conflicts in large standard-cell blocks is also proposed. Our experiments show that the percentage area increase for making standard-cell blocks phase-assignable ranges from 1.7% to 9.1%.

IV.A Introduction

As advanced wafer manufacturing technologies push the patterning processes toward lower k_1 sub-wavelength printing, reticle based resolution enhancement techniques (RET) have played a critical and enabling role. Alternating-Aperture Phase Shift Masking is a form of strong RET that uses phase modulation at the mask level to enhance the resolution limit of current lithography equipment. At advanced technology nodes, it will be widely used to image features of the polysilicon layer. Among several variants of AAPSM, *Bright-Field* AAPSM is the most viable technology for the polysilicon layer [33]. In a simple model of Bright-Field AAPSM, each critical feature, which is a shape in the design whose width is below a certain threshold value, must be flanked by two phase shifters of opposing phases that create destructive interference between them. The shifters must obey additional constraints of size and spacing in order to ensure a manufacturable mask.

A layout with shifters inserted around each critical feature is *phase-assignable* if and only if there is a phase-assignment solution which meets the following requirements:

1. Shifters on opposite sides of every critical feature are assigned opposite phases (0^o and 180^o); and

2. Shifters that are separated by less than the minimum shifter spacing should be merged and assigned the same phase. Two shifters separated by less than the minimum shifter spacing will be referred to as *overlapping shifters*.

Two shifters are in *phase conflict* if they violate the above conditions in a phase assignment solution in which each shifter is assigned a phase. Figure IV.1 illustrates an example where the above conditions are violated due to a cyclic sequence of shifters that cannot be properly mapped. The *Phase Conflict Detection Problem* seeks to find a minimum set of phase conflicts, which when corrected will result in a phase-assignable layout. The *Phase Conflict Correction Problem* corrects a given set of phase conflicts by layout/mask modification with minimum increases in area or mask complexity.

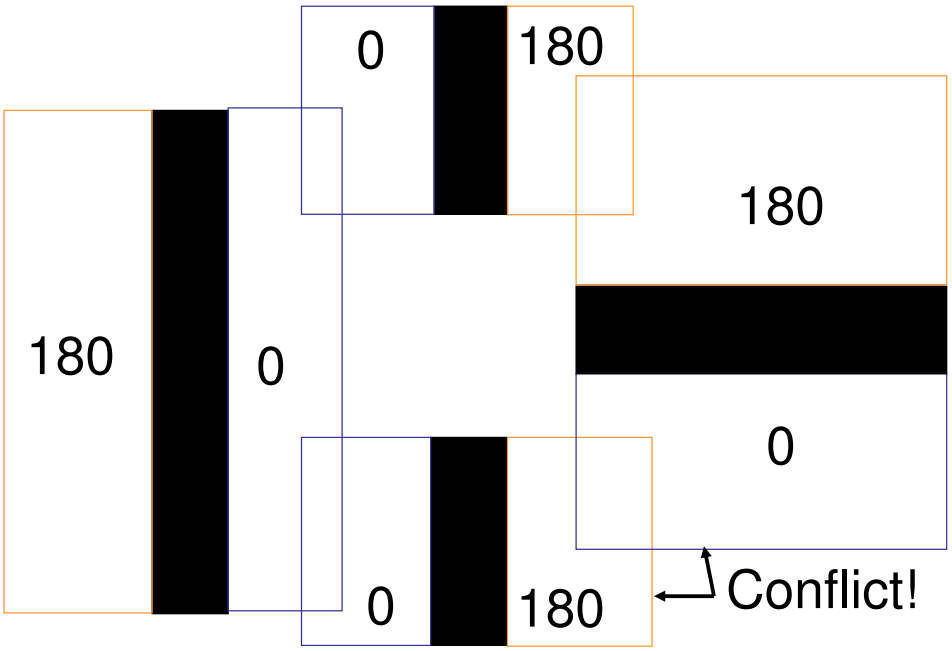


Figure IV.1: Example of incorrect phase assignment.

Our contributions are summarized as follows:

- We develop a new and computationally efficient algorithm for detecting phase conflicts, which when corrected will render the given layout phase-assignable. Unlike the bipartization formulation which is the basis of all previous work, we formulate the conflict detection problem as a conflict cycle removal problem. This leads to a substantial reduction in the number of nodes/edges in the constructed graphs and thereby produces average runtime speedups

of 5.9x, while maintaining the same quality of results as the best results available today [34].

- We provide a novel layout modification algorithm for correcting a selected set of phase conflicts, achieving small area increase and good scalability for large standard-cell blocks.[1] Compared to the approach in [34], which presents the only available results on layout modification for correcting phase conflicts for bright-field AAPSM, our new approach can reduce the maximum area increase from >100.0% to 9.1% for large designs.

This chapter is organized as follows. Section IV.B briefly reviews the previous work in phase conflict detection and correction. In Section IV.C, we provide a detailed discussion of the new theory of the proposed phase conflict detection flow. Section IV.D discusses our new conflict correction algorithm. Experimental results are presented in Section IV.E. We end with a summary in Section IV.F.

IV.B Previous Work

The phase conflict detection problem is addressed in [34] [36] [37] [38] [40]. The basic underlying principle in these works is the translation of the phase conflict detection problem to a graph bipartization problem of a suitably constructed graph. Thus, conflict detection involves identifying a set of edges such that the modified graph obtained after deleting the edges is bipartite. The work in [38][40] formulates the phase conflict detection problem as a minimum-weight graph bipartization problem to minimize the amount of layout modification necessary to render the layout phase-assignable. It is assumed that the constructed graphs will always be embedded planar graphs[2] and an optimal solution is provided for that case. The most recent work in this area is presented in [34]. This algorithm works

[1] *T-shaped phase conflicts* and other local phase conflicts can be easily detected with simple design rule checking and corrected with phase splitting [41], feature widening [47] or cell redesign. Like the approach in [34], we assume that T-shaped conflicts are already corrected by other methods. We only consider conflict correction by spacing, i.e., increasing space between features, due to the small impact of such perturbations on timing and mask complexity.

[2] An embedded planar graph is one that has no line crossings when embedded in a plane.

on general layouts and is a generalization of the scheme presented in [40]. The layout is represented as a graph called the *phase conflict graph*. We propose a new bipartization algorithm that does not require the input graph to be an embedded planar graph. The algorithm creates a planar subgraph of the given graph, applies a computationally efficient version of the optimal bipartization algorithm [38] on the planar subgraph to get an optimal solution and then combines this solution with a greedy solution for the edges deleted during planarization. The quality of results (in terms of number of conflicts selected for correction and runtime numbers) was significantly better than previous work in this area. This phase conflict detection algorithm will be used as our reference for comparison because it out-performs other existing work in the area.

Previous work in phase conflict correction falls into two major categories. Mask-level correction based approaches [41] split shifter regions whenever two shifters of opposite phases overlap to avoid layout modification. However, the mask complexity is increased and it is not always possible to split the shifter regions without negatively affecting process latitude. Layout modification based approaches remove the conflicts by increasing spacing between features or widening critical features [36] [37] [43] [42] [38] [34]. Most of these works focus on dark-field AAPSM[3]. The first layout modification scheme for correcting bright-field phase conflicts is presented in [34].[4] The key idea in this work is to add a minimal number of end-to-end spaces throughout the layout to separate all shifter pairs corresponding to the phase conflicts by the desired spacing. While this technique is suitable for standard cells and some macro blocks with a relatively small number of conflicts, experimental results show that it is highly unsuitable for standard-cell blocks in which a large number of phase conflicts must be corrected.

There are also cell-based solutions that propose to add blank space around each cell to avoid introducing phase conflicts between neighboring cells, or that introduce an additional requirement that all the boundary elements should have

[3]In dark-field AAPSM, phases are assigned to the critical features themselves. This form of phase is not likely to be used on the polysilicon layer.

[4]Although the work in [40] proposes the conflict detection methods for bright-field phase conflicts based on feature widening, it neglects the layout modification problem.

the same phase [44] [39]. No results were presented in the context of standard-cell blocks. However, each of these methods is somewhat conservative, and could lead to unnecessary increases in area since only a small fraction of the phase conflicts involve features of different cells.

IV.C Phase Conflict Detection Scheme

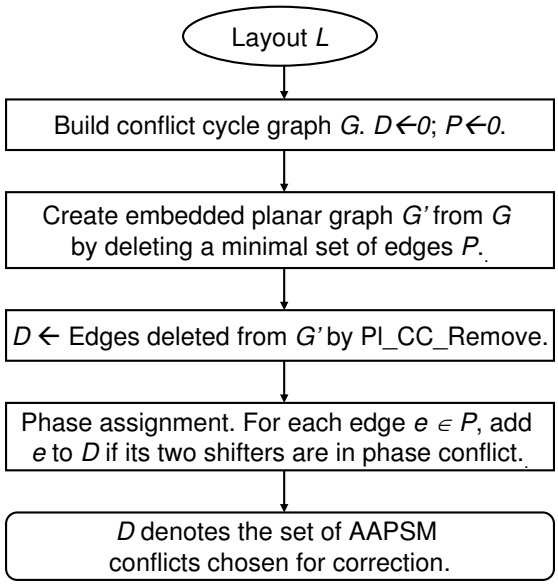


Figure IV.2: Phase conflict detection flow.

In this section, the proposed phase conflict detection scheme is presented. As shown in Figure IV.2, the proposed conflict detection flow is presented below:

1. *Conflict Cycle Graph Generation.* A *conflict cycle graph* G is constructed from a given layout L.

2. *Planar Graph Embedding.* The phase conflict graph G is not necessarily an embedded planar graph, which is required by the optimal algorithm. Hence, G is converted to an embedded planar graph G' by greedily removing minimum weight conflict edges that cross other edges. These conflict edges are added to a potential set of AAPSM conflicts P.

3. *Optimal Conflict Removal for Planar Graph.* An optimal minimum-weight conflict cycle removal algorithm *Bipartize*, described in Section III.B, is applied to G' for choosing the minimum set of AAPSM conflicts that when corrected will produce a *phase-assignable* layout. The list of edges deleted by the algorithm is added to D, which denotes a *minimal set of AAPSM conflicts* which when removed will ensure that G' is phase assignable.

4. *Computation of Final Set of AAPSM Conflicts.* It is necessary to check if any of the edges deleted during planar embedding, i.e., the conflict edges in P, lead to phase conflict. This is accomplished by 2-coloring G' after deleting the edges in D. If the two shifters of $e \in P$ are in phase conflict, e is added to the set D. At this point, D has a minimal set of edges or AAPSM conflicts which when removed will make G phase assignable.

Unlike previous formulations of the problem, the proposed solution does not reduce the problem to a bipartization problem. Instead, the phase conflict detection problem is reduced to a new problem called the minimum-weight conflict cycle removal problem (the problem will be introduced formally in the next section). This new reduction enables the construction of a much simpler graph from the layout. This graph, called the *conflict cycle graph*, removes all superfluous edges that were introduced in the phase conflict graph construction to make them bipartite for phase-assignable layouts. This simplification enables a significant reduction in the number of edges compared to the *phase conflict graph* [34] (an average reduction of 31% in the number of edges is achieved using the simpler graph).

A further advantage of this new formulation is that an optimal polynomial-time algorithm exists for the minimum-weight conflict cycle removal problem when the input graph is an embedded planar graph. The optimal algorithm is used as a subroutine in the proposed phase conflict detection algorithm. The use of the optimal algorithm ensures that the quality of results returned by our phase conflict detection algorithm is comparable to the best results returned by previous work [34], since large subgraphs of the input graph are solved using an optimal

algorithm. In addition, the new theory enables the removal of certain edges that are marked undeletable and cannot be selected by the phase conflict detection algorithm. This also results in significant speedups of the phase conflict detection algorithm. Experimental results on representative examples show average speedups of 5.9x using the proposed approach, while maintaining the same quality of results as the method in [34].

The novel and distinguishing features of the proposed phase conflict detection scheme, with respect to previous methods, can be summarized as follows:

- Representation of the layout as a conflict cycle graph and development of its relationship to phase-assignability of the layout.
- Reduction of the phase conflict detection problem to a minimum-weight conflict cycle problem (to be defined later) on the conflict cycle graph and an optimal polynomial-time algorithm for the same, when the graph is an embedded planar graph.
- Improvements to the intermediate reductions such that edges that cannot be selected by the conflict detection algorithm do not need to be explicitly represented.

The following sections give a detailed discussion of these points.

IV.C.1 Conflict Cycle Graph

The first step of the conflict detection algorithm is to build a *conflict cycle graph*. Given a layout L, the conflict cycle graph $G = (N, E \cup F)$ consists of shifter nodes N, conflict edges E and feature edges F.

1. For every shifter, create an edge shifter node $n \in N$.
2. For two overlapping shifters[5] s_1 and s_2, create a conflict edge $e \in E$ connecting n_1 and n_2. Here n_1 and n_2 are the edge shifter nodes for s_1 and s_2, respectively.

[5]Two shifters that are separated by less than the minimum shifter spacing are called *overlapping shifters*.

3. Create a feature edge $f \in F$ between the two shifters that are on opposite sides of a critical feature.

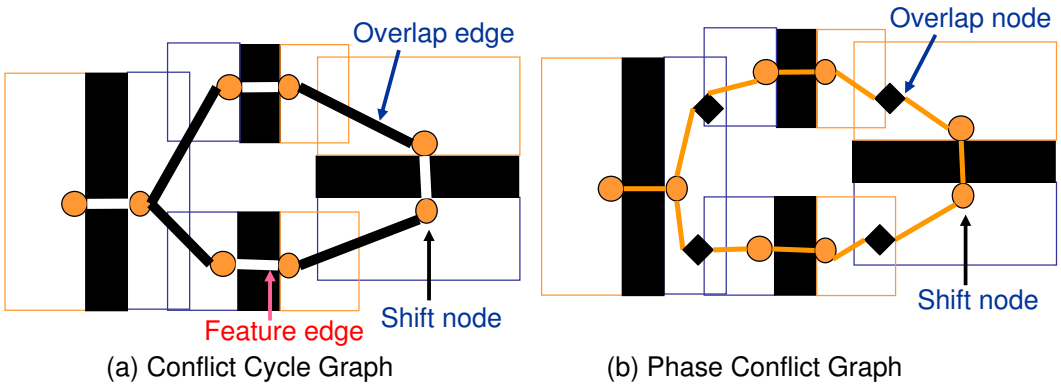


Figure IV.3: (a) Conflict cycle graph, (b) Phase conflict graph.

Figure IV.3(a) shows an example of a conflict cycle graph for the layout shown earlier in Figure IV.1. The conflict cycle graph has eight nodes and eight edges. By comparison, the phase conflict graph (shown in Figure IV.3(b)) of the same layout has 12 nodes and 12 edges. Experimental results in a later section show a substantial reduction in the node/edge count, which results in significant runtime improvements. However, unlike the phase conflict graph, the conflict cycle graph does not equate phase-assignability of its corresponding layout to bipartition. So, a conflict cycle graph may be bipartite even if its corresponding layout is not phase-assignable. Thus, a new criterion is needed for detecting phase conflicts using the conflict cycle graph.

It should be further clarified that in the conflict cycle graph, the feature edges and conflict edges play different roles: nodes connected by feature edges should be assigned different phases and nodes connected by conflict edges should be assigned the same phase. Later in this section, we discuss how in certain applications the feature edges are only used to appropriately classify the cycles they belong to and can be dropped during some intermediate graph constructions, thereby producing more speedups.

The general phase assignment algorithm is shown in Figure IV.4. A layout is *phase assignable* if and only if the boolean variable "failed" remains false at the end of the assignment process.

Input: Conflict cycle graph G
Output: Phase assignment of G

1. failed←False; all nodes are uncolored
2. **While** (failed==False AND ∃ uncolored nodes)
3. Pick a random uncolored node n_0 as the root and assign it color 0, $S_0 \leftarrow \{n_0\}$
4. Put all the nodes connected with at least one node in S_0 in a set S_1.
5. **For** (all colored nodes $n_1 \in S_1$)
6. Check nodes $n_2 \in S_0$ connected to n_1 with edge e for the two rules: (1) if e is a conflict edge, the color of n_1 should be the same as n_2; (2) if e is a feature edge, the color of n_1 should be different from n_2. If the rules are violated, failed←True
7. **If** (all nodes in S_1 are colored) return to Step 2
   Else
8. **For** (any uncolored nodes $n_1 \in S_1$)
9. Arbitrarily choose one node $n_2 \in S_0$ connected to n_1 with edge e: (1) if e is a conflict edge, the color of n_1 is the same as n_2; (2) if e is a feature edge, the color of n_1 is different from n_2.
10. $S_0 \leftarrow S_1$, return to Step 4
11. **If** (failed==True) G is not phase assignable

Figure IV.4: Phase assignment algorithm.

In the rest of this section, we present the theory for phase conflict detection using the conflict cycle graph.

Definition IV.1 *A* conflict cycle *is a cycle which contains an odd number of* feature edges.

Theorem IV.1 *A layout is phase assignable if and only if the corresponding conflict cycle graph has no conflict cycles.*

Proof: ($\rightarrow$) Assume L is phase-assignable. Let all the edge shifter nodes be colored with the same phases as the shifters in L. It is true that the node colorings of G satisfy the following two conditions: nodes connected by a feature edge have different colors and nodes connected by a conflict edge have the same color. Let us assume further that there exists a conflict cycle C and let $\{n_1, n_2, \ldots, n_k, n_1\}$ be a closed walk along C. By the definition of a conflict cycle, there are an odd number of feature edges in C. Hence, starting from n_1, the node phases will flip an odd number of times in C. Therefore, the node n_1 will be assigned two different phases, which is impossible. Hence our assumption that G, whose corresponding layout L is phase-assignable, has a conflict cycle is wrong.

($\leftarrow$) Assume G does not contain any conflict cycles and L is not phase-assignable. Then the phase assignment process specified in Figure IV.4 must violate the rules for two nodes n_1 and n_2 connected with the edge e. There must be two paths from the root to n_1 and n_2. Let n_0 to be the last common node on the two paths. Then there is a cycle $C = \{n_0, ..., n_1, n_2, ..., n_0\}$. There are two possible cases.

1. n_1 and n_2 have the same color and e is a feature edge: Since the colors are only changed across feature edges, if n_1 has the same color as n_0, then there must be an even number of feature edges from n_0 to n_1. By assumption, n_2 has the same color as n_1, and hence as n_0. Thus, there must be an even number of feature edges from n_0 to n_2. Then, C must contain an odd number of feature edges and hence C is a conflict cycle.

2. n_1 and n_2 have different colors and e is a conflict edge: If n_1 and n_0 have the same color, then there must be an even number of feature edges on the path from n_1 to n_0. By assumption, n_2 has a different color from n_1 and hence a different color from n_0. Thus, the path from n_0 to n_2 must have an odd number of feature edges. Hence C must contain an odd number of feature edges and is a conflict cycle.

This contradicts our initial assumption that G has no conflict cycles. Hence, our assumption that L is not phase-assignable is wrong. □

In order to make the layout phase assignable, it is necessary to remove all conflict cycles from the conflict cycle graph by deleting edges. The deleted edges directly correspond to phase conflicts that have to be corrected. Each edge has a given weight which reflects the negative effects of correcting the phase conflict.[6] A large number of phase conflicts selected for correction would imply large changes to the layout and/or mask, which is highly undesirable. Hence, it is essential to minimize the sum of weights of the edges to be deleted during conflict cycle removal.

The minimum-weight conflict cycle removal problem is defined as follows: Given a conflict cycle graph $G = (V, E)$, remove a minimum-weight set of edges E' such that the modified graph $G' = (V, E \setminus E')$ does not have any conflict cycles.

It can be easily proved that this problem is NP-hard for general graphs by doing a simple reduction to the minimum-weight bipartization problem. However, an optimal polynomial-time algorithm exists when the graph is an embedded planar graph. This optimal algorithm is referred to as `Pl_CC_Remove` in Figure IV.2 and will be discussed in detail in the next section.

IV.C.2 Optimal Minimum-Weight Conflict Cycle Removal Algorithm for Embedded Planar Graphs

The theory of the optimal polynomial-time algorithm for minimum-weight conflict cycle removal for embedded planar graphs is presented in this section. Let G denote an embedded planar graph for which we seek the optimal solution of the minimum-weight conflict cycle removal problem.

Definition IV.2 *A* conflict face *of G is a face corresponding to a conflict cycle in G. A face of G that is not a conflict face is a* legal face.

[6] The weighting scheme depends on the layout modification methods, which will be discussed in Section IV.D.

Definition IV.3 *The* dual graph G^D *of the conflict graph* G *is constructed by representing every face* f *of* G *with a node* n*. An edge* e *which belongs to faces* g_1 *and* g_2 *in* G *is represented with an edge* $e' = \{n_1, n_2\}$ *in* G^D*. A node* $n \in G^D$ *corresponding to a conflict face* $f \in G$ *is called a* conflict node*. A node that is not a conflict node is a* legal node*.*

Definition IV.4 *Two faces are* neighboring faces *if they share at least one common edge. The* merged face *of two neighboring faces is formed by deleting all common edges.*

Lemma IV.1 *The parity of the number of feature edges of the merged face is equal to the parity of the sum of the numbers of feature edges of two faces.*

Proof: Let the two faces have m_1 and m_2 feature edges and they share m_3 feature edges. Then the merged face has $m_1 + m_2 - 2m_3$ feature edges, which has the same parity as $m_1 + m_2$. □

Lemma IV.2 *A planar embedded graph* G *has no conflict cycles if and only if all faces are legal.*

Proof: For a planar embedded graph, any cycle is the result of merging n faces. If all faces are legal, we know that the number of feature edges in the merged face is even from Lemma IV.1. Therefore, by definition, the graph has no conflict cycles. If the original graph has no conflict cycles, then every face is legal by definition. □

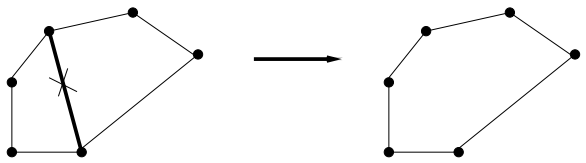


Figure IV.5: Deleting all common edges (in this case, only one) results in a merged face.

Theorem IV.2 *Removing an odd number of edges from every conflict face and an even number of edges from every legal face will generate a graph with no conflict cycles.*

Proof: Let G' be the graph obtained after the edge deletion. As shown in Figure IV.5, the deletion of one or more common edges results in the creation of a merged face. Any face in G' must be the result of merging a set S_1 of conflict faces and a set S_2 of legal faces in G. Let $S = S_1 \bigcup S_2$.

We first want to prove that the cardinality of S_1, $|S1|$, is even. Let $r(f)$ denote the number of removed edges for each face $f \in S$. Since all removed edges belong to two faces in S and are counted twice, $\sum_{f \in S} r(f)$ is even. $\sum_{f \in S} r(f) = \sum_{f \in S_1} r(f) + \sum_{f \in S_2} r(f)$. From the assumption, an even number of edges are removed from every legal face, i.e., $r(f)$ is even for $f \in S_2$. Therefore, $\sum_{f \in S_2} r(f)$ is even and hence $\sum_{f \in S_1} r(f)$ is even. Since $r(f)$ is odd for every $f \in S_1$, $|S1|$ must be even.

Then we want to prove that *the sum of the feature-edge numbers of all faces in S is even*. Since every face in S_1 has an odd number of feature edges and there are an even number of faces in S_1, the sum of the numbers of feature edges of all faces in S_1 is even. Also the sum of the numbers of feature edges of all faces in S_2 is even, since every face in S_2 has an even number of feature edges according to the definition of legal faces. Therefore, the sum of the feature-edge numbers of all faces in S is even.

According to Lemma IV.1, the feature-edge number of the merged face is even since the sum of the feature-edge numbers of all faces in S is even. Hence any merged face in G' is legal. Thus, G' has no conflict cycles according to Lemma IV.2. □

The problem of deleting a minimum-weight set of edges such that an odd number of edges are deleted from every conflict face and an even number of edges are deleted from every legal face of G translates to the following problem on its dual graph G^D:[7]

[7]Given a planar graph G, its geometric dual G^D is constructed by placing a vertex in each face of G (including the exterior face) and, if two faces have an edge in common, joining the

Find the minimum-weight set of edges S to be deleted in $G^D = (V, E)$ such that: (a) an odd number of edges in S are incident on every conflict node $u \in V$; (b) an even number of edges in S are incident on every legal node $v \in V$.

This is similar in spirit to the *T-join* problem [45] on a graph G which can be optimally solved. The *T-join* problem of a graph seeks a minimum-weight edge set S such that a node u is incident to an odd number of edges of S if and only if u belongs to the node subset T of the given graph. Our problem reduces to the *T-join* problem if and only if the set of all conflict nodes is denoted as the set T. Unlike the problem formulation in [34] in which T is the set of all nodes with odd degrees, in our formulation, T may include nodes with odd or even degrees.

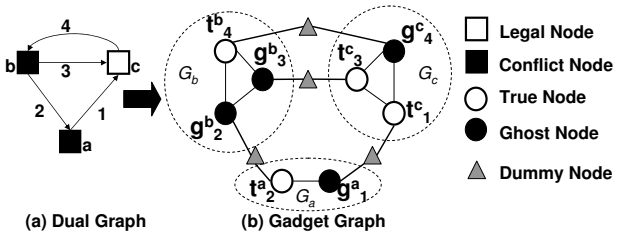


Figure IV.6: Gadget graph construction from dual graph. The directions on the edges in (a) are used to signify the edge assignment.

Next, we describe how the T-join problem can be reduced to a perfect matching problem on a suitably constructed gadget graph $\mathcal{G}$. The gadget graph construction consists of the following steps:

1. **Dual Edge Assignment.** Each edge e connecting v and v' in dual graph is assigned to v, v' or both. The assignment is done such that the following conditions are satisfied:[8]

   (a) For each conflict node v, the number of true nodes in G_v is odd.

   (b) For each legal node v, the number of true nodes in G_v is even.

corresponding vertices by a dual edge.

[8]To ensure that the conditions are satisfied, we use the edge assignment method in [38] to assign the edges such that for any gadget of n nodes, the number of ghost nodes is at least $\lfloor \frac{n}{2} \rfloor$. Then for any gadget G_v which violates the parity requirement, it is always possible to turn a ghost node g^v_e into a true node t^v_e (i.e., assign the edge e to both v and v') to meet the parity requirement at the cost of increasing the node number of the gadget graph by one.

In Figure IV.6 (a), directed edges are used to represent the assignment.

2. **Gadget Node Construction.** If a dual edge e connecting v and v' is assigned to v and not assigned to v', it will appear as a *true node* t_e^v in G_v and a *ghost node* $g_e^{v'}$ in $G_{v'}$.[9] As a result, each node v of degree k in the dual graph G^D becomes a gadget of k nodes in $\mathcal{G}$, which is denoted as G_v. The weight of $g_e^{v'}$ is $w(e)$ and the weight of t_e^v is 0. In other words, *the weight of any dual edge is always assigned to its corresponding ghost node.* Both nodes are connected to a *dummy node* with 0 weight edges. In any perfect matching solution for the gadget graph, *exactly one node in each pair of* t_e^v *and* $g_e^{v'}$ *will be matched within gadgets* since the other node will be matched with the dummy node.

3. **Complete Gadget Construction.** The nodes in G_v are connected to each other by weighted edges to form a complete graph. The weight of any edge in G_v is the total weight of its two nodes. Figure IV.6 (b) shows the gadget graph constructed from the dual graph of Figure IV.6 (a).

In summary, $\mathcal{G} = (V', E')$, where V' includes the true nodes, ghost nodes and dummy nodes, and E' is the set of edges between nodes in V'.

Theorem IV.3 *The T-join problem for a graph* $G^D = (V, E, w, T)$*, where* T *denotes the conflict nodes and* $V \setminus T$ *denotes the legal nodes in* V*, can be reduced to a minimum-weighted perfect matching on the gadget graph* $\mathcal{G} = (V', E', w')$.

Proof:($\rightarrow$) Mapping perfect matching solution of the gadget graph $\mathcal{G}$ to a valid solution of the T-join problem on G^D: For any node v in the dual graph G^D, divide the edges of node v into four sets:

- $S_1 = \{e | g_e^v$ matched within $G_v\}$.
- $S_2 = \{e | g_e^v$ not matched within $G_v\} = \{e | t_e^{v'}$ matched within $G_{v'}\}$.

[9]For example, edge 3 from node b to c means that edge 3 is assigned to node c and it appears as t_3^c in G_c and g_3^b in G_b.

- $S_3 = \{e | t_e^v$ matched within $G_v\}$.
- $S_4 = \{e | t_e^v$ not matched within $G_v\} = \{e | g_e^{v'}$ matched within $G_{v'}\}$.

We need to prove that the set $S = S_1 \bigcup S_4$ thus constructed is a valid solution to the T-join problem. Let the cardinality of S_1, S_2, S_3 and S_4 be a, b, c and d, respectively. In any perfect matching solution, the number of nodes matched within G_v, $(a+c)$, is even. If v is a conflict node, the number of true nodes in G_v, $c+d$, is odd by construction and $(a+c)+(c+d) = (a+d)+2c$ is odd. Therefore, the number of edges in S, $a+d$, is odd. Similarly, if v is a legal node, the number of edges in S is even. Therefore, the solution S is a valid solution of the T-join problem.

Since the weight of any edge e in the dual graph is always assigned to its corresponding ghost node g_e^v, the total weight of the edges in the T-join solution, $S = S_1 \bigcup S_4$, is equal to the total weight of all ghost nodes matched within gadgets.

On the other hand, the total weight of the matching solution, i.e., the total weight of the matched edges, = the total weight of the matched edges within gadgets (since the weights of edges incident to dummy nodes are all 0), = the total weight of all nodes matched within gadgets (since the edge weight is the total weight of its two nodes), and hence = the total weight of all ghost nodes matched within gadgets (since the weights of all true nodes are 0). Therefore, the total weight remains the same during the mapping.

($\leftarrow$) Mapping a solution S of the T-join problem to a solution of the perfect matching problem of $\mathcal{G}$ can be done as follows: For any node v in the dual graph G^D, divide the true nodes and ghost nodes in G_v into four sets:

- $S_1 = \{g_e^v | e \in S\}$.
- $S_2 = \{g_e^v | e \notin S\}$.
- $S_3 = \{t_e^v | e \notin S\}$.
- $S_4 = \{t_e^v | e \in S\}$.

Let the cardinality of S_1, S_2, S_3 and S_4 be a, b, c and d, respectively. We need to prove that there is a perfect matching solution in which the a ghost nodes in S_1 and the c true nodes in S_3 are matched within G_v and the remaining nodes are matched outside G_v. If v is a conflict node, since S is a valid solution of the T-join problem, the number of edges $\in S$, $a+d$, is odd. The number of true nodes $(c+d)$ is odd by construction. Thus, $((a+d)+(c+d)) = (a+c)+2d$ is even. Hence, $(a+c)$ is even. Similarly, we can prove that $(a+c)$ is even when v is a legal node in G^D. It is always possible to match an even number of nodes in a complete graph.

Since the edge weight in the dual graph is always assigned to its corresponding ghost node, we only need to consider the nodes in S_1 and S_2 (true nodes in S_3 and S_4 have corresponding ghost nodes in other gadgets). Among them, only the nodes in S_1 are matched within gadget and their weights are included in the matching solution. In other words, the ghost node weight is included in the matching solution if and only if its corresponding dual edge is in the T-join solution. Therefore, the matching solution has the same weight as the T-join solution.

□

The perfect matching problem can be optimally solved in polynomial time. In our implementation, we integrate the code of Cook and Rohe [46].

It should be noted that using the proposed conflict cycle graph and the T-join formulation implies that the classification of a face does not rely on its edge number. Therefore, further simplifications can be done on the dual graph, when it is converted to the gadget graph that is input to the perfect matching problem. For instance, feature edges are only needed to classify the faces as conflict faces or legal faces and can be dropped during the dual graph construction if they cannot be picked by the phase conflict detection algorithm (in the next section we discuss why this might be the case). This simplification results in a further reduction of the number of nodes and edges in the gadget graph without affecting the correctness of the above reductions. However, this simplification could not be done with the previous bipartite formulation in [38] [40] [34], which in turn resulted in the increased complexity of the constructed gadget graphs in those previous works.

IV.C.3 Gadget Decomposition With Divide Nodes

For a large gadget of n nodes, the number of edges is $O(n^2)$ since a gadget is a complete graph. Therefore, we propose a method to decompose a large gadget into a set of small complete gadgets with *divide nodes* to reduce the edge number.

The gadget graph construction with divide nodes consists of the following steps:

1. **Dual Edge Assignment and Gadget Node Construction.** These two steps are the same as Step 1 and 2 of the construction without divide nodes.

2. **Gadget Construction with Divide Nodes.** The nodes in each gadget G_v are divided into $2i + 1 (i \geq 0)$ subsets $G_{v,j}$ $(j = 1...2i + 1)$, which are linked with $2i$ *divide nodes* with 0 weight. All nodes in $G_{v,j}$ and $G_{v,j+1}$ are connected to *divide nodes* $d_{v,j}$, $(j = 1...2i)$. Each pair of neighboring divide nodes $\{d_{v,j}, d_{v,j+1}\}$ $(j = 1...2i - 1)$, are connected. The weight of any edge in G_v is the total weight of its two nodes. Figure V.6 shows one example of dividing a big gadget into three small subsets with divide nodes.

The edge numbers can be greatly reduced with divide nodes for large gadgets. For example, a complete gadget of n nodes has $\frac{n(n-1)}{2}$ edges. If the nodes are divided into subsets with divide nodes such that each subset has at most three nodes and at most one subset has less than three nodes, the edge number is at most $\frac{10(n+2)}{3}$. Therefore, the edge number is reduced from $O(n^2)$ to $O(n)$ while the node number is still $O(n)$. This leads to significant reduction in perfect matching runtime for large gadgets.

The constructed gadget with divide nodes has the following important property.

Lemma IV.3 *For any subset* $S_1 \subseteq \{G_{v,1} \bigcup G_{v,2}, ... \bigcup G_{v,2i}\}$ *(*$i \geq 1$*), there is a perfect matching solution to match all the nodes in* S_1 *and the divide nodes* $d_{v,1}, ..., d_{v,2i-1}$.

Proof: We prove this lemma inductively. For $i = 1$, there are three possible cases:

- Both $G_{v,1}$ and $G_{v,2}$ have even nodes $\in S_1$. We can match those nodes within $G_{v,1}$ and $G_{v,2}$ since we can always match an even number of nodes within a complete graph. Then $d_{v,1}$ can match $d_{v,2}$.

- Both $G_{v,1}$ and $G_{v,2}$ have odd nodes $\in S_1$. We can match one node in $G_{v,1} \cup S_1$ with $d_{v,1}$ and one node in $G_{v,2} \cup S_1$ with $d_{v,2}$. Other nodes can be matched within $G_{v,1}$ and $G_{v,2}$.

- Either $G_{v,1}$ or $G_{v,2}$ has odd nodes $\in S_1$. $d_{v,1}$ can match one node in S_1, which is in the subset with odd nodes $\in S_1$. Then, the other nodes $\in S_1$ are matched within the subsets.

Therefore, the lemma is true for $i = 1$.
Suppose the lemma is true for $i = k$. For $i = k + 1$: if $d_{v,2k}$ is matched, then we only need to match the nodes to be matched in $G_{v,2k+1}$ and $G_{v,2k+2}$ and $d_{v,2k+1}$, which is the same case as $i = 1$ if we view $G_{v,2k+1}$ as $G_{v,1}$, $G_{v,2k+2}$ as $G_{v,2}$ and $d_{v,2k+1}$ as $d_{v,1}$.

If $d_{v,2k}$ is not matched, there are four cases:

- Both $G_{v,2k+1}$ and $G_{v,2k+2}$ have even nodes $\in S_1$. We can match those nodes within $G_{v,2k}$ and $G_{v,2k+1}$, and match $d_{v,2k}$ with $d_{v,2k+1}$.

- Both $G_{v,2k+1}$ and $G_{v,2k+2}$ have odd nodes $\in S_1$. We can match one node of $G_{v,2k+1} \cup S_1$ with $d_{v,2k}$ and one node of $G_{v,2k+2} \cup S_1$ with $d_{v,2k+1}$. Other nodes $\in S_1$ can be matched within $G_{v,2k+1}$ and $G_{v,2k+2}$.

- $G_{v,2k+1}$ has odd nodes $\in S_1$ and $G_{v,2k+2}$ has even nodes $\in S_1$. We can match one node of $G_{v,2k+1} \cup S_1$ with $d_{v,2k}$ and match $d_{v,2k+1}$ with $d_{v,2k+2}$. The other nodes are matched within the subsets.

- $G_{v,2k+1}$ has even nodes $\in S_1$ and $G_{v,2k+2}$ has odd nodes $\in S_1$. We can match one node of $G_{v,2k+2} \cup S_1$ with $d_{v,2k+2}$ and match $d_{v,2k}$ with $d_{v,2k+1}$. The other nodes are matched within the subsets.

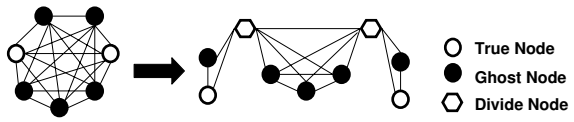


Figure IV.7: Decomposition of a complete gadget with divide nodes.

Therefore, the lemma is true for $i = k + 1$. □

Theorem IV.4 *The T-join problem for a graph $G^D = (V, E, w, T)$, where T denotes the conflict nodes and $V \setminus T$ denotes the legal nodes in V, can be reduced to a minimum-weighted perfect matching on the gadget graph with divide nodes $\mathcal{G} = (V', E', w')$.*

Proof: ($\rightarrow$) Mapping perfect matching solution of the gadget graph with divide nodes $\mathcal{G}$ to a valid solution of the T-join problem on G^D: For any node $v \in G^D$, let its gadget G_v be $2i + 1$ subsets $G_{v,j}$ $j = 1...2i + 1$ connected with divide nodes in the gadget graph. The edges of node v can be grouped into four sets:

- $S_1 = \{e | g_e^v$ matched within $G_v\}$.
- $S_2 = \{e | g_e^v$ not matched within $G_v\} = \{e | t_e^{v'}$ matched within $G_{v'}\}$.
- $S_3 = \{e | t_e^v$ matched within $G_v\}$.
- $S_4 = \{e | t_e^v$ not matched within $G_v\} = \{e | g_e^{v'}$ matched within $G_{v'}\}$.

We need to prove that the set $S = S_1 \bigcup S_4$ thus constructed is a valid solution to the T-join problem. Let the cardinality of S_1, S_2, S_3 and S_4 be a, b, c and d, respectively. In any perfect matching solution, the number of nodes matched within G_v (including $2i$ divide nodes), $(a+c+2i)$, is even. If v is a conflict node, the number of true nodes in G_v, $c+d$, is odd by construction and $(a+c+2i)+(c+d) = (a+d)+2c+2i$ is odd. Therefore, the number of edges in S, $a+d$, is odd. Similarly, if v is a legal node, the number of edges in S is even. Therefore, the solution S is a valid solution of the T-join problem.

Since the weight of any edge e in the dual graph is always assigned to its corresponding ghost node g_e^v, the total weight of the edges in the T-join solution, $S = S_1 \bigcup S_4$, is equal to the total weight of all ghost nodes matched within gadgets.

On the other hand, the total weight of the matching solution, i.e., the total weight of the matched edges, = the total weight of the matched edges within gadgets (since the weights of edges incident to dummy nodes are all 0), = the total weight of all nodes matched within gadgets (since the edge weight is the total weight of its two nodes), and hence = the total weight of all ghost nodes matched within gadgets (since the weights of all true nodes are 0). Therefore, the total weight remains the same during the mapping.

($\leftarrow$) Mapping a solution S of the T-join problem to a solution of the perfect matching problem of $\mathcal{G}$ can be done as follows: For any node $v \in G^D$ whose gadget is G_v, which includes $2i+1$ subsets $G_{v,j}$ $j = 1...2i+1$ linked with divide nodes, the true nodes and ghost nodes can be grouped into four sets:

- $S_1 = \{g_e^v | e \in S\}$.
- $S_2 = \{g_e^v | e \notin S\}$.
- $S_3 = \{t_e^v | e \notin S\}$.
- $S_4 = \{t_e^v | e \in S\}$.

Let the cardinality of S_1, S_2, S_3 and S_4 be a, b, c and d, respectively. We need to prove that *there is a perfect matching solution in which all divide nodes, the a ghost nodes in S_1 and the c true nodes in S_3 are matched within G_v and the remaining nodes are matched outside G_v.*

If v is a conflict node, since S is a valid solution of the T-join problem, the number of edges $\in S$, $a+d$, is odd. The number of true nodes $(c+d)$ is odd by construction. Thus, $((a+d)+(c+d)) = (a+c)+2d$ is even. Hence, the number of true nodes and ghost nodes to be matched within G_v, $(a+c)$, is even. Since all the $2i$ divide nodes should be matched within G_v, the total number of nodes to be matched, $(a+c)+2i$, is even. According to Lemma IV.3, all nodes to be matched in $G_{v,1}, ..., G_{v,2i}$ and the divide nodes $d_{v,1}, ..., d_{v,2i-1}$ can be matched in a matching solution. The remaining even number of nodes are located in a complete graph $G_{v,2i} \bigcup \{d_{v,2i}\}$. It is always possible to match an even number of nodes in a

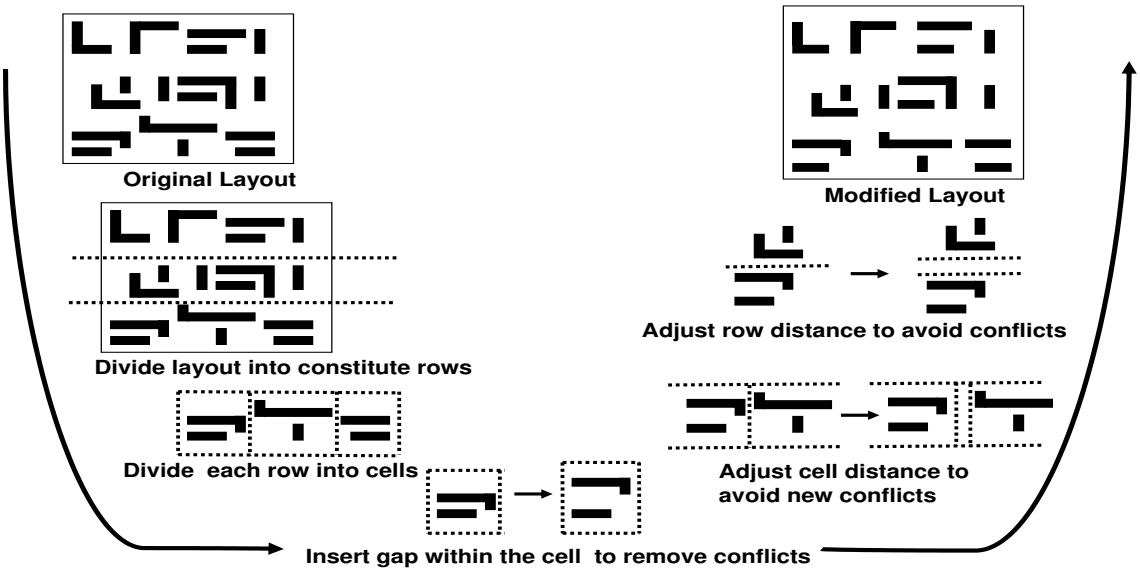


Figure IV.8: Details of layout modification algorithm.

complete graph. Similarly, we can prove that there is a perfect solution when v is a legal node in G^D.

Since the edge weight in the dual graph is always assigned to its corresponding ghost node, we only need to consider the nodes in S_1 and S_2 (true nodes in S_3 and S_4 have corresponding ghost nodes in other gadgets). Among them, only the nodes in S_1 are matched within gadget and their weights are included in the matching solution. In other words, the ghost node weight is included in the matching solution if and only if its corresponding dual edge is in the T-join solution. Therefore, the matching solution has the same weight as the T-join solution.

□

IV.D Layout Modification

The primary task of AAPSM-related layout modification is to correct the phase conflicts that the conflict detection algorithm selected in the previous step. Phase conflicts can be corrected either by adding space between shifters corresponding to a conflict (equivalent to correcting a conflict edge) or by widen-

ing critical features (equivalent to correcting a feature edge). However, widening critical features may introduce significant timing problems. Hence, in the present work, we only focus on phase conflicts that can be solved by increasing the spacing between features such that the corresponding shifters are separated by the required shifter spacing. However, merely increasing the spacing between the shifters corresponding to the phase conflict may cause DRC violations as well as introduce new phase conflicts as the relative locations of the neighboring features may change. The work presented in [34] solved this problem by adding end-to-end spaces throughout the layout. The spaces are inserted such that only the length of the poly interconnect is increased. This technique could only be applied to standard cells and macro blocks with a low density of phase conflicts. Experiments indicate that this method when applied directly to standard-cell blocks can cause large increases in area.

Our layout modification algorithm exploits the fact that standard-cell blocks can be naturally partitioned into rows and that phase conflicts in each row can be solved independently without introducing any DRC errors. The overall flow of the algorithm is presented in Figure IV.8. The algorithm consists of the following steps:

1. First the standard-cell block is partitioned into rows and its constituent cells. The rows are identified by locations of the power grid lines.

2. Next the phase conflicts that are strictly between features of a cell are corrected by adding a minimal number of end-to-end spaces in the cell as illustrated in Figure IV.9. In this scheme, horizontal and vertical spaces of variable width are added along a cut line throughout the cell to correct the chosen AAPSM conflicts. As shown in Figure IV.9 (a), the space insertion is equal to moving all features on the right side of the cut line to the right by a distance B. For any feature across the cut line: if it is not connected with the features on the right side of the cut line (Figure IV.9 (a)), it will not be moved; otherwise, if it is not connected with the features on the left side of the cut line (Figure IV.9 (b)), it will be moved to the right by a distance B;

if it is connected with the features on both sides (Figure IV.9 (c) (d)), the spaces are added such that only the gate widths are increased but the gate lengths remain the same. Therefore, the straight cut line will be replaced by the dashed line as shown in Figure IV.9 (d). This prevents any major timing problems after layout modification.[10]

3. The modified cells in a row are now assembled such that no phase conflicts exist between any two features of adjacent neighboring cells. The height of each row is equal to the height of the tallest cell and the width is equal to the sum of the widths of the standard cell plus the widths of the inserted spaces between the standard cells. The spaces that are occupied by filler cells are made available at this step to avoid any unnecessary area increase.

4. The final step consists of assembling the modified rows. Here, again horizontal space is added only as needed. Space is added only if there is an existing conflict between features of cells on adjacent rows or if relative locations of the features in adjacent rows is changed from the original configuration (this can only happen if vertical space is added at different locations on adjacent rows).

Hence, in this algorithm, end-to-end spaces are only added within a cell. The spaces between the cells and between the rows are smartly managed such that no phase conflicts remain or are introduced after the changes to the individual cells. This results in much smaller area increases for correcting phase conflicts when compared to the method in [34]. According to our layout modification algorithm, the weighting scheme of the conflict cycle graph is as follows. The weights of feature edges are assigned to be infinite, since we do not permit feature widening. The vertical conflicts (i.e., conflicts that can be solved by adding vertical

[10]In practice, the timing impact due to layout modification is negligible since (1) the cell is small; (2) a minimum-weighted set-covering problem (similar to the one proposed in [34]) is used to determine cut lines to avoid most cases like Figure IV.9 (c) and (d); and (3) if the cases like Figure IV.9 (c) and (d) cannot be avoided to correct one conflict, its corresponding edge in the conflict cycle graph will be assigned a large weight to prevent the edge from being selected for correction.

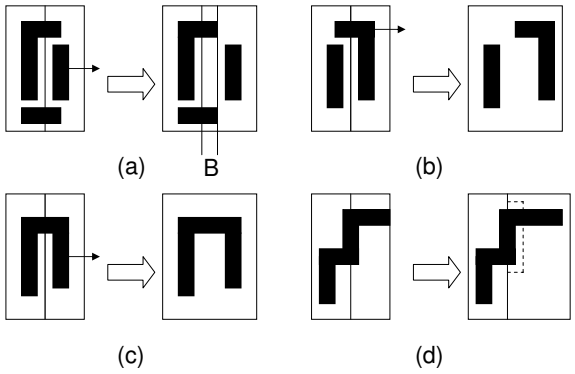
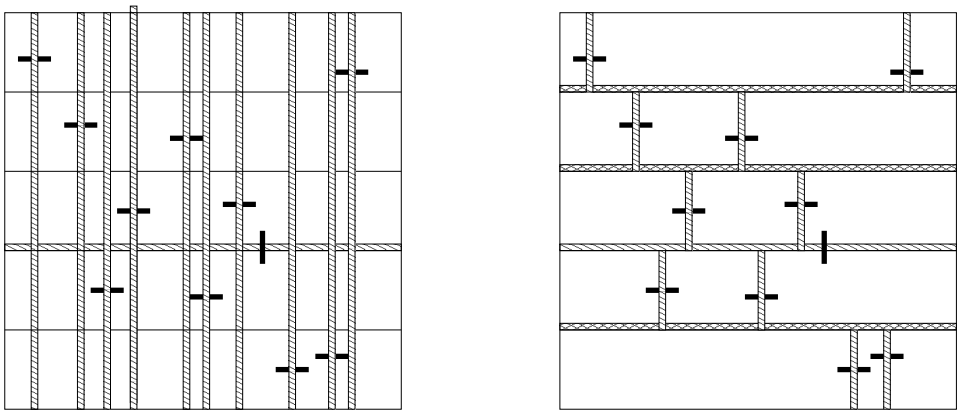


Figure IV.9: Layout modification with vertical space insertion.

end-to-end spaces) are assigned a much lower weight than horizontal conflicts (i.e., conflicts that can be solved by adding horizontal end-to-end spaces), since it is less disruptive to increase the width of the standard cells than their height. In our implementation, the weights of conflict edges of vertical conflicts are assigned as the width of the spacing to be added to solve the conflict, i.e., B in Figure IV.9, to reflect the area increase due to layout modification; the weights of conflict edges of vertical conflicts are assigned as $10\times$ spacing width. The weights of conflicts which may result in increased gate width are assigned as $50\times$ spacing width.

Figure IV.10 compares our layout modification algorithm with the one presented in [34] on a hypothetical example. The layout is a square and is composed of 5 rows of standard cells. Let l denote the length of each side of the layout. The shaded rectangles denote the spaces added in the layout and the bold dark lines are used to represent the locations of the phase conflicts being corrected. Let w denote the width of horizontal and vertical spaces added (assumed to be the same for simplicity). The area increase with the scheme in [34] is $11lw$, whereas the area increase with our scheme is only $6lw$.

The presented algorithm can be applied as an additional processing step during post-placement optimizations. The inserted spaces are integer multiples of the M1 routing pitch and hence the modifications introduced by the proposed flow does not introduce any additional complications for the router. This is also a difference from the method in [34] that does not match the inserted spaces to the



(a) Total Area Increase = l*11*w.

(b) Total Area Increase = (l/5)*10*w + 4*l*w = 6*l*w.

Figure IV.10: Comparing the area increases produced by the layout modification scheme in [35] with the proposed scheme.

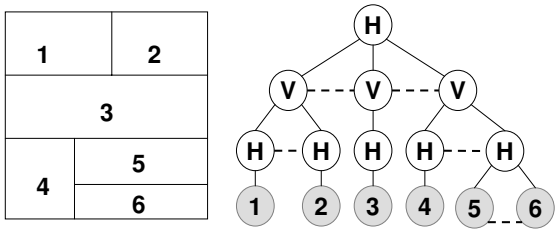


Figure IV.11: Hierarchical layout and its partition tree.

routing pitch. This solution needs to be applied even if the placement is done with AAPSM-compliant standard cells, i.e. cells that have no phase conflicts. This is because phase conflicts can exist between features of neighboring cells. The only difference is that Step 2 of the proposed layout modification algorithm may be omitted.

Our proposed algorithm can also be easily extended to solve phase conflicts in slicing hierarchical layouts, i.e., the layouts whose floorplan can be represented by slicing trees. As shown in Figure IV.11, a slicing hierarchical layout can be represented using the *partition tree*, where each leaf node represents a layout region. The H (or V) node represents a region which is partitioned into several child regions using horizontal (or vertical) cut lines; there is a dashed line between

any two neighboring child nodes which represents the cut line. The layout modification can be solved using a bottom-up algorithm. First, the phase conflicts are corrected within each leaf node by inserting end-to-end spaces. Then for each upper level, new phase conflicts are avoided by inserting gaps along the cut lines. The algorithm shown in Figure IV.8 is the special case when the input layout can be represented as a two-level tree.

IV.E Experimental Results

This section presents the experiments we conducted to test the benefits of the proposed ideas. All our examples are *90 nm* designs and assume typical values of threshold width for critical features, shifter dimensions and shifter spacing. This work focuses mainly on phase conflicts that can be solved by increasing the spacing between features in the layout. Thus, phase conflicts caused by T-shapes are not handled. These can be corrected by feature widening or mask splitting [41]. Phase conflicts caused by line-end conflicts between neighboring features are not considered since they can be efficiently detected and corrected using additional DRC checks during layout generation [48].

IV.E.1 Phase Conflict Detection Results

Table IV.1 compares the runtime and the quality of results (number of edges deleted, or in other words, number of phase conflicts selected for correction) of the proposed flow with other state-of-the-art approaches. The flow presented in [34] is our main comparison point since their results are best in terms of the number of phase conflicts chosen for comparison and runtime, when compared to other state-of-the-art approaches [38] [40]. In the table, Columns 1 and 2 give the design names and design statistics (number of polygons and number of shifter overlaps). The results obtained after applying the flow in [34] are grouped under the columns "Flow in [34]". The results obtained using our new phase conflict detection method are grouped under "Proposed Flow". Notably, Columns 4 and 8

give the runtimes of the flow in [34] and our proposed flow, respectively[11]. As can be seen, our runtimes are significantly better than those obtained using the flow in [34] with an average improvement of 5.9x. This can be primarily attributed to the significant reduction in the number of edges in the conflict cycle graph compared to the phase conflict graph used in [34] and the removal of undeletable edges (in our case, feature edges) during intermediate graph constructions. This is reflected in the number of nodes and edges of the gadget graph constructed during perfect matching. The gadget graphs constructed in our flow are significantly smaller than the ones constructed in [34]. While the examples presented are not very large, we believe the same trend of speedups should also be present in much larger examples. The limitations of the current code prevented us from testing our idea on larger examples.

The table also shows that the quality of our results (in terms of number of phase conflicts chosen for correction) is also better than the results obtained using the method in [34] (see the column labeled "# Conflicts" under the subgroup "Flow in [34]" for the results obtained using the method in [34], and the column labeled "# Conflicts" under subgroup "Proposed Flow" for the results obtained with our method). This improvement is primarily due to the fact that the number of edges deleted during planarization of the conflict cycle graph (second step in Figure IV.2) is smaller than the number of edges deleted during the corresponding planarization step of the phase conflict graph [34]. Hence, the optimal algorithm can be applied to a larger subgraph of the original graph.

IV.E.2 Phase Conflict Correction Results

Table IV.2 reports the results of using the proposed layout modification scheme for correcting the phase conflicts chosen by the detection step on the same layouts. Column *Area* reports the area of the designs in square microns. Column

[11]It should be noted that only the time spent in solving the perfect matching problem is reported in both cases as this is the most compute-intensive portion of the algorithm. The gadget graph construction was also sped up by 2x using the new graph, but the perfect matching has a greater contribution to the total runtime.

Table IV.1: Phase conflict detection results. Experiments were run on a 4X400 Mhz Ultra-Sparc II with 4.0 GB of RAM.

		Flow in [34]			Proposed Flow		
Design	# Plgns/#Ovlps	#edges	CPU(s)	# Conflicts	#edges	CPU(s)	# Conflicts
1	10274/24580	98347	2.53	938	66875	0.38	910
2	13630/32257	112599	1.90	963	79672	0.22	946
3	21868/53749	182809	3.33	1558	128836	0.55	1534
4	20425/50059	173319	3.18	1678	121546	0.48	1664
5	25784/63760	216691	3.87	1854	152655	0.60	1839
6	48787/157668	484999	12.17	6330	325769	2.78	5989
7	44121/142707	436297	12.30	5010	295001	2.47	4865
8	72101/237557	729980	20.05	10275	489922	4.23	9631
9	105882/376707	1133279	41.35	18148	757581	7.95	17463
10	159070/552767	1667581	66.40	27308	1115928	11.58	26349

Conflict specifies the number of phase conflicts selected by the detection algorithm for each design (the numbers are slightly different from those in Table IV.1 due to the use of different weighting schemes). Column *Outside* reflects the number of phase conflicts that are selected for correction and occur between features of neighboring cells. As can be seen, it is a very small fraction of the total number of phase conflicts in any design. This strengthens our view that it is too conservative to leave blank space around all the cells or force the boundary features of each cell to have the same phase, because this could cause large area increases. The fifth column reports the percentage area increase for these layouts as a result of the added spaces. The area increase for these layouts ranges from 1.7-9.1%, with an average increase of 6.1%. The area increase goes up slightly with the size of the testcases. For comparison, the layout increase caused by the method in [34] is also reported in the last column. As can be seen, the area increases caused by the method in [34] are very large.

IV.F Summary

In this chapter, we have presented a new theory for Bright-Field phase conflict detection. The proposed method greatly simplifies the graph constructed

Table IV.2: Layout modification results for standard-cell blocks.

Design	Area	Conflict	Outside	% Area Inc.	% Area Inc. [34]
1	25173.96	937	61	1.7	18.1
2	16397.82	995	197	5.4	23.1
3	31416.21	1589	284	5.8	26.8
4	25715.23	1724	238	6.1	28.8
5	40409.68	1720	322	4.7	32.12
6	61705.52	6257	770	5.8	57.47
7	58414.06	5100	586	6.1	59.1
8	94178.09	10141	512	7.3	80.2
9	148231.77	18657	2672	9.1	>100.0
10	249210.41	28121	4224	9.0	>100.0

from the layout, which results in a substantial reduction in its edge count. Unlike previous constructions, the proposed graph does not equate phase-assignability of its corresponding layout to its bipartition. Rather, a new property of the graph called *conflict cycles* is introduced, and an optimal algorithm for removing conflict cycles in embedded planar graphs is presented. The algorithm is also generalized such that a minimal solution may be obtained for nonplanar graphs. Supporting experimental results confirm huge runtime improvements with the same quality of results (in terms of number of phase conflicts chosen for correction) when compared against the best previous work in this area.

We also present a novel layout modification algorithm for standard-cell blocks. Experimental results confirm that the new method leads to much smaller increases in area than previous approaches to the phase conflict resolution problem. The small area increases make the new method suitable for use in a post-placement optimization step within production industry flows. The current algorithm does not assume that the standard cells used in the placement are phase-assignable. However, the proposed method can be applied to a placement of AAPSM-compliant cells and will produce much smaller area increases, when compared to other methods currently under consideration for building phase-assignable placements.

The material presented in this chapter is based on the following publica-

tion.

- C. Chiang, A. B. Kahng, S. Sinha, X. Xu and A. Zelikovsky, "Fast and Efficient Bright-Field AAPSM Conflict Detection and Correction", *IEEE Transactions on CAD*, (2006) accepted and to appear.

The dissertation author was the primary researcher and author. My coauthors (Prof. Andrew B. Kahng, Dr. Charles Chiang, Dr. Subarna Sinha and Prof. Alex Zelikovsky) have all kindly approved the inclusion of the aforementioned publication in my thesis.

Chapter V

Optimal Post-Routing Redundant Via Insertion for Manufacturing and Timing Yield Improvement

Via doubling, or redundant via insertion, is an effective DFM technique for yield improvement, electromigration alleviation, and performance enhancement. In this chapter, we propose maximum matching based post-route via doubling that achieves optimum yield improvement. We also exploit the insertion of redundant interconnect or "short loops" in addition to traditional minimum-spacing redundant via insertion, so as to achieve a maximum number of doubled vias. We further propose timing-driven redundant via insertion based on performance sensitivity computation and a weighted maximum matching algorithm for timing yield improvement. Our experimental results show that our perfect matching based redundant via insertion reduces the number of undoubled (i.e., critical) vias by 99.4% and 98% compared with the two best previous post-routing via doubling heuristics, and increases the via doubling coverage from 94.5% and 98.2% to 99.97%. One interesting observation is that nearly 100% via doubling coverage can be achieved with simultaneously optimal redundant via and short loop insertion in the post-route stage. Our timing-driven redundant via insertion achieves up to 3.3% timing yield improvement compared with timing-oblivious redundant via

insertion with the same number of doubled vias on the critical paths.

V.A Introduction

VLSI manufacturing techniques have in the past achieved low defect densities which enable integration of increasingly larger numbers of devices on a single chip, along with continuous performance increases. However, in the latest nanometer-scale VLSI processes, traditional low defect densities are not so easily achieved. Design techniques are needed for VLSI yield enhancement.

VLSI defect density and yield estimation is usually based on critical area computation, which captures surface defects, i.e., opens and shorts in on-chip interconnect or device layers. However, subsurface defects, or via voids, cannot be ignored.

Causes of failed vias include airborne particles, electromigration, or thermal stress induced voiding. For example, an airborne particle may block the via from functioning and lead to via opens. In the electroplating process, due to the presence of overhanging sidewalls in the narrow features, profiles are prone to premature pinch-off at the opening of the feature and result in via voids. If a via is partially blocked, its resistance greatly increases and becomes a resistive via. As a result, its performance is inhibited and the timing yield is decreased since the via resistance is relatively high, e.g., the resistance of one nominal copper via is already approximately equal to that of 15 routing tracks.

The effects of via voids or via opens can be minimized by inserting a second "backup" via. In other words, via doubling is "insurance", which is beneficial to both random defects (particle defects) and parametric defects (timing related defects) [57]: (1) the backup via will perform if the original one is completely blocked and (2) the parametric yield loss due to resistive vias can be reduced by providing an alternative current path with the second via. Therefore, it is desirable to add redundant vias to a ground-rule-correct VLSI design for the purpose of increasing manufacturing yield. In fact, via doubling (along with via enclosure augmentation) has been strongly recommended as a yield enhancement rule

in recent 90nm and below processes. Major EDA vendors have also included via doubling functionality in their latest routing tools.

Unfortunately, via doubling may conflict with other layout optimization objectives, e.g., minimum spacing between neighboring layout features and availability of OPC features (i.e., scattering bars). Via doubling should not increase die size or perturb circuit performance, nor should it affect design convergence. As a result, via doubling is performed wherever possible, with minimum layout perturbation, e.g., in terms of coupling capacitance variation and the resultant performance variation, as well as minimum additional routing.

In the detailed routing literature, Yao et al. proposed improved detailed routing with redundant via insertion [59]. Xu et al. proposed a Lagrangian relaxation solution for simultaneous redundant via insertion and maze routing [58]. Redundant via insertion has also been proposed for post-route layout optimization. Lee et al. [52] find redundant via insertions which may lead to design rule violations, then formulate post-route redundant via insertion as a maximum independent set problem, and propose a greedy heuristic to solve the NP-hard MIS problem along with a fast heuristic for improving the ratio of on-track vias to minimize wirelength overhead. Redundant interconnects may also be used to connect to redundant vias. Bickford et al. [50] introduced local interconnect redundancy with short loops that include redundant vias; this technique reduces short fault critical area and wrong-way routing, improves yield and timing performance, and fixes antenna rule violations. In the method of Bickford et al., short loops are inserted after minimum-spacing redundant via insertion to minimize layout perturbation and wirelength increase. However, a number of potential directions for improvement remain:

- no previous post-route via doubling technique optimally finds a maximum number of doubled vias;
- inserting short loops only after, rather than concurrently with, minimum spacing via insertion leads to suboptimal solution quality; and

- no previous via doubling technique targets timing yield improvement (a simple timing analysis for via doubling is presented in [54], but timing yield analysis is not presented in an previous works).

In this chapter, we study post-route redundant via insertion for maximum via doubling coverage and yield improvement. Our contributions are as follows.

1. We propose a perfect matching based *optimal* redundant via insertion algorithm. To the best of our knowledge, our proposed method is the first algorithm that can optimally solve the post-route via doubling problem in practical runtime. Existing techniques are either greedy heuristics or non-scalable techniques for NP-hard problem formulations. By contrast, our approach finds all conflict routes which may violate design rules, constructs gadgets that enable perfect matching to produce a DRC-clean via doubling solution, and thus achieves improved efficiency and increased redundant via insertion.

2. We extend both the traditional minimum spacing redundant via insertion [52] and the local redundant loop insertion [50] methods. We insert redundant vias and interconnects wherever possible (while taking wirelength overhead into account), and achieve near 100% redundant via coverage in a post-route stage.

3. We consider performance impact of redundant via insertion. We observe that redundant via insertion reduces via resistance and produces potential hold time violation, and perform Monte Carlo SPICE simulation to assess timing yield impacts as a guide for redundant via insertion.

The remainder of this chapter is organized as follows. We present necessary background and our formulation of the via doubling problem in Sections V.B and V.C, and propose a minimum weighted perfect matching algorithm in Section V.D. Our timing-driven via doubling method is presented in Section V.E. We present experimental results in Section V.F and conclude in Section V.G.

V.B Background

A simple VLSI yield estimation method is based on particle size distribution and critical area computation. The occurrence probability for a particle can be characterized as a piecewise linear function on a log scale in term of the particle size. For each particle size, critical areas are computed which are the unions of the locations where a particle strike would lead to an open or short defect. Such a model captures surface defect occurrence probability. For a complete yield estimation, one also needs a via defect model [56].

Via formation starts with an etching process at the via locations, followed by metal vapor deposition which yields the vias and the upper layer metal routes. Therefore, via defects can be due to (1) the etching process leaving an irregular via site, or (2) atmospheric pressure in metal vapor deposition leaving a via void. Via voids disconnect a via in extreme cases, and vary via resistance otherwise. A high-resistance via is susceptible to electromigration (i.e., the mass shifting of metal atoms in an interconnect due, in particular, to unidirectional currents) in power/ground networks, and thus implies degraded circuit lifetime. In a regular signal interconnect, where current flow is bidirectional, a high-resistance via leads to increased interconnect delay and degraded circuit performance. A high-resistance via distribution is presented in [51].

V.C Problem Formulation

We consider redundant via insertion in a routed layout, where the minimum distance between two adjacent routes is given by the minimum distance design rule between a metal route and a via, which is smaller than the minimum distance design rule between two vias (Fig. V.1) [58].

We make the following definitions.

- A $k \times m \times n$ 3-D *routing grid* is given by $m \times n$ grids on each of k routing layers.

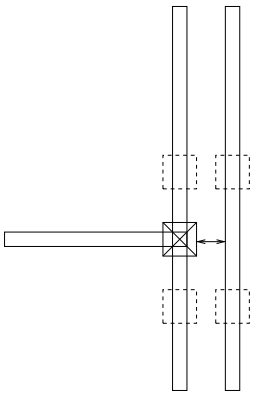


Figure V.1: Two adjacent routes are separated by the minimum distance between a via and a metal wire, which is smaller than the minimum spacing rule between two vias, such that two vias cannot be inserted at two adjacent locations.

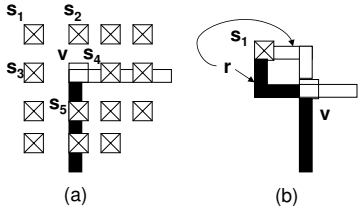


Figure V.2: Candidate sites (a) for one via v and (b) for the routing r to connect the redundant site s_1 for via v.

- A *via site* s is a vertical edge in the 3-D routing grid.
- A via is *critical* if an open defect of the via disconnects the interconnect.
- A via is *doubled* if an open defect of the via does not disconnect the interconnect.
- A *candidate doubling site* s doubles an existing via v when inserted with its associated additional route r. Figure V.2 (a) shows the candidate doubling sites for a via v. Figure V.2 (b) shows the route r to connect the redundant site s_1 for via v.

We formulate redundant via insertion in a routing grid as follows.

Problem 1 *Given*

1. *a $k \times m \times n$ routing grid H,*
2. *existing routes R,*
3. *existing vias V,*
4. *via-to-via minimum spacing S_{v-v},*
5. *via-to-wire minimum spacing S_{v-w},*
6. *maximum length of routing segment on a layer (antenna rules) L_{ant}, and*
7. *possible timing criticality C_T for each net*

find redundant vias and associated routes which achieve

1. *a minimum number of critical vias, and/or*
2. *maximum timing yield or probability to satisfy critical path delay constraints.*

V.D Optimal Redundant Via Insertion

Intuitively, redundant via insertion can be formulated as a bipartite matching problem, e.g., of finding a maximum number of matches in a graph $G = (V \cup S, E)$, where V are nodes for the existing vias, S are nodes for the possible redundant via insertion sites, E are edges connecting an existing via v and a possible redundant via insertion site s, if redundant via insertion at s doubles the existing via v with the additional route r. However, this simple matching formulation does not take into account (1) mutually exclusive redundant vias and routes, and (2) implied simultaneous via occupation, e.g., in a short loop. The inserted redundant vias given by this simple matching algorithm may violate DRC rules.

In this section, we construct a ***via graph*** G and apply weighted perfect matching for redundant via insertion, which finds the weighted maximum number of redundant vias while taking into account mutually exclusive redundant vias and routes and simultaneous via occupation implications. In other words, the via

graph G allows us to reduce an instance of via doubling to an instance of perfect matching which can be optimally solved.

Problem 2 (Weighted Maximum Matching) *Given a graph $G = (V, E, W)$ of nodes V, edges E, and edge weights W, find a subset of edges $E' \subseteq E$ such that no two edges in E' share a common endpoint and the sum of weights for the edges in E' is maximized.*

The via graph G is constructed in four steps as shown in Algorithm 1: (1) including all vias, possible via insertion sites and additional routes for via doubling, (2) constructing conflict gadgets which guarantee DRC correctness, (3) constructing gadgets for simultaneous via occupation implications, and (4) augmenting the graph G for guarantee of the existence of a perfect matching solution. We present each step in details in the following subsections.

Algorithm 1: Perfect Matching Based Redundant Via Insertion

Input: routing grid H, existing routes R, existing vias V,
Output: Redundant vias V_R

---

1. Construct initial graph
2. Construct gadgets for mutually exclusive redundant vias and routes
3. Construct gadgets for implied simultaneous redundant via occupation
4. Construct gadgets for existence of perfect match
5. Apply weighted maximum perfect matching algorithm

V.D.1 Construction of Initial Graph

In this step, we construct and initial graph G_i with via nodes V, redundant via insertion site nodes S, and edge $(s, v) \in E$ if there exists a route r between existing via $v \in V$ and possible redundant via insertion site $s \in S$ such that insertion of a redundant via at s doubles via v. As shown in Figure V.2 (a), only the neighboring sites are considered for doubling in order to minimize the wirelength overhead.

Algorithm 2: Construct Initial Graph

Input: routing grid H, existing routes R, existing vias V
Output: Initial Graph $G_i = (V \cup S, E)$

1. For each via
2. Construct a via node $v \in V$
3. For each redundant via insertion site
4. Construct a site node s
5. For each route r which bi-connects a via site s and a via v
6. Construct an edge $e(v, s) \in E$.

The weight of an edge $e(v, s)$ is a function $w(e) = f(v, r)$ of v and r, and is given by the optimization objectives, e.g., increased wirelength, timing criticality or via failure rates.

V.D.2 Construction of Gadgets for Mutual Exclusiveness

In this step, we perform design rule check (DRC) and find the conflicting redundant interconnects and vias for which we must ensure solution legality, i.e., that the selected routes in the perfect matching solution will not cause any design rule violations. We scan the layout and check three minimum spacing rules: (1) via to via, (2) via to wire, and (3) wire to wire on each routing layer for each local area.[1]

Definition V.1 *Two routes r_1 (connecting v_1 and s_1) and r_2 (connecting v_2 and s_2) are* conflict routes *if (1) $v_1 \neq v_2$, $s_1 \neq s_2$[2] and (2) r_1 and r_2 cannot be simultaneously taken.*

One example of two conflict routes is shown in Figure V.3(a). In this example, two stacked via sites s_1 and s_2 cannot be simultaneously taken by r_1 and r_2 without resulting in a short circuit failure. We replace the two via-site edges

[1] We use a method similar to that of [52] for checking possible design rule violations, i.e., construct an R-tree for range query, create expanded boxes for via sites and additional routes, and check box intersections for possible design rule violations.

[2] *The edges sharing the same via or site node cannot be simultaneously taken in the perfect matching solution and hence will not result in any design rule violation.*

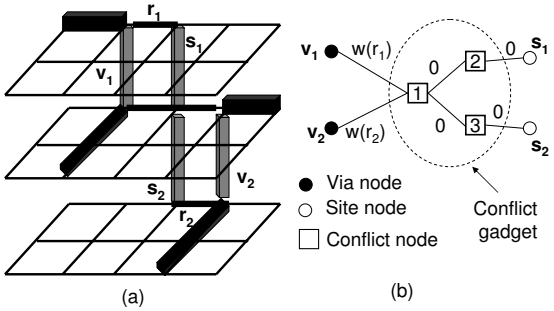


Figure V.3: An example of two exclusive routes (a) and the corresponding conflict gadget (b). In this example, the stacked via sites s_1 and s_2 cannot be simultaneously occupied.

$e(v_1, s_1)$ and $e(v_2, s_2)$ in the initial graph G by a gadget shown in Figure V.3(b). The solution legality is guaranteed since the nodes v_1 and v_2 cannot both match node 1. If node 1 matches v_1, r_1 is taken and nodes s_1 and s_2 have to match nodes 2 and 3 respectively; (2) if node 1 matches v_2, r_2 is taken; (3) otherwise, node 1 matches node 2 or 3, and neither r_1 nor r_2 will be taken.

Algorithm 3 summarizes our gadget construction method for mutually exclusive redundant vias and routes.

Algorithm 3: Gadget Construction for Mutual Exclusivity

Input: routing grid H, existing routes R, existing vias V, Initial Graph $G_i = (V \cup S, E)$
Output: Graph $G_c = (V \cup S \cup C, E)$ with conflict gadgets

---

1. For each route r_1 between v_1 and s_1
2. For each route r_2 (between v_2 and s_2) near r_1
3. If r_1 and r_2 are conflict routes
4. Replace two via-site edges with a conflict gadget

V.D.3 Construction of Gadgets for Implied Simultaneous Occupation

Traditional minimum spacing redundant via insertion cannot double all the vias, especially, the "dead vias".

Definition V.2 *A* dead via *is a via that cannot be doubled by a single inserted redundant via.*

For example, the via v shown in Figure V.4 is a dead via since it is blocked by four segments of other nets and hence cannot be doubled by a single inserted redundant via. Bickford et al. [50] proposed to insert a "short loop" with three vias to double the dead vias as shown in Figure V.4.

Definition V.3 *A* short loop *is a route which connects the top and bottom segments of an existing via and provides an alternate path with three additional vias.*

It is reported that via doubling coverage can be greatly improved (up to 97.5%) with the introduction of short loops. Short loops are also desirable in 45nm designs since they avoid wrong-way wiring that can challenge optical lithography [50]. However, one important constraint associated with short loops is that in order to insert one redundant via at the site s_1, two other sites s_2 and s_3 must also be occupied. To reflect this constraint, we create a conflict gadget which includes four conflict nodes and five edges as shown in Figure V.5(a). If v matches node 1 and s_1 matches node 3 (i.e., v is doubled at the site s_1), node 2 must match s_2 to avoid being unmatched and node s_3 must match node 4 (i.e., both s_1 and s_2 are occupied).

In addition, if the short loop r is in conflict with another route r_0 and they do not share the same via nodes or site nodes, we must combine the conflict gadget for short loop paths and the conflict gadget for conflict routes to form a new gadget as shown in Figure V.5(b). To minimize the timing issues and antenna problems associated with short loops, we limit the searching space to 10 tracks around v.

V.D.4 Construction of Gadgets for Existence of Perfect Match

Finally, we include additional nodes and edges in G_c to ensure the existence of a perfect matching solution in G.

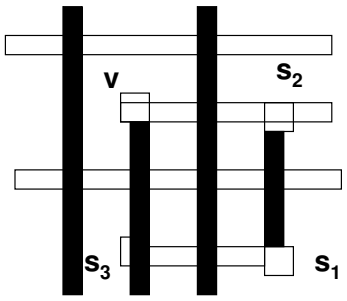


Figure V.4: A short loop (v, s_2, s_1, s_3) which doubles a dead via v.

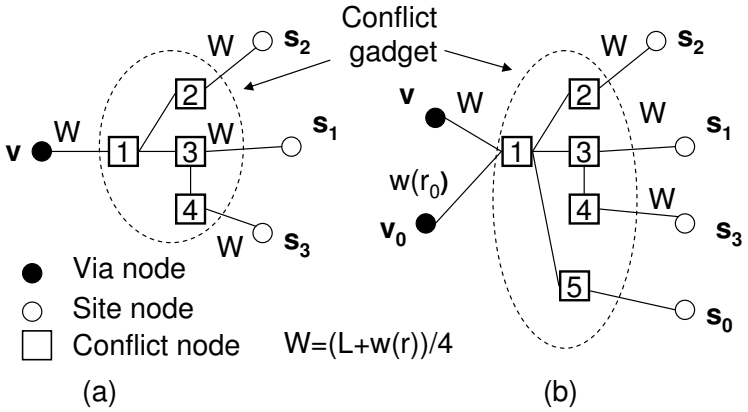


Figure V.5: (a) A gadget for a short loop, and (b) a gadget for a short loop in conflict with another route r_0.

Definition V.4 *A graph $G(V, E)$ is* matchable *if for any selected set of nodes $S_1 \subseteq V$, we can match the nodes in S_1 such that the number of unmatched nodes is at most one.*

Fact: *A complete graph is matchable* since we can pair up the nodes in S_1 until zero or one node is left.

Therefore, a straightforward way to guarantee the perfect matching feasibility is to connect every pair of via nodes and every pair of site nodes to form two complete graphs as shown in Figure V.6. However, a complete gadget of n nodes has $\frac{n(n-1)}{2}$ edges, and perfect matching runtime significantly increases with edge cardinality. Hence, we propose to insert *divide nodes* into G to reduce the number

of edges. As shown in Figure V.6, all via nodes in V are divided into $2i+1 (i \geq 0)$ gadgets GV_j $(j = 1...2i+1)$, which are linked with $2i$ *divide nodes*. Any pair of nodes in GV_j are connected to form a small complete graph. All nodes in GV_j and GV_{j+1} are connected to *divide nodes* dv_j, $(j = 1...2i)$. Each pair of neighboring divide nodes $\{dv_j, dv_{j+1}\}$ $(j = 1...2i-1)$ is also connected. To ensure that the matching by edges in G_c always has priority to the matching by the edges not in G_c in a minimum-weight perfect matching, the weights of all the edges not in G_c are set to be a sufficiently large number L. All the site nodes can be similarly divided into small gadgets GS_j $(j = 1...2i'+1)$ as shown in Figure V.6. The total edge cardinality of the constructed gadgets is greatly reduced compared with that of the complete graph. For example, if we limit the number of nodes in each gadget to be at most three, the edge cardinality is at most $\frac{10(n+2)}{3}$. Therefore, edge cardinality is reduced from $O(n^2)$ to $O(n)$ while node cardinality is still $O(n)$, which implies considerable reduction in perfect matching runtime. In the following, we will prove that the constructed gadget graph is still *matchable*.

Lemma V.1 *For any subset $S_1 \subseteq \{GV_1 \bigcup GV_2, ... \bigcup GV_{2i}\}$ $(i \geq 1)$, there is a perfect matching solution to match all the nodes in S_1 and the divide nodes $dv_1, ..., dv_{2i-1}$.*

Proof: We prove the lemma by induction.
For $i = 1$, there are three possible cases:

- Both GV_1 and GV_2 have an even number of nodes $\in S_1$. We can match those nodes within GV_1 and GV_2, respectively, and match dv_1 with dv_2.

- Both GV_1 and GV_2 have an odd number of nodes $\in S_1$. We can match one node in $GV_1 \cup S_1$ with dv_1 and one node in $GV_2 \cup S_1$ with dv_2. Then the remaining nodes $\in S_1$, which are even in number, can be matched within GV_1 and GV_2.

- Either GV_1 or GV_2 has an odd number of nodes $\in S_1$. Suppose GV_1 has an odd number of nodes $\in S_1$; we can then match one node in $GV_1 \cup S_1$ with dv_1 and the other nodes $\in S_1$ are matched within the gadgets.

Therefore, the lemma is true for $i = 1$.

Suppose the lemma is true for $i = k$.

For $i = k + 1$: since the lemma is true for $i = k$, we can first match the nodes in $S_1 \cap \{GV_1 \bigcup GV_2, ... \bigcup GV_{2k}\}$ and the divide nodes $dv_1, ..., dv_{2k-1}$.

If dv_{2k} is already matched, then we only need to match the nodes in S_1 which are in $GV_{2k+1} \cup GV_{2k+2} \cup \{dv_{2k+1}\}$, which is the same case of $i = 1$ if we treat GV_{2k+1} and GV_{2k+2} as GV_1 and GV_2 and dv_{2k+1} as dv_1.

Otherwise, there are four cases:

- Both GV_{2k+1} and GV_{2k+2} have even numbers of nodes $\in S_1$. We can match these nodes within GV_{2k} and GV_{2k+1}, and match dv_{2k} with dv_{2k+1}.

- Both GV_{2k+1} and GV_{2k+2} have an odd number of nodes $\in S_1$. We can match one node of $GV_{2k+1} \cap S_1$ with dv_{2k}, and one node of $GV_{2k+2} \cap S_1$ with dv_{2k+1}. Other nodes $\in S_1$ are matched within GV_{2k+1} and GV_{2k+2}.

- GV_{2k+1} has an odd number of nodes $\in S_1$ and GV_{2k+2} has an even number of nodes $\in S_1$. We can match one node of $GV_{2k+1} \cup S_1$ with dv_{2k} and match dv_{2k+1} with dv_{2k+2}. Other nodes $\in S_1$ are matched within the gadgets.

- GV_{2k+1} has an even number of nodes $\in S_1$ and GV_{2k+2} has an odd number of nodes $\in S_1$. We can match one node of $GV_{2k+2} \cap S_1$ with dv_{2k+2} and match dv_{2k} with dv_{2k+1}. Other nodes $\in S_1$ are matched within the gadgets.

Therefore, the lemma is true for $i = k + 1$. □

Lemma V.2 *For any subset $S_1 \subseteq \{GV_1 \bigcup GV_2, ... \bigcup GV_{2i+1}\}$ ($i \geq 1$), there is a perfect matching solution which leaves at most one node in $(S_1 \cap GV_{2i+1}) \cup \{dv_{2i}\}$ unmatched.*

Proof: From Lemma 1, all nodes $\in S_1$ which also belong to $(\{GV_1 \bigcup GV_2, ... \bigcup GV_{2i})$ and divide nodes $dv_1, ..., dv_{2i-1}$ can be matched. Then all the unmatched nodes in S_1 and dv_{2i} are located in the complete graph $GV_{2i+1} \cup \{dv_{2i}\}$, which is matchable.

□

The final step is to add one or two *pseudo nodes* to G to make the total number of nodes in G even. If the total number of nodes in G is even, we add two connected pseudo nodes: one connects $GV_{2i+1} \cup \{dv_{2i}\}$, and the other connects $GS_{2i'+1} \cup \{ds_{2i'}\}$ (as shown in Figure V.6); otherwise, we add one pseudo node which connects $GV_{2i+1} \cup \{dv_{2i}\} \cup GS_{2i'+1} \cup \{ds_{2i'}\}$. The perfect matching solution is guaranteed with the following lemma.

Lemma V.3 *There is always a perfect matching solution for any subset* $S_1 \subseteq \{GV_1 \bigcup GV_2, ... \bigcup GV_{2i+1}\} \bigcup \{GS_1 \bigcup GS_2, ... \bigcup GS_{2i'+1}\}$ *($i \geq 1$ and $i' \geq 1$).*

Proof: From Lemma 2, at most one via node $\in S_1 \cap (GV_{2i+1} \cup \{dv_{2i}\})$ is unmatched and at most one site node $\in S_1 \cap (GS_{2i'+1} \cup \{dv_{2i'}\})$ is unmatched. There are three cases.

- One via node $\in S_1 \cap (GV_{2i+1} \cup \{dv_{2i}\})$ is unmatched and one site node $\in S_1 \cap (GS_{2i'+1} \cup \{dv_{2i'}\})$ is unmatched. The total number of nodes in G must be even since the number of matched nodes is always even, and hence there are two pseudo nodes. We can match the two pseudo nodes with the unmatched nodes respectively.

- One node is unmatched. The total number of nodes in G must be odd, and there is one pseudo node. We can match it with the unmatched node.

- All nodes are matched. The total number of nodes in G must be even, and there are two connected pseudo nodes. We can simply match them.

In summary, there is always a perfect matching solution. □

Theorem V.1 *The maximum redundant via insertion problem can be reduced to the minimum weight perfect matching problem in G.*

($\rightarrow$) To map a perfect matching solution in G to a valid via doubling solution, we can delete all the edges except via-site edges and the edges with conflict nodes. Then (1) for all the matched via-site edges, we insert a second via at the site s

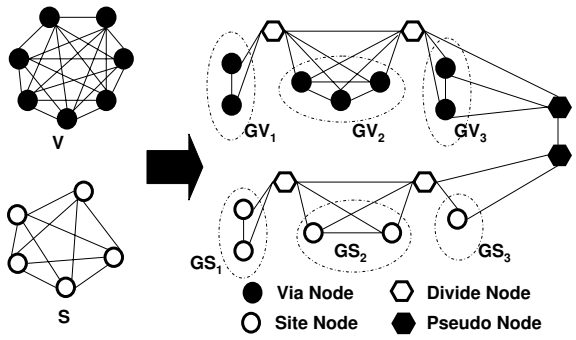


Figure V.6: An example of via and site gadgets construction.

with additional routing r; (2) for the conflict gadgets for exclusive routings shown in Figure V.3, we insert a second via at the site s_1 (s_2) with additional routing r_1 (r_2) if the node 1 is matched with v_1 (or v_2)); and (3) for the conflict gadgets for short loops shown in Figure V.5, we insert redundant vias at s_1, s_2 and s_3 with the additional routing r if both v and s_1 are matched with conflict nodes. The via doubling solution is legal since no conflict routing can be used in the perfect matching solution.

($\leftarrow$) To map a valid via doubling solution to a perfect matching solution in G, suppose the via v is doubled at the site s with the additional routing r. Then, (1) if r corresponds to a via-site edge, match v and s; (2) if r corresponds to r_1 in the conflict gadget shown in Figure V.3, match v_1 with node 1, node 2 with s_1, and node 3 with s_2 (to have a valid via doubling solution, s_2 cannot be matched with other via-site edges); and (3) if r corresponds to a conflict gadget for a short loop shown in Figure V.5, match v with node 1, s_2 with node 2, s_1 with node 3, and s_3 with node 4. Then, for the remaining unmatched via nodes and site nodes, we can match all of them according to Lemma V.3. □

Once the via graph G is constructed, the perfect matching problem can be optimally solved. In our implementation, we include the source code from Cook and Rohe [46].

V.E Timing-Driven Redundant Via Insertion

In this section, we extend our proposed redundant via insertion technique to timing yield improvement, e.g., critical path delay minimization. We compute sensitivity of via resistance to circuit performance, or, the critical path delay variation due to via resistance variation, as a measure of timing criticality of via insertion. The objective of minimum critical path delay is approximated by maximum weighted redundant via insertion, where via insertion weights correspond to timing criticalities. The weighted sum of redundant vias with these weights provides a first-order approximation of critical path delays for the circuit. I.e.,

$$\begin{aligned} \text{Maximize} \quad & \sum_i S_i x_i \\ \text{s. t.} \quad & S_i = \frac{\partial D}{\partial x_i} \\ & x_i = \{0, 1\} \\ & \sum_{i \in C_j} x_i \leq 1 \end{aligned}$$

where D is the critical path delay of the circuit, $x_i = 1$ indicates an inserted via, $x_i = 0$ indicates a non-inserted via site, C_j are mutually exclusive via insertion sites, and $\sum_i S_i x_i$ gives a first-order approximation of critial path delay reduction due to redundant via insertion.

An inserted redundant via reduces interconnect resistance and interconnect delay. It also reduces the resistive shielding effect and implies a possibly increased effective load capacitance, hence a possibly increased gate load delay for the driver of the interconnect. Overall, inserted redundant vias reduce path delay.

For the most accurate delay calculation, we extract RC parasitics for a routed layout and run SPICE simulation to determine sensitivity of circuit performance to via insertion. To avoid hold time violations, we exclude redundant via insertion in the shortest signal propagation paths. To achieve minimum critical path delay, we run timing analysis and find the longest timing-critical paths.

Table V.1: Characteristics of test cases.

Design	#Routs	#Vias	#Layers	WL(m)	#Dead Vias
1	162504	134587	8	1.56	9985
2	297863	381038	8	1.27	831
3	362023	460829	8	1.61	2323
4	640122	521400	8	1.37	6569

For the redundant via insertion sites, we compute the sensitivity of the critical path delays to each possible redundant via insertion, and assign these sensitivies as weights in the minimum perfect matching instance for redundant via insertion.

V.F Experiments

We conduct our experiments on four industry designs in a 90nm process technology with eight routing layers. Table V.1 gives the test case characteristics. We run Cadence SOC Encounter to conduct placement and routing and report timing analysis results. We implemented our proposed algorithm as well as two previous methods for comparison in C++. We first convert the LEF/DEF files into Open Access 2.2.0 database. We then read the Open Access database and perform via doubling. The resulting design is saved, and is converted back to the LEF/DEF format. All experiments are run on an Intel Xeon 2.4GHz system.

Table V.2 compares runtime and solution quality in terms of the number of undoubled vias (in the column "UDV") and increased wirelength overhead (in the column "WL), of the proposed matching flow and the two best previous post-route via doubling algorithms "H2K" and "SLP". "H2K" is the heuristic based on the MIS formulation proposed in [52], and "SLP" is the greedy method with short loop insertion proposed in [50]. Results of our proposed method are given under under the heading "Match". In order to reduce the additional routing overhead (for example, on-track vias are preferred to the off-track vias since they introduce smaller wirelength overhead), we choose the weight for a routing r as $w(r) = 1 + \alpha WL(r)$, where $WL(r)$ is the increased wirelength of r, and α is a chosen

Table V.2: Via doubling results. "H2K" is the heuristic based on the MIS formulation proposed in [52], "SLP" is the greedy method with short loop insertion proposed in [50], and "Match" is our proposed perfect matching method. "UDV" is the number of undoubled vias, "VDC" is the via doubling coverage expressed as a percentage, and "WL" is the percentage increase in wirelength.

Designs		1	2	3	4	Average
H2K	UDV	13255	16326	6213	38300	18523
	VDC (%)	91.5	95.3	98.6	92.6	94.5
	WL(%)	1.3	18.9	20.5	14.6	13.8
	CPU(s)	101	168	193	242	176
SLP	UDV	4942	4378	2188	8821	5082
	VDC (%)	96.3	98.8	99.5	98.3	98.2
	WL (%)	3.2	22.3	22.4	17.5	16.3
	CPU(s)	145	179	234	277	209
Match	UDV	78	49	130	156	103
	VDC (%)	99.942	99.987	99.972	99.970	99.970
	WL (%)	3.2	22.4	22.4	18.4	16.6
	CPU (s)	1512	2253	2816	5273	2613

constant such that $\alpha WL(r)$ is far smaller than 1. Therefore, if a via can be doubled at two sites, the routing with smaller increased wirelength will be chosen. We can see that our proposed optimal perfect matching algorithm outperforms the previous heuristics. The number of undoubled vias is reduced on average by 99.4% and 98% compared with the previous heuristics "H2K" and "SLP" respectively, which increases the via doubling coverage from 94.5% and 98.2% to 99.97% on average. The wirelength overhead of our method is slightly high since our new method produces more doubled vias. Also, any method using short loops will have higher wirelength overhead.

To assess the impact of our proposed timing-driven via doubling, we perform timing analysis and obtain the top 10 critical paths for the four routed designs. For a given via doubling percentage, we compare timing-driven doubling (TD) and random via doubling (RD) effects on the critical path delays. We adopt a 2.5% via failure rate [55], and a via resistance distribution from [51]. For timing-driven via insertion, we compute sensitivity of critical path delay to each possible via

Table V.3: Timing yields of timing-driven (TD) and random (RD) redundant via insertion for different percentage levels of redundant via coverage.

Design	Method	% Redundant Vias					
		0%	20%	40%	60%	80%	100%
1	TD	81.2	92.6	94.6	96.3	97.8	99.7
	RD	81.2	91.8	93.2	95.1	97.0	99.7
2	TD	77.9	87.6	93.2	96.9	98.2	99.6
	RD	77.9	86.4	91.4	94.7	96.8	99.6
3	TD	82.8	93.2	95.6	98.1	99.4	99.8
	RD	82.8	92.3	94.1	96.8	97.8	99.8
4	TD	77.0	90.1	94.3	97.9	98.8	99.6
	RD	77.0	88.3	92.4	94.6	96.7	99.6

doubling, and insert redundant vias in decreasing order of critical-path delay sensitivity. For random via insertion, vias are inserted with equal probability into the critical paths. We then run 1000 Monte Carlo SPICE simulations for the top 10 critical timing paths for each design, and obtain timing yield (i.e., occurrence probability for a critical path delay to be larger than the constraint) for timing-driven versus random via doubling. We observe that with the same number of doubled vias on the timing critical paths, timing-driven via doubling improves timing yield by up to 3.3% compared with random via doubling.

V.G Summary

Via doubling is an effective technique for yield improvement, electromigration alleviation, and performance enhancement. This thesis proposes perfect matching based post-routing redundant via insertion, along with insertion of redundant interconnects for connections to redundant vias, and achieves a substantially increased level of via doubling coverage. We propose timing-driven redundant via insertion, based on performance sensitivity of redundant via insertion. Our experimental results show that our proposed perfect matching based redundant via insertion reduces the number of undoubled (or critical) vias by 99.4% and

98% compared with two best previous post-routing via doubling heuristics, and increases the via doubling coverage from 94.5% and 98.2% to 99.97% on average. Our timing-driven redundant via insertion achieves up to 3.3% timing yield improvement compared with timing-oblivious redundant via insertion, with the same number of doubled vias on the critical paths.

The material presented in this chapter is based on the following submitted draft.

- A. B. Kahng, B. Liu and X. Xu, "Perfect Matching Based Optimal Post-Routing Redundant Via Insertion for Manufacturing and Timing Yield Improvement".

The dissertation author was the primary researcher and author. My coauthors (Prof. Andrew B. Kahng and Dr. Bao Liu) have all kindly approved the inclusion of the aforementioned work in my thesis.

Chapter VI

Conclusions and Future Work

Currently the topics of VLSI design for manufacturability (DFM) and IC manufacturing attract a tremendous amount of interest. The key issue is to design chips that can be physically manufactured and that will work as planned. The purpose of this thesis is to explain the manufacturability and manufacturing problems associated with current 90nm and 65nm technology nodes as well as future technology nodes, and to develop new solutions to satisfy the manufacturability and manufacturing requirements of today's ultra-deep submicron technologies.

In reality, the core problem underlying the vast majority of manufacturability and manufacturing issues is the fact that the feature sizes on the silicon chip are now smaller than the wavelength of the light used to create them. As a result, we need to post-process the drawn layout data file with a variety of resolution enhancement techniques (RET), such as optical proximity correction (OPC) and phase shift mask (PSM). A number of associated mask and wafer manufacturability and manufacturing issues are addressed in this thesis.

- **Fracturing** An optimal integer linear programming formulation and fast heuristics based on the ray-segment selection formulation are suggested for the fracturing problem in variable shaped-beam mask writing. Our new approach improves both shot count and, very substantially, sliver count, in comparison to leading commercial fracturing tools. We have suggested a fast gain-based ray segment selection method with auxiliary rays for the fractur-

ing problem in VSB mask writing. In particular, our fracturing solutions on three industry testcases dramatically reduce sliver count (which reflects the risk of mask critical-dimension errors) by 83.7% and 60.5% compared to two commercial fracturing tools while also reducing shot count (which reflects write time and mask cost) by 5.5% and 0.6% with negligible runtime overhead. Our results reveal significant headroom that can be exploited by future design-to-mask tools to reduce the manufacturing variability and cost of IC designs. Future work includes: (1) taking into account any unavoidable partitioning of slant edges and critical CDs, and, e.g., incorporating into the ILP objective the minimization of slant edge and critical CD slicing; (2) fast heuristics for reverse-tone fracturing; and (3) adjusting our approach to target non-rectilinear (e.g., X-style [16]) layouts with multiple slant edges.

- **MPW** We have proposed improved algorithms for multi-project reticle floorplanning, wafer shot-map definition, and wafer dicing. Experiments on industry testcases show that our methods significantly outperform previous methods in the literature as well as floorplans manually designed by experienced engineers. Our methods can also be extended to handle additional constraints such as die-alignment constraints imposed by the use of die-to-die mask inspection [29] by merging two copies of a die in a single "super-die". We have also explored the use of multi-project wafers for production under demand uncertainty. We have proposed novel algorithms and methodologies for robust multi-project reticle floorplanning and on-demand wafer dicing, and have shown that our algorithms come close in solution quality to algorithms relying on a priori knowledge of production volumes. In ongoing work we investigate the use of multiple die copies in the reticle, as well as multi-layer reticles, for further reductions in the manufacturing cost of prescribed die production volumes.

- **PSM** A new theory for Bright-Field phase conflict detection has been presented in this thesis. The proposed method greatly simplifies the conflict graph constructed from the layout, with substantial reduction in edge count.

A new property of the graph called *conflict cycles* is introduced, and an optimal algorithm for removing conflict cycles in embedded planar graphs is presented. The algorithm is also generalized so that a minimal solution can be obtained for non-planar graphs. Supporting experimental results demonstrate huge improvements in runtime while maintaining the same quality of results (in terms of number of phase conflicts chosen for correction) as the best available previous work in this area.

A novel layout modification algorithm for standard-cell blocks has also been presented. Experimental results confirm that the new method produces much smaller increases in area than previous work on this problem. The small area increases make the algorithm suitable for use as a post-placement optimization step in true industry production flows. Existing algorithms do not assume that the standard cells used in the placement are phase-assignable. However, the proposed method can also be applied to a placement done with AAPSM-complaint cells and will produce much smaller area increases, when compared to other methods being considered for building phase-assignable placements. The proposed method must be extended to allow feature widening for certain phase conflicts that cannot be solved by increasing the spacing between features. It will also be desirable to integrate the layout modification method with a timing engine since the layout modifications produced by the method can potentially modify timing and cause post-layout timing violations.

- **Redundant Vias** Via doubling is an effective technique for yield improvement, electromigration alleviation, and performance enhancement. This thesis proposes perfect matching based post-routing redundant via insertion, including the use of redundant interconnects for connections to redundant vias, so as to achieve a significantly increased number of doubled vias in a post-detailed routing flow. We propose timing-driven redundant via insertion, based on performance sensitivity of candidate redundant via insertions. Our experimental results show that our proposed perfect matching based

redundant via insertion reduces the number of undoubled (or critical) vias by 99.4% and 98% compared with the two best previous post-routing via doubling heuristics, and increases the via doubling coverage from 94.5% and 98.2% to 99.97%. Our timing-driven redundant via insertion achieves up to 3.3% timing yield improvement compared with timing-oblivious redundant via insertion with the same number of doubled vias on the critical paths.

Bibliography

[1] H. Nakao, M. Terai and K. Moriizumi, "A New Figure Fracturing Algorithm for Variable-Shaped EB Exposure-data Generation," *Electronics and Communication in Japan, Part 3* (83), 2000, pp. 87-102.

[2] S. Shulze, E. Sahouria and E. Miloslavsky, "High Performance Fracturing for Variable Shaped Beam Mask Writing Machines," *Proc. SPIE*, Volume 5130, 2003, pp. 648-659.

[3] T. Asano, T. Asano and H. Imai, "Partitioning a Polygonal Region into Trapezoids," *J. ACM* 33 (1986), pp. 290-312.

[4] H. Imai and T. Asano, "Efficient Algorithms for Geometric Graph Search Problems," *SIAM J. Computing* 15 (1986), pp. 478-494.

[5] T. Ohtsuki, "Minimum Dissection of Rectilinear Regions," *Proc. ISCS*, 1982, pp. 1210-1213.

[6] N. B. Cobb and E. Sahouria, "Hierarchical GDSII Based Fracturing and Job Deck System," *Proc. SPIE*, Volume 4562, 2002, pp. 734-762.

[7] N. B. Cobb and W. Zhang, "High Performance Hierarchical Fracturing," *Proc. SPIE*, Volume 4754, 2002, pp. 91-96.

[8] Y. Granik and N. B. Cobb, "MEEF as a Matrix", *Proc. SPIE*, Volume 4562, 2002, pp. 980-991.

[9] W. Maurer, "Mask Error Enhancement Factor", *Proc. SPIE*, Volume 3996, 2000, pp. 2-7.

[10] F. M. Schellenberg and C. A. Mack, "MEEF in Theory and Practice", *Proc. of SPIE*, Volume 3873, 1999, pp. 189-202.

[11] CPLEX Mixed Integer Optimizer, ILOG. `http://www.cplex.com/`.

[12] Photronics Inc. `http://www.photronics.com/`.

[13] Calibre Fracturem, Mentor Graphics. http://www.mentor.com/calibre/datasheets/mdp/html/.

[14] CATS, Synopsys. `http://www.synopsys.com/products/ntimrg/.`

[15] Mask EDA workshop. `http://www.sematech.org/meetings/archives /litho/mask/20010711/A_INTRO.pdf.`

[16] X Initiative. `http://www.xinitiative.org/.`

[17] M. Andersson, C. Levcopoulos and J. Gudmundsson, "Chips on Wafers", *Proc. WADS (Workshop on Algorithms and Data Structures)*, August 2003, pp. 412-423.

[18] M. Andersson, C. Levcopoulos and J. Gudmundsson, "Chips on Wafers", *Computational Geometry - Theory and Applications* 30(2) (2005), pp. 95-111.

[19] A. Balasinski, "Multi-Layer and Multi-Product Masks: Cost Reduction Methodology", *Proc. SPIE*, Volume 5567, 2004, pp. 351-359.

[20] A. Caprara, A. Lodi and M. Monaci, "An Approximation Scheme for the Two-Stage, Two-Dimensional Bin Packing Problem", *Lecture Notes in Computer Science 2337*, Springer-Verlag, 2002, pp. 320-334.

[21] S. Chen and E. C. Lynn, "Effective Placement of Chips on a Shuttle Mask", *Proc. SPIE*, Volume 5130, 2003, pp. 681-688.

[22] C. Chien and J. Deng, "Optimization of Wafer Exposure Patterns Using A Two-dimensional Cutting Algorithm", *Intl. Trans. in Operational Research* 8(5) (2001), pp. 535-545.

[23] A. B. Kahng, I. I. Mandoiu, Q. Wang, X. Xu and A. Zelikovsky, "Multi-Project Reticle Floorplanning and Wafer Dicing", *Proc. Intl. Symp. on Physical Design*, April 2004, pp. 70-77.

[24] A. B. Kahng, I. I. Mandoiu, X. Xu and A. Zelikovsky, "Yield-Driven Multi-Project Reticle Design and Wafer Dicing," *Proc. SPIE*, Volume 5992, 2005, pp. 1247-1257.

[25] A. B. Kahng and S. Reda, "Reticle Floorplanning With Guaranteed Yield for Multi-Project Wafers", *Proc. International Conference On Computer Design*, October 2004, pp. 106-110.

[26] R. D. Morse, "Multi-Project Wafers: Not Just for Million Dollar Mask Sets", *Proc. SPIE*, Volume 5043, 2003, pp. 100-113.

[27] M.-C. Wu and R.-B. Lin, "A Comparative Study on Dicing of Multiple Project Wafers", *Proc. ISVLSI*, 2005, pp. 314-315.

[28] M.-C. Wu and R.-B. Lin, "Reticle Floorplanning and Wafer Dicing for Multiple Project Wafers", *Proc. Intl. Symposium on Quality Electronic Design*, 2005, pp. 610-615.

[29] G. Xu, R. Tian, M. D. F. Wong and A. Reich, "Shuttle Mask Floorplanning", *Proc. SPIE*, Volume 5256, 2003, pp. 185-194.

[30] G. Xu, R. Tian, D. Z. Pan and M. D. F. Wong "A Multi-Objective Floorplanner for Shuttle Mask Optimization", *Proc. SPIE*, Volume 5567, 2004, pp. 340-350.

[31] G. Xu, R. Tian, D. Z. Pan and M. D. F. Wong "CMP Aware Shuttle Mask Floorplanning", *Proc. Asia South Pacific Design Automation Conference*, 2005, pp. 1111-1114.

[32] Circuits Multi-Projets, `http://cmp.imag.fr/`.

[33] L. W. Liebmann, T. H. Newman, R. A. Ferguson, R. M. Martino, A. F. Molless, M. O. Neisser and J. T. Weed, "A Comprehensive Evaluation of Major Phase Shift Mask Technologies for Isolated Gate Structures in Logic Designs," *Proc. SPIE*, Volume 2197, 1994, pp. 612-623.

[34] C. Chiang, A. B. Kahng, S. Sinha, X. Xu and A. Zelikovsky, "Bright-Field AAPSM Conflict Detection and Correction," *Proc. Design Automation and Test Europe*, 2005, pp. 908-913.

[35] C. Chiang, A. B. Kahng, S. Sinha and X. Xu, "Fast and Efficient Phase Conflict Detection and Correction in Standard-Cell Layouts," *Proc. IEEE Intl. Conf. on Computer-Aided Design*, 2005, pp. 149-156.

[36] A. Moniwa, T. Terasawa, N. Hasegawa and S. Okazaki, "Algorithms for Phase-Shift Mask Design with Priority on Shifter Placement," *Japan J. of Applied Physics* (34), 1993, pp. 6584-6589.

[37] K. Ooi, S. Hara and K. Koyama, "Computer-Aided Design Software for Designing Phase-Shift Masks," *Japan J. of Applied Physics* (32), 1993, pp. 5887-5891.

[38] P. Berman, A. B. Kahng, S. Mantik, I. L. Markov and A. Zelikovsky, "Optimal Phase Conflict Removal for Layout of Dark Field Alternating Phase Shifting Masks," *IEEE Trans. on CAD* 9 (1999), pp. 1265-1278.

[39] A. B. Kahng and Y. C. Pati, "Subwavelength Lithography and its Potential Impact on Design and EDA," *Proc. ACM/IEEE Design Automation Conference*, 1999, pp. 799-804.

[40] A. B. Kahng, S. Vaya and A. Zelikovsky, "New Graph Bipartizations for Double-Exposure, Bright Field Alternating Phase-Shift Mask Layout," *Proc. Asia South Pacific Design Automation Conference*, 2001, pp. 133-138.

[41] C. Pierrat, F. A. Driessen and G. Vandenberghe, "Full Phase-Shifting Methodology for 65nm Node Lithography," *Proc. SPIE*, Volume 5040, 2003, pp. 282-293.

[42] K. Ooi, K. Koyama and M. Kiryu, "Method of Designing Phase-Shifting Masks Utilizing a Compactor," *Japan J. of Applied Physics* 32 (1993), pp. 6774-6778.

[43] A. Moniwa, T. Terasawa, K. Nakajo, J. Sakemi and S. Okazaki, "Heuristic Method for Phase-Conflict Minimization in Automatic Phase- Shift Mask Design," *Japan J. of Applied Physics* 34 (1995), pp. 6584-6589.

[44] K. Cao, J. Hu and M. Cheng, "Layout Modification for Library Cell Alt-PSM Composability," *Proc. SPIE*, Volume 5379, 2004, pp. 253-259.

[45] W. J. Cook, W. H. Cunningham, W. R. Pulleyblank and A. Shrijver, *Combinatorial Optimization*, New York, Wiley Inter-Science, 1998.

[46] W. Cook and A. Rohe, "Computing Minimum-Weight Perfect Matchings," *INFORMS Journal on Computing* 11(2) (1999), pp. 138-148.

[47] L. Liebmann, J. Lund, F. L. Heng and I. Graur, "Enabling Alternating Phase Shifted Mask Designs for a Full Logic Gate Level: Design Rules and Design Rule Checking," *Proc. ACM/IEEE Design Automation Conference*, 2001, pp. 78-84.

[48] P. Ghosh, C. Kang, M. Sanie and J. Huckabay, "PMSLint: Bringing AltPSM Benefits to the IC Design Stage," *Proc. SPIE*, Volume 5042, 2003, pp. 314-325.

[49] R. J. Allen, J. D. Hibbeler and G. E. Tellez, "Use of a Layout-Optimization Tool to Increase the Yield and Reliability of VLSI Designs," *United States Patent* 6,941,528 B2, 2005.

[50] J. Bickford, M. Buhler, J. Hibbeler, J. Koehl, D. Muller, S. Peyer and C. Schulte, "Yield Improvement by Local Wiring Redundancy," *Proc. International Symposium on Quality Electronic Design*, 2006, pp. 473-478.

[51] R. L. Guldi, J. B. Shaw, J. Ritchison, D. L. Corum, S. Oestreich, K. Sherman, J. H. Lin and R. Firdalice, "Characterization of Copper Voids in Damascene Processes," *IEEE Trans. On Semiconductor Manufacturing* 17(4) (2004), pp. 597-602.

[52] K.-Y. Lee and T.-C. Wang, "Post-Routing Redundant Via Insertion for Yield/Reliability Improvement," *Proc. Asia South Pacific Design Automation Conference*, 2006, pp. 303-308.

[53] H. K. S. Leung, "Advanced Routing in Changing Technology Landscape", *Proc. Intl. Symposium on Physical Design*, 2003, pp. 118-121.

[54] D. Z. Pan and M. D. F. Wong, "Manufacturability-Aware Physical Layout Optimizations", *Proc. Intl. Conf. on IC Design and Technology*, May 2005, pp. 149-153.

[55] T. J. Pricer, M. J. Kushner and R. C. Alkire, "Monte Carlo Simulation of the Electrodeposition of Copper II. Acid Sulfate Solution with Blocking Additive", *Journal of The Electrochemical Society* 149(8) (2002), pp. 406-412.

[56] L. Scheffer, "Recommended Rules Not Recommended," *Proc. Electronic Design Processes Workshop*, Monterey, April 2006.

[57] J. Wilson and W. Ng, "Via Doubling to Improve Yield", *Mentor Graphics White Paper*, August 2005.

[58] G. Xu, L.-D. Huang, D. Z. Pan and M. D. F. Wong, "Redundant-Via Enhanced Maze Routing for Yield Improvement," *Proc. Asia South Pacific Design Automation Conference*, 2005, pp. 1148-1151.

[59] H. Yao, Y. Cai, X. Hong and Q. Zhou, "Improved Multilevel Routing with Redundant Via Placement for Yield and Reliability," *Proc. Great Lakes Symposium on VLSI*, 2005, pp. 143-146.